rede e-Tec Brasil

Web Design

Danielle Freitas
Roberto Douglas da Costa

Curso técnico nível médio subsequente
em Informática para Internet

INSTITUTO FEDERAL
Rio Grande do Norte
Campus Avançado Natal - Zona Leste

editoraifrn

Didáticos

Web Design

Danielle Freitas
Roberto Douglas da Costa

Curso técnico nível médio subsequente
em Informática para Internet

Instituto Federal de Educação, Ciência e Tecnologia
do Rio Grande do Norte.

editoraifrn Didáticos

Natal-RN

2022

Presidente da República
Luiz Inácio Lula da Silva

Ministro da Educação
Camilo Sobreira De Santana

Secretário de Educação
Profissional e Tecnológica
Getúlio Marques Ferreira

INSTITUTO FEDERAL
Rio Grande do Norte
Campus Avançado Natal - Zona Leste

Reitor
José Arnóbio de Araújo Filho

Pró-Reitor de Pesquisa e Inovação
Avelino Aldo de Lima Neto

Caderno elaborado em parceria entre o Instituto Federal de Educação, Ciência e Tecnologia e o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

Comitê Editorial da Diretoria de Educacão a Distância e Tecnologias Educacionais - Campus Avançado Natal Zona Leste/IFRN

Presidente
Wagner de Oliveira

Membros
José Roberto Oliveira dos Santos
Albérico Teixeira Canario de Souza
Glácio Gley Menezes de Souza
Wagner Ramos Campos

Suplentes
João Moreno Vilas Boas de Souza Silva
Allen Gardel Dantas de Luna
Josenildo Rufino da Costa
Leonardo dos Santos Feitoza

Equipe | Produção de Material Didático

Equipe de Elaboração
Cognitum

Coordenação Institucional
COTED

Projeto Gráfico
Eduardo Meneses e Fábio Brumana

Revisão ABNT
Francisco de Assis Noberto

Revisão linguística
Maria Tânia Florentino de Sena Nascimento

Revisão Pedagógica
Kalina Alessandra Rodrigues de Paiva

Revisão tipográfica

Diagramação
Alexandre Baccelli
Georgio Nascimento
Joaci De Paula
Leonardo dos Santos Feitoza
Luã Santos
Luanna Canuto
Rômulo França

Ficha catalográfica

I43 Informática para internet: web design. / Organização: Danielle Freitas, Roberto Douglas da Costa, -- 2022.
30 f. ; 30cm.

Guia (Curso Técnico Nível Médio Subsequente). Campus Zona Leste - Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Natal (RN), 2022.

ISBN: 978-65-84831-36-0

1. Educação 2. Guia 3. Educação Profissional 4. Curso Técnico I. Título. II. Vários Autores.

CDU: 004.4’27

Catalogação na publicação pelo Bibliotecário-Documentalista
Ezequiel da Costa Soares Neto CRB15/613
Biblioteca Sebastião Názaro do Nascimento (BSNN) / Campus Natal-Zona Leste – IFRN

Sumário

Aula 1 - Fundamentos de Internet e Desenvolvimento Web

Objetivos

Ao final desta aula, você deverá ser capaz de:

Aprender os fundamentos do desenvolvimento Web;

Entender como funciona a internet e os *sites*;

Compreender o que são linguagens de desenvolvimento.

GLOSSÁRIO

GNU - General Public License

Desenvolvendo o conteúdo

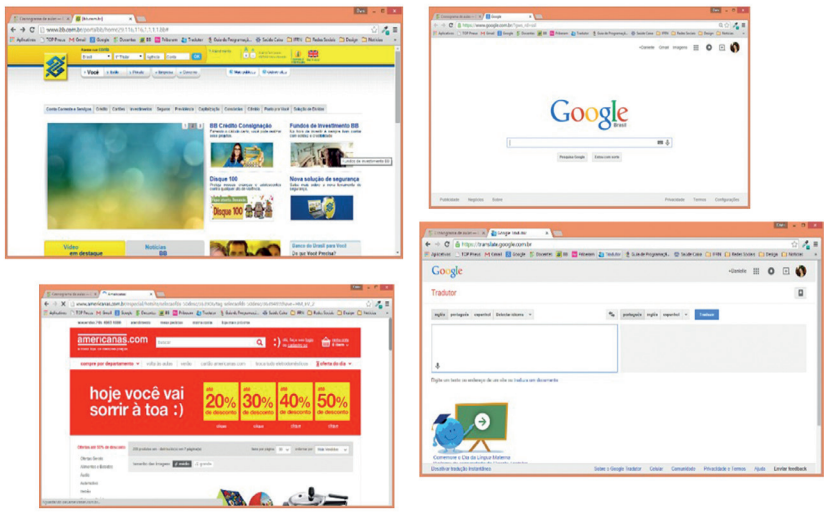
A importância do desenvolvimento *web*

Nos últimos tempos, a internet está mais popular, mais acessível às pessoas. Com isso, muitas aplicações que antes estavam disponíveis para serem instaladas apenas em seus computadores, hoje estão disponíveis para serem acessadas no *site* da própria instituição ou empresa.

Exemplos de aplicações web: comércio eletrônico, serviço de *e-mail*, bate-papo, redes sociais, acesso à conta de banco, serviço de programas de escritório, tradutor, entre outros.

Hoje, o usuário da internet não precisa sair de casa para efetuar pagamentos, para comer, para assistir a filmes de alta qualidade de imagem, por exemplo. Pela internet, é possível pagar todas as contas pelo *site* do banco; comprar um lanche pelo *site* da lanchonete; baixar filmes, seriados; e assistir a programas antigos em alta qualidade na resolução da imagem, através de um serviço

especializado com baixo custo. Veja, abaixo, imagens de sistemas *web* com serviços muito utilizados pelos usuários: <http://google.com>, pesquisa de conteúdo; *site*; <http://translate.google.com.br>, para tradução de texto de diferentes línguas para diferentes línguas; <http://americanas.com>, para compras de produtos; e <http://www.bb.com.br/> para acesso a sua conta bancária e efetuar pagamentos, visualizar extratos, transferir valores, entre outros.



Fonte: http://www.google.com.br/, http://translate.google.com.br/, http://americanas.com/ e http://www.bb.com.br/

Figura 1: Aplicações Web

Design

O *design* pode ser definido como elaboração, concepção, criação, definição de um projeto voltado à determinada função. O termo deriva, originalmente, de *designare*, palavra em latim que logo foi substituída e adaptada para o inglês *design*, porém, esse vocábulo não está voltado, necessariamente, para *web*, pois há *design* para produtos impressos, como convites, cardápios, panfletos; *design* para embalagens; *design* para roupas; *design* para produtos; *design* de interiores; *design* para materiais didáticos por exemplo.

O *design* nos possibilita enviar uma mensagem para o usuário, ou seja, é um meio de comunicação. Pensemos juntos: ao ver um novo produto, você quer testá-lo, entendê-lo, e, dependendo de como ele foi desenhado, estruturado, ele pode ser mais acessível, usável, sendo visivelmente atraente.

Designer

Profissional habilitado para elaborar trabalhos relacionados a design. Exemplos de especializações: *web*, produtos, gráfico, *design* de *moda*, *designer* instrucional etc.

Como a internet funciona

A internet funciona através de padrões que são abertos e permitem que cada rede se conecte a todas às outras redes. Assim, torna-se possível a propagação de conteúdo sem qualquer prévia avaliação.

Para nos conectarmos à internet precisamos de: um computador, um *modem* e uma conexão por rádio, telefone ou cabo. A conexão também pode ser feita por uma rede de celular. Para carregar uma *webpage*, é necessário, primeiramente, a existência de uma URL.

1 - as páginas de internet são encontradas através do uso do protocolo <http>.

2 - <www.google.com.br> é o nome do servidor. A convenção de se usar <www> para se referir a um servidor de internet está lentamente sendo eliminada, mas ainda é utilizada em muitos endereços.

Para encontrar uma página, o computador primeiro conecta-se uma rede de servidores de *Domain Name System* (DNS, sistemas de nomes de domínios, em português) que funciona como lista telefônica. Assim, é listada a localização de milhões de *websites*. Dessa forma, uma vez encontrado o servidor DNS, podemos localizar o servidor de internet que tem os dados da página. O servidor de internet é apenas um entre milhões que contém bilhões de páginas de internet no mundo. Após localizar a página, o DNS envia sua localização de volta ao computador do usuário.

O acesso à internet permite aos, usuários fazerem *download* e *upload*. O *download* — também chamado de "baixar", "descarregar", "obter", "pegar" — ocorre quando se obtém dados através da rede. Quando se baixa, por exemplo, uma música, um filme, um vídeo ou um programa para ser instalado na máquina. Inclusive, as páginas que são visualizadas pelos navegadores são "baixadas". Já o *upload* é o contrário. É quando se envia algo para rede, como anexar um documento por *e-mail.* Faz-se

upload, também, quando se quer enviar uma foto para alguém ou arquivos em geral. Todos os *sites* disponíveis na *web* foram enviados, ou seja, feito o *upload*, por alguém que enviou os arquivos do *site* para um **servidor**, termo que será explicado mais à frente.

Uma Internet local ou LAN é como pode ser chamada a intranet, termo comum quando se trabalha em empresas. Trata-se de uma rede privada, cujos dados/documentos da empresa só podem ser acessados pelos usuários dessa rede, pois o objetivo é ter segurança. Todos os funcionários têm acesso à internet, mas, para visualizar os documentos da empresa, há restrição. Isso apenas pode ser feito de dentro da instituição ou com chave de acesso. Com esse tipo de rede, todos os computadores ficam interligados, criando-se uma minirrede. Abaixo, são ilustrados vários computadores interligados, criando uma rede local e um computador, ao centro, também chamado servidor, ligado à internet.

Fonte: Imagem adaptada do site http://br.freepik.com/

Figura 2: Servidor ao centro ligado à Internet e computadores em uma rede local

Atividade de aprendizagem 1

1. Além dos serviços proporcionados pela plataforma *web*, os quais vimos nesta aula, liste mais três serviços que podem ser oferecidos apenas na referida plataforma.

2. Explique qual a importância da atividade do *designer* para o usuário.

3. Para se acessar um *site*, em casa, pelo computador, o que é necessário?

World Wide Web (WWW)

Engana-se quem pensa que, para se ter internet, é necessário ter um navegador. Podemos ter acesso via outros programas, tais como programas de bate-papo, rádio ou computação nas nuvens. Enfim, há várias outras formas de acessarmos a internet. A WWW permite, aos usuários de computador, localizar e visualizar documentos baseados em multimídia: textos, imagens, animações, áudios ou vídeos.

Navegadores

A navegação pelas informações dá-se pelo *download* de arquivos de servidores (computadores espalhados por vários lugares do mundo, no qual contém essa informação). Para fazermos uso das páginas *web*, é necessário um programa capaz de exibir os arquivos de forma ordenada e organizar esses arquivos de forma visual. Por exemplo:

Opera, Mozilla Firefox, Internet Explorer, Safari e Google Chrome.

Fonte: http://www.opera.com/pt-br, https://www.mozilla.org/pt-BR/firefox/new/, http://windows.microsoft.com/pt-br/internet-explorer/download-ie, https://www.apple.com/br/safari/ e http://www.google.com.br/chrome/browser/desktop/index.html

Figura 3: Logotipos de navegadores

Site e páginas

Uma página contém informações com texto e/ou imagens que são acessadas pelos navegadores. Já *site* ou *website* é um conjunto de páginas. A página

inicial é chamada de *Home*, página principal ou mesmo página inicial. Quando colocamos uma URL sem utilizar a barra "/" no endereço do navegador, por exemplo, <www.google.com>, entramos na página inicial do *site*. Muitas vezes, ao fazemos uma pesquisa nos robôs de busca, como o *google*, eles nos levam para uma página interna do site, por isso, para acessarmos a página inicial, clicamos ou no logotipo ou no "*menu*".

Quando navegamos por uma página de um *site* específico, tem-se a mesma URL para todas as outras páginas. Quando se clica em um *link* no qual a URL é diferente da URL do *site* atual, obtemos uma página de outro *site*.

Dispositivos

Atualmente, os *sites* devem ser adaptados para serem visualizados em diferentes dispositivos, chamados de *websites* responsivos. Eles têm esse nome porque o *site* identifica o dispositivo e exibe a diagramação de forma que o usuário tenha uma visualização mais confortável possível. Abaixo, você encontrará uma imagem que ilustra bem essa adaptação. Em uma tela maior, como no computador e *notebook*, tem-se um destaque e três caixas abaixo; na visualização do *tablet*, só se tem duas caixas; e, na visualização do celular, as caixas não ficam lado a lado, mas abaixo uma das outras, inclusive sem texto. Existe desenvolvedor que apenas diminui a proporção, o que não é tão interessante, pois a navegação não fica tão agradável. Além disso, tem desenvolvedor que modifica as imagens, pois diminuindo-as de tamanho, temos pouca percepção dos detalhes, e substitui a imagem por uma mais coerente com o espaço. Podemos desenvolver um *website* responsivo apenas com as linguagens HTML e CSS, as quais estudaremos no curso.

Para entender ainda melhor, entre no *site* <http://www.awwwards.com/websites/responsive-design/> e abra uma página da galeria. Após abrir, na janela do navegador, clique em "restaurar tamanho". Caso esteja "maximizada", no canto da janela do navegador, altere o tamanho da janela. Ao diminuir, você irá perceber que o *site* se adaptará ao novo espaço disponível.

Fonte: http://br.freepik.com/

Figura 4: Interface responsiva

Atividade de aprendizagem 2

1. Explique a diferença entre internet e WWW.

2. Liste mais três navegadores diferentes dos já citados na aula e elabore um pequeno texto sobre o desempenho deles.

3. Considere a seguinte afirmativa: "Quando se acessa um *site* de pesquisa, às vezes, o *link* nos leva para um *site* dentro da página". Agora, comente se essa afirmação está correta e justifique sua resposta.

4. Existem diferentes dispositivos, *tablet*, *smartphone*, monitores, entre outros. Sabendo disso, responda: os *sites* são obrigados a exibirem o conteúdo diagramado para melhor visualização? Justifique.

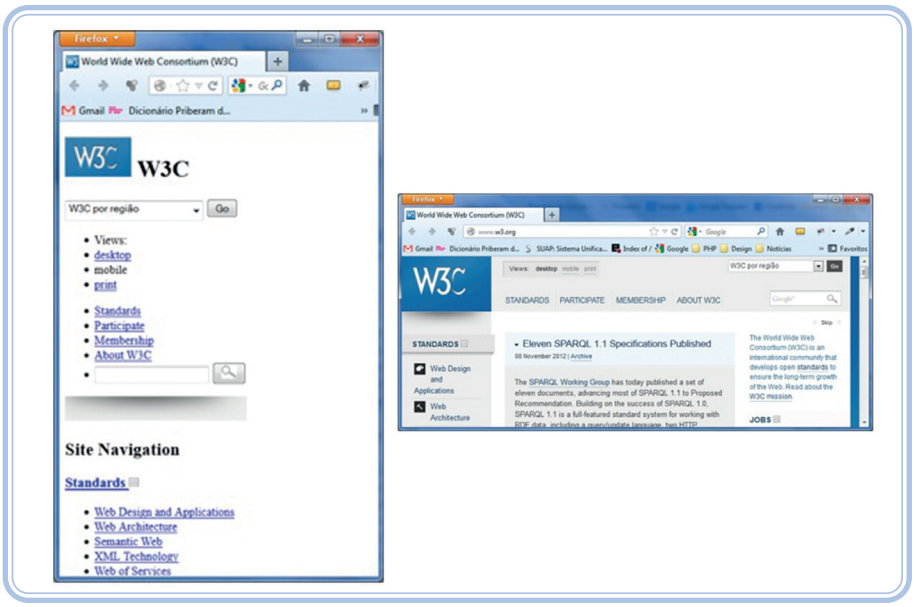
Linguagens HTML e CSS

HTML significa Linguagem de Marcação de Hipertexto — em inglês, *Hyper Text Markup Language*. Essa linguagem tem por objetivo descrever a estrutura, o conteúdo e a relação com outros documentos. Todos os *sites* são programados com a linguagem HTML, além de outras linguagens.

LEMBRE-SE

Na disciplina, trabalharemos com HTML5, que é o HTML na versão e com formatação em CSS. Mas, para simplificar, será chamada apenas de HTML.

A linguagem CSS, que significa Folha de Estilo em Cascata (em inglês, *Cascading Style Sheet*), também é validada pela W3C. A função do CSS é formatar as páginas. Essas formatações contemplam: cores de texto, tamanho da fonte, diagramação da página, alinhamento do texto, cores de fundo, imagens de fundo, entre outros. Veja nas imagens.



Fonte: http://www.w3c.br

Figura 5: Página da W3C com e sem CSS

Para o desenvolvimento em HTML e CSS, é necessário um *software* de edição de texto. Pode ser um simples bloco de notas, mas há outras opções de *softwares* que ajudam no desenvolvimento. Alguns deles são:

- Aptana – *software* gratuito,
- Dreamweaver – *software* pago,
- Notepad++ - *software* gratuito,
- Eclipse – *software* gratuito e
- TopStyle – *software* pago.

Sites estáticos x dinâmicos

Os *sites* podem ser dinâmicos ou estáticos, porém a maioria é dinâmica. Para se programar para *web*, precisamos basicamente do HTML, porque assim teremos como estruturar nossos dados. O CSS nos dará a possibilidade de colocarmos formatação, cores, organizar a diagramação, espaçamentos etc.

Outra linguagem que pode ser usada também é o JavaScript, que possibilita interatividade. Hoje, com o HTML5, temos interatividade também sem necessidade do JavaScript, mas o JavaScript nos dá um leque maior de opções. Essas três linguagens HTML, CSS e JavaScript trabalhando juntas nos possibilitam um bom *site*, porém ainda será necessário trabalhar com uma linguagem servidor para que o desenvolvimento seja profissional, uma vez que, embora utilizando três linguagens, ainda teremos um *site* estático. Essas linguagens são executadas no lado cliente, isto é, no navegador do usuário que acessa a página. Faça um teste você mesmo, seguindo estes passos:

- abra um site qualquer;
- clique com o botão direito sobre a página, em uma área que não haja fotos, vídeos, nem texto, ou seja, uma área vazia;
- procure, nas opções que aparecem, um item similar a "exibir código fonte" ou somente "código fonte", caso não tenha essa opção, tente novamente clicando em outra área.

Depois desse procedimento, você irá visualizar um código cheio de sinais, como estes "<" e ">", mais o texto da página. Esse código disponível é o código HTML, CSS e JavaScript. É uma programação CLIENTE, que deve ser bem escrita, pois ela será interpretada pelo navegador do usuário.

Como cada usuário tem uma preferência de navegador e alguns navegadores interpretam de forma diferente esses códigos, é importante que essa programação CLIENTE seja bem escrita e testada para não haver problemas na hora de visualizar as páginas. Um complicador é que esses códigos podem ser vistos e copiados por qualquer pessoa, existem até algumas formas de dificultar esse acesso, mas, de modo geral, são acessíveis.

Em contrapartida, existem as programações executadas, mas não

interpretadas pelo lado SERVIDOR. Essas são também trabalhadas em conjunto com as linguagens citadas acima, não necessariamente as três juntas, mas, no mínimo, com a linguagem HTML.

Funciona assim: quando o usuário acessa uma página que utiliza uma linguagem servidor, todo o código é executado no servidor e os resultados são enviados para o navegador. Esses *sites* são DINÂMICOS e o navegador exige a página já processada. Essas linhas de código de programação SERVIDOR não podem ser vistas por ninguém, já que não aparecem no código fonte do navegador.

Sobre o servidor, é o local onde ficam armazenado os arquivos que compõem os *sites*. É um supercomputador, na maioria das vezes, que executa esse código. Na realidade, qualquer pessoa pode transformar o seu computador em um servidor, mas serão necessárias algumas configurações e instalações de alguns programas. Você pode colocar o *site* na *web* e ele ser acessado pelo seu próprio computador. Claro que isso tem implicações, como: seu computador precisará ficar ligado direto e, assim, ter ataques malignos, isto é, vírus, e outros problemas.

Na imagem abaixo, aparece o servidor à esquerda com uma página ASPX sendo executada e, à direita, a página em HTML no navegador.

Fonte: Autoria própria

Figura 6: Servidor e página *web*

São exemplos de linguagens servidor: PHP, ASP, JSP, ASPX, ASP.NET etc. Não existe a melhor linguagem, porém a que o desenvolvedor mais se identifica e domina, desde que seja a mais coerente com o nível do projeto. Não

adianta uma linguagem super robusta para desenvolver apenas um sistema de notícias, pois aquela linguagem requer mais código para segurança e organização das camadas de desenvolvimento.

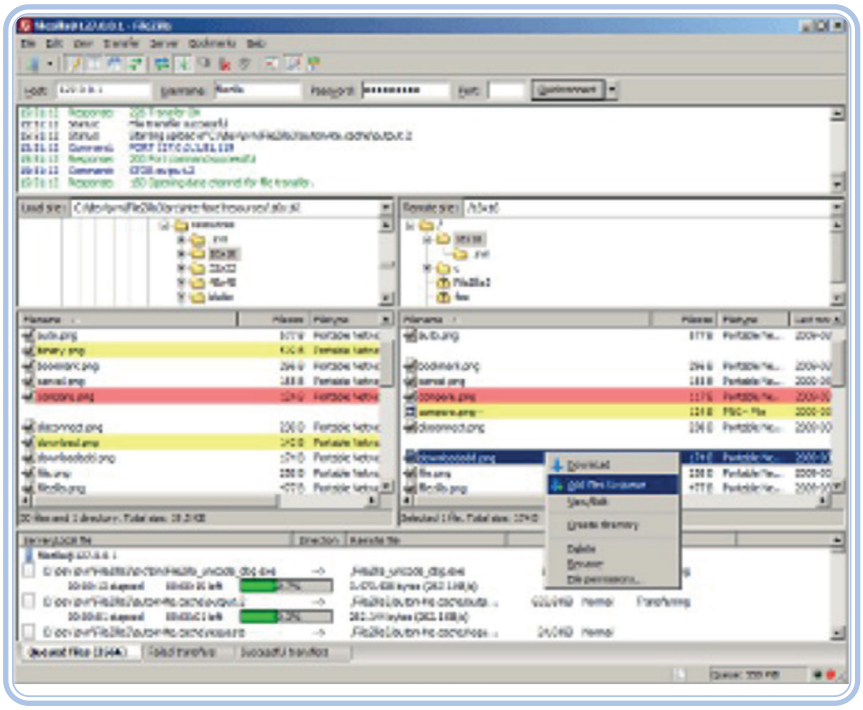
Como colocar um *site* na Web

Para colocar um *site* na internet e permitir que todos tenham acesso, é muito simples: basta contratar um serviço de hospedagem e de registro. Dependendo do serviço de hospedagem, eles já oferecem o serviço de registro. O registro é o nome, url do *site*, por exemplo, <paulofernando.com>. Com isso, é possível verificar a disponibilidade de uma url pelo *site*: registro.br.

O serviço de hospedagem é o servidor no qual o *site* ficará. Lembre-se de que o servidor é esse supercomputador que fica ligado direto e que tem uma supervelocidade e uma boa capacidade de armazenamento, isto é, tem uma espaço bem maior que o de um computador pessoal para guardar dados, arquivos. Se essa máquina for desligada, seu *site* sairá do ar. Existem vários servidores, uns muito bons e mais caros e outros, proporcionalmente, simples e mais baratos. Devemos adequar o perfil do *site* ao servidor, até porque não há necessidade de pagar algo tão robusto se não for usado.

Existem inúmeros serviços de hospedagem, inclusive, com vários preços diferentes, uns com o mesmo serviço que outros e mais barato – algo normal no mercado. Há também o serviço de hospedagem gratuita, mas esses são mais trabalhosos, aparecerão no seu *site* algumas propagandas e, claro, demorará mais para carregar, visto que o serviço é gratuito. Para encontrar esse serviço, é só buscar "hospedagem de *sites*" em um *site* de busca.

Após contratar um serviço de hospedagem e registro, é necessário colocar o *site* no servidor. É possível fazer isso pelo navegador, porém o mais comum é fazer por um programa de FTP, já que, com ele, é possível se conectar com o servidor, colocando os dados de acesso, que são passados pelo serviço de hospedagem e transferir os arquivos do seu computador para o servidor. Depois de colocar os dados de conexão, os arquivos do servidor são visualizados no lado direito enquanto os arquivos do computador são visualizados do lado esquerdo. Logo abaixo, veremos um exemplo de uma tela de um programa de FTP, o Filezilla, que é gratuito e muito simples.



Fonte: https://filezilla-project.org/download.php?type=client

Figura 7: Interface do Filezilla

Para passar um arquivo do computador para o servidor, basta clicar duas vezes no arquivo. Se quiser passar vários arquivos, é só selecionar os arquivos/pastas e arrastar para a direita. Também é possível capturar os arquivos do servidor, já que a ação é a mesma, mas, claro, da direita para a esquerda. Para entender, veja que na imagem acima, há duas colunas: na primeira linha, tem dados de acesso; na segunda, tem o navegador de pastas; na terceira, tem os arquivos, que vocês transferirão e na quarta linha, um retorno do envio dos arquivos. Devemos ter cuidado apenas ao sobrescrever os arquivos, porque, após sobrescritos, não poderão ser resgatados.

Há também ferramentas para se criar o *site* de forma *online*, sem precisar programar, e já hospedar. Essas têm uma estrutura mais fechada, já que não possibilita muitas adaptações, como os *blogs*. Existem serviços bons e gratuitos. Eles já ajudam a desenvolver os *sites* por uma ferramenta *online* e própria deles e hospedam; entretanto, a URL não fica tão exclusiva.

Muitos usuários colocam os *sites* na internet e querem que ele apareça no topo dos resultados dos programas de busca. Existem vários profissionais que vendem garantindo esse serviço e, para conseguir, não é muito difícil.

Para ficar no topo dos resultados de pesquisas, por exemplo, os primeiros resultados que aparecem no Google.com quando você faz uma busca, é

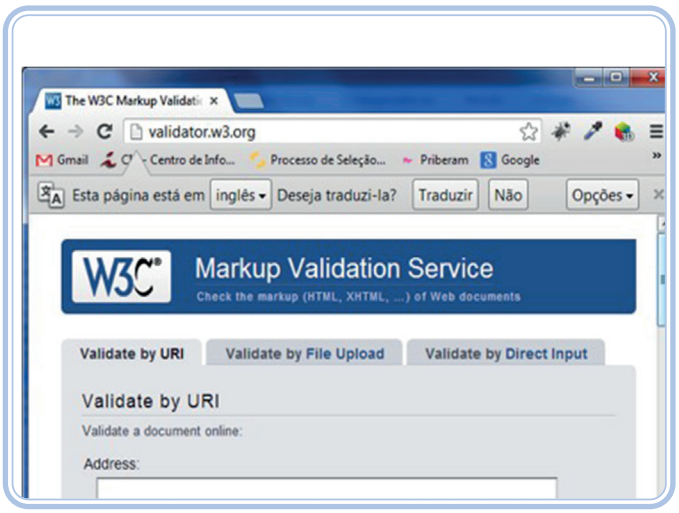
necessário: tempo de registro do domínio, o *site* ser muito acessado, escrever um código padronizado, quantidade de *link*s externos do *site* entre outros pontos. Para saber mais, procurem sobre o SEO.

W3C

A W3C[1] é uma comunidade internacional dedicada a desenvolver tecnologias interoperáveis não proprietárias para a WWW. Um dos objetivos principais é tornar a *web* universalmente **acessível** — independente de deficiência, limitações, linguagens ou culturas dos usuários.

As tecnologias de *web* padronizadas pela W3C são chamadas de Recomendações. Elas definem regras para as tecnologias, que podem ser, por exemplo, HTML, CSS, XML, XSL etc.

Para testar essas recomendações, a W3C disponibiliza um validador (validator.w3.org). Essa validação é recomentada, mas não é perfeita. Por exemplo, uma recomendação é fazer uma descrição textual nas imagens inseridas nas páginas, mas se o desenvolvedor digitar como descrição "imagem", será validado, porém, de fato, essa descrição não cumpriu com sua função, ou seja, a página será validada, mas não será acessível, que é o mais importante. Trabalharemos sempre com essa meta de cumprir as regras da W3C para garantirmos acessibilidade em nossas páginas. Logo, é importante que o desenvolvedor tenha consciência de que essas recomendações são importantes e devem ser cumpridas. Podemos enviar o código de três formas (na ordem das abas da imagem abaixo): pela URL, por *upload* do arquivo ou colando o código HTML.



Fonte: http://www.w3c.br

Figura 8: Validador da W3C

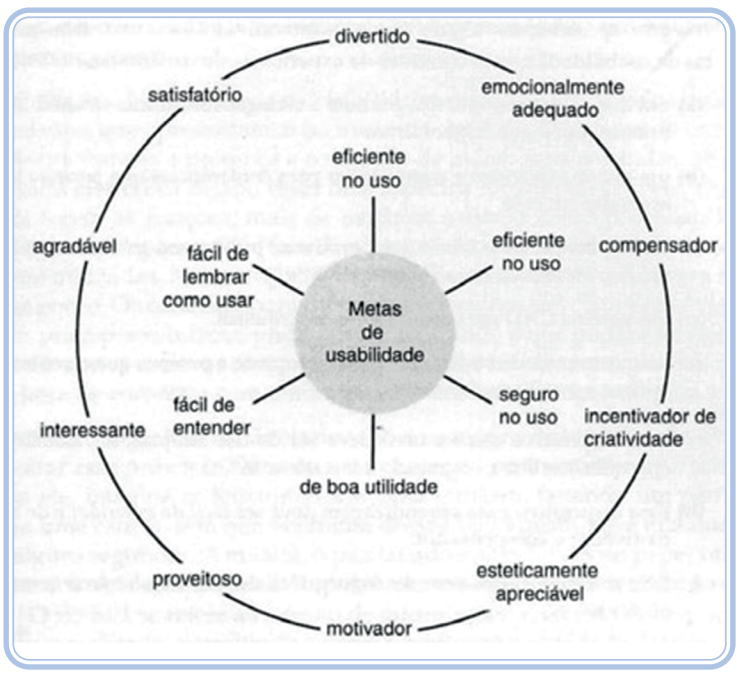
1 <http://www.w3.org/Consortium/>.

Usabilidade

São práticas que devem ser implementadas em tudo: maçaneta de porta, óculos, garrafa de qualquer coisa, página *web*, sistema de computador, caixa de produtos etc. Deve assegurar que um produto, não necessariamente concreto, por exemplo uma página *web*, seja fácil de usar, eficiente e agradável para o usuário. Esse usuário não são todas as pessoas do mundo, e sim, as que fazem uso daquele determinado produto, que pode um público muito diversificado, como por exemplo, bebês, no caso de um produto físico como as fraldas descartáveis, pois fraldas também têm qualidades a serem atingidas.

Todas as pessoas podem passar por situações desagradáveis por não conseguirem usar ou usarem de forma incorreta algum produto ou serviço. Exemplos claros são: sachês de maionese e *ketchup* que não abrem, fone de ouvido que sai da orelha, tampa do *pen drive* que se perde, maçaneta de porta que escorrega, controle remoto que não tem o botão de ligar e desligar sem cor diferente e até mesmo aquele *site* que não responde ao clicar em algum item. Enfim, vários pontos que podem ser estudados para melhorar nosso dia a dia.

Na imagem abaixo, temos explicitadas as metas de usabilidade no centro e as metas decorrentes da experiência do usuário ao redor.



Fonte: Preece, Rogers e Sharp (2005).

Figura 9: Metas de usabilidade e Experiência do usuário

Atividade de aprendizagem 3

1 Após estudarmos sobre os fundamentos da internet, explique qual o papel da linguagem HTML e da linguagem CSS em um *site*.

2 Para desenvolver um *site* dinâmico, quais linguagens são necessárias? Justifique.

3 A W3C obriga os desenvolvedores a trabalharem dentro dos padrões? Por quê?

3 Liste três produtos, objeto(s) e/ou *sites*, com problema(s) de usabilidade e sugira procedimentos para corrigi-los.

RESUMINDO

Nesta aula, aprendemos a importância da internet para a sociedade, partindo dos conceitos de *design* e *designer*, como também a diferença entre internet e WWW e o que são navegadores e como eles nos auxiliam na *web*. Estudamos, ainda, como os *sites* e páginas funcionam, suas diferenças, suas adaptações para diferentes dispositivos, quais linguagens são necessárias para desenvolvê-los. E, ainda, aprendemos como colocar um conteúdo na internet e como permitir que outras pessoas tenham acesso a ele. Para finalizar, tivemos conhecimento sobre o que é necessário para fazer um *site* dinâmico e sobre qual a importância de um *site* com acessibilidade e usabilidade.

Leituras complementares

Para conhecer melhor a W3C, entre no *site* <w3c.org> ou no *site* brasileiro <www.w3c.br> e conheça a organização, os cursos oferecidos, os padrões definidos e as várias publicações a respeito.

Outro assunto muito importante é a usabilidade. No livro KRUG, Steve. **Não me faça pensar**: uma abordagem de bom senso à usabilidade na web. Rio de Janeiro: Altabooks, 2008, você verá a importância do desenvolvedor e do *designer*. Entenda como a usabilidade pode melhorar sua *interface*.

Avaliando seus conhecimentos

Pesquise sobre o termo SEO e discorra sobre a relação desse termo com os Padrões *web*.

Faça uma lista de produtos/*sites* com problemas de usabilidade e explicite solução(ões) para cada um deles.

Faça uma pesquisa sobre o termo *website* responsivo e explique a sua relação com a acessibilidade.

Aula 2 - Sintaxe html, títulos e parágrafos

Objetivos

Ao final desta aula, você deverá ser capaz de:

entender a importância de se trabalhar com HTML dentro dos padrões *web*;

entender o que é a linguagem HTML na versão 5;

aprender as marcações básicas para iniciar uma página *web* simples.

Desenvolvendo o conteúdo

Linguagem HTML
Introdução

HTML é uma abreviação de Hypertext *Markup Language*, que, traduzindo de forma literal, significa Linguagem de Marcação de Hipertexto. Esse termo "hipertexto" significa um texto que leva a outros textos. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc) para a *web*.

Construir *sites* acessíveis a um maior número de usuários e com dispositivos melhor estruturados, com redução de custos de produção e com maior facilidade de manutenção é o objetivo da W3C e também o nosso objetivo. O principal no código sempre é a SEMÂNTICA, a qual atribui coerência ao código escrito, ou seja, significado. Trabalharemos toda a disciplina sob essa perspectiva. Então, fique sempre consciente de que o HTML, responsável pelo desenvolvimento dos *sites web*, pode ser simples de ser programado, mas, se não for bem escrito, esse conteúdo disponibilizado na internet pode não ser tão acessível para os usuários os quais pretende atingir.

HTML é uma linguagem de marcação que descreve a estrutura, o conteúdo e a apresentação do documento e sua relação com outros documentos. É possível, com ela, interligar diferentes tipos de dados multimídia como imagens, sons, vídeos, gráficos, entre outros. Apesar do HTML poder descrever também a formatação, faremos isso apenas com CSS, pois é a forma mais moderna de programar em HTML.

HTML5

O HTML tem várias versões e as mais conhecidas são: HTML 4 e XHTML. Estas foram desenvolvidas pela W3C. Em 2004, um grupo formado por desenvolvedores de navegadores trabalharam em uma nova versão do HTML. Em 2006, a W3C reconheceu o esforço desses desenvolvedores e anunciou a padronização do HTML 5, que é a versão 5, posterior à versão 4, isto é, a versão XHTML que é a junção do HTML e XML, que estava sendo trabalhada na versão 2, mas foi deixada de lado pela W3C, organização esta que define padrões, mas não obriga os desenvolvedores de *sites*, nem de navegadores, nem de programas de desenvolvimento a seguirem com essas regras. Mas são fortemente recomendadas.

Um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva, de maneira que seja transparente para o usuário final. Isto é, uma sintaxe mais simples com mais recursos interativos. Seria mais semântica com menos código. Seria mais interatividade sem a necessidade de instalação de *plugins* e perda de *performance*. Por exemplo, a utilização de arquivos "Flash" dentro da página está cada vez mais escasso, pois, para o mesmo funcionar, é necessária a instalação do *plugin*.

Tags

Os elementos que compõem a estrutura de um documento HTML são denominados de *tag*. As *tags* são também chamadas de marcações HTML. Um documento é composto de dois tipos de texto: as informações que serão exibidas e as *tags* (elementos de marcação) que definirão como os navegadores irão apresentar as informações contidas nas páginas do *site*. Essa apresentação poderá ser, posteriormente, formatada com CSS. Resumindo, *tags* são rótulos usados para informar ao navegador como deve ser apresentado o *website*.

Uma *tag* é sempre precedida de um caractere "<" (menor que) e seguida de

Fonte: Autoria Própria (2015).

um caractere ">"(maior que). Na maioria das vezes, uma *tag* deve ter um correspondente, chamado de *tag* de fechamento. Ou seja, uma *TAG* indica onde começa sua área de influência, enquanto a sua correspondente, *tag* de fechamento, indica onde termina a área de abrangência.

No código abaixo, temos a *tag* que demarca um texto comum, parágrafo, divisão de um texto. Devemos utilizar a *tag* de abertura e a *tag* de fechamento .A *tag* de fechamento se diferencia da de abertura, pelo uso da barra. Dentro delas é colocado o texto "Olá", esse é o texto que será mostrado no navegador, essas marcações/*tags* são apenas para o navegador entender que é um tipo de texto. Há sempre uma forma de delinear uma informação no HTML, títulos, tabela, formulário.

```
1 -     <p>Olá!</p>
```

Outro exemplo abaixo, tempo a *tag* de abertura <h1> e a *tag* de fechamento com a barra. Dentro das marcações é colocado o texto "Web Design. Sempre usar a barra nesse caso.

```
2 -     <h1>Web Design</h1>
```

Todas as páginas na Internet são programadas com HTML. Para exemplificar melhor, abra uma página qualquer, clique com o botão direito do *mouse* e procure a opção "Exibir código fonte", ou algo parecido com o termo "código fonte". Bem, provavelmente, você não entenderá muito, pois algumas páginas têm muitos códigos, mas apenas perceba que existem inúmeras marcações com os caracteres "<" e ">".

Algumas *tags* podem possuir atributos que definem suas características ou propriedades. Sempre incluídos na *tag* inicial, utilizando a sintaxe **nome_atributo="valor"**. Uma *tag* pode ter vários atributos e não há uma ordem específica e esses devem ser separados por espaço, conforme o exemplo abaixo:

```
1 -     …<img src="imagem.gif" alt="bebê" />…
```

Sintaxe básica do HTML

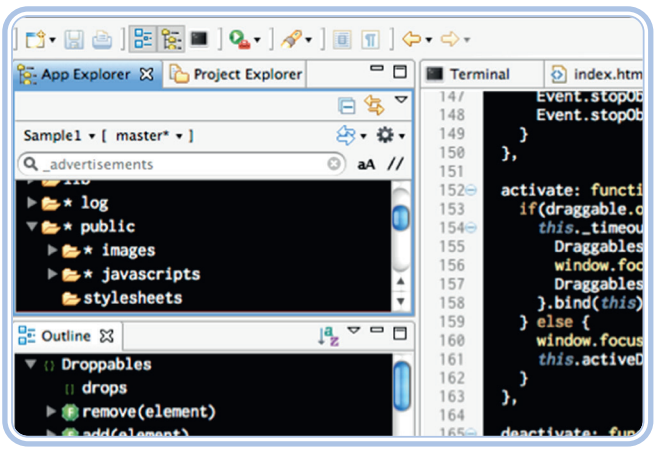
Comparada às versões anteriores, a versão 5 do HTML tem regras menos formais. Por exemplo, as *tags* não precisam mais ser fechadas, mas é

interessante manter esse padrão. Os atributos não precisam ter "=" e os valores entre aspas, entre outros pontos, mas trabalharemos sempre com melhores práticas na nossa disciplina.

As regras mais importantes são: definir o tipo do documento na primeira linha do código, respeitar a semântica das marcações, isto é, utilizá-las de forma coerente com seu significado, por exemplo, a marcação de parágrafo para um texto equivalente a um parágrafo e, por último, formatar as páginas com a linguagem CSS, a qual será explicada melhor na aula 3.

Primeiro código

Para começar, abra um programa coerente com a linguagem, pode ser o notepad ++, aptana ou o próprio bloco de notas, lista de programas descritos no módulo anterior. Sobre isso, veja a imagem a seguir



Fonte: http://www.aptana.com/

Figura 1: Aptana

Digite o código do quadro abaixo no programa escolhido, sem os números das linhas, pois esses foram colocados para ajudar na explicação. É tentador copiar o código e colar no programa, mas se fizer isso estará diminuindo o aprendizado; portanto, sugerimos que diminua o tamanho das janelas, deixando a janela do material na metade da sua tela e a janela do programa na outra metade.

```
1 -     <!DOCTYPE html>
2 -     <html>
3 -     <head>
4 -     <title>Minha primeira página</title>
5 -     </head>
6 -     <body>
7 -     <p>Minha primeira página</p>
8 -     </body>
9 -     </html>
```

1. Crie uma pasta para a disciplina de Web Design, organize os módulos em pastas também, assim será mais fácil de encontrar os arquivos.

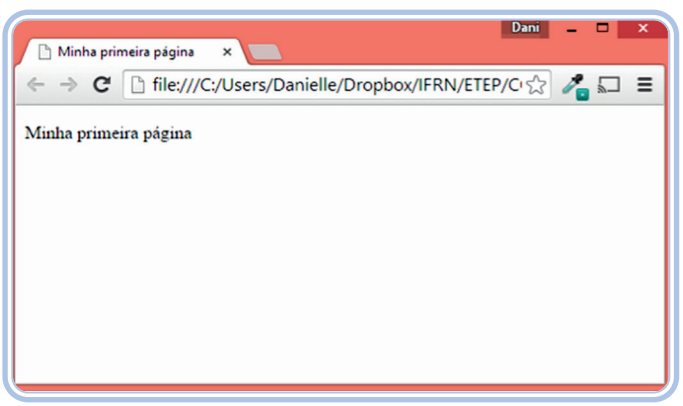
2. Salve o arquivo como "primeirapagina.html", (se for usar o bloco de notas no campo "tipo" escolha a opção "todos os arquivos") sem as aspas. Não colocar acentos e nem espaços nunca, para nenhuma de nossas páginas. Salve na pasta do módulo2.

3. Abra a pasta onde está o arquivo.

4. Clique duas vezes no arquivo.Dessa forma, provavelmente, abrirá no seu navegador padrão e, se você visualizar o texto "minha primeira página", é porque deu certo. Perceba que tudo é em preto e branco, sem alinhamento, sem cores. Todas essas formatações serão feitas com CSS na próxima aula.

A imagem abaixo ilustra a visualização da nossa primeira página no navegador Chrome, mas pode ser usado qualquer navegador.

Vamos entender cada linha do código:



Fonte: Autoria Própria

Figura 2: Primeira página

- Linha 1: definição da versão usada no HTML, mas o HTML 5 tem um cabeçalho bem mais simples que antigamente e é importante manter a coerência de maiúsculas/minúsculas.

- Linha 2: *tag* principal do código HTML que contém todas as informações da página. Perceba que dentro dessa *tag* há outras, mas não há outra tag depois dela. Essa *tag* é fechada na linha 9.

- Linha 3: *tag* de cabeçalho. Nessa *tag* são colocadas informações de importantes para o *site*, por exemplo, descrição, palavra-chave, que ajudarão na indexação do conteúdo pelos *sites* de busca, assim como as chamadas dos arquivos de formatação e de *javascript*. Inúmeras outras informações podem ser colocadas dentro dessa marcação, claro que todas essas informações devem ser postas nas marcações de forma correta. Essa *tag* é fechada na linha 5. A figura abaixo mostra um texto que é definido dentro da *tag* de cabeçalho, esse texto fica na *tag* TITLE, dentro da *tag* HEAD.

- Linha 6: *tag* de corpo. A *tag* de corpo deve ser dentro da *tag* HTML e ser definida depois da *tag* de cabeçalho, exatamente como o código acima. Essa estrutura é básica e deverá ser feita para todas as páginas, cada página com uma estrutura assim. Para entender melhor, ver códigos fonte de *sites*. Todo o conteúdo que é visualizado no navegador deve ficar dentro dessa *tag* de corpo. Essas marcações de informação serão vistas nas próximas seções. Essa *tag* deve ser fechada antes da *tag* HTML, pois as *tags* devem respeitar a hierarquia. O fechamento é na linha 8.

- Linha 7: *tag* de parágrafo. Assim como uma carta, o HTML permite criar parágrafos, para cada parágrafo um par da marcação que é a "<p>".

Fonte: http://portal.ifrn.edu.br

Figura 3: HEAD da página

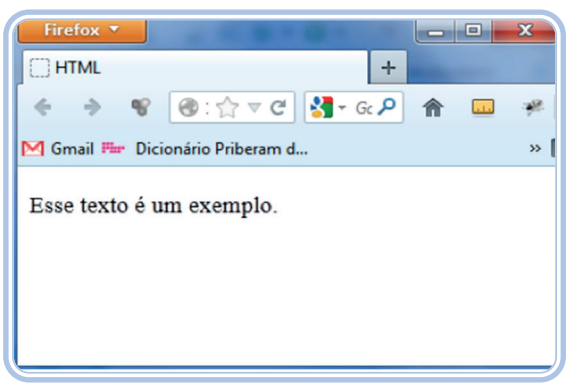
Fonte: http://portal.ifrn.edu.br

Figura 4: BODY da página

No HTML não são considerados vários os espaços ou "*enters*" (chamado de retornos de carro) no código. Por exemplo, no código abaixo mesmo com vários espaços entre as palavras assim como um *enter* separando as duas frases, o texto fica contínuo no navegador. Para que seja dada uma quebra de linha, há uma *tag* específica
.

```
1 -     <p>Esse          texto

2 -               é um exemplo.

3 -     </p>
```

Analise agora como fica, verificando a imagem que segue.



Fonte: Autoria própria

Figura 5: Retorno de carro

Os arquivos HTML podem ter várias extensões, geralmente quando se trabalha a página com linguagens de programação, mas para trabalhar apenas com HTML, a extensão do arquivo é ".html", sempre minúsculo. O nome do arquivo deve sempre ser definido sem espaços e sem acentos, isso porque quando sua página é colocada na internet pode ocorrer erro (não há URL com espaços ou acentos).

Quando se trabalha com linguagens, é importante que o código fique organizado. Para isso é necessário indentar, mas a indentação não fará sua página melhor visualmente e nem influencia na validação, apenas ajuda a entender o código, principalmente, quando há muitas linhas.

Exemplo de dois códigos: um mal indentado e um bem indentado, respectivamente. Perceba o alinhamento das *tags* de iniciais e finais.

```
1 -     …
2 -     <ol>
3 -     <li><img src="imagem.jpg" />
4 -     </li>
5 -     <li>Item teste</li></ol>
6 -     ....
```

```
1 -     …
2 -     <ol>
3 -             <li><img src="imagem.jpg" /></li>
4 -             <li>Item teste</li>
5 -     </ol>
6 -     ....
```

Boas práticas do HTML:

- Todas as *tags* devem ser fechadas, mesmo as *tags* que não têm *tag* de fechamento.

```
1 -     <p>Texto </p>
```

```
2 -     <br />
```

- Todas as *tags* devem ficar em minúsculo (em maiúsculo não influencia em um erro na página, mas não cumprir a sintaxe da linguagem é interessante)

- As *tags* não podem ser sobrepostas, a última aberta deve ser fechada.

```
1 -   <p><a href="http://google.com">Google</p></a>
2 -   <p><a href="http://google.com">Google</a></p>
```

ERRADO

CERTO

- Os atributos devem ter um "=" e o valor entre aspas.

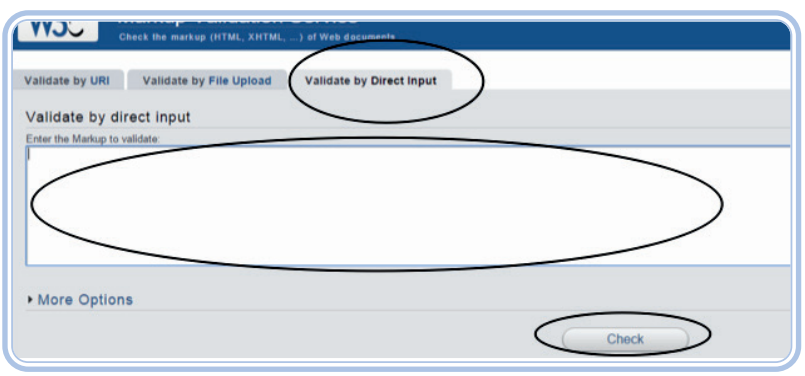
```
1 -   <html xmlns="http://www.w3.org/1999/xhtml"… >
```

Atividade de aprendizagem 1

1. Faça uma página HTML com as *tags* básicas e coloque o seu nome completo na área do corpo.

2. Entre no *site* http://validator.w3.org/

- clique na aba "Validate by direct input";
- copie todo o seu código;
- cole no campo de mensagem da página;
- clique no botão "Check";
- veja se aparece uma tarja verde com o texto: "This document was successfully checked as HTML5!"



Fonte: <http://w3c.org>

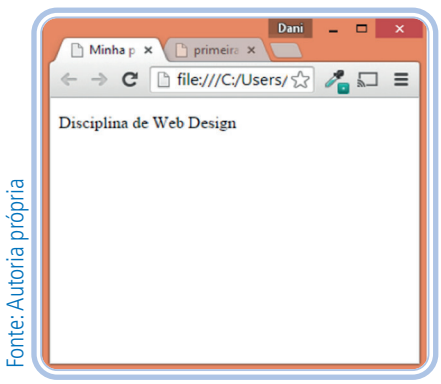
Figura 6: site da w3c.org

Comentários

Informações adicionais que podem ser colocadas no código para melhor compreensão, podem vir em qualquer lugar do código. Essas informações não ficarão visíveis na visualização do navegador, apenas no código da página. Faça o teste. Pegue nosso primeiro código e adicione esse comentário. Verifique, no navegador, se esse texto é mostrado e, depois, verifique o código fonte da página.

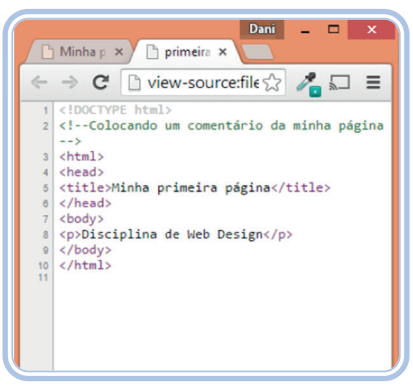
1 - …
2 - <!— colocar aqui o comentário -->
3 - …

Confira nas imagens que seguem.



Fonte: Autoria própria

Figura 7: Página com comentário



Fonte: Autoria própria

Figura 8: Código fonte

Elementos

Considerar que todos os elementos abaixo devem ser colocados dentro da tag BODY.

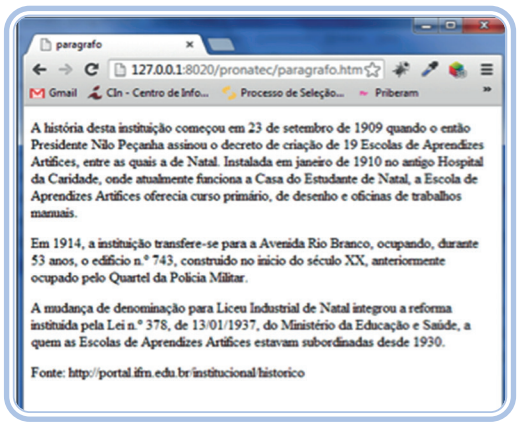
Parágrafo

As *tags* ** ... ** definem um bloco de texto como um parágrafo. Há uma quebra de linha antes e depois dessas *tags*, por padrão.

Para cada parágrafo deve haver um par de *tags " " para início e " "* para o final.

1 - A história desta instituição ...
2 - Em 1914, a instituição...
3 - A mudança de denominação para Liceu...
4 - Fonte: http://portal.ifrn...

Confira na imagem.



Fonte: Autoria própria

Figura 9: parágrafos

Perceba que, por padrão, os navegadores formatam os parágrafos para que eles fiquem distantes um dos outros, Essa formatação padrão é definida pelo navegador, isto é, podemos, com o CSS, configurar qual a distância entre eles, assim como definir um recuo na primeira linha como é padrão de usarmos quando escrevemos no papel. Na realidade, essa prática não é muito comum em páginas *web*. Veja um exemplo na imagem abaixo, retirada do *site* da <globo.com>.

Fonte: http://g1.globo.com/politica/operacao-lava-jato/noticia/2015/03/cpi-da-petrobras-convoca-graca-foster-gabrielli-cervero-e-zelada.html>

Figura 10: site da globo.com

Quebra de linha

A quebra de linha não é uma *tag* que recebe texto, logo ela segue uma sintaxe um pouco diferente do usual. Apenas se usam um espaço e uma barra após a marcação. Essa *tag* tem somente um objetivo: QUEBRAR LINHA, em inglês, *break*, significa quebrar em português; então é só fazer a associação,
 equivale a BREAK....

```
1 -     <p>Olá <br /> Mundo</p>
2 -     ...
```

LEMBRE-SE

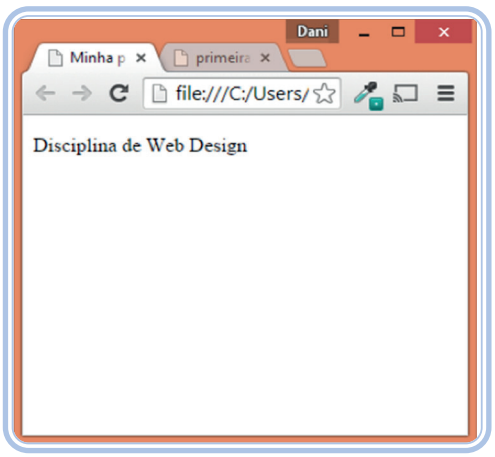
As quebras de linhas não podem ser usadas diretamente BODY, isto é, devem ficar dentro de outras *tags*.

Não é correto colocar várias quebras de linha para se ter espaçamento. Isso deve ser feito com CSS.

Linha horizontal

Linha horizontal em inglês é "*horizontal row*", assim como a quebra de linha, essa *tag* não tem marcação final, apenas o "espaço e uma barra". Seu objetivo é organizar o conteúdo com linhas horizontais. Confira nas imagens a seguir.

```
1 -   ...
2 -   <hr />
...
```



Fonte: Autoria própria

Figura 11: Página Web

LEMBRE-SE

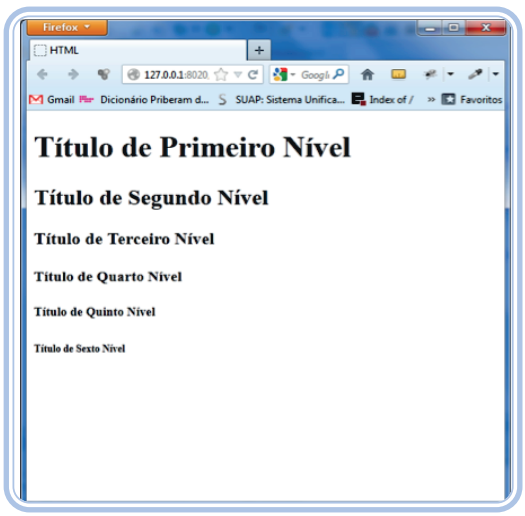
Hoje em dia, as linhas horizontais são substituídas por bordas que são aplicadas com CSS.

Títulos

As *tags* de <h*> ...</h*>, onde * pode ser 1, 2, 3, 4, 5, ou 6, servem para identificar seis níveis de títulos diferentes em um documento, assim como nos livros ocorrem as subdivisões por seções e subseções.

Por consequência e padrão dos navegadores, a formatação dessas marcações de título demonstra uma alteração de fonte, mas essa formatação é uma consequência de sua importância e não deve ser considerado o tamanho da fonte ao delimitar a *tag* e sim a definição de importância

```
1 -   ...
2 -   <h1>Título de Primeiro Nível</h1>
3 -   <h2>Título de Segundo Nível</h2>
4 -   <h3>Título de Terceiro Nível</h3>
5 -   <h4>Título de Quarto Nível</h4>
6 -   <h5>Título de Quinto Nível</h5>
7 -   <h6>Título de Sexto Nível</h6>
8 -   ...
```

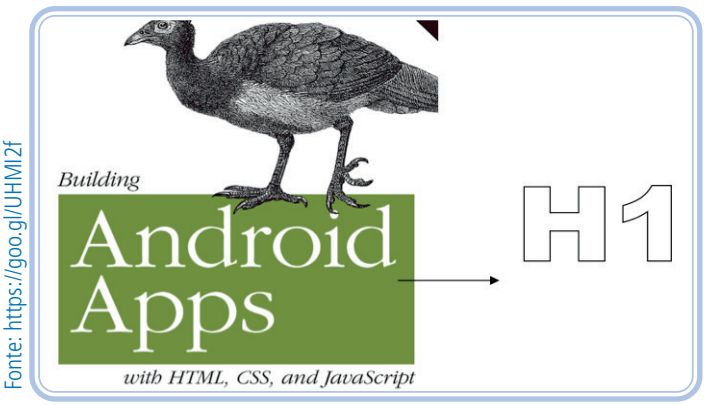


Fonte: Autoria própria

Figura 12: Títulos

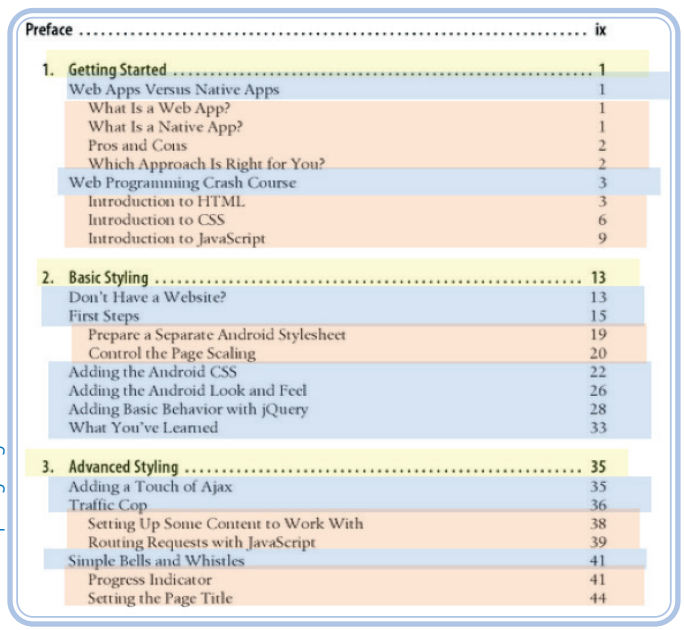
Essas *tags* são muito importantes e nem sempre são utilizadas da forma correta. Alguns desenvolvedores se atrapalham porque o tamanho do texto de cada título que é exibido no navegador, reflete, às vezes, no espaço onde ele é utilizado. MAS ESSA FORMA DE TRABALHAR É ERRADA, pois essa formação padrão do navegador, como já foi dito, pode ser modificada pelo CSS.

Para explanar melhor como usar essas *tags*, veremos dois exemplos. No primeiro exemplo, temos um livro. O título principal seria como a *tag* H1, os outros títulos acompanhariam os outros níveis. Na imagem abaixo, há a capa do livro, onde o título seria o "H1"; na imagem abaixo da capa, há o sumário, em que temos 3 cores. A área destacada com amarelo seria o título "H2", a área destacada com azul seria o nível "H3" e, por fim, a área destacada com laranja seria o título "H4".



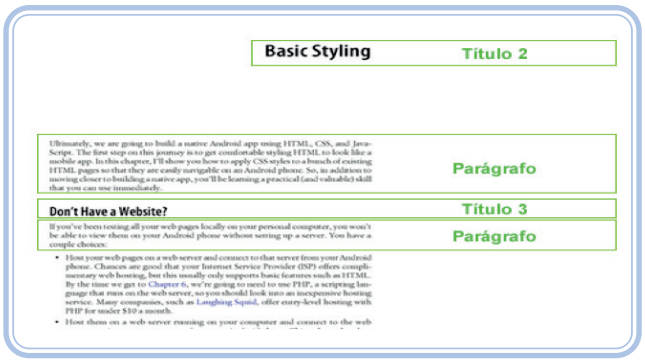
Fonte: https://goo.gl/UHMI2f

Figura 13: Capa de um livro



Fonte: https://goo.gl/UHMI2f

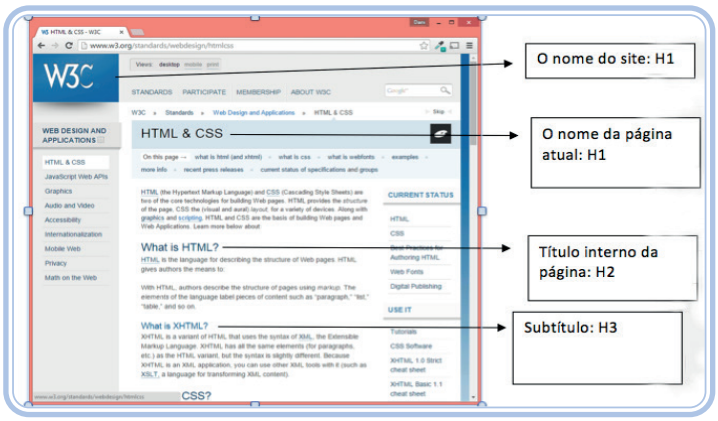
Figura 14:



Fonte: https://goo.gl/UHMI2f

Figura 15: Início de uma seção de um livros

Agora, fazendo uma relação com páginas *web*, na página abaixo, da w3c Brasil, temos o nome do *site* que deve ser sempre no H1, o nome de uma página interna, também H1, e os títulos abaixo que têm tamanhos e cores diferentes, que deixam nítidos que há hierarquia, respectivamente H2 e H3. A página é ampla e há mais texto e títulos abaixo.



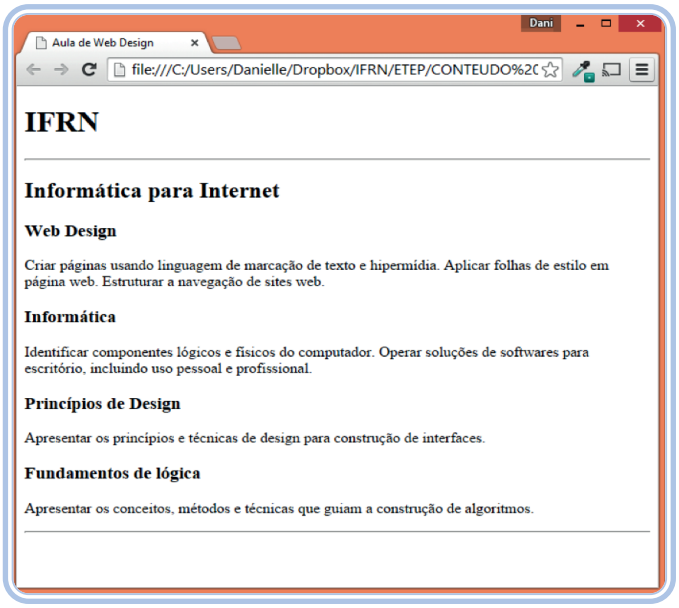
Fonte: http://www.w3.org/standards/webdesign/htmlcss

Figura 16: Página da W3C

Atividade de aprendizagem 2

1. Desenvolva uma página similar à disposta na imagem abaixo.

2. Após a criação, valide a página, conforme foi explicado na atividade anterior.



Fonte: Autoria própria

Figura 17: Página web com títulos e parágrafos

RESUMINDO

Nesta aula, aprendemos a importância da linguagem HTML e a diferença entre tal linguagem e a versão 5. Aprendemos o que são marcações, comentários e as marcações básicas para desenvolver uma página Web: html, body, head, title, p, br, hr, h1, h2, h3, h4, h5 e h6.

Leituras complementares

A importância no envolvimento *web* bem escrito reflete na acessibilidade das páginas, esses padrões são definidos pela W3C. Leia mais sobre acessibilidade no *link* abaixo:

W3C BRASIL. **Cartilha de acessibilidade na web**. [201-?]. Disponível em: <http://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.html>. Acesso em: 24 jun. 2015.

Avaliando seus conhecimentos

Desenvolva uma página simples com informações de uma cidade, visando o desenvolvimento do turismo. Inclua apenas texto, organize com títulos, parágrafos e linhas horizontais. Inclua informações como: história da cidade, política, pontos turísticos, entre outros.

Aula 3 - Sintaxe CSS

Objetivos

Ao final desta aula, você deverá ser capaz de:

entender a importância de se separar a informação da formatação;

incluir a formatação nas páginas;

realizar formatações básicas de fontes, texto, cores de fundo, espaçamentos, bordas e dimensões.

Desenvolvendo o conteúdo

Introdução

CSS significa Cascading Style Sheets (Folhas de Estilo em Cascata). É uma linguagem desenvolvida para formatar os documentos HTML/XHTML/HTML5, em qualquer versão. Tem como maior característica a possibilidade de controle sobre os atributos tipográficos do *site*, como tamanho, cores, fontes, espaçamentos, dentre outros. Com o CSS também é possível trabalhar a diagramação, isto é, definir elementos que ficarão ao lado de outros, como em colunas, e definir elementos sobrepostos.

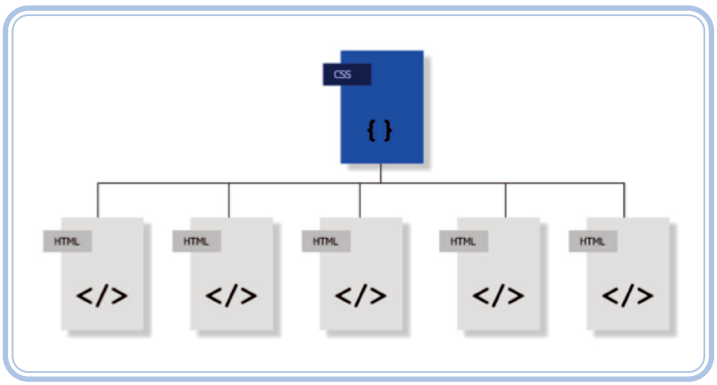
A nova versão do CSS, que é a terceira, traz novos atributos, isto é, novas formatações. Essas acompanham o objetivo do HTML 5, que é o de aumentar a interatividade e tomar as páginas menos dependentes de plug-ins e outras linguagens para o desenvolvimento mais aprimorado, como, por exemplo, efeitos de animação de um objeto.

O CSS trabalha vinculando seu arquivo de formatação a todas as páginas

do seu *site*. Um *site* pode ter várias páginas e pode ter vários arquivos CSS vinculados a ele. No exemplo abaixo, é ilustrado um *site* com apenas 5 páginas e um arquivo CSS. Perceba que todas as páginas estão vinculadas a este arquivo. Se você mudar uma linha deste arquivo CSS, as consequências de formatação podem ser visualizadas em todas as páginas.

LEMBRE-SE

Aprenderemos, nesta aula, como vincular CSS ao HTML. Para o exemplo da imagem abaixo, o tipo de inclusão é externa.



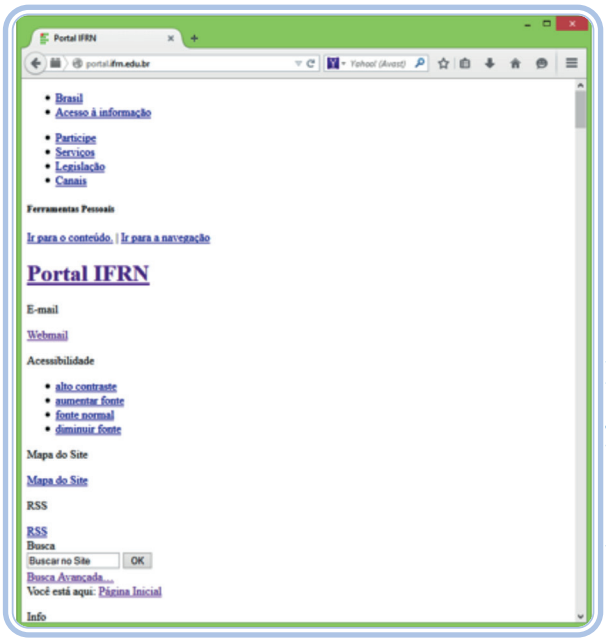
Fonte: Autoria própria.

Figura 1: Páginas com vínculo no CSS

O CSS vai separar toda a informação da formatação do *site*. Para exemplificar melhor, vamos visualizar o *site* do IFRN com a formatação CSS e sem a formatação CSS, respectivamente, nas imagens abaixo.

Fonte: <http://portal.ifrn.edu.br>

Figura 2: Interface do portal do IFRN com CSS

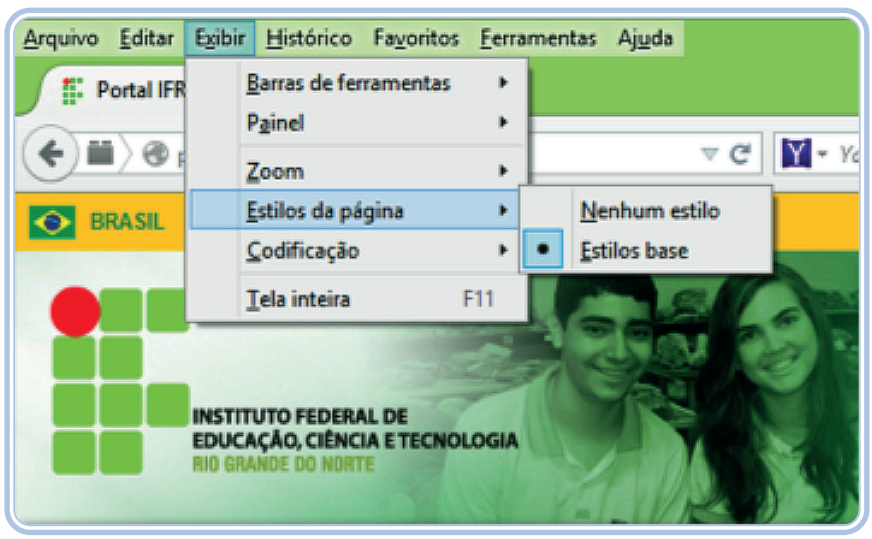


Fonte: <http://portal.ifrn.edu.br>

Figura 3: Interface do portal do IFRN sem CSS

Para fazer o teste sugerido a seguir, o aluno precisará do navegador Mozilla Firefox, cujo ***download*** é gratuito.

Para fazer o teste, abra o navegador, digite o endereço de qualquer *site* e clique no botão "alt" do teclado. Quando fizer isso, aparecerá um *menu* na parte superior direita da janela, como mostra a imagem abaixo. Procure a opção "Exibir" e, em seguida, "Estilos da página". Para exibir a página com CSS, clique em "Estilo base" e, para exibir a página sem CSS, clique em "Nenhum estilo".



Fonte: <http://portal.ifrn.edu.br>

Figura 4: Menu do navegador para desativar CSS

Faça o teste em outros *sites*. Perceba que alguns deles, quando sem CSS, não mudam muito. É importante observar que, o que fica de formatação nativa é do próprio navegador. Por exemplo, os *link*s sempre ficam sublinhados e na cor azul. Além disso, em todos os navegadores, os títulos ficam com o tamanho de texto diferente e negrito. Isso mostra que há uma formatação nativa dos navegadores para cada componente do HTML.

As imagens das páginas também não mudam quando tiramos o CSS, elas continuam com as mesmas cores. No entanto, pode mudar o tamanho, o alinhamento, aliás, toda a informação da página fica a esquerda, percebeu? Isso porque não há no HTML, exceto na versão 4, uma *tag* que coloque o texto no centro. Já no CSS, podemos colocar o texto no centro, à direita, justificado, etc.

Ademais, perceba que o CSS diagrama a página. Os *sites* muitas vezes são organizados em três colunas, quando tiramos a formatação, só visualizamos texto abaixo de texto. Isso significa que a diagramação de colunas, de largura e de espaçamentos é toda desenvolvida com CSS.

Agora que aprendemos o que é o CSS, vamos aprender quais as vantagens de utilizar essa linguagem, que é totalmente diferente da linguagem HTML.

As principais vantagens são:

Podemos modificar completamente o *design* de um *site* apenas alterando o CSS dele. Para *designs* muito robustos, algumas alterações HTML poderão ser necessárias, mas, se houver uma comunicação entre o desenvolvedor e o *designer*, todos os documentos HTML poderão ser aproveitados para o novo *design*. Para melhor ilustrar, vamos observar os três designs a seguir, que contêm a mesma informação, mas *designs* totalmente diferentes.

Fonte: <http://goo.gl/sK7R4Q> <http://goo.gl/0xncr0> <http://goo.gl/DGRmGA>

Figura 5: Interfaces com o mesmo código HTML e diferentes códigos CSS

Você poderá acessar mais informações no *site* <http://www.csszengarden.com> que disponibiliza um HTML para designers e/ou desenvolvedores criarem um *design* e um CSS.

- A manutenção de um *site* que faz a formatação com CSS é muito mais ágil. Antigamente, trabalhava-se com formatação no próprio código HTML. Comparando as duas formas de desenvolvimento, pode-se afirmar com precisão que a manutenção de um *site* com CSS é muito mais simples e rápida. Vejamos um exemplo prático: Você desenvolveu um *site* e seu cliente não gostou da cor de plano de fundo. Considerando a logística usada para desenvolvimento, alterando apenas um arquivo, especificamente, uma linha, podemos atualizar a cor de plano de fundo do *site*, isto é, todas as páginas.

- Consistência no *design* das páginas. Essa vantagem reflete-se na usabilidade do *site*. Quando entramos na página inicial do *site* e clicamos em um *link* no *menu*, é interessante que o *design* do *site* permaneça semelhante ao da página inicial. Quando isso não acontece, pensamos: "será que mudei de site?". Essa sensação é bastante comum. Por isso, é importante que todas as páginas do *site* tenham uma diagramação e cores similares. Nesse caso, a formatação com CSS ajudará nessa consistência, já que temos como vincular o mesmo CSS para várias páginas e, assim, se alterarmos uma formatação, essa modificação será efetuada em todas.

Sintaxe CSS

A sintaxe CSS é muito simples e totalmente diferente da HTML. Vamos a um exemplo:

```
1 -  body{
2 -  font-size: 12px;
3 -  text-align: center;
4 -  }
5 -  /* para fazer um comentário no código, use esses carateres */
```

Explicando linha a linha:

- Linha 1: Seletor "body": o seletor é a *tag* HTML, sem os sinais "<" e ">", que iremos formatar. No código, a *tag* "body" é a parte principal da página.

- Linha 2: atributo "font-size" e o valor "12px". Esse atributo define que o tamanho da fonte será de 12px. Existem inúmeros atributos CSS, cada um faz uma formatação diferente e tem diferentes valores para o tamanho da fonte. Usa-se um valor inteiro e uma unidade, que pode ser px, de pixels, cm, de centímetros, pt, de pontos, entre outros.

- Linha 3: atributo "text-align" e o valor "center". Segue a mesma sintaxe da linha 2, mas a formatação é outra. Aqui, aplica-se um alinhamento centralizado no texto. Há também outros valores, veremos mais adiante.

- Linha 4: fechamento do bloco com o caractere "}".

- Linha 5: para se fazer um comentário em HTML usa-se os caracteres "<!--->", mas em CSS usa-se "/* comentário aqui */". Esse comentário pode vir em qualquer lugar no documento e não tem limite de quantidade.

A sintaxe CSS é um pouco mais rígida em relação a erros de escrita se comparada à sintaxe HTML. Quando erramos algumas coisas simples em HTML, a página é exibida mesmo assim e, às vezes, nem se nota o erro. Na sintaxe CSS, se houver algum caractere a mais ou a menos, uma formatação pode não ser aplicada.

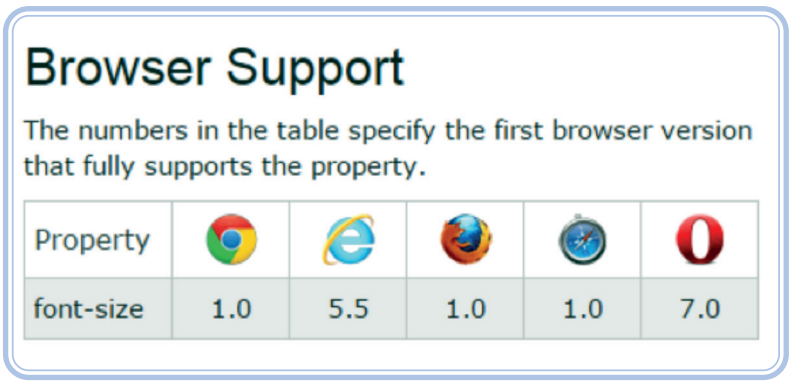
Perceba que a sintaxe usa chaves "{" e "}" para abrir e fechar o bloco. Elas não podem ser esquecidas. Já os seletores, que são as *tags* em HTML, também o são no CSS na maioria das vezes. Contudo, existem exceções. Por exemplo, os seletores não podem vir com "<" e nem com ">" e devem ser sempre válidos, isto é, não podem ser inventados.

Os atributos também não podem ser inventados, já que há uma lista de atributos fixa, que é redefinida e alargada a cada nova versão do CSS. Após o termo do atributo, deve-se usar o caractere ":" e, logo após esse último, o valor, que também deve seguir a sintaxe equivalente ao atributo. Por exemplo, "font-size" não aceita "big", mas aceita "larger". Todos os atributos e seus possíveis valores estão disponíveis no *site* w3c.org e também em w3schools.com, onde a visualização é melhor.

LEMBRE-SE

Como já foi dito na primeira aula, as nossas páginas HTML são interpretadas pelos navegadores, isto é, o navegador tem de entender a linguagem, mas

nem sempre isso acontece. Trabalharemos neste curso com HTML na versão 5 e CSS na versão 3. Nessas versões, há muitas marcações novas para o HTML e também formatações novas para o CSS. Infelizmente, nem todas essas novidades são interpretadas pelos navegadores. No *site* 3Wschools.com, é possível ver como funciona a compatibilidade em relação ao navegador. Na imagem abaixo, há uma parte da página. Ele mostra que o atributo "font-size" é totalmente compatível com os navegadores relacionados. Verifique que são informados os navegadores compatíveis, e, também, a partir de qual versão se dá a compatibilidade.



Fonte: http://www.w3schools.com/cssref/pr_font_font-size.asp

Figura 6: Suporte dos navegadores para os atributos

Por fim, faltam citar as definições de estilo, que são classificadas com relação a sua abrangência e podem ser aplicadas de três formas diferentes: *inline*, incorporada e externa. É o que veremos nas próximas sessões.

Formas de inclusão CSS local: inline e incorporada

Inclusão *inline*

Essa forma de inclusão é pouco usada, pois mistura o código HTML com o CSS. Como nossa meta nesta disciplina é SEMÂNTICA e como, nesse contexto, o CSS tem a finalidade de formatação, é importante separarmos HTML de CSS.

Essa forma de inclusão é mais usada com o Javascript, para interação de comportamento, como no exemplo do quadro abaixo.

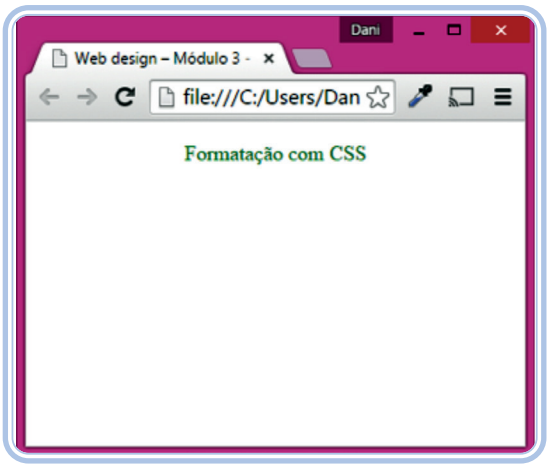
```
1 -  <!DOCTYPE html>
2 -  <html>
3 -  <head>
4 -  <title>Web design – Módulo 3 - CSS</title>
5 -  </head>
6 -  <body>
7 -  <p style="color: green; text-align: center;">Formatação com CSS</p>
8 -  </body>
9 -  </html>
```

Arquivo "inclusaoinline.html"

A seguir, salve essa página como "css1.html" (esse nome é uma sugestão, mas lembre-se de não usar espaços nem caracteres especiais para o nome do arquivo) e coloque em uma pasta que conterá todas as páginas do módulo 3.

Vamos entender o código acima: para formatação *inline,* colocamos a sintaxe CSS na linha da *tag* do HTML. A inclusão do CSS é apenas na *tag* "P". Na Linha 7, adicionamos o atributo "style" e, dentro das aspas, foram colocados os atributos usando a sintaxe CSS, que inclui o caractere ":", o valor e o caractere ";". Nesse trecho, só foram colocados dois atributos: cor de texto, "color", definida como verde (*green*) e alinhamento do texto, "text-align", que definimos como centralizado (*center*). Não há restrição de quantidade, isto é, podemos colocar vários atributos. Perceba que, nesse caso, não usamos seletor, pois é a própria *tag* que recebe a formatação.

É importante que o aluno observe o quão desorganizado fica o código com a definição desse atributo em linha (*inline*). Essa é uma das justificativas para a sua utilização moderada. Abaixo a visualização do código no navegador.



Fonte: Autoria própria.

Figura 7: Página HTML com formatação

Inclusão incorporada

A inclusão incorporada é um pouco mais utilizada que a inclusão *inline*. Vamos a um exemplo.

```
1 -  <!DOCTYPE html>
2 -  <html>
3 -  <head>
4 -  <title>Web design – Módulo 3 - CSS</title>
5 -  <style type="text/css">
6 -  p{
7 -  color: green;
8 -  text-align: center;
9 -  }
10 - </style>
11 - </head>
12 - <body>
13 - <p>Formatação com CSS</p>
14 - </body>
15 - </html>
```

Arquivo "inclusaoincorporada.html"

O modelo incorporado utiliza a *tag* "style", que deve ser definida obrigatoriamente dentro da *tag* "head". Na *tag* "style", escrevemos o código CSS de acordo com sua sintaxe. Na linha 6, temos o seletor "p", que define que a formatação está sendo especificada para o texto do parágrafo. Na linha 7 e 8, aplicamos a cor e o alinhamento do texto.

LEMBRE-SE

Podemos ter mais seletores dentro da *tag* "style", e, dentro do seletor "p", podemos ter mais atributos.

Tanto o código da inclusão *inline* como o da inclusão incorporada têm o mesmo resultado visual. Mas, e se colocássemos mais parágrafos no código HTML, o que aconteceria?

É importante entender que a inclusão *inline* é muito específica. Se nós apenas incluirmos mais parágrafos, sem o atributo "style", eles não terão formatação. No quadro abaixo, suprimimos o código padrão do HTML. Crie esta página com o código completo e salve com o nome "css2.html".

```
1 -   ...
2 -   <body>
3 -   <p style="color: green; text-align: center;">Formatação com CSS com inclusão
inline</p>
4 -   <p>Texto sem atributo style, logo, sem formatação.</p>
5 -   </body>
6 -   ...
```

Fonte: Autoria própria.

Figura 8: Página HTML com formatação CSS inline

Perceba, na figura acima, que apenas o parágrafo de cima, que tinha o atributo "style", ficou formatado. Podemos, em uma página, ter vários parágrafos e destacar algum deles, com uma cor diferente ou com negrito por exemplo. Para isso, podemos usar a formatação *inline* ou uma classificação. Veremos mais adiante como usar essa última.

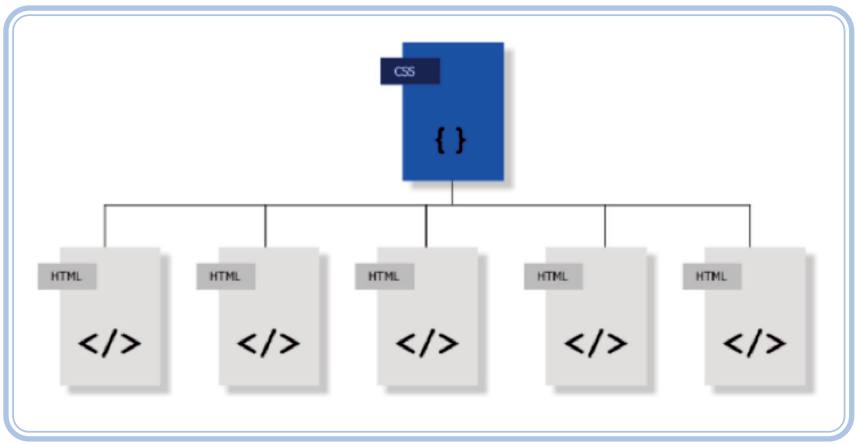
LEMBRE-SE

Evite copiar o código do material. Para um melhor aprendizado, divida a tela do computador em duas janelas e digite o código você mesmo. Atente para o fato de que, ao se copiar o código, o caractere de aspas, às vezes, entra em conflito com o programa utilizado. Logo, se o código não funcionar, isto é, se a formatação não for exibida, simplesmente delete as aspas e digite-as novamente. No entanto, vale a pena repetir: evite copiar o código e colar no programa.

Forma de inclusão externa

O modo de inclusão externa é a única forma que permite aplicação em múltiplas páginas, isto é, temos um arquivo CSS e vários arquivos HTML

com o vínculo para este arquivo, como mostra na imagem abaixo. Para isso, devemos criar um arquivo e salvá-lo com a extensão ".css",



Fonte: Autoria própria.

Figura 9: Páginas HTML vinculadas com um arquivo CS

Existem dois modos de fazer referência ao arquivo CSS: através da *tag* "*link*" e através da regra "@import". Vamos trabalhar mais com a primeira, que é mais usada. Vejamos os exemplos de código abaixo.

```
1 -  <!DOCTYPE html>
2 -  <html>
3 -  <head>
4 -  <title>Web design – Módulo 3 - CSS</title>
5 -  <link href="principal.css" rel="stylesheet" type="text/css" />
6 -  </head>
7 -  <body>
8 -  <p>Formatação com CSS através de vínculo externo</p>
9 -  </body>
10 - </html>
```

Salvar o arquivo como "inclusaocssexterna1.html"

No código acima, na linha 5, temos a *tag* "*link*", que deve vir, obrigatoriamente, dentro da *tag* "head". Aquela tem três atributos: "href" (hiper*link* de referência), que recebe o endereço e nome do arquivo css; "rel" (relação), que descreve a relação do documento HTML atual com o documento de CSS, neste caso folha de estilo e, por último, o atributo "type", "tipo" que especifica o tipo de documento.

```
1 -  <!DOCTYPE html>
2 -  <html>
3 -  <head>
4 -  <title>Web design – Módulo 3 - CSS</title>
5 -  <style type="text/css">
6 -  @import url(principal.css);
7 -  </style>
8 -  </head>
9 -  <body>
10 - <p>Formatação com CSS através de vínculo externo</p>
11 - </body>
12 - </html>
```

Salvar o arquivo como "inclusaocssexterna2.html"

Nesta segunda proposta de inserção, trabalhamos com a *tag* "style", na linha 5. Já na linha 6, é utilizada a regra "@import". Não há diferença entre os dois modos, considerando a parte prática, já que no primeiro modo podemos também criar a *tag* "*link*" e "style" na mesma página.

No exemplo abaixo, o arquivo "principal.css" é um exemplo de nome de arquivo vinculado aos dois códigos. Ele deve ser um único arquivo com o código css.

```
1 -  body{
2 -  font-family: arial, verdana;
3 -  font-size: 12px;
4 -  }
5 -  p{
6 -  color: green;
7 -  }
```

Arquivo "principal.css"

Crie os três arquivos: "inclusaocssexterna1.html", "inclusaocssexterna2.html" e "principal.css" e salve na mesma pasta, "Módulo 3". Teste no navegador e observe se os dois arquivos html aparecerão com a formatação definida: tipo de fonte Arial, tamanho de fonte 12px e cor de texto verde.

LEMBRE-SE

Dentro do arquivo CSS não podemos ter nenhum código HTML. O código deve ser escrito exatamente como acima, pois qualquer erro de digitação pode fazer com que a formatação não funcione.

Quadro 01: Atributos que formatação as fontes

Definição	Atributo	Valores
Define o tipo da fonte (mais detalhes adiante).	font-family	Nomes de fontes entre vírgulas. Exemplo: Arial, Verdana;
Define o tamanho da fonte.	font-size	Valores não usados comumente: medium, xx-small, x--small, small, large, x-large, xx-large, smaller, larger e em %. Mais comumente usado: tamanho e unidade, que pode ser px, cm, etc. Exemplo: 12px;
Define o estilo, basicamente se itálico ou normal.	font-style	"italic" para itálico, "oblique", que é igual a itálico, e "normal".
Permite aplicar a fonte com caixa alta, mas diferenciando as maiúsculas das minúsculas.	font-variant	"normal" e "small-caps".
Aplica negrito a fonte.	font-weight	"normal" e "bold". Há outros valores disponíveis, mas que não mudam a formatação e, por isso, foram desconsiderados.
Especifica todas as propriedades de fonte em uma só declaração.	font	font-style, font-variant, font-weight, font-size, font-family.

Fonte: Adaptado de <http://www.w3schools.com/css/css_font.asp>

Para uma melhor compreensão, aplique a seguinte formatação a uma página e, após isso, teste todos os outros atributos e visualize cada diferença.

```
1 -  body{
2 -  font-family: verdana, "Trebuchet MS", Arial;
3 -  font-size: 12px;
4 -  }
```

Há algo peculiar no atributo "font-family" ("família de fontes" em tradução livre): ele deve receber uma lista de fontes similares. Como você já deve ter aprendido em "princípios de ***design***", no estudo da tipografia, as fontes são organizadas por categorias. Por exemplo, há fontes sem serifas, como Arial; há fontes com serifas, como ***Times New Roman***; há outras fontes que imitam a caligrafia; etc. Como já foi dito, o HTML e CSS são linguagens-clientes, logo, deve-se considerar que a fonte precisa estar instalada no computador do usuário que está acessando um determinado *site*. Assim, as famílias de fontes permitem-nos oferecer mais de uma opção, se a primeira não estiver disponível no computador do usuário, o navegador carrega automaticamente outra opção de fonte.

As fontes mais usadas em páginas *web* são: Arial, Verdana, Helvetica, Times, Courier, Palatino, Garamond, Georgia, Trebuchet MC, Arial Black, Impact, entre outras.

Na versão 3 do CSS, temos a possibilidade de adicionar fontes fora do padrão do sistema. Essas fontes deverão ficar disponíveis no servidor e serem relacionadas com CSS. Veja o exemplo:

```
1 -  @font-face {
2 -  font-family: helveticaneue;
3 -  src: url(helveticaNeueLTStd-UltLt.otf);
4 -  }
5 -  p {
6 -  font:36px helveticaneue, Arial, Tahoma, Sans-serif;
7 -  }
```

Na linha 1, temos a nova regra "font-face", que recebe o atributo "src" (*source*). Na linha 3, tem-se o endereço da fonte, que, nesse caso, está na mesma pasta do arquivo atual de CSS. Caso a fonte esteja em um *site* da internet, deve-se especificar toda a URL do *site*; caso esteja em outra pasta, deve-se especificar o endereço da pasta. Anteriormente, na linha 2, renomeamos a fonte como "helveticaneue" e, mais adiante, na linha 5, nós a definimos para o parágrafo. Já na linha 6, temos a primeira opção de fonte que será carregada para o texto, que é a fonte inserida pela nova regra. Perceba que, mesmo disponibilizando o endereço da fonte, definimos também outras opções de fonte.

Por fim, observe que alguns atributos têm nos valores a opção "normal". Você poderá se perguntar: "- Por que 'normal', se eu quero formatar?" Bem, o que acontece é que, às vezes, alguns elementos já estão formatados por padrão ou foram formatados globalmente. Com a opção "normal", podemos reverter tais formatações. Por exemplo, as *tags* de título, h1, h2, h3... h6, por padrão, são formatadas em negrito. Usando a opção "normal", podemos destacá-las de outra forma, como itálico, letra grande, plano de fundo, etc. Outro exemplo: podemos aplicar negrito ao BODY da página, assim teremos todo o texto do *site* em negrito. Mas, e se quisermos algo em especial, sem negrito? É só aplicar a formatação "font-weight: normal;".

LEMBRE-SE

Em casos de nomes de fontes com mais de uma palavra, deve-se usar as aspas. Por exemplo: "Times New Roman".

Formatando os textos

Abaixo, temos uma lista dos atributos para formatação dos textos.

Quadro 02: Atributos de formatação de texto

Definição	Atributo	Valores
Cor do texto	color	Cor em hexadecimal, nome da cor e outros valores.
Direção do texto (se da direita para a esquerda ou vice-versa).	Direction	Ltr ou Rtl
Espaçamento das letras	letter-spacing	Normal ou valor inteiro seguido da unidade (px, pt, cm, em, etc.). Aceita valor negativo.
Distância entre as linhas	line-height	Valor inteiro seguido da unidade (px, pt, cm, em, etc.) ou em porcentagem.
Alinhamento do texto: esquerda, direita, centro ou justificado.	text-align	Left, Right, Center ou Justify.
Decoração do texto: efeito de sublinhado, linha sobre (superior) o texto, em cima do texto (tachado) ou sem efeito.	text-decoration	Underline, Overline, Line-through ou None.
Espaçamento de margem na primeira linha.	text-indent	Valor inteiro seguido da unidade (px, pt, cm, em, etc.) ou em porcentagem.
Efeito de sombra no texto disponível na versão 3 do CSS.	text-shadow	Valor para deslocamento horizontal, valor para deslocamento vertical, efeito de desfoque, cor e sem sombra. Os valores são inteiros seguidos da unidade (px, pt, cm, em, etc.).
Transforma todos os caracteres em maiúsculos, minúsculos ou as primeiras letras de todas as palavras em maiúsculas.	text-transform	None, capitalize, uppercase ou lowercase.
Espaço entre as palavras.	word-spacing	Valor inteiro seguido da unidade (px, pt, cm, em, etc.) ou em porcentagem.

Fonte: Adaptado de <http://www.w3schools.com/cssref/default.asp>

Algumas observações a respeito do quadro acima:

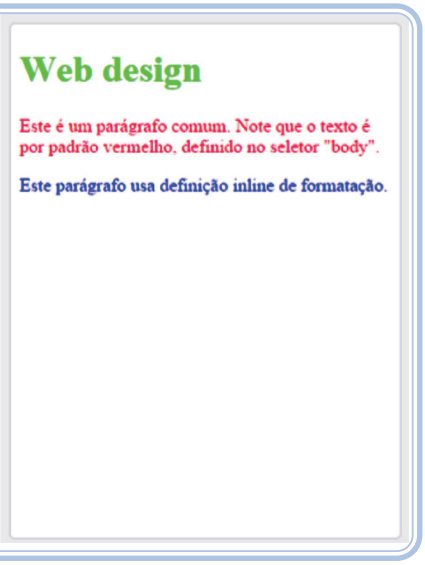
- Os atributos mais utilizados estão destacados em negrito.

- A maioria dos atributos tem mais dois valores que não foram mencionados: "initial" e "inherit", sendo que o primeiro define o valor padrão do elemento e o segundo, o valor do "elemento pai".

- O "text-shadow" faz parte dos novos atributos da versão 3 do CSS. Esses novos atributos trazem mais interatividade para as páginas. Mas é necessário ter cautela ao usá-los, pois alguns deles ainda não são interpretados por algumas versões dos navegadores.

Agora, passemos a alguns testes práticos de uso desses atributos:

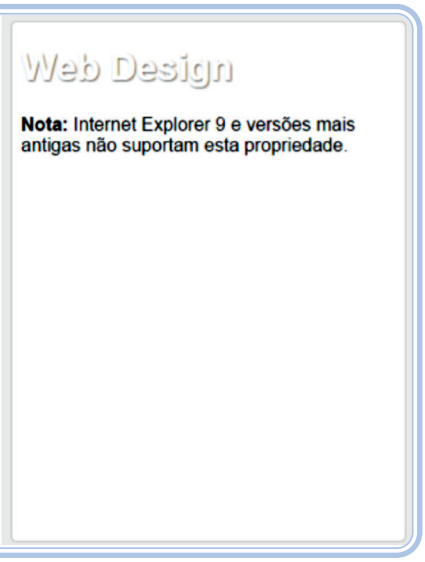
```
<!DOCTYPE html>
<html><head><title>Web design</title>
<style>
body {
    color: red;
}
h1 {
    color: #00ff00;
}

</style>
</head>
<body>

<h1>Web design</h1>
<p>Este é um parágrafo comum. Note que o
texto é por padrão vermelho, definido no
seletor "body".</p>
<p style="color: rgb(0,0,255)">Este
parágrafo usa definição inline de
formatação.</p>

</body>
</html>
```



Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

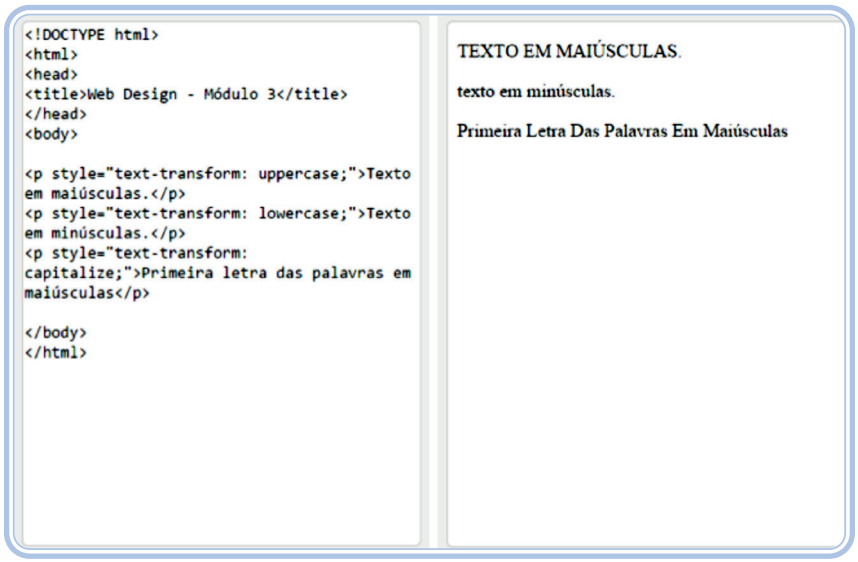
Figura 10: Formatação com o atributo "color"

```
<!DOCTYPE html>
<html>
<head>
<style>
body{font-family: Arial, verdana;}
h1 {
    color: white;
    text-shadow: 1px 1px 2px gray;
}
</style>
</head>
<body>

<h1>Web Design</h1>

<p><b>Nota:</b> Internet Explorer 9 e
versões mais antigas não suportam esta
propriedade.</p>

</body>
</html>
```



Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 11: Formatação com o atributo "text-shadow"



Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 12: Formatação com o atributo "text-transform"

Formatando as cores de plano de fundo

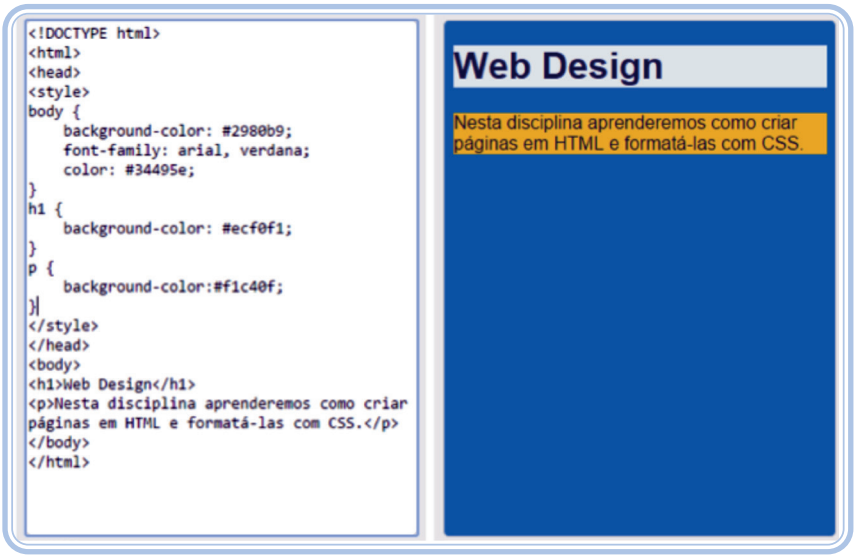
Existem vários atributos para controlar o plano de fundo, geralmente é definido com uma cor sólida, é possível usar uma imagem de plano de fundo e uma cor sólida ao mesmo tempo.

Quadro 03: Atributos para formatação do plano de fundo

Definição	Atributo	Valores
Cor de plano de fundo.	background-color	Valores de cores e transparente
Define imagem de plano de fundo.	background-image	URL da imagem
Repetição da imagem de plano de fundo.	background-repeat	Repeat, repeat-x, repeat-y e no-repeat.
Define se a imagem fica fixa ou se ela rola com o "scroll" da página.	background-attachment	Scroll, fixed e local
Define o posicionamento da imagem.	background-position	Recebe dois valores para: Posição X e Posição Y. Valores: top, bottom, left, right e center.
Recebe os valores dos atributos acima.	background	Na ordem: cor, posição/tamanho, repetição, "attachment" e url da imagem.

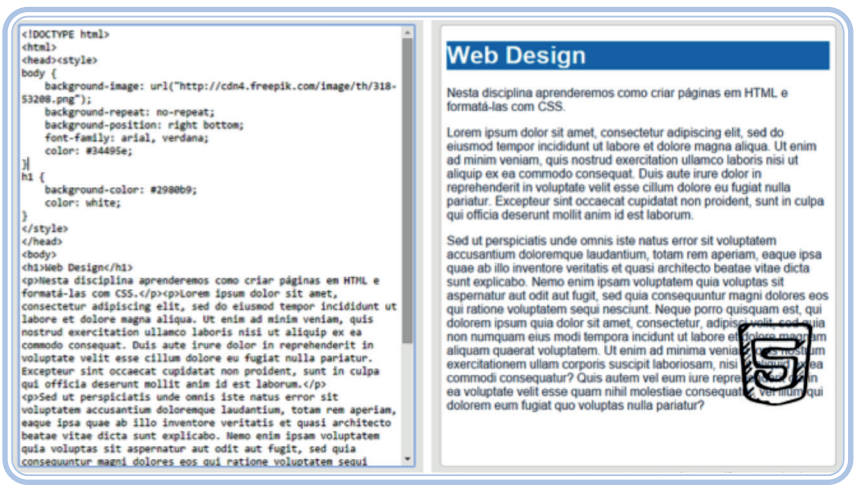
Fonte: Adaptado de <http://www.w3schools.com/cssref/default.asp>

Há mais duas propriedades da nova versão: "background-clip" e "background-origin". Na nova versão, também é possível definir várias imagens de plano de fundo para o mesmo elemento.



Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 13: Formatação com o atributo "background-color"



Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 14: Formatação com os atributos de plano de fundo

Parte do código HTML foi cortado, mas após finalizar o texto, o "p" o "body" e o "html" são fechados.

Para aplicar uma imagem de plano de fundo, pode-se fazer um relacionamento direto com uma imagem da internet ou com uma imagem do computador. Para isso, é preciso colocar as imagens na mesma pasta dos arquivos de página. Fica, ainda mais organizado, colocá-las em uma pasta de imagens e relacioná-las colocando o nome do arquivo "url(nomedoarquivo.jpg)" e sua extensão, JPG, JPEG, GIF ou PNG dentro dos parênteses após o termo "URL".

Com a propriedade "background-position", posicionamos a imagem no lado direito e embaixo. Podemos também centralizar a imagem, aplicando "center". Podemos também definir no topo e à direita, usando "top right". Os valores NÃO podem vir entre vírgulas, devem ser definidos com espaços entre os termos.

Para definirmos que o plano de fundo não deve se repetir aplicamos o atributo "background-repeat" e o valor "no-repeat", caso altere para "repeat-y", a imagem será repetida no eixo Y, verticalmente. No atributo "position", só precisamos definir no valor "left" ou "right".

Observe que, para preencher a página, foi usado um texto muito comum na área do *design*, "Lorem ipsum", esse texto é apenas para preencher a área. Na internet, tem o texto para você copiar.

No HTML e CSS, ainda não é possível transformar a imagem de plano de fundo. Caso queira diminuir, cortar uma parte, modificar cores, entre outros efeitos, é necessário fazer o *download* da imagem e utilizar um programa de edição de Imagem como o ***photoshop***, ***gimp***, entre outros.

LEMBRE-SE

Vocês podem baixar as imagens da internet e salvar no seu computador. Geralmente, as imagens de plano de fundo são imagens elaboradas modificadas em programas de *design*. Tenham cautela ao fazer o *download* de algumas imagens da internet, essas podem ter direitos autorais. Na dúvida, sempre utilize *sites* que tenham imagens gratuitas.

Sites com imagens gratuitas:

FREEPIK. Disponível em: <http://www.freepik.com/>

FREEIMAGES. Disponível em: <http://www.freeimages.com/ >

Formatando com bordas

Todas as marcações podem receber uma borda para delinear sua área. Existem vários atributos CSS para controlar essas bordas, veja na tabela abaixo.

Quadro 04: Atributos para formatação das bordas

Definição	Atributo	Valores
Largura da borda	border-width	Valor inteiro com unidade em px, pt, cm, etc ou em porcentagem.
Estilo da borda	border-style	None, hidden, dotted, dashed, **solid**, double, groove, ridge, inset e outset
Cor da borda	border-color	Nome da cor, RGB da cor ou hexadecimal da cor.
Borda em todos os lados	Border	Recebe os três valores dos 3 atributos "border-width", "border-style" e "border-color" separados por espaço.
Define a borda de um lado específico	Border-right, border-left, border-top e border-bottom	Recebe os três valores dos 3 atributos "border-width", "border-style" e "border-color" separados por espaço.

Fonte: Adaptado de <http://www.w3schools.com/cssref/default.asp>

Para aplicar a borda, é possível trabalhar com os atributos "border-width", "border-style" e "border-color" ou, de forma mais prática, apenas com o atributo "border". Isto é, podemos trabalhar assim:

```
1 -  P{
2 -  border-width: 2px;
3 -  border-style: solid;
4 -  border-color: blue;
5 -  }
```

Ou assim:

```
1 -  P{
2 -  border: 2px solid blue;
3 -  }
```

As duas formas darão o MESMO resultado. Se pudermos economizar linha, melhor.

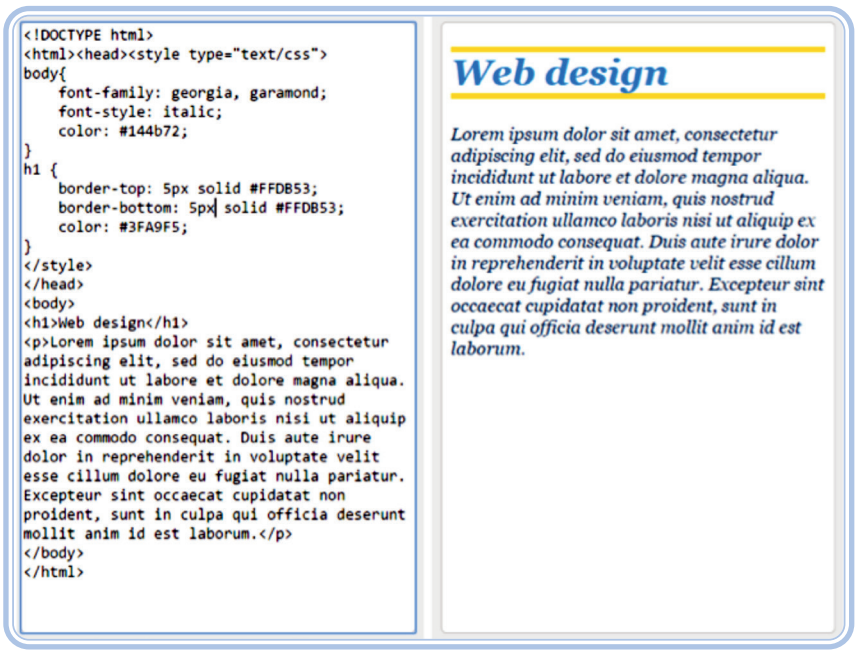
Podemos ser mais específicos e só definir borda superior, borda direita, borda inferior e borda esquerda ou mesclar e definir apenas borda inferior e superior, enfim, é possível definirmos apenas um lado das bordas ou alguns lados da borda.

Existem vários estilos de borda. Mas o estilo mais usado é o "solid", define uma linha sem detalhes, simples.

A *tag* que usávamos para linha horizontal, "hr", agora poderá ser substituída por "border". Imagine um projeto em que você coloca um "hr" no código, depois de um tempo você quer modificar esse *design*, o que é mais vantajoso,

abrir UM arquivo CSS e modificar o atributo "border" ou abrir TODAS as páginas que tem o "hr" e removê-lo?

Podemos aplicar bordas em qualquer marcação do HTML, inclusive a marcação BODY. Abaixo, um exemplo de utilização do atributo, aplicado apenas ao elemento H1.



```
<!DOCTYPE html>
<html><head><style type="text/css">
body{
    font-family: georgia, garamond;
    font-style: italic;
    color: #144b72;
}
h1 {
    border-top: 5px solid #FFDB53;
    border-bottom: 5px solid #FFDB53;
    color: #3FA9F5;
}
</style>
</head>
<body>
<h1>Web design</h1>
<p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.</p>
</body>
</html>
```


Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 15: Formatação com o atributo de bordas

Formatando com dimensões

A maioria das marcações aceitam a formatação de dimensões, as marcações que não aceitam por padrão são as marcações do tipo "inline", mas toda marcação "inline" pode ser definida como "block". Veremos melhor como esse atributo trabalha na próxima aula.

Quadro 05: Atributos de dimensões

Definição	Atributo	Valores
Altura do elemento	height	Todos recebem um valor inteiro com unidade em px, pt, cm, etc ou em porcentagem.
Altura máxima	max-height	
Largura máxima	max-width	
Altura mínima	min-height	
Largura mínima	min-width	
Largura	width	

Fonte: Adaptado de <http://www.w3schools.com/cssref/default.asp>

Fonte: Adaptado de < http://www.w3schools.com/cssref/default.asp>

Figura 16: Formatação com bordas

No exemplo acima, foram aplicados largura e altura no parágrafo e para visualizarmos a área do parágrafo foi aplicada uma borda.

Podemos definir uma área para as marcações, delimitando altura e largura, além da possibilidade de definirmos um valor de largura máxima, largura mínima, assim como altura máxima e altura mínima.

Essas novas propriedades permitem a flexibilidade do conteúdo. Perceba, no momento em que colocamos largura em um texto, limitamos sua área e se o texto aumentar? Se for uma página dinâmica onde o texto inserido por programação, isto é, o conteúdo vem do banco de dados e não for possível prever o tamanho desse texto? Por isso, a altura mínima, por exemplo, pode ser aplicada. O texto ficará na área definida.

Antigamente, era comum ter uma página com conteúdo delimitando área fixa e tínhamos de usar uma barra de rolagem dentro dessa caixa para poder navegar, lembra? Fizemos uma adaptação no *site* do IF só para vocês entenderem. Essa área do conteúdo tem uma barra de rolagem para visualizarmos o conteúdo completo. Essa técnica foi muito usada, mas hoje não tem sentido termos uma área tão grande na tela e as páginas limitarem nossa área de leitura. Compreendem? Se estamos visualizando a página em um computador, temos uma área considerável para leitura. Mas, mesmo assim, alguns *sites* limitam nossa visualização. Com o atributo de altura mínima, podemos organizar o ***layout*** e, ao mesmo tempo, adaptar o texto à área que temos.

Fonte: http://portal.ifrn.edu.br/campus/reitoria/noticias/search?SearchableText=resultado+de+requerimento+de

Figura 17: Demarcação de altura em uma página

Mas então, para que trabalhar com limite de largura ou altura? Temos de unir o bom *design* a um bom código. É importante fazermos uma diagramação que seja confortável para leitura, pois assim como é desagradável ter de usar barra de rolagem para visualizar todo o conteúdo, também é desagradável ter uma linha de leitura muito extensa, pois quando temos, em uma leitura de livro, uma linha muito extensa, nossos olhos perdem o foco na hora de voltar e continuamos, às vezes, na mesma linha ou pulamos uma linha. Da mesma forma, pode acontecer na diagramação. Por isso, é importante delimitarmos as áreas.

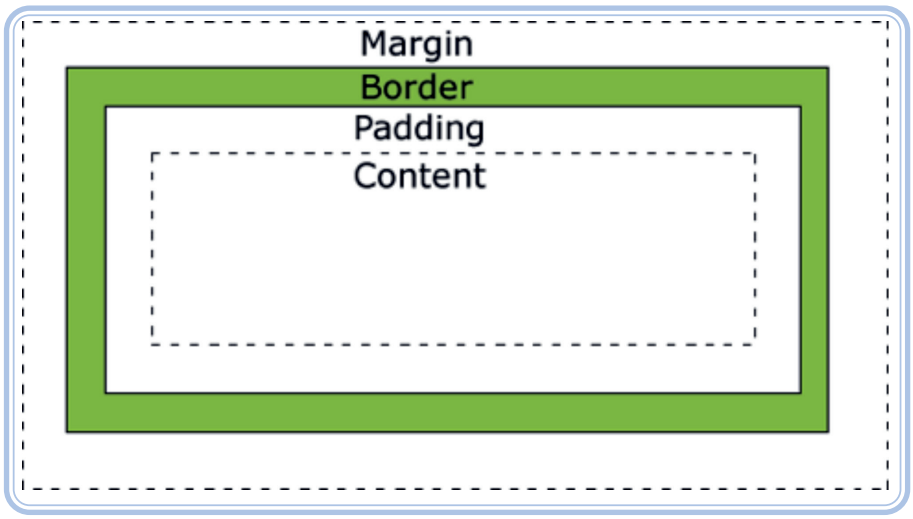
Todo *design* deve ter uma diagramação bem estudada, coerente com o projeto. E, em toda boa diagramação, utilizamos caixas para definir essas áreas. Por exemplo, este *site* abaixo tem uma largura fixa, para valorizar o *design* e temos várias "caixas" que têm largura e altura definidas. Nas próximas aulas, aprenderemos como fazer os elementos ficarem uns ao lado dos outros.

Fonte: http://www.culturalsolutions.co.uk/

Figura 18: Diagramação

Formatando com espaçamentos

Na imagem abaixo, temos um exemplo da organização dos espaçamentos em uma marcação. *Margin* é um espaçamento externo, *Border* é um efeito de borda, *Padding* é espaçamento interno e *Content* é uma referência ao conteúdo da caixa. Podemos aplicar esses atributos para qualquer marcação do tipo "*block*", isto é, parágrafos, divisões, títulos, entre outros.



Fonte: http://www.w3schools.com/css/css_boxmodel.asp

Figura 19: Identificação de margem, borda, padding e conteúdo

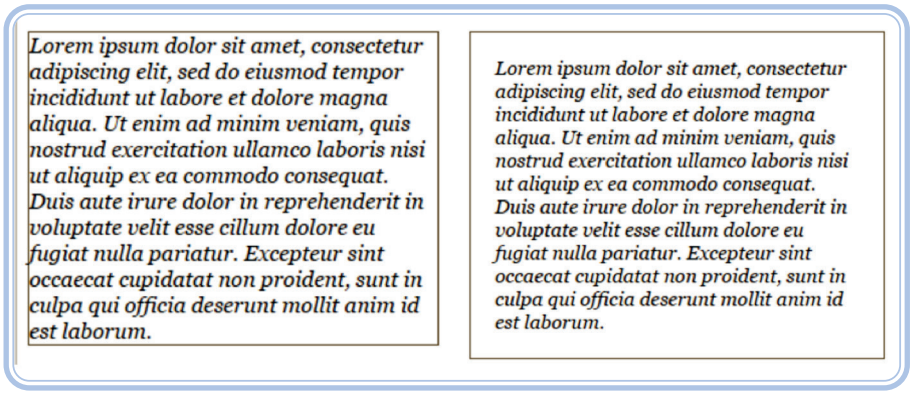
Quadro 06: Atributos de espaçamento

Definição	Atributo	Valores
Espaçamento interno	padding	Recebem um valor inteiro com unidade em px, pt, cm, etc ou em porcentagem.
Espaçamento externo	margin	

Essas propriedades são muito utilizadas na diagramação. Precisamos unir o *design* a um bom código. Para isso, é importante entendermos a importância de cada formatação. Mesmo que você não se identifique muito com a área de *design*, é importante entender a teoria dele para avaliar bons *layouts*, assim como é importante entender quando se tem um código bem programado.

Essas configurações estão diretamente relacionadas com *design* porque precisamos de espaço para os elementos do *layout*. Observe as imagens abaixo. Qual imagem permite mais conforto visual? A primeira que tem a borda colocada no texto ou a segunda que tem a borda mais distante do texto? Quando pegamos em um papel em branco para escrever, temos o hábito de deixar um espaço entre a borda do papel e nosso texto. No *design web* é do mesmo jeito. Esses espaços são importantes para legibilidade dos

elementos, não necessariamente dos textos, mas de todos os elementos que temos no nosso *layout*.



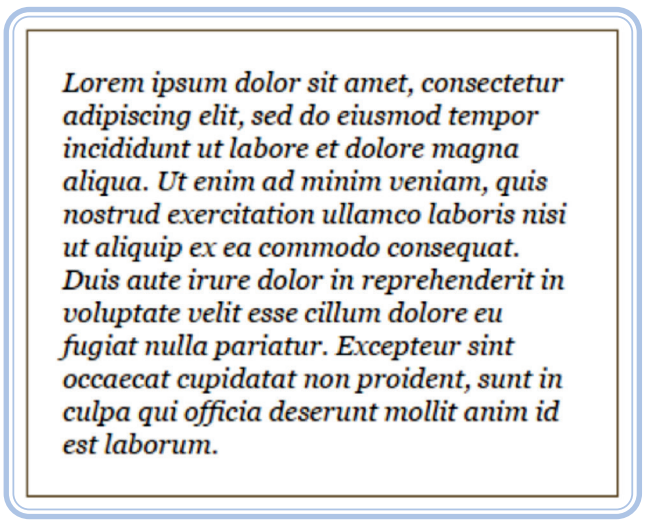
Fonte: Autoria própria.

Figura 20: Caixas de texto com padding e sem padding

Podemos trabalhar com os lados separadamente também, por exemplo, *margin-left*, *margin-bottom*, *margin-top* e *margin-right*, assim com *padding-top*, *padding-left*, *padding-right* e *padding-bottom*. É possível com apenas um atributo definir diferentes valores para os lados, tanto para "padding" quanto para "margin". Vejamos:

- Um valor: aplica para todos os lados

```
1 -  p{
2 -  padding: 20px;
3 -  }
```

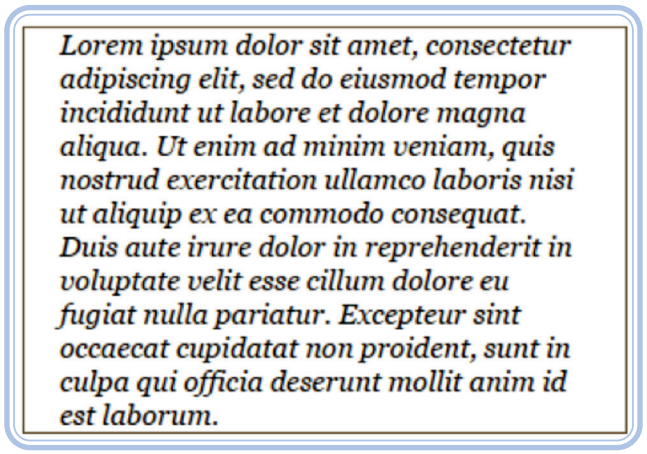


Fonte: Fonte: Autoria própria.

Figura 21: Caixa de texto com padding

- Dois valores: top-bottom e left-right

```
1 -    p{
2 -    padding: 0 20px 50px;
3 -    /* aplicamos: 0 para o topo, 20px para direita e esquerda e 50px para a parte inferior, perceba como o texto está bem mais distante da borda inferior. */
4 -    }
```

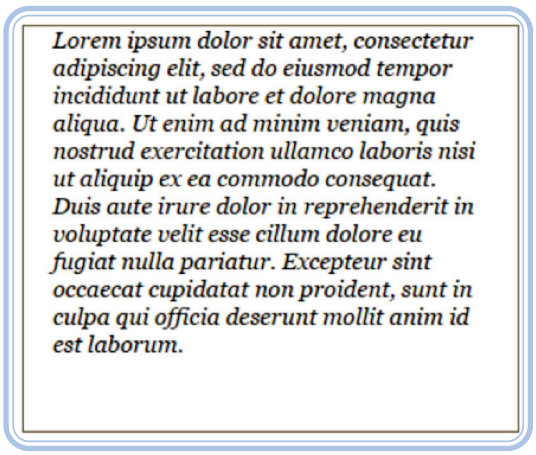


Fonte: Fonte: Autoria própria.

Figura 22: Caixa de texto com *padding* à esquerda e à direita

- Três valores: top, left-right e bottom

```
1 -    p{
2 -    padding: 0 20px 50px;
3 -    /* aplicamos: 0 para o topo, 20px para dir eita e esquerda e 50px para a parte inferior, perceba como o texto está bem mais distante da borda inferior. */
4 -    }
```

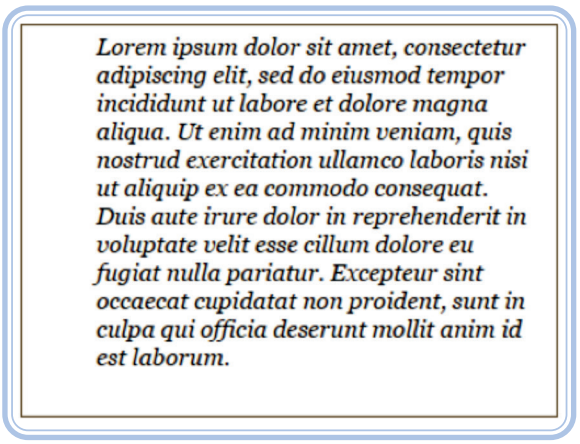


Fonte: Fonte: Autoria própria.

Figura 23: Caixa de texto com *padding* à direita, esquerda e abaixo

- Quatro valores: aplica para cada lado nesta ordem: top, right, bottom e left

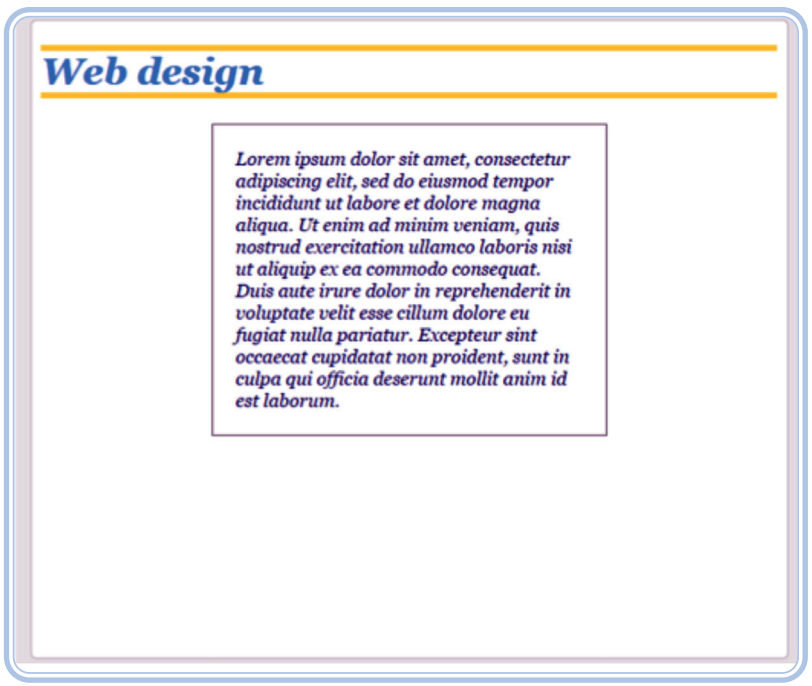
```
1 -  p{
2 -  padding: 5px 10px 30px 50px;
3 -  }
```



Fonte: Fonte: Autoria própria.

Figura 24: Caixa de texto com *padding* para os 4 lados

Com a propriedade "margin" também temos o valor "auto" que permite centralizar o elemento.



Fonte: Fonte: Autoria própria.

Figura 25: Caixa de texto centralizada à página

Código completo:

```
1 -  <!DOCTYPE html>
2 -  <html><head><style type="text/css">
3 -  body{
4 -        font-family: georgia, garamond;
5 -        font-style: italic;
6 -        color: #144b72;
7 -  }
8 -  h1 {
9 -        border-top: 5px solid #FFDB53;
10 -       border-bottom: 5px solid #FFDB53;
11 -       color: #3FA9F5;
12 -  }
13 -  p{
14 -       width: 300px;
15 -       height: 100px;
16 -       padding: 20px;
17 -       border: 1px solid gray;
18 -       margin:  0 auto;
19 -  }
20 -  </style>
21 -  </head>
22 -  <body>
23 -  <h1>Web design</h1>
24 -  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</p>
25 -  </body>
26 -  </html>
```

Atividade de aprendizagem 1

Desenvolva o *layout* a seguir de acordo com as marcações HTML já estudadas e as formatações da aula de hoje. Para auxiliar, siga as seguintes orientações:

- Fontes usada no corpo do texto: "trebuchet MS", arial.
- Cor da borda superior: #858585
- Plano de fundo cinza: #EAE4E4
- Fontes do título: georgia, garamond.
- Cor de plano de fundo dos parágrafos: #25A9AF

Fonte: Fonte: Autoria própria.

Figura 26: Atividade

RESUMINDO

Nesta aula, aprendemos a importância de trabalhar com formatação CSS, como incluir CSS na página e formatações simples para fontes, textos, cores de fundo, bordas, dimensões e espaçamentos.

Leituras complementares

Veja a lista dos atributos de formatação CSS:

W3SCHOOLS. Disponível em: <http://www.w3schools.com/cssref/default.asp>.

Avaliando seus conhecimentos

Aproveite a atividade da aula passada que contém informações sobre uma cidade e aplique as formatações aprendidas na aula.

Aula 4 - Listas, Imagens e Âncoras

Objetivos

Ao final desta aula, você deverá ser capaz de:

entender e criar listas ordenadas e não ordenadas;

incluir imagens nas páginas;

criar *links* de referências para outras páginas e documentos;

criar *links* em imagens.

Desenvolvendo o conteúdo

Lista

As listas são usadas sempre que queremos criar um grupo de itens. Por exemplo: *menu* de navegação, lista de informações relativas ao texto de uma página e outros exemplos. Essa lista poderá ser formatada para que não tenha os marcadores ou para que tenha outros marcadores no CSS. O importante é sempre entender quando utilizar essas marcações. Como explicitado em todas as aulas, a semântica do código torna seu *site* dentro dos padrões *web*.

As listas são organizadas em duas categorias: ordenadas e não ordenadas. Ambas definem uma lista que pode ou não mostrar os marcadores ou, ainda, modificar os marcadores utilizando CSS. O importante é sempre entender quando utilizá-las.

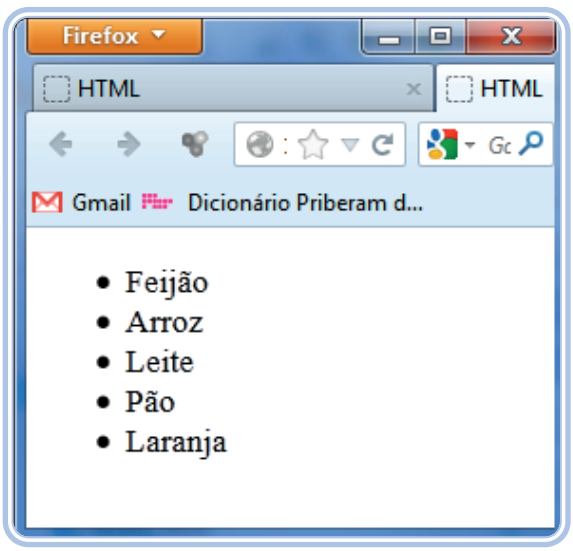
De acordo com as regras da W3C para listas, todos os itens devem vir com a marcação LI e, dentro das marcações de OL e UL, só pode conter a *tag* LI.

Isto é, não é permitido colocar a marcação parágrafo dentro da marcação OL ou UL, mas é permitido colocá-la dentro da marcação LI.

Listas não ordenadas

As tags <ul>... definem os limites de uma lista não ordenada — que consiste em vários itens aninhados, sem ordenação, acompanhados de um símbolo (*bullet*) na frente de cada item. Para definição dos itens, são utilizadas as *tags* ..., que também são usadas para listas ordenadas. A semântica de uma lista não ordenada é que esses itens não têm uma ordem. Essa ordem pode ser: cronológica, alfabética, de prioridades, entre outras. Na maioria dos *sites*, no *menu* de navegação, existem os itens organizados por prioridades. Perceba que, geralmente, o "contato" é um dos últimos itens. Observe, no exemplo a seguir, a disposição dessas *tags* e uma figura com a visualização no navegador.

```
1 -   ...
2 -   <ul>
3 -           <li>Feijão</li>
4 -           <li>Arroz</li>
5 -           <li>Leite</li>
6 -           <li>Pão</li>
7 -           <li>Laranja</li
8 -   </ul>
9 -   ...
```



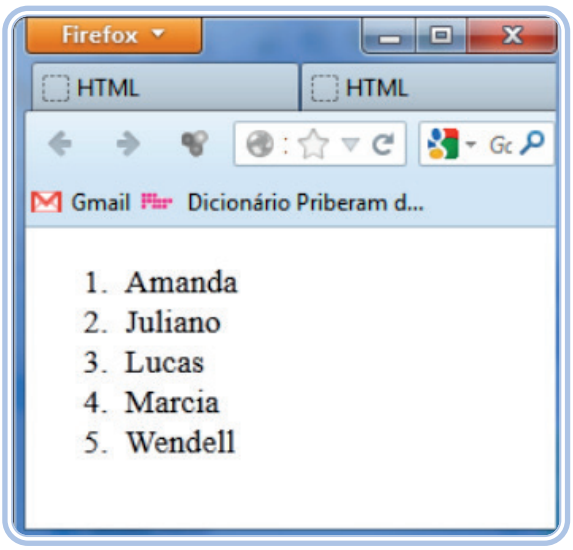
Fonte: Autoria própria.

Figura 1: Lista não ordenada

Lista ordenada

As *tags* ... definem os limites de uma lista ordenada, que consiste em vários itens aninhados acompanhados de uma numeração na frente de cada item. Para a definição dos itens, são utilizadas as *tags* .... Observe, no exemplo a seguir, a disposição dessas *tags*, em código, e a visualização da página no navegador.

```
1 -   ...
2 -   <ol>
3 -           <li>Amanda</li>
4 -           <li>Juliano</li>
5 -           <li>Lucas</li>
6 -           <li>Marcia</li>
7 -           <li>Wendell</li>
8 -   </ol>
9 -    ...
```



Fonte: Autoria própria.

Figura 2: Lista ordenada

LEMBRE-SE

A lista HTML não é ordenada automaticamente, as informações já devem ficar ordenadas no código. Por exemplo, se criarmos uma lista de itens, esses não aparecerão em ordem alfabética, até porque existem várias formas de ordenação, cronológica, alfabética, prioridades, entre outros.

Esse tipo de lista, ordenada <ol>, deve ser usada quando os itens seguirem uma ordem, por exemplo, alfabética, lógica, numérica etc.

As listas são muito importantes e muito utilizadas, mas nem sempre corretamente, pois seu uso "incorreto" não traz prejuízos à exibição. Aliás, qualquer marcação pode ser "mal" utilizada e pode não gerar prejuízos na exibição da página para o usuário no navegador, entretanto, como já foi dito nas aulas anteriores, a utilização correta traz benefícios de acessibilidade para dispositivos, para pessoas com deficiência. Enfim, mais do que isso, traz garantia de um código limpo e padronizado.

Formação CSS para listas

O CSS tem três atributos específicos para listas:

Quadro 1: Atributos para formatação de tabelas

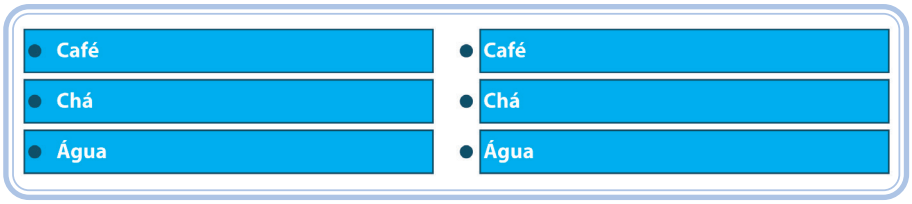
Definição	Atributo	Valores
Disponibiliza diferentes "*bullets*" para as listas	*List-style-type*	*Disc*, *circle*, *georgian*, decimal, *hiragana*, *katakana*, *lower-latin*...
Especifica se o conteúdo do item ficará dentro ou fora da área	*List-style-position*	*inside*, *outside*
Largura máxima Define uma imagem para o "*bullet*". Este atributo anula o atributo "*list-style-type*"	*List-style-image*	url com o endereço da imagem.
Define todas as propriedades acima	*List-style*	

Fonte: <http://www.w3schools.com/html/html_lists.asp>.

```
1 -   <!DOCTYPE html>
2 -   <html>
3 -   <head>
4 -   <style>
5 -   ul li{
6 -       list-style-position: outside;
7 -       border: 1px solid black;
8 -       margin: 3px;
9 -    }
10 -  </style>
11 -  </head>
12 -  <body>
13 -  <ul>
14 -      <li>Cefé</li>
15 -      <li>Chá</li>
16 -      <li>Água</li>
17 -  </ul>
18 -  </body>
19 -  </html>
```

As imagens abaixo exemplificam a utilização do atributo "*list-style -position*". A lista da imagem da esquerda tem o valor de "*list-style-position*" igual a

"*inside*" e a lista da imagem da direita, o valor igual a "*outside*", o código a¬cima foi usado para o exemplo.



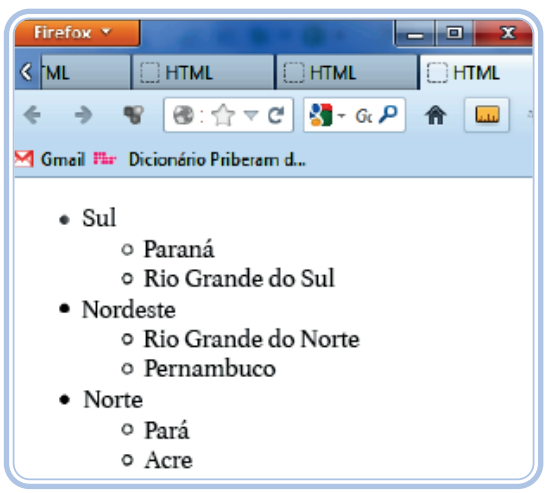
Fonte: Autoria própria.

Figura 3: Listas com posição "*inside*",esquerda, e "*outside*", direita

Lista aninhada

Uma lista aninhada é uma lista com outra ou outras dentro dela. É utilizada quando se quer definir subitens. Abaixo há um exemplo, em código, e na visualização do navegador.

```
1 -  ...
2 -  <ul>
3 -          <li>Sul
4 -                  <ul>
5 -                          <li>Paraná</li>
6 -                          <li>Rio Grande do Sul</li>
7 -                  </ul>
8 -          </li>
9 -          <li>Nordeste
10 -                 <ul>
11 -                         <li>Rio Grande do Norte</li>
12 -                         <li>Pernambuco</li>
13 -                 </ul>
14 -         </li>
15 -         <li>Norte
16 -                 <ul>
17 -                         <li>Pará</li>
18 -                         <li>Acre</li>
19 -                 </ul>
20 -         </li>
21 - </ul> ...
```



Fonte: Autoria própria.

Figura 4: Lista aninhada

Há uma regra, nas listas, que foi dita acima: as marcações OL e UL só podem conter LI; portanto, para criar uma lista aninhada, ou seja, uma lista dentro da outra, só é possível colocar a lista dentro de um LI.

Exemplo:

```
1 -   ...
2 -   <ul>
3 -   <li>MPB</li>
4 -       <ul>
5 -       <li>Caetano Veloso</li>
6 -       <li>Gilberto Gil</li>
7 -       <li>Lulu Santos</li>
8 -       </ul>
9 -   </ul>
10 -  ...
```

ESTE CÓDIGO ESTÁ SEMANTICAMENTE ERRADO!!

```
1 -   ...
2 -   <ul>
3 -   <li>MPB
4 -       <ul>
5 -       <li>Caetano Veloso</li>
6 -       <li>Gilberto Gil</li>
7 -       <li>Lulu Santos</li>
8 -       </ul>
9 -   </li>
10 -  </ul>
11 -      ...
```

Na linha 3, de ambos os códigos, temos um item da lista. Se quisermos criar uma lista dentro desse item, temos de fechar o LI apenas depois de fechar a lista interna. Perceba que a lista interna também deve seguir a sintaxe de marcação OL ou UL, assim como os itens com LI.

Podemos criar diversos níveis de hierarquia, mas temos de seguir a regra das listas. Sempre as listas devem vir dentro dos itens.

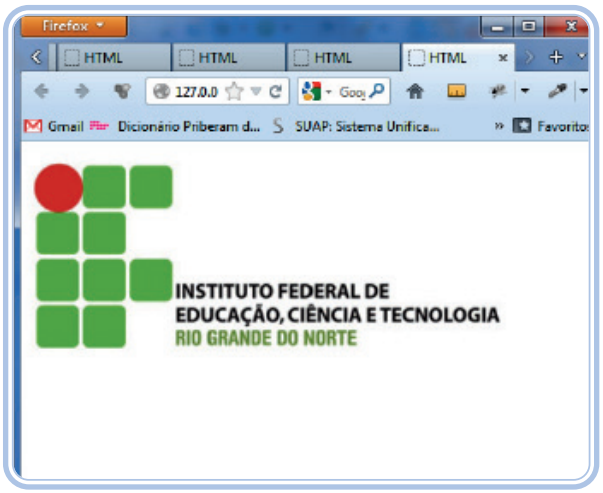
Imagens

É possível inserir imagens em um documento através da *tag* <img/>. Para usar essa marcação, devemos incluir dois atributos obrigatoriamente: SRC, o qual recebe o endereço da imagem, que pode ser local ou de um *site* externo, e o atributo ALT, que recebe a descrição textual da imagem. Esse

atributo é uma recomendação de grande prioridade da W3C, pois é pelo texto colocado no ALT que os deficientes visuais, que acessam as páginas por *software,* que a leem conseguem entender a imagem. Essa descrição também é exibida quando a imagem não é mostrada no navegador. Se isso acontecer, os motivos podem ser: falha no servidor, computador sem *internet*, endereço da imagem errado, erro no código ou outros.

Podemos inserir imagens de diferentes tipos: GIF, JPG, JPEG e PNG. Logo a seguir, há um código para incluir uma imagem do tipo ".jpg" e uma visualização da imagem no navegador.

```
1 -   ...
2 -   <p>
3 -   <img src="ifrn.jpg" alt="logo do IFRN" />
4 -   </p>
5 -   ...
```



Fonte: Autoria própria.

Figura 5: Imagem

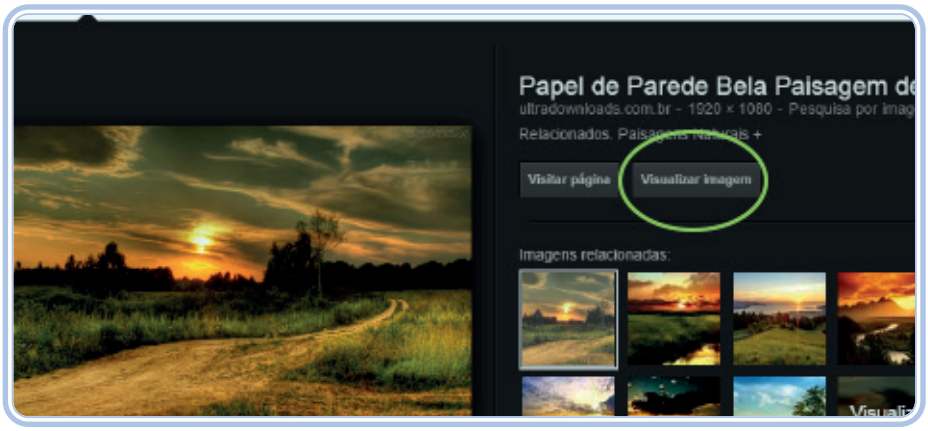
O atributo SRC tem no valor "ifrn.jpg", ficando src="ifrn.jpg", nele tem o nome da imagem, esta deve estar na mesma pasta do atual documento HTML. Abaixo, há um passo a passo para criar essa referência.

Passo a passo:

1. abra o navegador;

2. faça uma busca por uma imagem em um *site* de busca, como o Google.com;

3. clique na imagem e terá um *link* ao lado com o texto "visualizar imagem";



Fonte: Google imagens.

Figura 6: "visualizar imagem"

4. clique com o botão direito do *mouse* sobre a imagem e clique novamente em "salvar imagem como..." ou algo similar;

5. salve a imagem na mesma pasta que está a imagem HTML, altere o nome do arquivo para remover os espaços e acentos, por exemplo, coloque apenas "paisagem", verifique que será mostrado o tipo de imagem, se JPG, GIF, PNG, JPEG...;

6. no seu código HTML, defina, no atributo SRC, o nome da imagem e a extensão, que é o tipo da imagem.

Também é possível colocar a URL da imagem, mas, nesse caso, se a imagem do servidor original ficar indisponível, futuramente, a imagem não será visualizada na página. Por exemplo, ao buscar por IFRN, temos em um dos resultados essa URL, que ficaria assim:

```
1 -    ...
2 -    <img src=" http://portal.ifrn.edu.br/institucional/logomarcas/jpg/Logo%20
IFRN%20-%20Campus%20Santa%20Cruz.jpg" alt="logo ifrn" />
3 -    ...
```

Perceba que a linha ficou extensa e ultrapassou o limite, porém não há problema, contanto que o valor do SRC não tenha espaços. Esses códigos "%20" são justamente os espaços que há no nome do arquivo, mas não haverá problema na visualização.

É importante entender a diferença de imagens em código e imagem em plano de fundo. As imagens que devem vir com a marcação IMG são imagens com informações. Esse logotipo do IF, por exemplo, é uma informação, uma

paisagem e não tem texto, mas é uma informação. Imagens que representam texturas, gradientes, são imagens que não têm valores informacionais e devem ser definidas como planos de fundo.

Às vezes, queremos colocar uma paisagem como plano de fundo. Isso é semântico, se ela não fizer parte do contexto atual da página, se for apenas para compor o *design*. Isto é, se você está fazendo um *site* para uma agência de turismo e insere uma imagem de fundo que é uma junção de várias cidades de forma bem artística, fica interessante e semântico, mas, em uma página que fala de Paris, ter uma linda imagem da cidade em plano de fundo não é coerente, porque é uma imagem que corresponde com o contexto informativo da página atual, logo deverá ser usada com a *tag* IMG no código HTML e não como plano de fundo.

O HTML5 tem novas definições para as imagens. Foi criada uma nova marcação FIGURE que recebe imagens e assim organiza melhor as imagens na página. Leia mais em: **W3 SCHOLLS**. HTML <figure> Tag. [20--?]. Disponível em: <http://www.w3schools.com/tags/tag_fi gure.asp>. Acesso em: 26 jun. 2015.

Âncoras

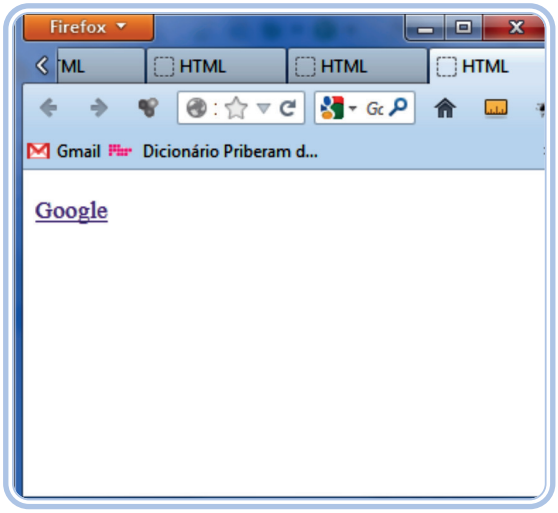
As *tags* <a>... definem um *link* que pode ser clicado e que estabelece uma ligação para seções de um mesmo documento ou para outros documentos ou recursos externos: uma imagem, um arquivo em PDF, um aplicativo, entre outros.

Âncoras externas

É quando faz uma ligação com outros *sites*, que utilizam uma URL diferente. Nesse caso, é obrigatório o uso do protocolo de internet: <HTTP://>. Sem ele, a ligação ficaria como internos.

```
1 -  ...
2 -  <p>
3 -  <a href="http://google.com">Google</a>
4 -  </p>
5 -  ...
```

A formatação padrão do *link* é azul com sublinhado, mas essa pode ser alterada utilizando CSS.



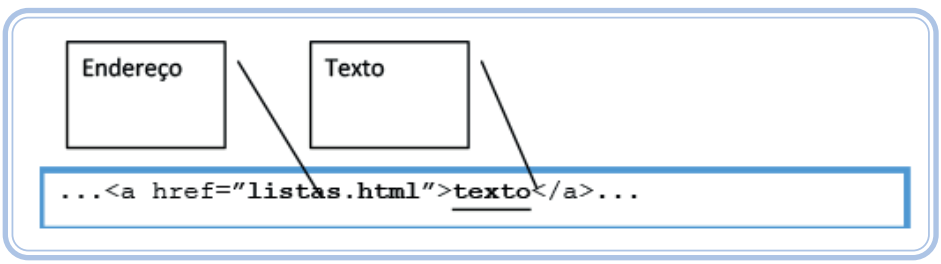
Fonte: Autoria própria.

Figura 7: Âncoras

Âncoras internas

Faz uma ligação com as páginas ou arquivos que estão no *site*, isto é, têm a mesma URL da página na qual se encontra. Por exemplo, você está na página inicial e clica em um *link* do *menu*, irá para uma página do próprio *site*. Parece óbvio, mas podemos ter, inclusive no menu, um *link* para um outro *site*, um *link* externo, como visto acima.

Dentro das marcações das âncoras, coloca-se o texto que será o *link* e, dentro do atributo HREF, coloca-se o nome da página que se quer fazer a ligação.



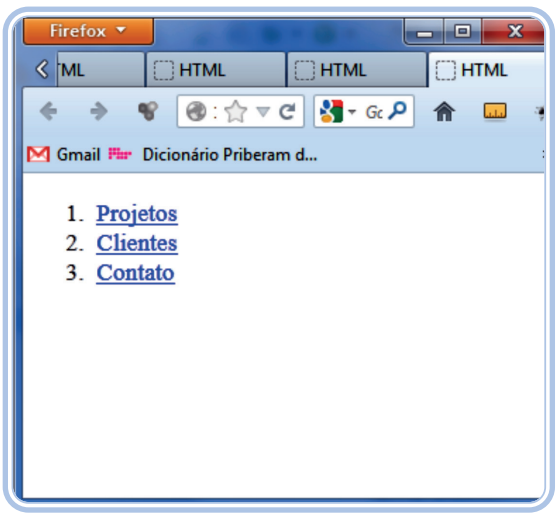
No código abaixo, o *link* está dentro da *tag* de parágrafo, mas pode vir dentro das *tags* de título, de lista, de tabelas, entre outras.

```
1 -   ...
2 -   <p>Minhas atividades de HTML:
3 -   <a href="titulos.html">Títulos</a> e
4 -   <a href="listas.html">Listas</a>
5 -   </p>...
```

Lista de links

```
1 -   ...
2 -   <ol>
3 -           <li><a href="projetos.html">Projetos</a></li>
4 -           <li><a href="clientes.html">Clientes</a></li>
5 -           <li><a href="contato.html">Contato</a></li>
6 -   </ol>
7 -   ...
```

Pode-se fazer uma ligação para uma página na qual o texto do *link* não condiz. Por exemplo, como são usados nos *link*s de vírus enviados por *e-mail*.



Fonte: Autoria própria.

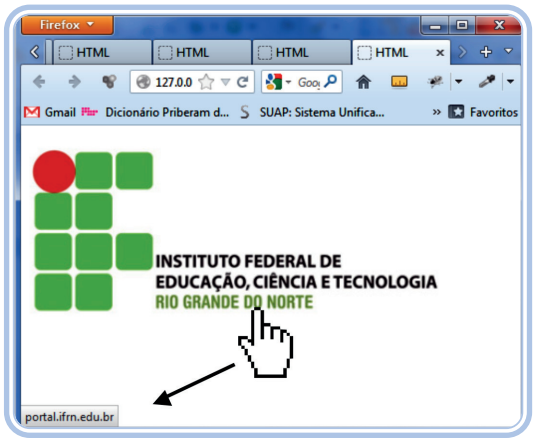
Figura 8: Menu

A lista usada na imagem acima foi ordenada, porque há uma ordem de prioridades dos itens.

Links nas imagens

Algumas vezes, temos imagens como substituição de texto, as quais, às vezes, precisam direcionar o usuário para alguma outra informação como, por exemplo, o *menu* com imagens, o nome do *site* para a página inicial, entre outros.

```
1 -   ...
2 -   <h1>
3 -   <a href="http://portal.ifrn.edu.br">
4 -       <img src="ifrn.jpg" alt="logo do IFRN" />
5 -   </a>
6 -   </h1>
7 -   ...
```



Fonte: Autoria própria.

Figura 9: Imagem com link para o site portal.ifrn.edu.br

LEMBRE-SE

1 - As marcações de estruturas iniciais de alguns códigos acima foram suprimidas. Ao criar suas páginas, use todas as marcações. Caso não se lembre, veja nas aulas anteriores todas as marcações necessárias.

2- Para todos os exemplos de código das aulas, crie as páginas e salve em pastas organizadas por aulas. Não defina o nome do arquivo HTML e CSS com espaços e caracteres especiais.

RESUMINDO

Nesta aula, aprendemos a importância de como trabalhar com listas ordenadas, não ordenadas e aninhadas, incluir imagens e âncoras, assim como a relação entre essas marcações, como imagens com *link*s.

Leituras complementares

Veja todos os atributos que podem ser usados nas marcações de:

a) Listas: **W3 SCHOOLS**. HTML Links. [20--?]. Disponível em: <http://www.w3schools.com/html/html_*link*s.asp>. Acesso em: 26 jun. 2015.

b) Âncoras: **W3 SCHOOLS**. HTML Links. [20--?]. Disponível em: <http://www.w3schools.com/html/html_*link*s.asp>. Acesso em: 26 jun. 2015.

c) Imagens: **W3 SCHOOLS**. HTML Images. [20--?]. Disponível em: <http://www.w3schools.com/html/html_images.asp>. Acesso em: 26 jun. 2015.

Avaliando seus conhecimentos

Crie uma pasta chamada "projetoturismobr", dentro da pasta do "Módulo 4" e, nela, crie 6 páginas para um *site* sobre turismo no Brasil. Veja as orientações.

A página inicial será "index.html". Defina-a com títulos, parágrafos e listas de informações sobre o que teremos no *site* de turismo, como, por exemplo, uma síntese das informações. Após o código de título, coloque *link*s para as páginas das regiões do Brasil, essa lista terá em todas as páginas das regiões.

Nas páginas "nordeste.html", "norte.html", "sul.html", "sudeste.html" e "centrooeste.html", defina a mesma estrutura da página de "index.html" com a lista de *link*s para as páginas, além de parágrafos, imagens e listas com informações sobre a respectiva região.

Aula 5 - Tabelas e Formulários

Objetivos

Ao final desta aula, você deverá ser capaz de:

criar tabelas e formulários semanticamente;

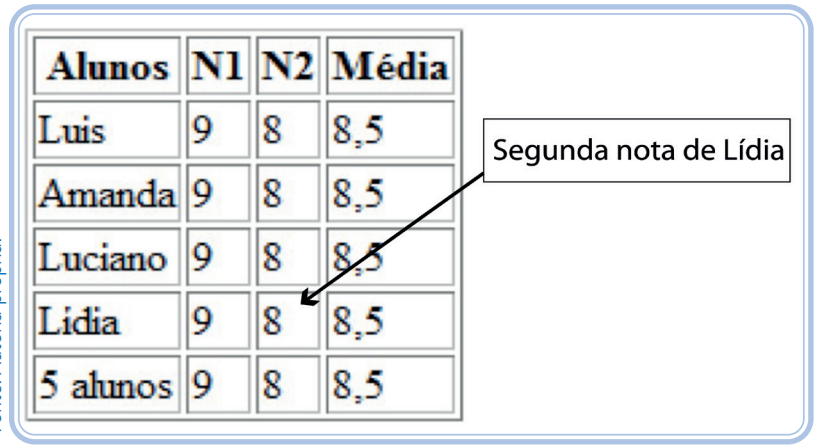
diagramar os formulários.

Desenvolvendo o conteúdo

Tabelas

As tabelas são usadas nas páginas para tabular dados, que é quando se tem dados relacionados. Por exemplo, a tabela abaixo tem notas e essa nota destacada está relacionada com a segunda nota da aluna Lídia. A forma correta de estruturar as tabelas é um ponto de observação da W3C, pois usuários com limitações de tela ou visuais podem não conseguir acessar esses dados ou ter dificuldades para acessá-los.

Antigamente, a utilização de CSS era limitada e muitos desenvolvedores utilizam as tabelas para diagramar páginas. Um bom exemplo disso são as páginas terem aquela estrutura topo, *menu* à esquerda, conteúdo à direita e rodapé abaixo? Então, essa estrutura é como uma tabela com três linhas, sendo a linha do topo e rodapé com duas células mescladas. Enfim, o importante é entender que essa forma de trabalhar é semanticamente incorreta, porque as tabelas têm como objetivo diagramar informações.



Fonte: Autoria própria.

Figura 1: Tabela com dados tabulados de notas

O código é extenso, mas as *tags* seguem uma estrutura lógica. Para se estruturar, é necessário seguir a ordem de células da tabela: de cima para baixo e da esquerda para a direita. As tabelas são compostas de linhas e cada linha tem células. Então, ao criar a linha, inserimos as células, que, visualmente, formarão colunas. Abaixo, temos as marcações usadas nas para o desenvolvimento de tabelas.

- *Table*: *tag* que delimita a área da tabela.
- *Border*: atributo da *tag table* para criar uma borda na tabela e que pode ser substituído por formatação em CSS.
- *Caption*: para colocar o título da tabela (única *tag* que não começa com 'T').
- *Thead*: organiza as linhas de cabeçalho da tabela.
- *Tfoot*: organiza a(s) linha(s) de rodapé da tabela, mas o rodapé é **opcional** (obs.: o *tfoot*, no código, fica após o *thead*, porém, quando a tabela é visualizada no navegador, os dados ficam do rodapé abaixo do corpo da tabela).
- *Tbody*: organiza as linhas do corpo da tabela.
- *Tr*: linha da tabela (*row* é linha em inglês).
- *Th*: célula de título (por padrão os dados ficam em negrito e centralizado).
- *Td*: célula de dados.

```
1 -     ...
2 -     <table summary="Notas da disciplina de Autoria Web I" bor-
der="1">
3 -     <caption>Notas de Autoria Web I</caption>
4 -             <thead>
```

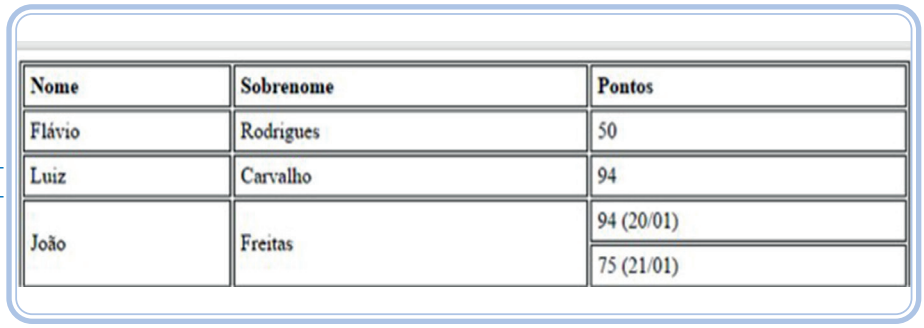
```
5 -             <tr>
6 -             <th>Alunos</th><th>N1</th><th>N2</th><th>Média</
th>
7 -           </tr>
8 -       </thead>
9 -       <tfoot>
10 -            <tr>
11 -            <td>5 alunos</td><td>9</td><td>8</td><td>8,5</td>
12 -          </tr>
13 -      </tfoot>
14 -      <tbody>
15 -            <tr>
16 -            <td>Luis</td><td>9</td><td>8</td><td>8,5</td>
17 -          </tr>
18 -            <tr>
19 -            <td>Amanda</td><td>9</td><td>8</td><td>8,5</td>
20 -          </tr>
21 -            <tr>
22 -            <td>Luciano</td><td>9</td><td>8</td><td>8,5</td>
23 -          </tr>
24 -            <tr>
25 -            <td>Lídia</td><td>9</td><td>8</td><td>8,5</td>
26 -          </tr>
27 -      </tbody>
28 -  </table>…
```

Tabelas com células mescladas

Para mesclar células, usamos os atributos "*colspan*" e "*rowspan*". O primeiro mescla colunas, e o segundo, linhas.

```
1 -   …
2 -   <style>
3 -   table, th, td {
4 -       border: 1px solid black;
5 -   }
6 -   th, td {
7 -       padding: 5px;
8 -       text-align: left;
```

```
9 -     }
10 -    </style>
11 -    </head>
12 -    <body>
13 -
14 -    <table style="width:100%">
15 -    <thead>
16 -      <tr>
17 -        <th>Nome</th>
18 -        <th>Sobrenome</th>
19 -        <th>Pontos</th>
20 -      </tr>
21 -    </thead>
22 -    <tbody>
23 -      <tr>
24 -        <td>Flávio</td>
25 -        <td>Rodrigues</td>
26 -        <td>50</td>
27 -      </tr>
28 -      <tr>
29 -        <td>Luiz</td>
30 -        <td>Carvalho</td>
31 -        <td>94</td>
32 -      </tr>
33 -
34 -      <tr>
35 -        <td rowspan="2">João</td>
36 -        <td rowspan="2">Freitas</td>
37 -        <td>94 (20/01)</td>
38 -      </tr>
39 -      <tr>
40 -
41 -
42 -        <td>75 (21/01)</td>
43 -      </tr>
44 -    </tbody>
45 -    </table>
46 -    ...
```



Nome	Sobrenome	Pontos
Flávio	Rodrigues	50
Luiz	Carvalho	94
João	Freitas	94 (20/01)
		75 (21/01)

Fonte: Autoria própria.

Figura 2: tabela com células mescladas

Perceba que, na última linha, não tem três marcações de células, mas apenas uma, pois, na linha acima, as duas primeiras células têm o atributo "*rowspan*", o qual mescla as duas linhas. Logo, a linha após essa já tinha duas células "ocupadas", se fosse inserida mais uma célula nessa linha, a tabela ficaria com uma célula à direita como se houvesse uma quarta coluna, isto é, deformaria a tabela.

Para o atributo "*colspan*", usamos o atributo na marcação de célula também, é um pouco mais fácil que o atributo "rowspan", pois as mesclagens não influenciam as linhas abaixo, apenas as células da própria linha que têm o atributo. Faça o teste. Coloque quantidades diferentes de células nas linhas.

Como os formulários funcionam

Os formulários trazem possibilidades de envio e recebimento de dados. Como exemplos: formulário de cadastros, formulários de *logins*, formulários para enquete, formulário para contato, entre outros.

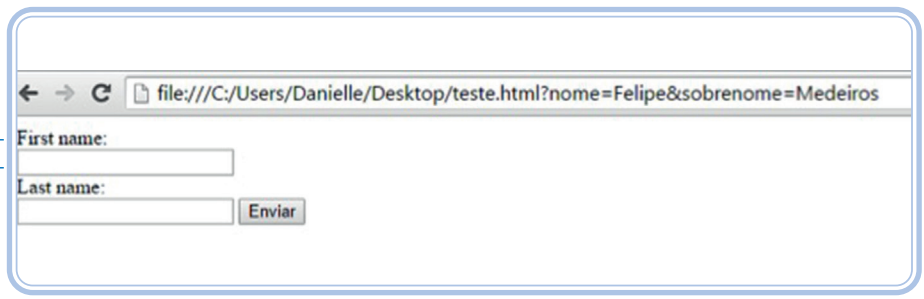
Os dados do formulário são enviados pelo navegador, mas, para que sejam manipulados, é necessária uma linguagem servidor. A linguagem HTML é chamada de cliente e não de servidor. Uma linguagem cliente tem o código interpretado pelo Navegador e uma linguagem servidor tem o código interpretado pelo servidor e depois pelo navegador.

Os formulários são delimitados pela *tag* <*form*> e todos os componentes devem ficar dentro dele. Essa *tag* tem dois atributos: *action* que, deve conter o endereço do arquivo que receberá os dados e, o atributo *method* que, pode ter dois valores: GET ou POST. O primeiro envia os dados e exibe-os na URL e o segundo também envia os dados, mas não os exibe na URL.

Para cada formulário na página, é necessário usar a marcação <FORM>. Se, na página, tiver um formulário de busca, cadastro e enquete, serão marcações de FORM.

```
1 -     …
2 -     <form action=”” method=””>
3 -     …
4 -     </form>
5 -     …
```

Ao utiliza o formulário com método GET:



Fonte: Autoria própria.

Figura 3: Formulário com método GET

Componentes para o desenvolvimento de um formulário

Rótulo

O rótulo, *tag label*, é um item que não implica em nada no visual do formulário, mas é muito importante para a acessibilidade da página. Deve ser usado em todos os itens de entrada de dados, nos campos de texto, para cada *checkbox*, para cada *radiobuttom*, para o *combobox* e etc.

Pode ser aplicado de duas formas:

1° - Com o campo dentro das *tags label* (conferir o final da linha três, que será explicado mais a frente).

```
1 -     …
2 -     <p>
3 -     <label>Nome: <input type=”text” name=”nome” /></label>
4 -     <p>
5 -     …
```

2º - Com o campo em outra área do código, mas referenciado pelo atributo *for* do *label* com o *id* do campo (conferir linha quatro, que será explicada mais a frente).

```
1 -     ...
2 -     <p>
3 -     <label for="nome">Nome: </label>
4 -     <input type="text" name="nome" id="nome" />
5 -     <p>
6 -     ...
```

Entrada de dados

Agora vamos ver como criar um componente que recebe dados. Os campos de entrada de dados devem sempre ter o atributo *name*; opcionalmente, o atributo id, este depende de dois pontos: se é usado *javascript* ou CSS no componente e em relação ao rótulo (*label*); e, em alguns casos, são usados o atributo *value*, que equivale ao valor associado ao componente. Esses atributos são utilizados na programação do tipo servidor, mas pode implicar no mau funcionamento do componente. Todos os componentes *input* devem ter um atributo *type* que vai defini-lo. Todos serão explicados a seguir.

LEMBRE-SE!

Colocar nos valores de "*name*", "*id*", "*type*" sempre termos sem caracteres especiais e sem espaços.

Texto simples

Cria um campo de entrada de texto de uma só linha.

```
1 -     ...
2 -     <p>
3 -     <label for="nome">Nome: </label>
4 -     <input name="nome" id="nome" type="text" />
5 -     </p>
6 -     ...
```

Fonte: Autoria própria.

Figura 4: Entrada de texto

Senha

Cria um campo de entrada de texto, mas, para a segurança do usuário, os dados não são exibidos.

Em alguns computadores, não aparecem "círculos" e sim asteriscos. Essas configurações estão relacionadas com o sistema operacional ou navegador utilizado.

```
1 -     …
2 -     <p>
3 -     <label for="senha">Senha: </label>
4 -     <input name="senha" id="senha" type="password" />
5 -     </p>
6 -     …
```

Fonte: Autoria própria.

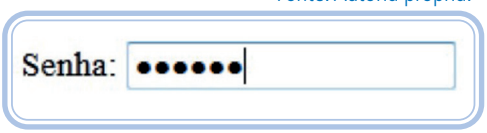


Figura 5: Campo para senha

Check Box

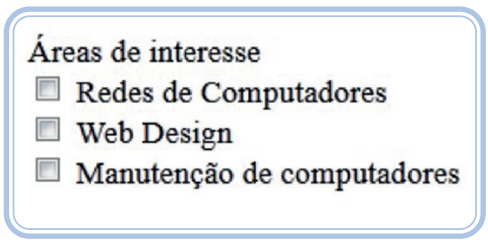
Cria uma caixa de seleção onde é possível marcar ou desmarcar uma opção, permite um, zero, todas, duas, três ... opções marcadas. Deve ser usado como atributos: *type*, *name* e *value*. O atributo *name* tem o mesmo valor em todos os itens do grupo (podem ter, no mesmo formulário, vários *check boxes*), e o atributo value deve ter o valor que condiz com o texto visível da opção. Nos textos do *name* e do *value*, não podem conter espaços e/ou caracteres especiais (acentos). Por padrão, colocamos tudo em minúsculo com *underline*, separando duas palavras, se o desenvolvedor achar melhor.

Nesse componente, a utilização do rótulo faz uma grande diferença. Experimente clicar sobre o texto da opção com e sem o rótulo.

Nas linhas 2, 5 e 8, existem os atributos *name* com o mesmo valor, cujo texto é grafado em minúsculas, sem acento e com *underline* separando as duas palavras.

Nas linhas 2, 5 e 8, também há os atributos *value* com os valores condizentes com as opções: "redes", "*web*" e "*manutencao*", sem acentos, com letras minúsculas e de forma mais reduzida.

```
1 -     …
2 -     <p>Áreas de interesse<br />
3 -     <label><input type="checkbox" name="area_interesse"
value="redes" />
4 -             Redes de Computadores
5 -     </label><br />
6 -     <label><input type="checkbox" name="area_interesse"
value="web" />
7 -             Web Design
8 -     </label><br />
9 -     <label><input type="checkbox" name="area_interesse"
value="manutencao" />
10 -            Manutenção de computadores
11 -    </label><br />
12 -    </p>…
```



Fonte: Autoria própria.

Figura 6: *Check boxes*

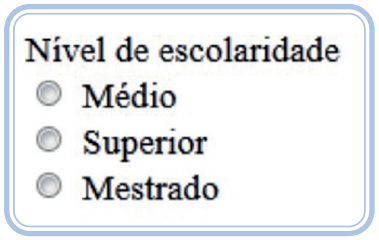
Radio Button

Cria botões com opções, similar ao *check box*, mas só permite a seleção de uma opção. Segue a mesma estrutura de necessidade para inserção dos atributos *type*, *name* e *value*.

Perceba que, nas linhas 2, 5 e 8, os atributos *name* têm o mesmo valor: "ne". E, nas mesmas linhas, o atributo *value* tem o valor que equivale a opção: "medio", "superior" e "mestrado", sem usar acentos.

```
1 -     …
2 -     <p>Nível de escolaridade<br />
3 -     <label><input type="radio" name="ne" value="medio" />
4 -             Médio
5 -     </label><br />
6 -     <label><input type="radio" name="ne" value="superior" />
7 -             Superior
8 -     </label><br />
```

```
9 -     <label><input type="radio" name="ne" value="mestrado" />
10 -            Mestrado
11 -    </label><br />
12 -    </p>...
```



Fonte: Autoria própria.

Figura 7: *Radio buttons*

Tanto o *checkbox* quando o *radiobutton* trabalham com grupos de itens. Esses devem ter o atributo "*name*" com o mesmo valor, como foi dito, pois podem existir os grupos de *checkbox's* e *radiobutton's* na mesma página. E o que os diferenciam é justamente o valor do atributo "*name*".

O atributo "*value*" não pode ficar sem ser definido. Essas estruturas têm um texto que é visível apenas para o usuário que visualiza a página pelo navegador. Quando os dados do formulário forem enviados para o servidor, o que será recebido é justamente o que foi colocado no atributo "*value*". Faça uma página e teste o envio do formulário, utilizando o "*method*" com valor "GET" e verifique o campo de endereço do navegador.

Envio de arquivos

Cria um controle de seleção de arquivos, que permite o envio de arquivos. É necessário colocar no atributo *accept* na *tag form* os tipos de arquivos que devem ser reconhecidos (para o caso de programação).

```
1 -     ...
2 -     <p>
3 -     <label for="foto">Foto:</label>
4 -     <input name="foto" id="foto" type="file" />
5 -     </p>...
```

Fonte: Autoria própria.

Figura 8: Componente para envio de arquivo

Cada navegador exibe esse componente diferente. O navegador usado na imagem ao lado é o Chrome.

Botão de submissão

Submeter os dados é o ato de enviar para o servidor os dados que foram digitados ou selecionados. Para o botão de submissão, não é necessário usar rótulos. O atributo *value* é usado para definir o texto que ficará visível no botão, por isso pode ser digitado com espaços, letras maiúsculas e acentos. O valor no *type* que o define é *submit*.

```
1 -     ...
2 -     <p>
3 -     <input type="submit" value="Enviar" />
4 -     </p>
5 -     ...
```

Fonte: Autoria própria.

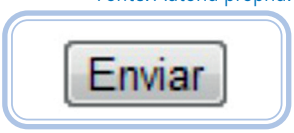


Figura 9: Botão para submissão dos dados

Botão de limpar dados

Vamos agora criar um botão que, ao ser clicado, restaura os valores dos campos do formulário para o original, geralmente limpa os valores que foram digitados pelo usuário no formulário. Se o formulário já tiver dados, como quando clicamos em "editar dados", os dados são restaurados e quando o formulário se inicia vazio, os dados que foram digitados, são apagados.

Não é necessário usar rótulos, e o atributo "*value*" segue com a mesma coerência do botão *submit*. O valor no *type* que o define é *reset*. Esse recurso é pouco usado porque não há ainda um padrão de localização, se ao lado direito do botão de enviar ou do lado esquerdo. Às vezes, o usuário clica sem querer. Então, caso utilize, uma dica é modificar a formatação do botão de enviar e limpar, deixando o botão de enviar mais visível e o botão de limpar mais opaco.

```
1 -     ...
2 -     <p>
3 -     <input type="reset"  value="Limpar" />
4 -     </p>...
```

Fonte: Autoria própria.

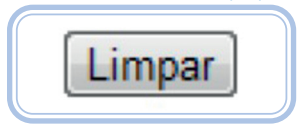


Figura 10: Botão para "resetar" os dados

Botão de submissão do tipo Imagem

Agora, veremos como criar um botão que não tem a aparência padrão de um botão do navegador, mas também é um botão de submissão. No lugar de texto é exibida uma imagem, que deve ter seu caminho especificado no atributo src.

```
1 -     ...
2 -     <p>
3 -     <input type="image" src="botao.gif" />
4 -     </p>
5 -     ...
```

Fonte: Autoria própria.

Figura 11: Botão com imagem para submeter os dados

Essa imagem "botão.gif" deve estar no mesmo diretório da página. Para testar esse código, é necessário tê-la. Segue a mesma lógica da marcação de IMG.

Combo Box ou Drop Down

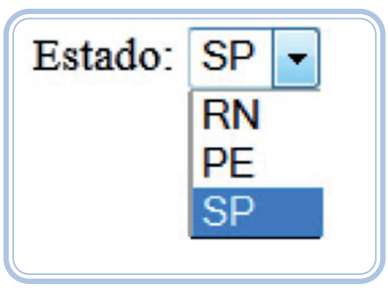
São componentes que possibilitam, ao desenvolvedor, agrupar vários itens que são comuns e não ocupar espaços na área da página. Por exemplo, agrupar estados, meses, dias, anos, cidades, entre outros dados. A vantagem é que possibilita, ao usuário, selecionar a opção dentre as muitas disponíveis e, para a página, há a vantagem de muitos itens ocupando pouco espaço, além de padronizar o dado selecionado, pois, cada usuário poderia digitar seu estado de uma forma diferente, como, a sigla, o nome por extenso e minúsculo ou abreviado e maiúsculo etc. Por isso temos opções limitadas. Esse componente é bem similar ao *radio buttom*, pois também só permite

a seleção de uma opção. A diferença entre eles é visual, no *select*, porque não podemos ver todos os itens, apenas após clicar, e, no radio *Button*, visualizamos todos os itens, portanto, é necessário avaliar para definir qual a melhor opção.

O elemento para criar o *combo box* ou *drop down* é o *select* no qual deve ser definido, assim como em todos os componentes, com o atributo *name*, o atributo *id*, como dito em rótulos, depende da forma aplicada.

Para cada opção no *combo box*, coloca-se um *option*, que deve ter o atributo *value*, o qual é usado no envio dos dados. Logo, visualmente, esse atributo não altera em nada a parte visual assim como também não altera em nada a parte de acessibilidade. É possível escolher um *option* para ficar ativo por padrão. Para isso, colocamos o atributo selected="selected".

```
1 -    ...
2 -    <p>
3 -    <label for="estado ">Estado:</label>
4 -    <select name="estado" id="estado">
5 -    <option value="rn">RN</option>
6 -    <option value="pe">PE</option>
7 -    <option selected="selected" value="sp">SP</option>
8 -    ...
9 -    </select>
10 -   </p>
11 -   ...
```



Fonte: Autoria própria.

Figura 12: *Combo box*

Há, também, a possibilidade de fazer o *combo box* aberto, assim o usuário tem como selecionar várias opções: utilizando o CTRL do teclado. Para isso, usamos o atributo multiple="multiple" na *tag select*.

```
1 -     ...
2 -     <p>
3 -     <label for="uf">Estado:</label>
4 -     <select name="uf" id="uf"  multiple="multiple">
5 -     ...
6 -     </select>
7 -     </p>
```

Fonte: Autoria própria.

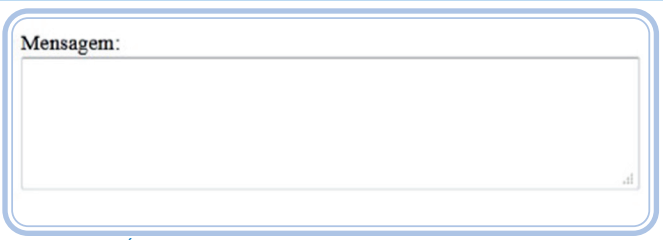
Figura 13: *Combo box* com opção de seleção múltipla

Esse visual do combo box não é comum. Muitos usuários não sabem que podem selecionar várias opções, por isso deixe sempre uma mensagem clara, informando que se pode apertar no CTRL e clicar em vários itens! Faça o teste!

Área extensa para entrada de dados

Há uma área para colocar dados que tenham mais texto e isso é possível ao quebrarmos linha, dando *enter*. A *tag* textarea tem *tag* de abertura e fechamento para facilitar que uma informação já seja colocada. Por mais que não tenha nada dentro, deve ter a marcação de abertura e fechamento. É obrigatório colocar três atributos: *name*; *cols*, que define a largura; e *rows*, que define a altura.

```
8 -     ...
9 -     <p><label>Mensagem
10 -    <textarea name="mensagem" cols="5"> </textarea>
11 -    </label></p>
12 -    ...
```



Fonte: Autoria própria.

Figura 14: Área de texto longo

Organizador de dados

Agrupa elementos relacionados do formulário e cria uma borda fina ao redor da área. A *tag legend* cria uma legenda para a área. Podem ter vários *fieldset* em um só formulário.

```
13 -    ...
14 -    <fieldset>
15 -    <legend>Dados Pessoais</legend>
16 -    ...
17 -    </fieldset>
18 -
19 -    ...
```

Fonte: Autoria própria.

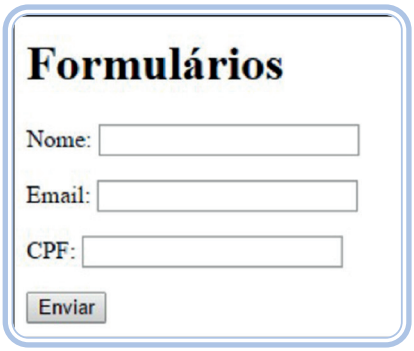
Figura 15: Organização dos campos do formulário

Formatação diagramada de formulário com CSS

Para o formulário da imagem a seguir, temos este código:

```
1 -     ...
2 -     <form action="resultado.html" method="get">
3 -             <p>
4 -                     <label for="nome">Nome:</label>
5 -                     <input type="text" id="nome" name="nome" />
6 -             </p>
7 -             <p>
8 -                     <label for="email">Email:</label>
9 -                     <input type="email" id="email" name="email" />
10 -            </p>
11 -            <p>
12 -                    <label for="cpf">CPF: </label>
13 -                    <input type="number" unselectable="off" id="cpf"
```

```
name="cpf" />
14 -            </p>
15 -
16 -            <p>
17 -                    <input type="submit" value="Enviar" />
18 -            </p>
19 -    </form>...
```



Fonte: Autoria própria.

Figura 16: Formulário não diagramado

Note que, na imagem acima, temos quatro componentes, três campos de texto e um botão de submissão. Para os campos de texto, temos os rótulos e as caixas, mas essas caixas estão desalinhadas. Alguns desenvolvedores pouco informados organizam esses componentes com tabelas. De fato, teríamos uma tabela com quatro linhas e duas colunas, porém essa forma de diagramação não é ideal, pois está fora do padrão, uma vez que as tabelas foram criadas para tabelar dados e não diagramar *layout*. Lembrando que, se trabalharmos da forma "incorreta" ou não ideal teremos a mesma exibição para o caso de trabalharmos dentro dos padrões. Vamos, então, à forma correta de utilização com CSS.

```
1 -     label{
2 -             float: left;
3 -             width: 100px;
4 -     }
```

Na linha 1, definimos o seletor "*label*". Apenas esse seletor será formatado. O atributo "*float*" faz o componente "flutuar" para a esquerda, valor "*left*", e, por último, o atributo "*width*" que define a largura do rótulo.

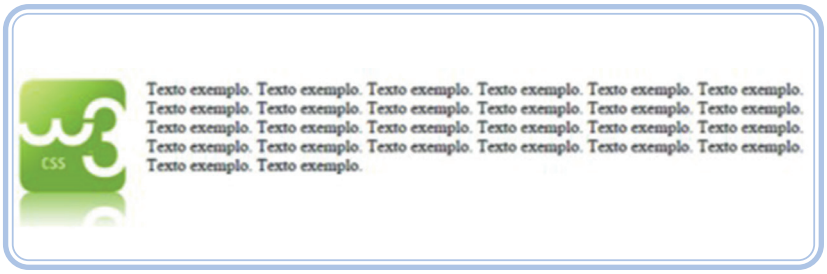
Fonte: Autoria própria.

Figura 17: Formulário diagramado com CSS

No código HTML, um pouco mais acima, trabalhamos com o campo de texto fora do rótulo. Vimos a definição do campo de texto dentro ou fora. Se o campo de texto for aplicado fora, temos de usar o atributo "*for*" no "*label*" e "id" no campo de texto com mesmo valor. Se você trabalhou, no seu formulário, o outro método, pode aplicar esse CSS a uma outra TAG, mas alterará o HTML. A marcação SPAN possibilita a formatação. Para o CSS, em vez do seletor ser o "*label*", será o SPAN. São marcações diferentes, não são concorrentes. Pois temos de usar a marcação LABEL, a marcação SPAN é apenas para formatarmos este rótulo.

```
1 -     ...
2 -     <p>
3 -       <label><span>Nome:<span>
4 -                     <input type="text" name="nome" />
5 -       </label>
6 -     </p>
7 -     ...
```

O atributo "*float*" pode ser usado em qualquer componente. Ele é muito usado em imagens. Se aplicarmos "*float*" na imagem, existindo texto após ela, essa imagem ficará diagrama para a direita ou esquerda e o texto ficará do outro lado e abaixo.



Fonte: Autoria própria.

Figura 18: Imagem com float

```
1 -     …
2 -     <style>
3 -     img {
4 -         float: left;
5 -         margin-right: 20px;
6 -     }
7 -     </style>
8 -     </head>
9 -     <body>
10 -    <p><img src="w3css.gif" width="100" height="140">
11 -    Texto exemplo. Texto exemplo…
12 -    </p>
13 -    …
```

LEMBRE-SE

As marcações de estrutura foram suprimidas dos códigos, mas devem ser colocadas nos seus códigos!

RESUMINDO

Nesta aula, aprendemos a importância das tabelas em páginas *web*, que elas não podem ser trabalhadas para diagramar nosso *layout*. Aprendemos também como estruturar dados e mesclar células, como também a importância dos formulários para os serviços na internet e como criá-los.

Leituras complementares

Nos *link*s abaixo, há uma especificação mais detalhada de como criar as tabelas e os formulários. Existem mais marcações do que as que trabalhamos na aula, inclusive marcações novas, HTML5, que são interessantes, assim como tem os atributos possíveis para usar em cada marcação. Por exemplo, há um atributo no campo de texto do formulário para limitar o tipo de dado que é digitado, um telefone que só tem número. Isso é uma regra básica.

Tabelas:

W3 SCHOOLS. **HTML Tables**. Disponível em: <http://www.w3schools.com/

html/html_tables.asp>. Acesso em: 08 jul. 2015.

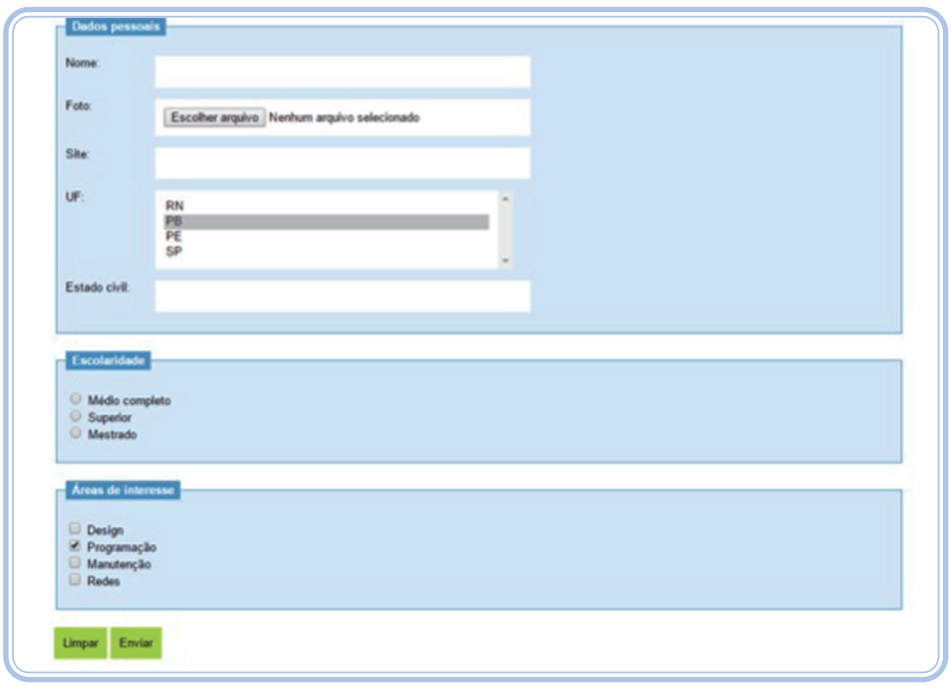
Formulários:

W3 SCHOOLS. **HTML Forms**. Disponível em: <http://www.w3schools.com/html/html_forms.asp>. Acesso em: 08 jul. 2015.

Avaliando seus conhecimentos

1) Crie uma página com notas de 10 alunos. Para tanto, defina uma coluna com nome, nota do primeiro bimestre, nota do segundo bimestre e média. No rodapé, coloque o total de alunos, a média do 1º bimestre, a média do 2º bimestre e a média das médias dos alunos. Lembre-se: esses cálculos são colocados de forma manual, pois o HTML não faz cálculos automáticos.

2) Crie uma página com o *design* do formulário de cadastro abaixo:



Aula 6 - Seções e posicionamento em HTML e pseudo-classes de CSS

Objetivos

Ao final deste módulo, você deverá ser capaz de:

identificar as marcações de seções em HTML;

trabalhar com posicionamento, identificação e classificação;

trabalhar com subseções em CSS.

Desenvolvendo o conteúdo

Meta informações

Há uma marcação HTML chamada META a qual tem como objetivo incorporar meta-informações do documento, isto é, informações sobre o conteúdo da página *web*. Elas são importantes para a catalogação, identificação e indexação do *site*, em motores de busca, por exemplo. Com ela, é possível definir informações sobre quem criou o documento, quando foi a última modificação, uma descrição do *site*, palavras-chave coerente com o seu conteúdo, entre outros. Todas essas informações, lógico, podem vir no *site*, mas, com a marcação META, nós podemos organizar melhor e tudo fica no código, não fica visível para o usuário pelo navegador, apenas se o usuário visualizar o código fonte.

Essas marcações devem vir sempre dentro da marcação HEAD. Não são obrigatórias e não há limite de uso. No código abaixo, há a marcação HEAD e, dentro, há quatro meta-informações: para definir a tabela de caracteres, para descrição da página, para palavras-chave e para definir o autor da página. A *tag* META tem dois atributos "name" e "content". No primeiro, colocamos o nome da informação a ser passada, existem vários que listamos abaixo, e,

no segundo atributo, o valor correspondente. Há também o atributo "http-equiv" que fornece um cabeçalho HTTP para a informação e o atributo "charset" que especifica a codificação dos caracteres do documento.

```
1 -     ...
2 -     <head>
3 -     <title>Seções e posicionamento em HTML e pseudo-classes em CSS</title>
4 -     <meta charset="UTF-8" />
5 -     <meta name="description" content="Aula de Web Design" />
6 -     <meta name="keywords" content="HTML,CSS " />
7 -     <meta name="author" content="EAD IFRN" />
8 -     </head>…
```

Valores para o atributo "name" da marcação META.

- Author: Responsável pelo desenvolvimento.
- Description: Descrição do *site*.
- Generator: Programa usada para o desenvolvimento
- Keywords: palavras-chave que definem o conteúdo do *site*.

Quando fazemos uma busca por "globo", no *site* de buscas Google.com, encontramos o seguinte resultado:

Fonte: Google.com

Figura 1: Pesquisa no google.com

Perceba que há uma frase bem organizada descrevendo o portal de notícias da Globo. Essa frase é justamente a frase colocada na meta-informação de

descrição no código do *site*. Não é uma frase que tenha na parte do corpo do *site*, ela tem apenas no código. Faça o teste. Entre no *site* do Google.com, busque por "Globo", veja a frase que aparecerá, abra o *site* da globo.com, procure a frase que apareceu, depois visualize o código fonte da página e busque novamente. Faça isso com outros *sites* e veja quais empresas têm um bom desenvolvimento de *sites*.

Utilizar essa técnica ajuda para que cada vez mais os resultados das nossas pesquisas sejam mais coerentes com a que nós buscamos. Se, no futuro, tivermos mais *sites* da *web* com essa organização da informação, todos ganharão, principalmente, o usuário.

Marcação DIV e SPAN

A marcação DIV tem como objetivo dividir a informação em áreas, essas divisões geralmente são feitas para facilitar na formatação CSS. Por exemplo, você tem um "banner" no topo da página e quer diagramá-lo à direita, ao lado do logotipo, uma das alternativas é colocar essa informação dentro da marcação de DIV e, no CSS, definir os atributos.

A marcação SPAN trabalha de forma similar, também não tem valor semântico, isto é, não há uma obrigatoriedade de qual tipo de conteúdo deve ter nestas marcações, mas há uma diferença grande entre elas. A marcação DIV é do tipo "bloco" e a marcação SPAN é tipo "linha".

Fonte: Autoria própria

Figura 2: Marcações de DIV e SPAN

Código HTML

```
1 -     …
2 -     <body>
3 -      <div>Exemplo</div>
4 -      <span>Exemplo</span>
5 -     …
```

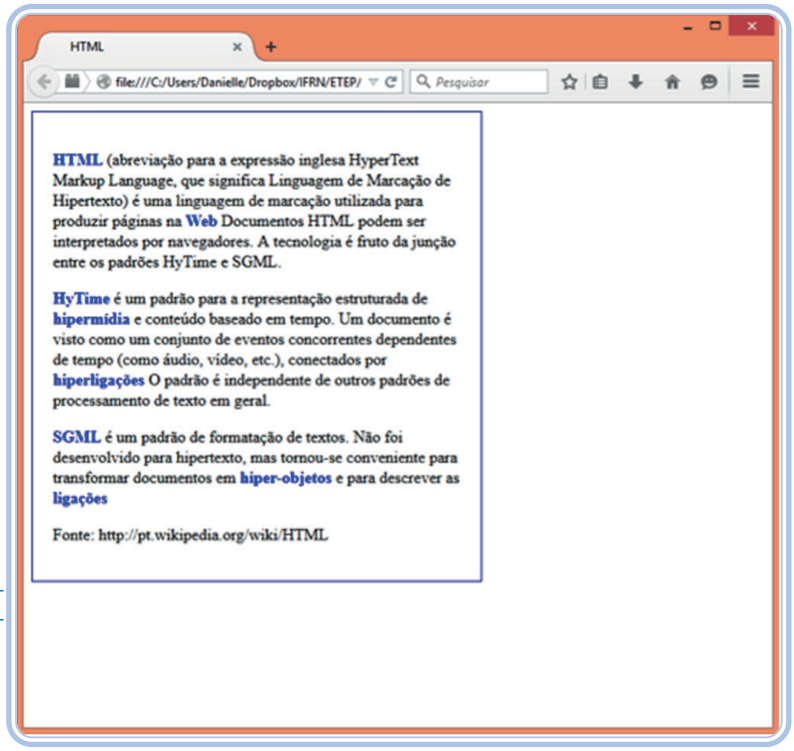
Código CSS

```
1 -     …
2 -     div, span{
3 -      background-color: gray;
4 -     }
5 -     …
```

DICA
Podemos trabalhar com o caracter "," vírgula no CSS para agrupar seletores. Quando usada, definimos esse(s) atributo(s) para todas as marcações. Não há limite de seletores, ou seja, podemos trabalhar com vários seletores e o(s) atributo(s) serão aplicados a todos.
É diferente de tralharmos com espaço.
p a{ atributos... } Aplicará os atributos apenas nas marcações âncora, A, que estiverem dentro das marcações de parágrafo, P.

Perceba que a marcação de DIV ficou com o plano de fundo ocupando toda a extensão da página, todo o bloco, e a marcação SPAN ocupou apenas a área que definia seus caracteres. Vê-se então que a marcação SPAN é mais usada para pequenas formações de palavras, termo etc e, a marcação, DIV para grandes áreas. Mas é possível alterar essa propriedade, a formatação padrão dessas marcações. O atributo DISPLAY no CSS tem como valores: BLOCK, INLINE... com isso, é possível transformar uma DIV em SPAN e vice-versa. Assim como qualquer outra marcação que tenha uma formatação diferente da padrão exibida no navegador. Por exemplo, a marcação A, âncora, é similar à marcação SPAN. Às vezes, é necessário que a área de hiper*link* seja maior para os usuários terem um acesso mais confortável e não apenas a área do texto, assim, é possível aplicar o atributo CSS DISPLAY e definir o valor BLOCK.

Neste exemplo, você pode utilizar a formatação para SPAN e DIV, para exemplificar.



Fonte: Autoria própria.

Figura 3: Formulário com método GET

Para o CSS:

```
1 -     …
2 -     div{
3 -                             border: 1px solid blue;
4 -                             padding: 20px;
5 -                             width: 400px;
6 -                     }
7 -                     span{
8 -                             font-weight: bold;
9 -                             color: blue;
10 -                    }
```

Para o HTML:

```
1 -     …
2 -     <div>
```

3 - HTML (abreviação para a expressão inglesa HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada para produzir páginas na Web Documentos HTML podem ser interpretados por navegadores. A tecnologia é fruto da junção entre os padrões HyTime e SGML.

```
4 -     <p><span>HyTime</span> é um padrão para a representação
estruturada de <span>hipermídia</span> e conteúdo baseado em tempo. Um
documento é visto como um conjunto de eventos concorrentes dependentes
de tempo (como áudio, vídeo, etc.), conectados por <span>hiperligações</
span> O padrão é independente de outros padrões de processamento de
texto em geral.</p>
5 -     <p><span>SGML</span> é um padrão de formatação de textos.
Não foi desenvolvido para hipertexto, mas tornou-se conveniente para
transformar documentos em <span>hiper-objetos</span> e para descrever
as <span>ligações</span></p>
6 -     <p>Fonte: http://pt.wikipedia.org/wiki/HTML</p>
7 -                     </div>…
```

Marcação HEADER

A nova marcação HEADER, nova porque não existia nas versões anteriores do HTML, foi criada para o HTML na versão 5. Os desenvolvedores trabalham as páginas com uma estrutura de "colunas" e, para dividir as informações, é necessário utilizar uma marcação. Antes, a marcação usada era a DIV, seção 2 deste material, mas essa marcação não tem valor semântico, por isso os desenvolvedores criavam uma divisão para o cabeçalho, outra para o rodapé, para o *menu* e assim sucessivamente. Percebendo esta inconsistência, o HTML5, definiu marcações específicas para área da página. Essa mudança também tem como ponto positivo a indexação da informação dessa página nos robôs de busca. O principal objetivo é tornar cada vez mais semântico o desenvolvimento.

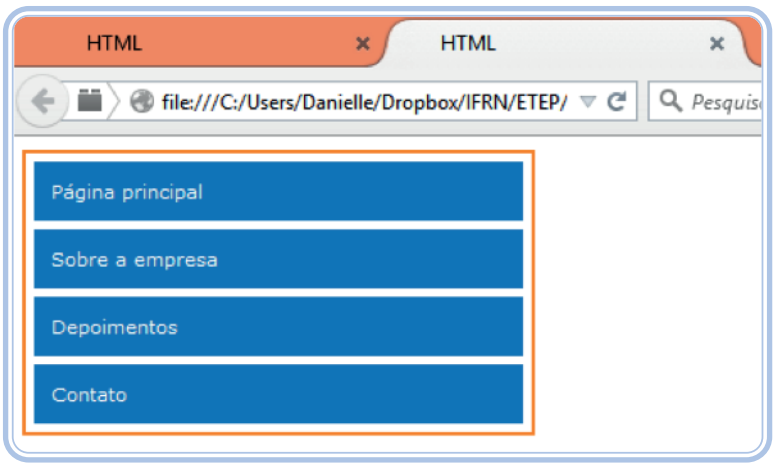
A marcação HEADER define a área do cabeçalho visual, pois há o cabeçalho HEAD que é diferente. O HEAD é o cabeçalho do documento, enquanto que, no HEADER, ficam informações como o nome do *site*.

```
1 -     …
2 -     <body>
3 -     <header>
4 -     <h1>Disciplina de Web design</h1>
5 -     </header>
6 -     …
```

Marcação NAV

A marcação NAV também foi criada com a mesma justificativa da marcação anterior. Quando definimos o design de uma página, sempre organizamos as informações com um menu de navegação. Para desenvolvê-lo, usamos a marcação NAV.

Para definirmos um menu de navegação, como todas as definições de marcação, temos de seguir uma semântica, assim, um menu é um grupo de itens que nos permite navegar pelas informações do *site*. E, como é uma lista, usamos a marcação de lista também. Como já dito, a lista OL é uma lista organizada que tem uma ordenação específica, logo, para o nosso menu, usamos uma lista ordenada e o tipo de ordem é PRIORIDADE! Pois é, para



Fonte: Autoria própria.

Figura 4: Entrada de texto

formularmos um menu, usamos a marcação NAV, a marcação de lista OL, consequentemente, LI e a marcação A que nos permitirá criar o hiper*link*. Vamos desenvolver um menu de navegação como a Imagem abaixo.

```
1 -     ...
2 -     <style type="text/css">
3 -
4 -             nav{
5 -                     border:  2px solid #ff9000;
6 -                     font-family: Verdana, Geneva, Arial, Helvetica, sans-
serif;
7 -                     font-size: 11px;
8 -                     width: 300px;
9 -             }
10 -            nav a{
11 -                    color: white;
```

```
12 -                    text-decoration: none;
13 -                    display: block;
14 -                    background-color: #1a84c1;
15 -                    margin: 5px;
16 -                    padding: 10px;
17 -            }
18 -            li{
19 -                    list-style-type: none;
20 -            }
21 -            ol{
22 -                    padding: 0;
23 -                    margin: 0;
24 -            }
25 -    </style>
26 -    </head>
27 -    <body>
28 -    <nav>
29 -            <ol>
30 -                    <li><a href="principal.html">Página principal</a></
li>
31 -                    <li><a href="sobre.html">Sobre a empresa</a></li>
32 -                    <li><a    href="depoimentos.html">Depoimentos</
a></li>
33 -                    <li><a href="contato.html">Contato</a></li>
34 -
35 -            </ol>
36 -    </nav>…
37 -
```

O código HTML, da linha 28 a 36, contém a marcação NAV para delimitar o início e fim do menu, a lista organizada e as marcações de hiper*link* com os respectivos direcionamentos, atributo HREF.

No código CSS:

- Linha 5 a 8: atributos para serem aplicados a NAV, uma borda, uma largura delimitada e formatação de texto.
- Linha 11: por padrão, o hiper*link* é de cor azul. Se quisermos alterar a cor, temos de aplicar essa formatação ao âncora. Não adianta definir no LI, ou no NAV, pois o seletor tem de ser o A.
- Linha 12: por padrão, o hiper*link* tem sublinhado, o que é bem interessante para o usuário do sistema saber que "ali" é uma ligação para outra página, entretanto, estamos em *menu*, então é comum ser uma área

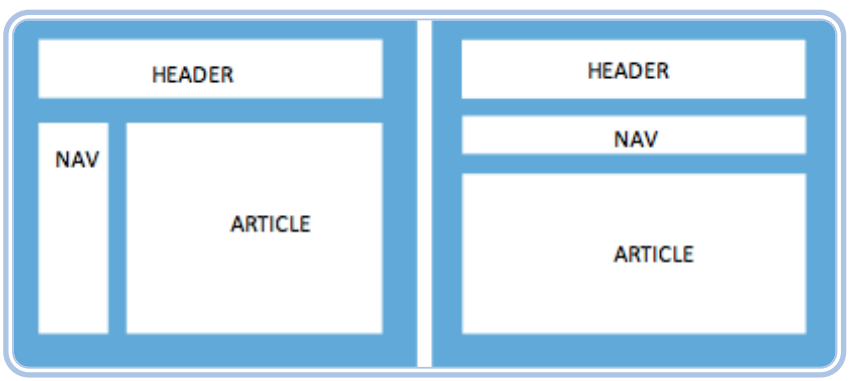
com itens "clicáveis". Podemos remover essa formatação definindo o valor do atributo "text-decoration" igual a "none".

- Linha 13: o atributo "display" já foi comentado nesta aula. Ele pode definir que uma marcação seja em "linha" ou em "bloco". O padrão dos âncoras é ser em "linha", então temos que defini-los em bloco e, assim, termos uma área para clicar maior aumentando a acessibilidade do menu.
- Linha 19: o atributo "list-style-type" retira a visualização dos "bullets" da lista.
- Linha 15 e 16: define espaçamentos para que os itens não fiquem colados.
- Linha 22 e 23: algumas marcações têm formatação padrão pelos navegadores, como, por exemplo, as listas têm os "bullets", os *link*s têm o sublinhado, os parágrafos têm margens, dessa forma, as listas também têm esses espaçamentos internos e externos que podem ser removidos com CSS. Os atributos "padding" e "margin" são definidos "zero" e assim temos maior controle sobre esses espaços.

Marcação ARTICLE

A marcação ARTICLE recebe um bloco de informações relacionadas com notícias, post de blog, comentários, fórum de discussão, enfim, informações corridas de texto comum, parágrafos.

Considerando as marcações já definidas nesta aula, temos então uma página com uma organização visual com cabeçalho, HEADER, navegação, NAV, e conteúdo, ARTICLE. A seguir, há dois *layouts* que ilustram uma organização visual de uma página organizada com essas três marcações.

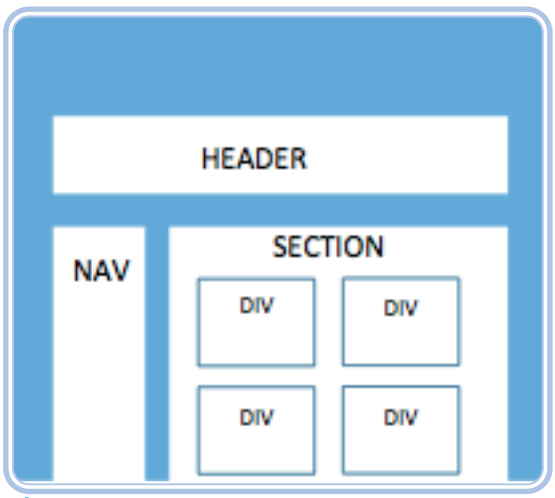


Fonte: Autoria própria.

Figura 5: Layouts com marcações HEADER, NAV e ARTICLE

Marcação SECTION

Essa marcação define um bloco de "chamadas" de um assunto específico. Por exemplo, na página principal, temos várias áreas que fazem referências com os conteúdos das páginas internas, mas não contém, lógico, todas as informações necessárias, só são "chamadas", como uma capa de jornal. Cada "chamada" é uma seção. As seções agrupam elementos como *link*s, conteúdo, imagens, listas, entre outros. Essa marcação é similar à marcação ARTICLE, mas a diferença é que a SECTION é usada para agrupar diferentes tipos de informações, enquanto que ARTICLE é um bloco de informações de um assunto específico.



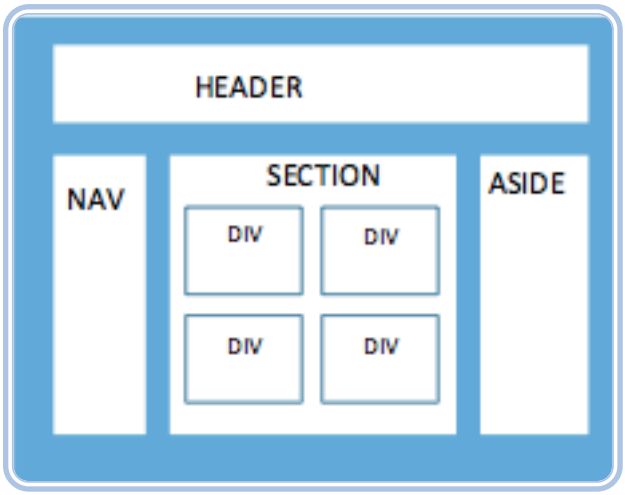
Fonte: Autoria própria.

Figura 6: Layout com SECTION

No exemplo que segue, tem um design com as marcações HEADER, NAV, SECTION e DIV. Esta última faz um papel de organizador dos grupos, cada DIV é um assunto.

Marcação ASIDE

Essa marcação trabalha com informações complementares que ficarão "ao lado", tradução literal de aside. A maioria das estruturas dos *sites* tem uma coluna que fica ao lado, contém: citações, banners, blocos de navegação, entre outros. Como você pode ver, essa marcação, assim como as anteriores, são para demarcar uma área.



Fonte: Autoria própria.

Figura 7: Layout com ASIDE

Marcação FOOTER

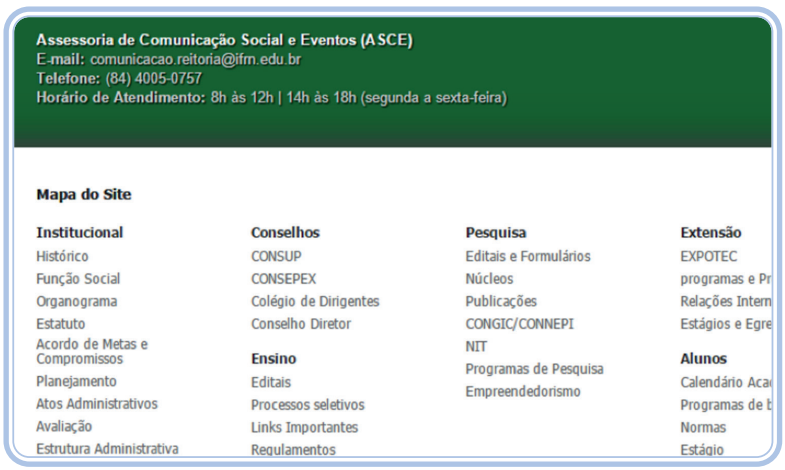
A marcação FOOTER recebe as informações de rodapé, endereço, *menu* de navegação, formulário de contato, *link*s para as redes sociais, nome do desenvolvedor ou da empresa que desenvolveu o *site*, copyright, entre outras informações. Às vezes, a página nem tem rodapé.

O *site* da Caixa, por exemplo, não tem rodapé na página inicial e, nas páginas internas, só tem *link*s para redes sociais.

Fonte: http://fgts.caixa.gov.br

Figura 8: *rodapé do site da caixa*

O *site* do IFRN tem um rodapé com informações de endereço, horários, redes sociais e o menu de navegação completo.



Fonte: Autoria própria.

Figura 9: Rodapé do site do IFRN

O *site* do Banco do Brasil tem menos informações, relacionadas com o contato, o que é mais comum.

Fonte: <http://bb.com.br>

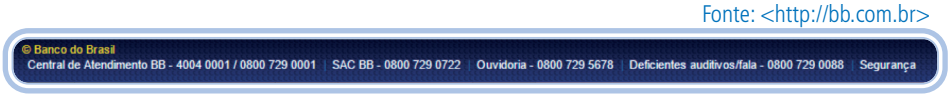
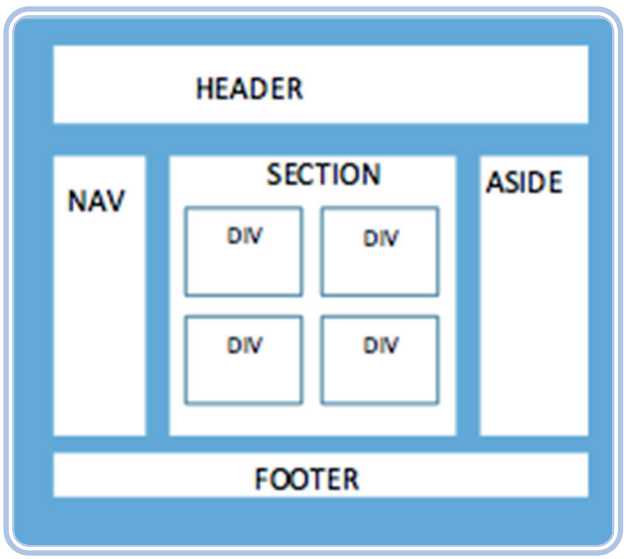


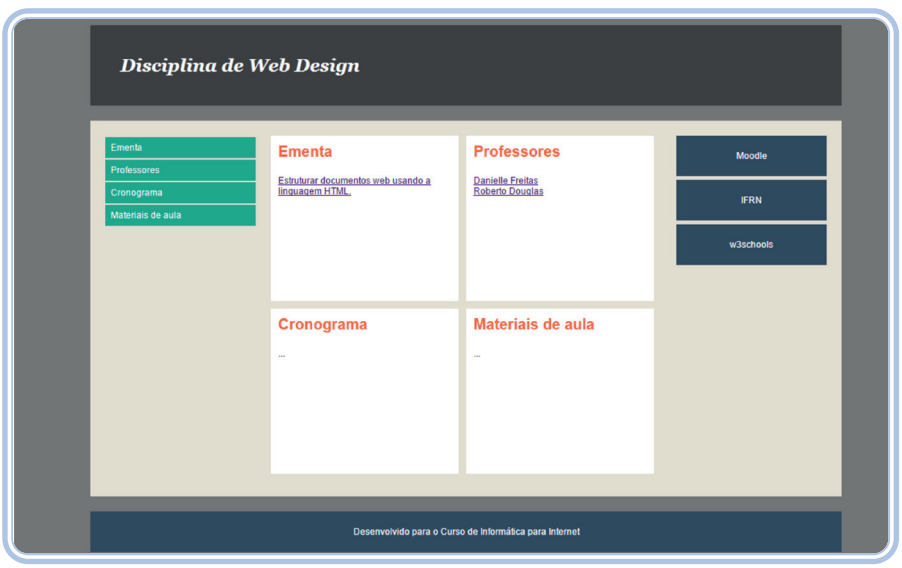
Figura 10: *Rodapé do site do Banco do Brasil*

Na Figura 10, temos a estrutura para uma interface Web. Abaixo, transformaremos em código HTML e o CSS irá estruturar.



Fonte: Autoria própria.

Figura 11: Estrutura com FOOTER e outras marcações para seções



Fonte: Autoria própria.

Figura 12: Protótipo da nossa página

```
1 -     <!DOCTYPE html>
2 -     <html lang="pt-br">
3 -         <head>
4 -             <meta charset="utf-8">
5 -             <title>Web Design</title>
6 -             <meta name="description" content="Página
desenvolvida para a disciplina de Web Design">
7 -             <meta name="author" content="Danielle Freitas -
IFRN">
8 -             <link href="principal.css" type="text/css"
rel="stylesheet" />
9 -         </head>
10 -
11 -        <body>
12 -            <div id="centro">
13 -                <header>
14 -                    <h1>Disciplina de Web Design</h1>
15 -                </header>
16 -                <div id="principal">
17 -                <nav>
18 -                    <ol>
19 -                        <li><a href="#">Ementa</
a></li>
20 -                        <li><a
href="#">Professores</a></li>
21 -                        <li><a
href="#">Cronograma</a></li>
22 -                        <li><a href="#">Materiais
de aula</a></li>
23 -
24 -                    </ol>
25 -                </nav>
26 -
27 -                <section>
28 -                    <div>
29 -                        <h2>Ementa</h2>
30 -                        <p><a href="">Estruturar
documentos web usando a linguagem HTML.</a></p>
```

```
31 -
32 -                                   </div>
33 -                                   <div>
34 -                                       <h2>Professores</h2>
35 -                                       <p><a       href="">Danielle
Freitas</a></p>
36 -                                       <p><a       href="">Roberto
Douglas</a></p>
37 -                                   </div>
38 -                                   <div>
39 -                                       <h2>Cronograma</h2>
40 -                                       <p>...</p>
41 -                                   </div>
42 -                                   <div>
43 -                                       <h2>Materiais de aula</h2>
44 -                                       <p>...</p>
45 -
46 -                                   </div>
47 -
48 -                               </section>
49 -                               <aside>
50 -                                   <ol>
51 -                                       <li><a href=" http://ead.ifrn.
edu.br/">Moodle</a></li>
52 -                                       <li><a     href="http://portal.
ifrn.edu.br">IFRN</a></li>
53 -                                       <li><a              href="http://
w3schools.com">w3schools</a></li>
54 -                                   </ol>
55 -                               </aside>
56 -                               </div>
57 -                               <footer>
58 -                                   <p>
59 -                                       Desenvolvido para o Curso de
Informática para Internet
60 -                                   </p>
61 -                               </footer>
62 -                           </div>
63 -                   </body>
64 -       </html>
65 -
```

Pontos importantes no código HTML:

- Na linha 2, há a definição do tipo de linguagem do código e todas as páginas têm uma principal. Se houver pequenos trechos com outras linguagens, esses podem ser definidos com o atributo "lang". É importante colocar para os robôs de busca relacionarem as pesquisas com a linguagem equivalente.
- Nas linhas 4, 6 e 7, temos as marcações META, estudadas acima.
- Na linha 8, tem a chamada para o CSS. Veja que foi definido o nome como "principal.css". Alguns desenvolvedores criam os arquivos com o nome "css.css", "estilo.css", mas não há nenhuma semântica nesse termo. De fato, é mais óbvio, porém, se colocarmos um termo mais coerente, nos ajudará no futuro ou mesmo com outras pessoas que acessarem o código. Lembrando que podemos colocar quantas marcações de LINK quisermos. Nesse caso, só usei uma, mas uma página pode conter vários arquivos. Por exemplo, "incial.css", formatação específica para a página inicial; "internas.css", formatação específica para as páginas internas; e assim por diante.
- Na linha 12, temos uma marcação de DIV. Essa tem um atributo ID, identificador, que utilizaremos para identificar essa DIV no CSS, com o termo "centro". O objetivo dessa divisão é colocarmos todo o conteúdo do *site* no centro da página, isto é, se você diminuir a janela do seu navegador, vai observar que a *interface* de texto ficará no centro até 1000 pixels, que foi o valor definido para a largura. Esse valor tem de sempre acompanhar a proposta do *site*, porque tem *sites* que são mais largos, outros menos, portanto o ideal é fazer uma análise dos usuários.
- Na linha 13, existe a marcação HEADER que contém uma tag de título de nível 1, H1, lembrando que podemos colocar imagens dentro dessa marcação H1.
- Na linha 16, há marcação DIV com o atributo identificador com o valor "principal", que é apenas para termos melhor controle das três colunas que contém na *interface*.
- Na linha 17, temos a marcação NAV com uma lista de itens organizados, pois há uma ordem de prioridades nos itens. No atributo HREF, da marcação A, tem um "#", mas é só porque não tem página interna para esse *link*, se não colocarmos nada o *link* não fica "clicável".
- Na linha 27, existe a marcação SECTION com as marcações DIV, H2 e parágrafo.
- Na linha 49, temos a marcação ASIDE com *link*s externos para outros *sites*. Perceba que há um protocolo no atributo HREF da marcação A, para ir a um *site* externo, o protocolo "http://".
- Na linha 57, há a marcação FOOTER com um parágrafo apenas.

Código do arquivo de formatação, "principal.css":

```
1 -     *{
2 -             margin: 0;
3 -             padding: 0;
4 -             list-style-type: none;
5 -     }
6 -     body{
7 -             font-family: arial, verdana;
8 -             font-size: 12px;
9 -             background-color: #8E8E8E;
10 -    }
11 -    header{
12 -            background-color: #565656;
13 -            padding: 40px;
14 -            margin: 20px 0;
15 -    }
16 -    header h1{
17 -            color: white;
18 -            font-family: georgia, garamond;
19 -            font-style: italic;
20 -    }
21 -    div#centro{
22 -            width: 1000px;
23 -            margin: 0 auto;
24 -    }
25 -    div#principal{
26 -            background-color: #ECE8DF;
27 -            display: table;
28 -    }
29 -
30 -    nav  , aside , section{
31 -            width: 200px;
32 -            float: left;
33 -            padding: 20px;
34 -    }
35 -    section{
36 -            width: 520px;
37 -            padding: 20px 0;
38 -    }
```

```
39 -    nav li a{
40 -            background-color:#01AF98;
41 -            color: white;
42 -            text-decoration: none;
43 -            display: block;
44 -            padding: 7px;
45 -            margin: 2px 0;
46 -    }
47 -
48 -    section div{
49 -            background-color: white;
50 -            padding: 10px;
51 -            margin: 0 10px 10px 0;
52 -            float: left;
53 -            width: 230px;
54 -            height: 200px;
55 -    }
56 -    section div h2{
57 -            margin-bottom: 20px;
58 -            color: #F5764B;
59 -            font-size: 20px;
60 -    }
61 -    aside li{
62 -            margin-bottom: 5px;
63 -            padding: 20px;
64 -
65 -            background-color: #445D73;
66 -            text-align: center;
67 -    }
68 -    aside li a{
69 -            color: white;
70 -            text-decoration: none;
71 -    }
72 -    footer{
73 -            background-color: #445D73;
74 -            padding: 20px;
75 -            clear: both;
76 -            margin: 20px 0;
77 -            color: white;
78 -            text-align: center;
79 -    }
```

Pontos importantes do código CSS:

- O seletor "*" na linha 1 formata TODAS as marcações do código HTML. Por padrão, algumas marcações vêm com espaçamentos internos e externos, então nós "zeramos" para que cada espaçamento seja colocado individualmente.
- Na linha 21, a marcação DIV com o identificador "centro" é formatada. Perceba que é importante usarmos o caracter "#" para relacionar o identificador. Podemos usar também apenas "#centro" sem a marcação, mas é menos semântica.
- Na linha 30, formatamos três marcações de uma só vez com o caractere ",". É definido uma largura, espaçamento interno, "padding", e "float" para os três. Esta última é o atributo principal para estruturarmos nosso código e nos permitirá criar o visual de colunas na *interface*.
- Na linha 48, temos o seletor "section div", formatamos todas as DIVs que estão dentro da marcação SECTION.
- Na linha 72, formatamos o FOOTER. Além das definições de formatação simples, tem o atributo "clear" que "limpa" a página em relação aos elementos que estão o atributo "float". Esses dois atributos trabalham juntos, enquanto que o "float" permite que outros elementos fiquem lado a lado, o atributo "clear" vai justamente limitar esse posicionamento, já que a definição "float" influencia nos elementos que estão após ele.
- Os valores usados nas formatações de largura, espaçamentos internos e externos fazem toda a diferença na *interface*. Perceba que a largura dos elementos DIV com identificação principal é 1000, logo as três colunas no meio não podem ter, na sua soma de larguras e espaçamentos, mais que esse valor. Os valores definidos no espaçamento interno, *padding*, influenciam na largura total, isto é, se um elemento tem 200 de largura e 20 de "padding" para os lados, então ele ocupará 240 pixels, 200+20 da direita + 20 da esquerda.

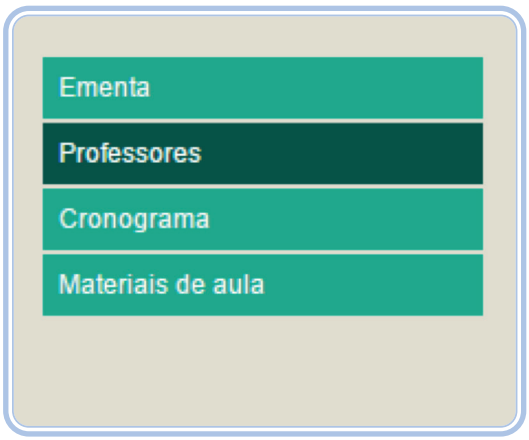
Formatando com pseudo-classe

Uma pseudo-classe é como uma classe dinâmica no elemento. Basicamente elas manipulam um determinado estado do elemento. Confira, a seguir, um quadro com as pseudo-classes mais usadas.

Quadro 1: Pseudo-classes mais usadas

Pseudo-Classe	Descrição de uso
:hover	Efeito sobre o objeto quando o *mouse* passa por cima
:visited	Quando o elemento foi visitado. Normalmente aplicado para *link*.
:nth-child(odd)	Seleciona as linhas ímpares de uma marcação, muito usado em tabelas para ficarem com cores diferentes nas linhas.
:last-child	Seleciona o último elemento.
:first-child	Encontra o primeiro elemento.

Fonte: Autoria Própria.



Fonte: Autoria própria.

Figura 13: Efeito hover no menu.

Se adicionarmos as linhas abaixo no CSS, teremos um efeito no menu, uma vez que, ao passar o *mouse* no item, o plano de fundo ficará com um tom mais escuro. Podemos colocar outras formatações, como cor de texto, espaçamento, imagem de fundo, enfim, muitas opções. Essa pseudo-classe também pode ser aplicada a outras marcações, claro que o efeito de clicar é da marcação de âncora.

```
1 -     nav li a:hover{
2 -             background-color: #01665A;
3 -     }
```

Formatando os posicionamentos

O atributo "position" no CSS especifica o método de posicionamento utilizado pelo elemento. Temos aqui a liberdade de posicionar um elemento em qualquer lugar na nossa página, inclusive sobre os objetos que já estão na página. Valores para o elemento "position":

Quadro 2: Valores para o atributo position

VALORES	DESCRIÇÃO
Absolute	Posição absoluta em relação aos atributos "top", "left", "right" e "bottom".
Relative	O elemento se mantém fixo mesmo que o documento seja movido pela barra de rolagem.
Relative	O elemento é posicionado em uma posição relativa à ocupada normalmente, mas permite o uso dos atributos de coordenadas: "top", "left", "right" e "bottom".
Static	Estado padrão do elemento

Fonte: Autoria Própria.

Se adicionarmos ao nosso código HTML:

```
1 -     ...
2 -     </nav>
3 -                         <div id="mensagem">
4 -                             <h2>Feriado</h2>
5 -                             <p>No dia 04/06 não haverá aula.</
p>
6 -                         </div>
7 -
8 -                         <section>
9 -     ...
```

E nosso código CSS:

```
1 -     div#mensagem{
2 -             position: absolute;
```

```
3 -             top: 200px;
4 -             left: 200px;
5 -             padding: 50px;
6 -             font-size: 40px;
7 -             color: white;
8 -             background-color: #445D73;
9 -     }
```

Temos essa alteração na *interface*. Confira na imagem a seguir.

Fonte: Autoria própria.

Figura 14: Aviso com posicionamento absoluto

Perceba que nosso *banner* ficou acima dos elementos já existentes na página. Esse é o efeito do atributo "position" com o valor "absolute" e com os outros atributos "top" e "left" podemos definir as coordenadas para essa mensagem. Mas essa caixa não ficou no centro, então temos de fazer uma outra modificação.

```
1 -     div#centro{
2 -             width: 1000px;
3 -             margin: 0 auto;
4 -             position: relative;
5 -     }
```

O atributo "position" com o valor "relative", na marcação DIV#centro, irá tornar a mensagem com as coordenadas RELATIVAS ao DIV#CENTRO e não mais ao corpo da página, que é o padrão. Agora as coordenadas "top" serão em relação à localização da DIV#CENTRO, assim como a coordenada "left". Logo, a mensagem está a 200 pixels do topo da DIV#CENTRO e 200 pixels à esquerda da DIV#CENTRO.

Fonte: Autoria própria.

Figura 15: Aviso com posicionamento absoluto e relativo

LEMBRE-SE!

As marcações de estrutura foram suprimidas dos códigos, mas devem ser colocadas nos seus códigos!

Atividade de aprendizagem 1

Com base no conteúdo estudado e em suas leituras prévias, crie uma página para cada código e organize-a na pasta do respectivo módulo.

RESUMINDO

Nesta aula, aprendemos como criar um *site* com a estrutura de blocos com conteúdo semânticos usando DIV, SPAN, HEADER, NAV, ARTICLE, SECTION, ASIDE e FOOTER. Além de aprendermos como usar pseudo-classes em CSS e posicionamentos.

Leituras complementares

Logo abaixo, há um endereço eletrônico, no qual você terá acesso à documentação do HTML5 que está bem mais interativa do que a das versões anteriores. Porém, infelizmente, nem todos os navegadores se ajustaram ainda a ela, mesmo assim, é importante estudá-la e entender como usar cada propriedade.

Documentação do HTML5

W3C RECOMMENDATION. **HTML5**. [20--?]. Disponível em: <http://www.w3.org/TR/html5/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements>. Acesso em: 17 jul. 2015.

Avaliando seus conhecimentos

Na aula de hoje, você criou uma página. Agora, você também criará páginas internas, modificando os textos e as formatações com pseudo-classes.

Aula 7 - Arquitetura da Informação

Objetivos

Ao final deste módulo, você deverá ser capaz de:

entender as preocupações mais amplas de como navegar para ajudar em uma solução mais apropriada de estruturar a informação;

aprender os mecanimos e os tipos de navegação;

enteder e aprender como rotular a navegação.

Desenvolvendo o conteúdo

A navegação tem um papel fundamental em adaptar nossas experiências na *web*. Ela fornece acesso à informação de uma maneira que melhora o entendimento, refletindo na credibilidade da empresa.

Quando a navegação *web* funciona bem, é pouco notada ou até nem é notada. Na imagem abaixo, temos o *site* da w3schools. Nele, não há nada de interessante sobre a navegação, pois ela está lá quando você precisa.

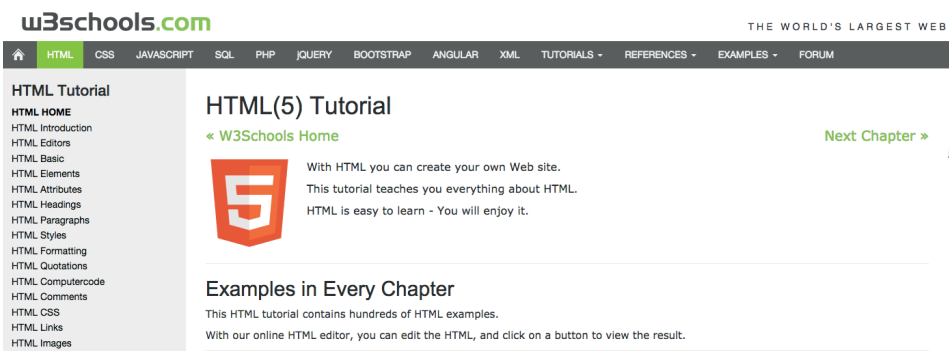


Figura 1: Site da w3schools

Onde seu foco de atenção fixou-se primeiro? Se você não está buscando algo específico, seus olhos perambulam pela página.

Sem tomar conhecimento, você criou um esquema para a página, de forma a ajudá-lo a entender sua navegação e conteúdo. De modo geral, os diversos elementos são compostos para criar um sistema de navegação. Em destaque, na imagem abaixo, a navegação principal está na horizontal e a navegação local, na vertical.

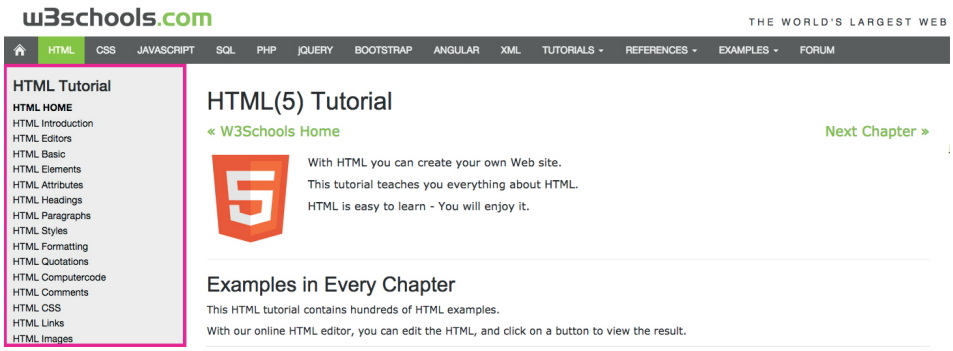
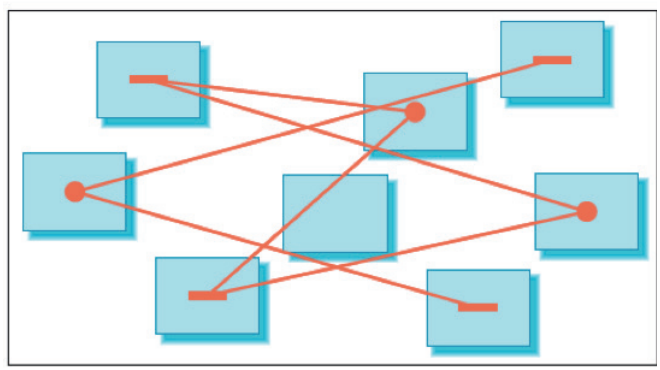


Figura 2: Navegações do *site* <www.w3schools.com/html>.

Funções da navegação *Web*:

Fornecer acesso à informação em um *site*

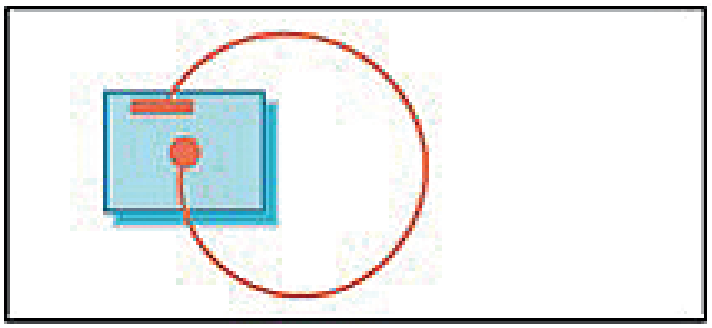
Abaixo, temos uma imagem que define uma coleção de páginas ligadas umas às outras sem organização hierárquica ou padrão de ligação particular. Esse modelo não facilita a busca por informações.



Fonte: Kalbach (2009)

Figura 3: Coleção de páginas ligadas

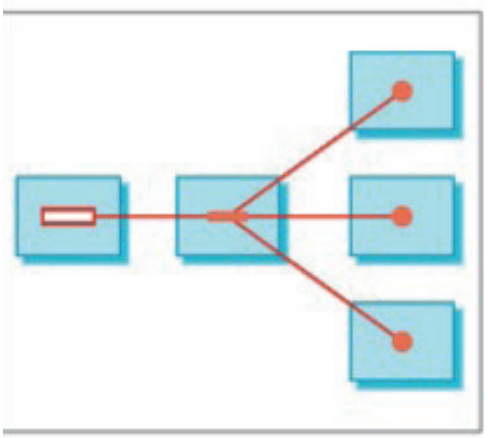
Abaixo, há um exemplo de uma página ligada a ela mesma. Sugere uma maneira potencialmente eficiente de acessar a informação, porém a experiência para visitantes leigos seria mais complicada.



Fonte: Kalbach (2009)

Figura 4: Página ligada a ela mesma

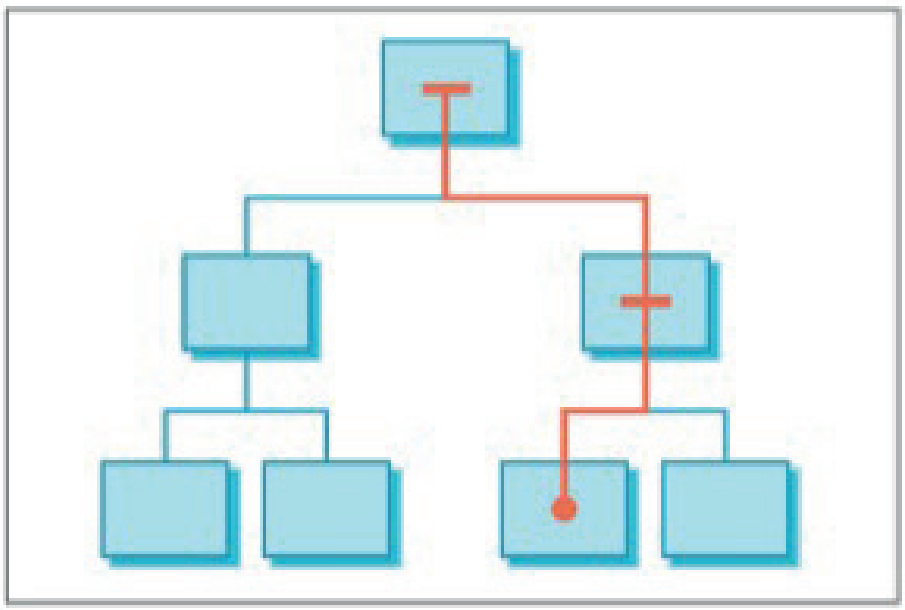
No modelo de busca, disposto abaixo, há uma página com os resultados para a consulta. Buscar, certamente, é uma maneira eficiente de acesso à informação.



Fonte: Kalbach (2009)

Figura 5: Modelo de busca

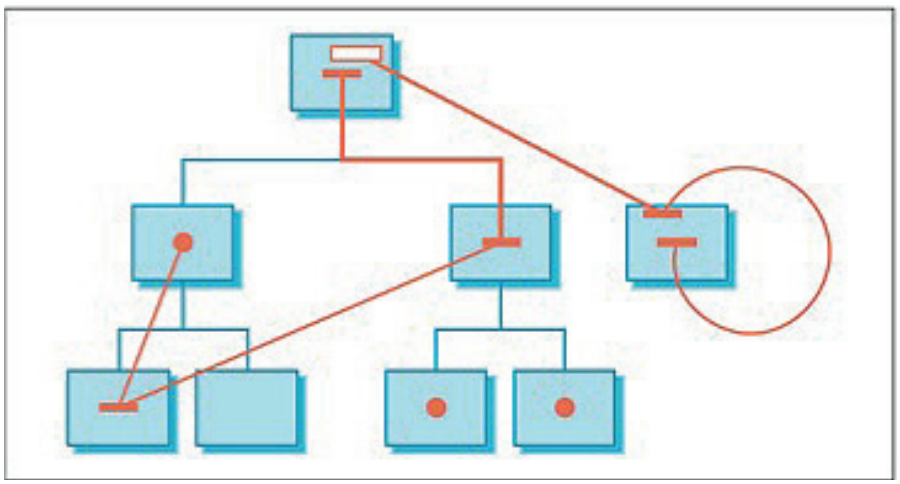
O modelo de navegação estrutural é um conjunto de *link*s. Por meio dele é possível clicar através de uma hierarquia de opções de navegação, atualizando o conteúdo da página. Não existem *link*s embutidos, nem funções de busca.



Fonte: Kalbach (2009)

Figura 6: Navegação estrutural

A navegação balanceada é uma mistura desses modelos, mais comum nos *sites*. De um modo geral, um sistema de navegação *web* deve fornecer um acesso à informação de forma eficiente e balanceada.



Fonte: Kalbach (2009)

Figura 7: Navegação híbrida

Mostra a localização de um *site*

A navegação não se trata apenas de ir de uma página a outra; trata-se também de prover orientação. Algumas vezes, as pessoas necessitam saber onde

elas estão em um *site web*. Muitos têm apontado para três necessidades básicas de orientação *web*. Enquanto navegam em um *site*, os usuários normalmente precisam saber: onde estão, o que têm e para onde ir, a partir de onde está. A navegação ajuda a definir o contexto juntamente com títulos de página e outros elementos.

Mostrar o assunto de um *site*

A navegação mostra a amplitude e o tipo do conteúdo de um *site web* e suas ofertas ou o "assunto" do *site*. Ela cria uma coerência geral, significativa, dos assuntos e conteúdos do *site* e cria expectativas. Dessa forma, o conhecimento dos tópicos principais de um *site* pode afetar a abordagem que as pessoas usam para encontrar informações neste *site*. Note que isso não implica que a navegação mostrará o escopo do *site* em termos de quantidade de páginas. Ao invés disto, ela reflete a profundidade do assunto de um *site*.

Reflete a marca, a credibilidade e a retabilidade

Uma marca é, frequentemente, pensada em termos de sua manifestação visual: logotipo, cores, fontes etc. Mas uma marca é muito mais do que isso. A posição de uma marca afeta essencialmente cada aspecto de um produto ou serviço, incluindo a navegação *web*. Por último, uma marca é uma promessa ao consumidor sobre os produtos e serviços oferecidos. Um objetivo comum no *design* de *sites web* é tornar um *site* mais confiável possível. Isso ajuda a passar a mensagem adiante. Se um visitante não consegue encontrar a informação que precisa, isso pode ser custoso para seu negócio.

Mecanismo de Navegação

A estrutura de um *site* sugere um mecanismo específico de navegação, como uma barra de navegação horizontal, que em outros *sites* não podem ser aplicados por ter um limite de espaço, já que colocar duas linhas de itens de *menu* na horizontal torna a navegação confusa. Logo abaixo, temos os mecanismos existentes de navegação.

Navegação por passos

Consiste em rótulos de texto e/ou seta para navegação anterior e próxima. Esse mecanismo é importante quando a decisão em um dado passo afeta algo no passo seguinte. Exemplos: assistente, processo de compra, seções de um documento extenso, questionários, entre outros.

Figura 8: Navegação por passos

Fonte: <http://www.submarino.com.br>.

Figura 9: Navegação por passos no site da W3C

Fonte: <http://www.w3c.br/cursos/html5/conteudo/capitulo1.html>.

Navegação por paginação

Você conhece muito essa navegação. É similar à navegação por passos, mas com a diferença que nessa é exibido o número de páginas. É interessante quando a navegação por passos e a paginação são trabalhadas juntas. Exemplo na imagem abaixo do *site* do Google. Existe uma tendência de "página" infinita, ou seja, é uma página que exibe os resultados à medida que o usuário vai usando a barra de rolagem. Por exemplo, no Google Imagens, os resultados mostram várias imagens e a medida que usamos a barra de rolagem, outras imagens são exibidas.

Figura 10: Navegação por paginação

Fonte: <http://www.google.com>.

Trilha de migalhas de pão

Esse termo é uma analogia às migalhas de pão de João e Maria. As migalhas de pão no *site* têm um objetivo similar, mostrar ao usuário por onde ele passou, assim como possibilitar o retorno. Todos os *sites* com muitas informações devem ter migalhas de pão e essas devem ser bem aplicadas, isto é, permitir que o usuário clique nas seções anteriores. No exemplo da imagem abaixo, temos as seções anteriores com um *link*. Essas seções são separadas por uma barra "/", usam-se também setas, dois pontos, traço, entre outros.

Fonte: <http://portal.ifrn.edu.br/campus/canguaretama/noticias/oportunidades-de-estagio>.

Figura 11: Navegação com migalhas de pão para a página do IF

Fonte: <http://www.americanas.com.br>. Acesso em: 30 jul. 2015.

Figura 12: Navegação com migalhas de pão para as páginas das Americanas.com

Árvore de navegação

A informação é organizada de forma hierárquica. Muito usada em sistemas operacionais para navegar no diretório de arquivos, essa árvore é simples e efetiva, mas devemos ter cautela com os ramos da árvore, pois, ao clicar, as opções devem ser exibidas sem recarregar toda a página.

Exemplo na imagem abaixo:

Fonte: <http://support.safaribooksonline.com/home >

Figura 13: Árvore de navegação

Mapas do *site*

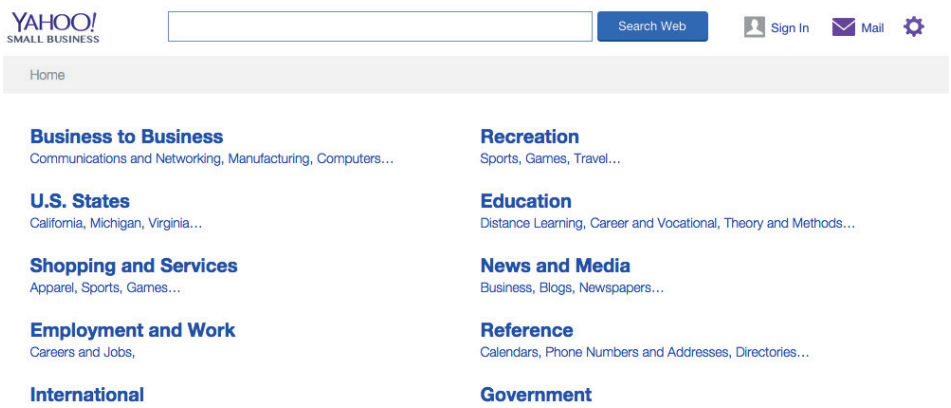
Representação estrutural do *site* que fornece uma visão de uma vez só, otimizando, inclusive, a indexação do conteúdo em motores de busca. São usados em *sites* com grande quantidade de informações. Abaixo, o mapa do *site* do portal do IFRN.

Fonte: <http://www.porta.ifrn.edu.br>

Figura 14: Mapa do *site*

Diretórios

Fornecem acesso a página via tópicos. São úteis para informações mistas, isto é, sem relacionamento hierárquico. Muitos *sites* de busca tinham esse método de organização.



Fonte: <https://business.yahoo.com/>

Figura 15: Navegação por diretórios

Nuvens de tags

Listam *link*s alfabeticamente organizados e, ao mesmo tempo, exibem o peso de cada termo por sua frequência no *site*, quanto maior o *link*, mais importante ele é. Esses termos são muito utilizados em *blogs* e são definidos pelos seus próprios autores. Esse mecanismo de navegação é limitado, portanto o usuário deve ter outras formas de acesso a informação, por paginação, por busca, entre outros.

Fonte:<http://www.usabilidoido.com.br/cat_arquitetura_da_informacao.html>

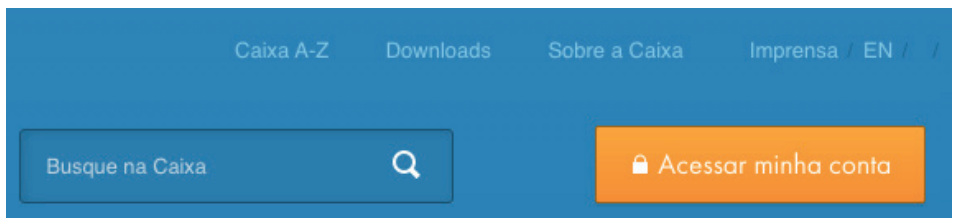
Figura 16: Nuvem de *tags*

Fonte:<http://www.usabilidoido.com.br/cat_arquitetura_da_informacao.html>

Figura 17: Nuvem de *tags*

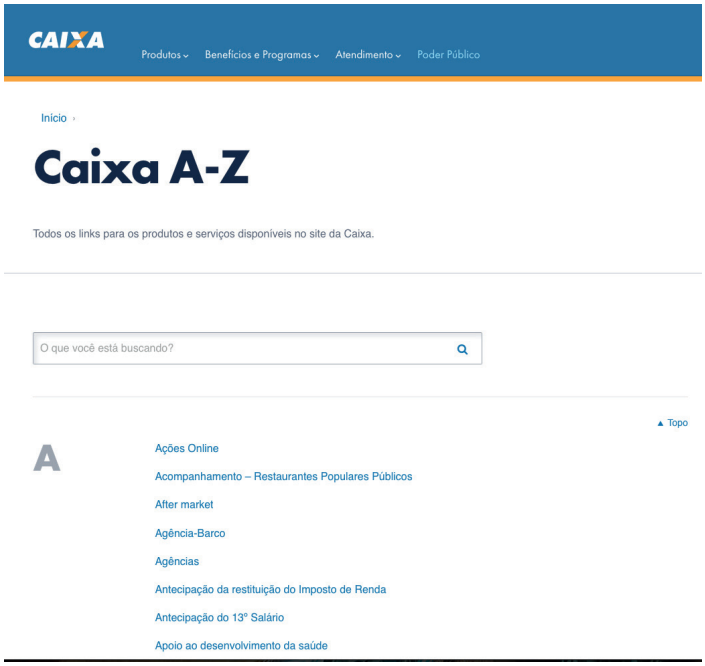
Índices de A-Z

Guia alfabético para os tópicos, similar ao índice no fim do livro. Indicado para *site* com muita informação, como o *site* do banco da Caixa Econômica. Na página inicial, tem um *link* "Caixa a-z", em destaque na imagem abaixo. Ao clicarmos, temos toda a informação organizada por ordem alfabética e com um campo de busca.



Fonte: <http://www.caixa.gov.br/Paginas/home-caixa.aspx>.

Figura 18: Navegação por índice



Fonte: <http://www.caixa.gov.br/caixa-a-z/Paginas/default.aspx>

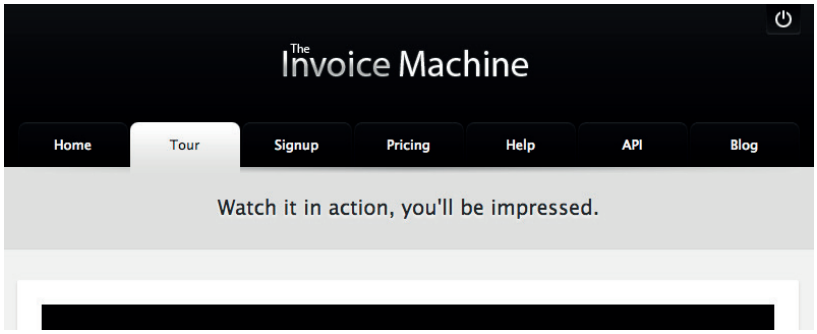
Figura 19: Índice no *site* da Caixa

Barras de navegação e abas

As abas físicas são muito utilizadas, pois permitem uma visualização clara de uma área, por exemplo, um fichário com documentos, papéis, as abas permitem o acesso rápido. As abas na interface são imitadas, mas exigem cautela, já que se usa, geralmente, na horizontal e não se tem muito espaço. Assim como as abas que foram clicadas, devem ficar em destaque, como no exemplo abaixo no *site* da Microsoft.

Fonte: <http://windows.microsoft.com/pt-br/windows-8/apps#cat=t0>

Figura 20: Navegação por abas



Fonte: <http://invoicemachine.com/tour>

Figura 21: Navegação por abas

Menus verticais

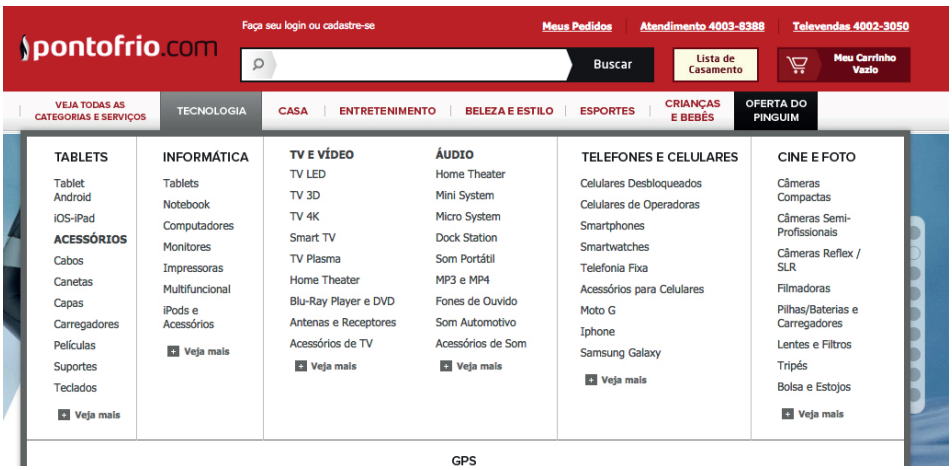
Mecanismo de navegação amplamente utilizado em *design* de navegação *web*. Geralmente, mais flexíveis que barras de navegação ou abas, pois há mais espaço. Indicado quando se tem muita informação para organizar e não indicado no caso contrário, já que a barra à esquerda ocupa uma área da interface que poderia ser usada para uma outra informação.

Fonte: <http://portal.ifrn.edu.br/>

Figura 22: Menus verticais

Menus drop-down

São itens exibidos ao passar o mouse sobre uma categoria específica, usados para *sites* com muito conteúdo, como o ponto frio, imagem abaixo, e para *sites* com quantidade menor de informações, mas que as organizam por categorias.



Fonte: <http://www.pontofrio.com.br/>

Figura 23: Menu *drop-down*

ATIVIDADE

Identifique cada mecanismo de navegação exposto acima, faça um *prints-creen* na tela e circule ou destaque a área que contém essa navegação. Por meio de um comentário, analise se a navegação foi coerente, se funciona de forma efetiva e se é necessária, porque nem sempre a navegação é necessária onde foi colocada.

Tipos de navegação

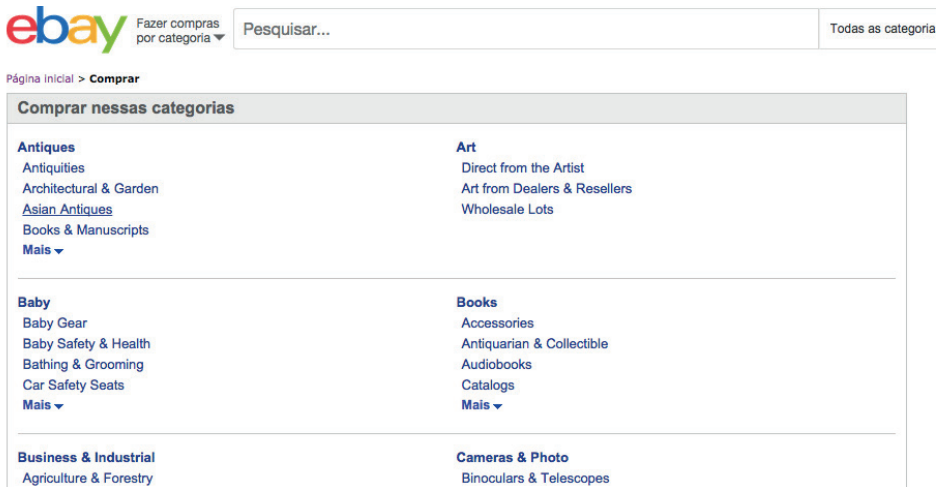
Agrupando as informações similares, é possível determinar o propósito e a importância da navegação. Não existe um conjunto de regras de uso, apenas variações.

Para termos uma boa organização, é necessário, incialmente, pensar como um visitante e não como Designer ou Arquiteto da Informação. Não se pode

refletir a estrutura organizacional de uma empresa na estrutura organizacional da informação do *site*. Por exemplo, todas as instituições de ensino são organizadas por diretorias, gerências e nessas existem as estruturas dos cursos. Seria interessante organizar dessa forma o *site* de uma instituição? Por que não? No IFRN, a DIATINF, Diretoria Acadêmica de Gestão e Tecnologia da Informação, tem o curso de Comércio Exterior. Será que é claro para o usuário procurar esse curso na DIATINF?

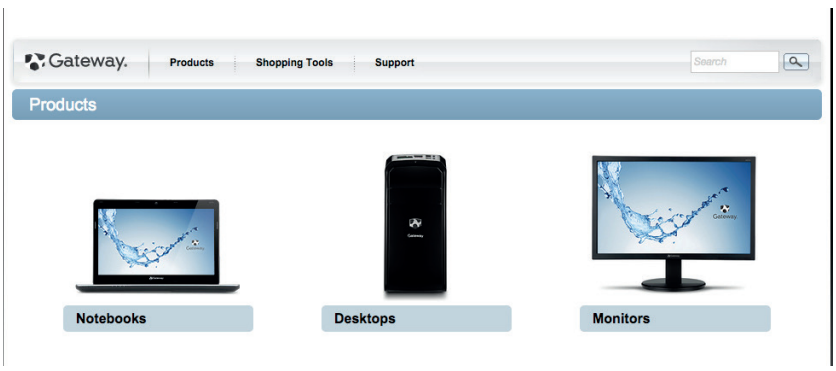
Para cada tipo de página, temos uma navegação mais adequada, não ideal, porém mais adequada. Existem três tipos de páginas: páginas navegacionais, páginas de conteúdo e páginas funcionais.

As páginas navegacionais direcionam o visitante para o seu objetivo principal: página inicial contém informações sobre as principais informações do *site*; página de aterrissagem são páginas de categorias e contêm uma visão geral de departamentos, por exemplo, *site* do ebay.com, imagem abaixo; páginas de galerias, similares às páginas de aterrissagem, mas fornecem uma visão específica de produtos ou fotos, imagem do *site* da Apple abaixo; e páginas de resultados de busca, baseadas em palavras-chave.



Fonte: <http://www.ebay.com>.

Figura 24: Página de aterrissagem



Fonte: <http://us.gateway.com>.

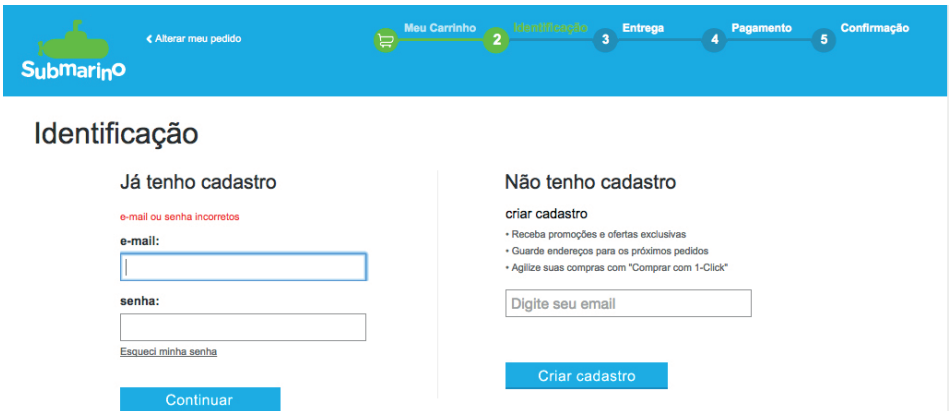
Figura 25: Página de galerias

As páginas de conteúdo são as informações que os usuários estão procurando: textos, histórias, notícias, artigo, currículo, entre outros. O foco dessas páginas deve ser o conteúdo, porque uma navegação extra ou gráficos, muitas vezes, sobrecarregam a página. A tendência atual é que essas sejam coerentes, isto é, o conteúdo deve ser objetivo para alguns casos, por exemplo, história da empresa, de modo geral, quem lê isso?! As exceções para colocar esse tipo de informação são os casos onde a história é importante, exemplo, Museu de Santos Dumont, imagem abaixo. Na dúvida se colocar ou não, deve-se fazer uma coleta de dados e ou investigar o histórico do *site*.

Fonte: <http://www.museusantosdumont.org.br>

Figura 26: Site do Museu de Santos Dumond

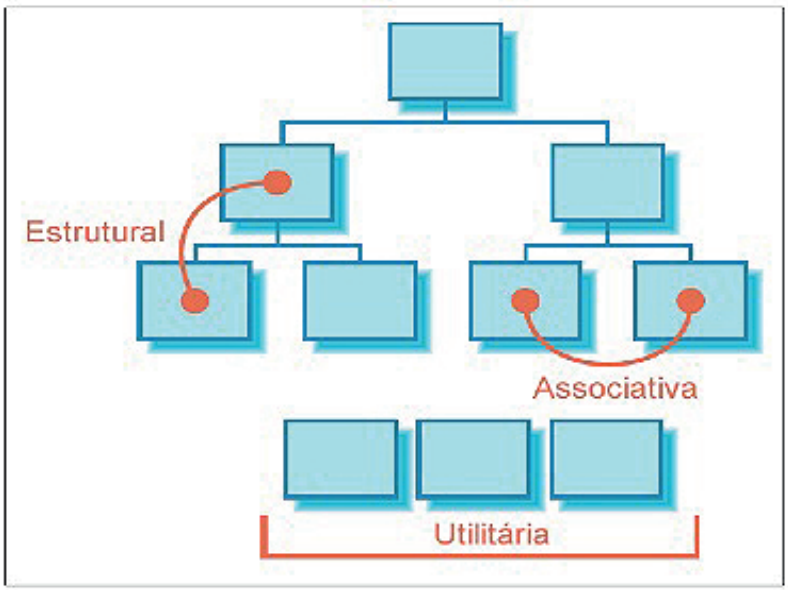
As páginas funcionais têm tarefas específicas para serem completadas *online*: enviar um e-mail, fazer uma compra, fazer uma busca, formulário de cadastro, *aplicativo* de documentos de escritório *online*, entre outras. Podem ter pouco ou nenhum texto. É importante manter o foco na *interface* para a tarefa para não tirar a atenção do usuário, uma vez que muitas informações com propagandas, menus extensos, atrapalham. No *site* da submarino. com, por exemplo, nem o menu principal é exibido. A página é totalmente orientada à tarefa, imagem abaixo.



Fonte: <http://carrinho.submarino.com.br>

Figura 27: Página orientada à tarefa - funcional

A figura abaixo ilustra as três categorias de navegação mais usadas no *site*: estrutural, associativa e utilitária.

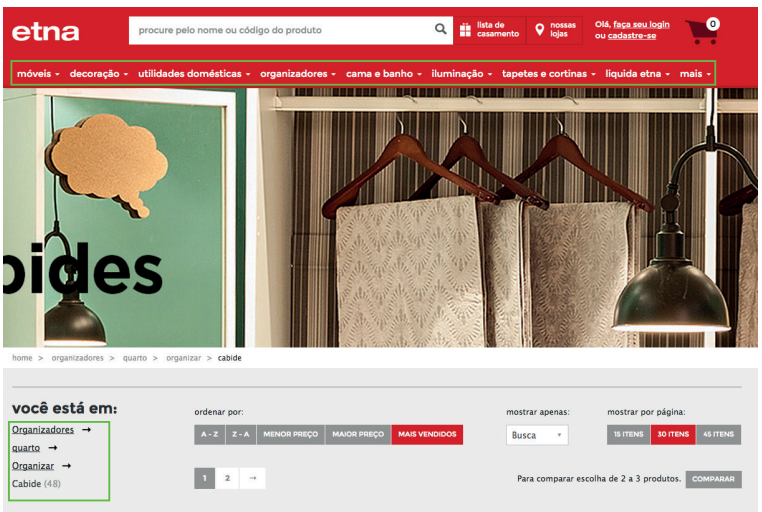


Fonte: Kalbach (2009).

Figura 28: Tipos de navegação

Navegação estrutural

Segue a estrutura do *site* e permite que os usuários se movam para cima e para baixo em diferentes pontos na hierarquia estrutural de como as informações foram organizadas. Pode ser dividida em dois tipos: principal, que representa as páginas em um nível mais alto; e a navegação local, que permite navegar pelos níveis mais baixos da estrutura. No *site* abaixo, da Etna, temos a navegação principal na horizontal e a navegação local na vertical.



Fonte: <https://www.etna.com.br/etna/c/organizadores/quarto/organizar/cabide>

Figura 29: Navegação principal e local

Fonte: Kalbach (2009).

Figura 30: Disposições para as navegações

Existem várias disposições para essas navegações. Na imagem abaixo, há três opções, mas é possível colocar a navegação local à direita também, não é muito comum, entretanto, dependendo do estilo do *site* e do seu público-alvo, é possível.

Navegação associativa

O ato de clicar, tanto para nós, desenvolvedores, como para os usuários, é prático, por isso vamos tornar a navegação mais prática aprendendo a técnica. A navegação associativa faz ligações entre os diferentes níveis de uma hierarquia e permite ao, usuário, acessar informações similares e que não estão na navegação local, por exemplo. Existem três formas de criar essa associação: (i) navegação contextual, com *link*s embutidos no próprio texto e também com *link*s relacionados, imagem Figura 10; (ii) navegação adaptativa, que traz *link*s a partir de um filtro colaborativo que se baseia no comportamento dos usuários, por exemplo, é muito comum, em comércio eletrônico, ter abaixo do produto selecionado "O usuário que comprou esse produto também comprou", Figura 11; e (iii) navegação de rodapés, localizada no fim da página, permite ao usuário que chegou até esse ponto acessar o *menu* sem precisar ir ao topo da página, muito comum para *sites* extensos, muitas vezes comporta um mapa do *site*.

Fonte: Kalbach (2009).

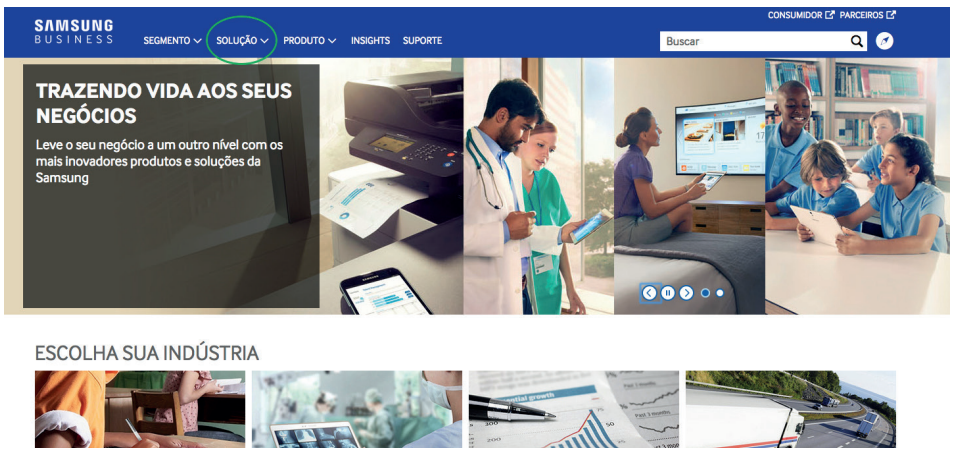
Figura 31: Links embutidos e links relacionados

Fonte: <http://pontofrio.com>.

Figura 32: Navegação adaptativa

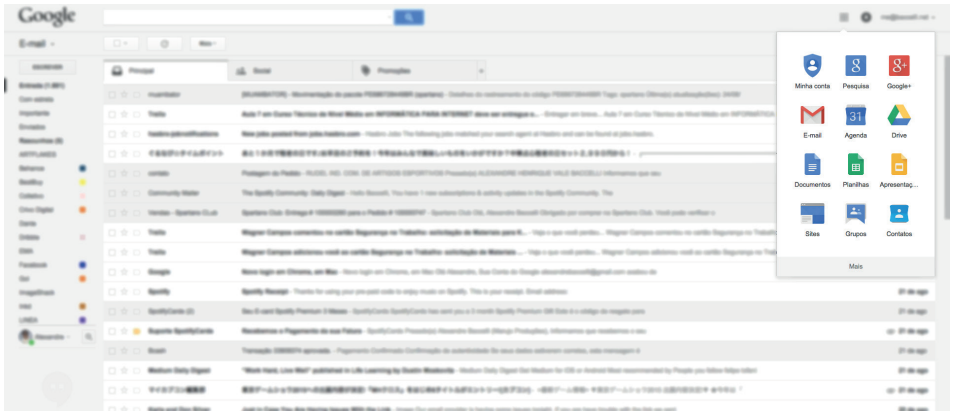
Navegação utilitária

Conecta ferramentas e funcionalidade com o objetivo de ajudar o usuário a navegar pelo *site*. Exemplos: *link*s para *sites* de outras companhias ou *sites* específicos de um produto da empresa, como na figura 12 que exibe um menu de ferramentas com opção de páginas funcionais e na figura 13 que apresenta logotipos com *link*s para as páginas iniciais, seletores de línguas e seletores de países.



Fonte: <http://www.samsung.com/br/home> e <http://www.samsung.com/br/business/>

Figura 33: Navegação extra-site



Fonte: <http://gmail.com>.

Figura 34: Menu de ferramentas do Google

Rotulando a navegação

Rótulos são os termos escolhidos nos itens do *menu*, no nome das páginas, entre outros. Essa é uma parte subestimada no processo de desenvolvimento, mas faz parte do conteúdo do *site*, sua funcionalidade e estrutura. Vamos, agora, aprender a importância dos rótulos e como criar bons rótulos.

Sistema de rotulagem

Os títulos dos navegadores são um rótulo importante para o usuário encontrar a "tab" do *site* no navegador, assim como, quando salvar o *site* nos favoritos e imprimir a página.

Atualmente, as URL são chamadas de amigáveis porque contém informações sobre a página de forma clara e que ajuda, inclusive, na indexação do conteúdo em *site* de busca. No *site* do Banco do Brasil, por exemplo, quando clicamos em "2° via do boleto de cobrança" a URL não contém informações da página. Diferente do *site* da Caixa, conforme você pode conferir nas imagens abaixo.

Fonte: <https://www63.bb.com.br/portalbb/boleto/boletos/hc21e,999,3322,10343.bbx>.

Figura 35: URL no site do BB

Fonte: <http://www.caixa.gov.br/atendimento/2-via-boleto/Paginas/default.aspx>

Figura 36: URL no site da Caixa

É padrão que todas as páginas tenham um título bem visível e coerente com o *link* clicado. Por exemplo, na imagem acima, do *site* da caixa, o título é grande e claro "2º via de boletos".

Bons rótulos

Os bons rótulos falam a língua do usuário, o qual deve entender naturalmente as informações. Devemos evitar abreviações, jargões da empresa, terminologia técnica. Por exemplo, o *site* da NIKON, imagem abaixo, organiza seus produtos de uma forma técnica e pouco comum para os fotógrafos não profissionais. O que seriam Câmeras Digitais SLR?! Qual a diferença entre câmeras Nikon 1 e as outras?!

Fonte: <http://www.nikon.com.br/index.page>

Figura 37: Rótulo no *site* da Nikon

O tamanho do rótulo também reflete no entendimento. Rótulos curtos com uma ou duas palavras é mais indicado, mas nem sempre é necessário seguir esse padrão, pois, às vezes, rótulos mais extensos ajudarão aos usuários a prever o que virá pela frente.

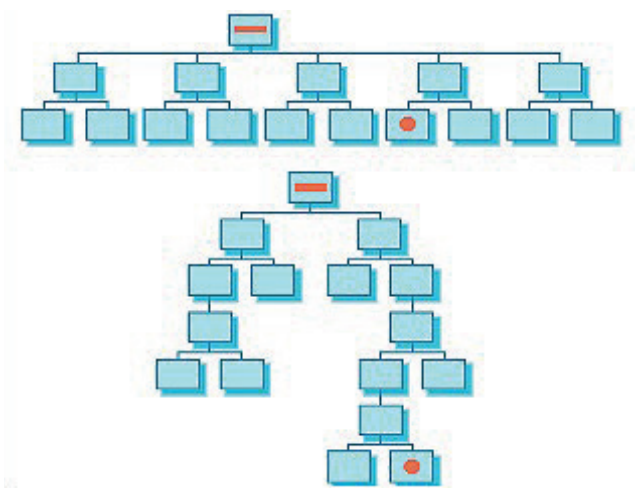
Avaliando a navegação

Engana-se quem pensa que trabalhar com *interface* de usuário é um assunto subjetivo. A avaliação do *site*, por parte do usuário, vai justamente provar se o que foi desenvolvido é coerente, porque, se o usuário, ao navegar pelo *site*, sentiu dificuldades, é porque o trabalho está ERRADO, mas, se ele conseguir fazê-lo com facilidade, está CERTO. Não é subjetivo, não é mais ou menos, é exato!

Uma boa navegação deve ser invisível! E é possível medir essa efetividade de várias formas e, comumente, se testa com usuários. Devemos avaliar na navegação: balanço, *feedback*, consistência e inconsistência, se é fácil de aprender, se os rótulos são claros e se há clareza visual.

Balanço

Equilíbrio entre o número de itens do *menu* versus sua profundidade. Quanto menos itens de navegação, mais profunda é a estrutura. Essa estrutura não funciona tão bem, apesar de organizar melhor a informação, em alguns casos. Na imagem abaixo, há dois tipos de estruturas, opção 1: estrutura mais ampla e menos profunda; e opção 2: estrutura menor e com maior profundidade, o que requer mais esforço dos usuários.

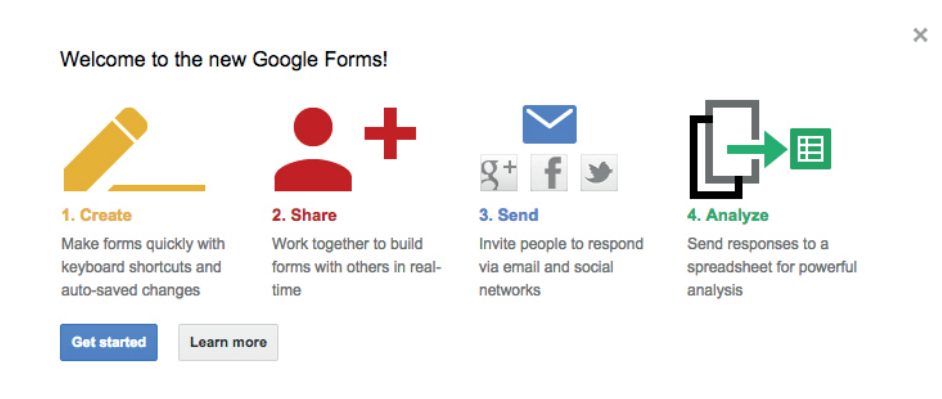


Fonte: Kalbach(2009)

Figura 38: Equilíbrio nos itens do menu

Fácil de aprender

Um *site* não deve ter um treinamento de como usar, porque a navegação deve ser clara e coerente com o tipo do usuário, ou seja, se está voltado para um público iniciante ou mais profissional. O google forms, por exemplo, exibe uma explicação sucinta que aparece apenas na primeira vez que abrimos a ferramenta, mas aparece também ao clicar em "ajuda". O texto que é exibido é um "tutorial" simples com imagens e bastante prático. Fácil de usar não necessariamente é necessário aprender, mas tem *sites* que precisamos aprender, aqueles que têm muitas funcionalidades.



Fonte: <https://docs.google.com/forms>.

Figura 39: Tutorial do Google Forms

Consistência e inconsistência

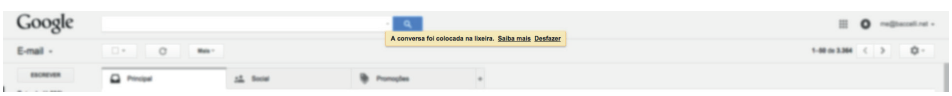
Essa recomendação é primária para o design de *interface*. Em termos de navegação, refere-se ao mecanismos e *link*s que aparecem em uma localização da tela. Os rótulos devem também ter a mesma aparência ao longo do *site*. No *site* do google, há um menu com ferramentas que é exibido em quase todos os seus *sites*. Na imagem abaixo, temos o gmail, google tradutor e o google.com para compararmos.

Fonte: <http://gmail.com>, <http://google.com> e <http://translate.google.com.br>

Figura 40: Consistência nas páginas

Feedback

Define que é necessário se certificar de que a *interface* informe ao usuário o que está acontecendo, ou seja, todas as ações precisam de *feedback* instantâneo para orientá-lo. Por exemplo, no gmail.com, ao excluir um *e-mail*, é exibida uma mensagem para o usuário com o objetivo de confirmar que a operação foi realizada com sucesso, ver imagem abaixo. Inclusive, antigamente, apenas era apresentada uma mensagem e agora é possível "desfazer". Nos sistemas de *e-mails*, em geral, é possível desfazer a operação, mas é necessário ir na lixeira e mover a mensagem para a caixa de entrada. Outra funcionalidade que demonstra bem o *feedback* é a tela pós *login* do gmail.com.



Fonte: <http://gmail.com>

Figura 41: Mensagem de *feedback* após excluir um *e-mail*

Fonte: <http://gmail.com>.

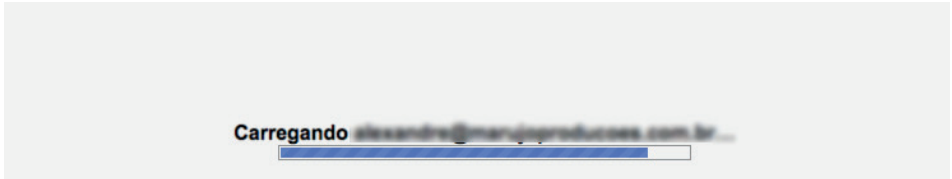


Figura 42: *Feedback* após o *login*

Rótulos Claros

Como foi explicado na aula anterior, é importante que os rótulos sejam avaliados também. Os usuários interagirão com a *interface* se entender o que eles significam e onde eles os levam.

Clareza visual

A cor, a fonte e a estrutura fazem parte da experiência do usuário. O *design* não tem apenas a função de atrair, mas tem também a função de criar um senso de orientação, melhorando assim, a usabilidade da navegação.

A tendência atual são *interfaces* adaptadas para monitores grandes, *smart phones* com telas pequenas e *tablets* que têm telas medianas. As interfaces devem se adaptar para a melhor experiência e, principalmente, adaptar a navegação, muitos dispositivos são *touchscreen*, como os usuários saberão onde é para clicar?! A navegação tem esse compromisso.

Assim, com esse conteúdo, terminamos nosso curso. Com todas essas técnicas, faremos uma *web* melhor para que os usuários acessem as páginas e não tenham dúvidas de como as coisas funcionam, evitando que eles experimentem o desconforto de uma página "não funcionar" porque o desenvolvedor não fez a programação direito.

ATIVIDADE

Pesquise sobre Wireframe e desenvolva três telas para um sistema de visualização de filmes na internet. Uma tela deve ser para a página incial, outra para a de busca e a última para a página de visualização de um filme específico. Existem muitos sistemas que fazem isso atualmente, porém a navegação deixa muito a desejar, como exemplo, o www.looke.com.br e o

www.netflix.com/br. Esses permitem definir favoritos, definir dependentes, visualizar informações do filme, entre outras funcionalidades coerentes e necessárias, mas não há a possibilidade de visualizar somente os filmes de comédia e drama ao mesmo tempo. Um sistema deve SEMPRE permitir que os usuários visualizem informações específicas, nem sempre isso é possível e o problema da navegação inflexível é por causa dos desenvolvedores.

RESUMINDO

Hoje, aprendemos a organizar a informação, melhorar a navegação, a importância dos rótulos e como avaliar uma *interface*. Todos esses conceitos não são subjetivos, pois a avaliação nos dirá se o que foi desenvolvido é coerente.

Leituras complementares

Existem diversos *sites* com informações dinâmicas sobre arquitetura da informação. Abaixo, um deles, de um autor mais conhecido é o kalbach, mas há também Dan Saffer, Jennifer Fleming, entre outros. Recomendamos, também, outro tópico muito importante para o desenvolvimento *web*: USABILIDADE! Sobre isso, procurem os livros de Steve Krug e Nielsen.

BLOG DE ARQUITETURA DA INFORMAÇÃO (AI). Disponível em: <http://arquiteturadeinformacao.com>. Acesso em: 16 jul. 2015.

Avaliando seus conhecimentos

Pesquise sobre métodos de avaliação de *interface* e, em seguida, crie um documento de texto com um resumo de cada método. Aplique um método em um *site* e descreva os resultados.

Referências

CSS (3) Tutorial. Disponível em: <http://www.w3schools.com/>. Acesso em: 04 mar. 2015.

FERREIRA, E.; DIEGO, E. **HTML5 e CSS3 com farinha e pimenta**. São Paulo: Clube dos Autores, 2012.

HTML (5) Tutorial. Disponível em: <http://www.w3schools.com/>. Acesso em: 04 mar. 2015.

KALBACH, J. **Design de Navegação Web.** Porto Alegre: Bookman, 2009.

MACEDO, M. S. **Construindo sites adotando padrões web**. Curitiba: Ciência Moderna, 2004.

PEREIRA, R. **União entre estratégia de conteúdo e arquitetura de informação.** 2012. Disponível em: <http://webinsider.com.br/2012/10/17/uniao-entre-estrategia-de-conteudo-e-arquitetura-de-informacao/>. Acesso em: 24 jun. 2015.

INSTITUTO FEDERAL
Rio Grande do Norte
Campus Avançado Natal - Zona Leste