**Aula1 - Introdução ao HTML**

**A história do HTML**

HTML (Hypertext Markup Language) é uma linguagem para publicação na Web baseada no conceito de hipertexto, ou seja, elementos que se conectam entre si formando uma rede de informação. Esses elementos podem ser áudio, vídeo, texto etc.

O hipertexto possibilita a comunicação de dados e a organização de conhecimentos e informações. Nesse contexto, a HTML surge como proposta de linguagem universal, entendida por vários meios de acesso, para distribuição global dessa informação.

Tim Berners-Lee desenvolveu a linguagem HTML. A linguagem se tornou popular na década de 1990, quando desenvolvedores e fabricantes de browsers a transformaram em uma linguagem base. Em 1997, com a versão 3.2 desenvolvida pelo W3C, ela se torna uma linguagem padrão.

Desde sua criação, a HTML tem como uma de suas principais características a interoperabilidade, ou seja, pode ser usada em vários dispositivos, plataformas ou outros meios de acesso.

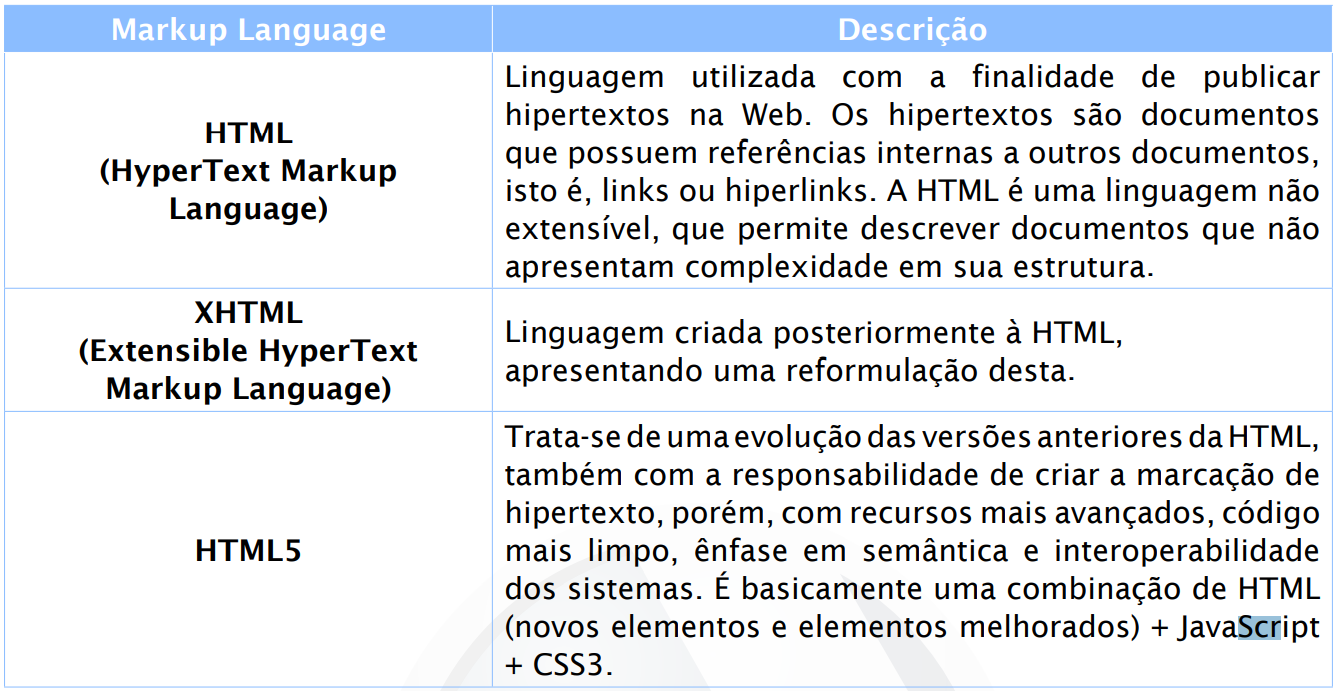
Em 2004, surge o WHATWG (Web Hypertext Application Technology Working Group), grupo formado por especialistas das grandes corporações de TI, que discordava dos caminhos da Web até então e que decide desenvolver uma nova linguagem, uma linguagem mais flexível para Web: HTML5.

Em 2006, houve o reconhecimento do trabalho do grupo. O próprio Tim Berners-Lee se juntou ao WHATWG e, oficialmente, em 2009, foi anunciado o desenvolvimento conjunto da HTML5.

**Markup Languages**

As Markup Languages, também conhecidas como MLs, são linguagens que têm o objetivo de fornecer elementos adicionais a um texto, denominados marcações, a fim de definir uma estrutura ou um layout de informações.

A tabela a seguir descreve algumas Markup Languages cujos códigos são compostos não apenas por conteúdo, mas também por tags, que são marcas utilizadas para enfatizar a estrutura de um documento. Observe:



**HTML5**

Esta linguagem é uma evolução dos padrões HTML que já existiam, ela associa as tags de marcação utilizadas em HTML com novos conceitos semânticos e uma coleção de novos recursos nativos que antes eram possíveis somente com plugins.

Quando um site funciona normalmente em qualquer navegador, chamamos de cross-browser. Quando um site funciona normalmente em qualquer dispositivo, como desktop, notebook, tablet, smartphone, TV, carro, óculos, chamamos de cross-device.

Quando trabalhamos com a linguagem XHTML (Extensible HyperTex Markup Language), contamos com as tags, as quais são definidas como códigos desta linguagem que têm a função de representar um comando. As tags são colocadas entre os sinais de menor e de maior, da seguinte forma: <nome\_tag>.

**Interpretação e transformação do código fonte**

Ao abrirmos o browser, inserirmos uma URL em seu campo Endereço e pressionarmos a tecla ENTER, a resposta obtida é um arquivo de texto simples que pode ser interpretado de várias maneiras, de acordo com a extensão que ele apresenta. Caso a extensão desse arquivo seja .html, o browser aciona seu mecanismo de renderização.

A engine é responsável não apenas pela interpretação do código fonte, mas também por sua transformação em elementos gráficos.

O processo de renderização é ato da transformação do código fonte. Este termo é bastante

utilizado para falar a respeito de métodos que permitem calcular cores, sombras e texturas, seja em um documento baseado em MLs ou em uma imagem vetorial.

**Navegadores**

Existem atualmente centenas de browsers no mercado e qualquer desenvolvedor que tenha

conhecimento da estrutura de renderização e interpretação de códigos HTML pode criar o seu navegador.

Podemos mencionar os principais navegadores existentes no mercado. O que deve ficar claro é que todo navegador de Internet possui em seu núcleo um motor de renderização, que é responsável por determinar como o navegador interpreta a árvore de elementos de um código HTML.

**Motor de Renderização**

Existem alguns motores de renderização que são softwares utilizados no núcleo dos navegadores e clientes de e-mail responsáveis por interpretar a linguagem HTML, folhas de

estilo (CSS), imagens e gerar um código final. A seguir, apresentamos os principais motores de renderização utilizados no mercado.

**WebKit**

É um motor de renderização criado pela Apple, utilizado inicialmente no Safari, conservado

como um projeto de código aberto, é escrito em linguagem C++ e hoje é mantido por um grande grupo de desenvolvedores. É importante mencionar que existem vários tipos de WebKit e o fato de dois navegadores utilizarem WebKit como motor de renderização não necessariamente os torna idênticos em seu resultado final. É utilizado nos seguintes navegadores:

• Safari;

• Google Chrome (além de WebKit, utiliza o Google V8 JavaScript Engine). Em abril de 2013, o Google anuncia que está desenvolvendo seu próprio motor de renderização, denominado Blink e que será adotado em seu navegador Google Chrome;

• Konqueror;

• Opera (Em 2013, o Opera mudou de Presto para WebKit e também irá utilizar o Google V8 JavaScript Engine). Em abril de 2013, o Opera anunciou que também mudará para o Blink, projeto de código aberto criado pelo Google. A ideia é o aumento de desempenho e o multiprocessamento.

**Gecko**

É um motor de renderização criado pela Fundação Mozilla. Inicialmente chamado de NGLayout. Embora seja mantido pela Fundação Mozilla, Gecko é uma marca registrada da Netscape Communications Corporation, empresa pertencente ao grupo AOL. É utilizado nos seguintes navegadores:

• Firefox;

• Mozilla Suite;

• Camino;

• Flock;

• K-Meleon;

• Thunderbird (e-mail).

**Trident**

É um motor de renderização proprietário da Microsoft e embora seja um dos primeiros a implantar recursos de formatação de estilos, acabou por gerar inúmeros problemas de incompatibilidade. Cada um dos navegadores Internet Explorer de versões 6 a 8 interpretava o código à sua maneira, o que tornava o trabalho do profissional responsável pelo layout um tanto quanto maçante, uma vez que um mesmo site tinha várias formas de comportamento para um mesmo mecanismo. Com a versão 6.0 do Trident utilizada no Internet Explorer 10, a tentativa da Microsoft é manter uma maior compatibilidade com o Padrão Web (Web Standard). É utilizado nos seguintes navegadores:

• Internet Explorer;

• Microsoft Outlook (e-mail);

• Visual Studio (IDE).

**Aula 2 - O Universo HTML 5**

**A linguagem HTML5**

A linguagem de marcação HTML5 facilita o trabalho de desenvolvedores, tornando possível a manipulação de elementos e a modificação de características dos objetos de maneira leve

e funcional. HTML5 possui ferramentas para CSS e JavaScript, bem como cria novas tags e

modifica a função de outras. Assim, elementos e atributos que foram modificados podem ser usados de maneira mais eficaz.

Nas versões anteriores do HTML, não existia um padrão para criação de seções comuns e específicas (rodapés, cabeçalhos etc.), como também não havia um padrão de nomenclatura de IDs, classes ou tags.

O HTML5 facilita a maneira como o código é escrito e como as informações são organizadas na página, possibilitando mais interatividade sem a necessidade de utilização de plug-ins e sem perda de performance.

Dois aspectos são relevantes no HTML5:

• **Interoperabilidade**: Facilita a reutilização de informação em novos dispositivos;

• **Retrocompatibilidade**: Compatibiliza sites já existentes com os browsers recentes, ou

seja, não é preciso refazer sites para que estes se adaptem a novos conceitos e regras.

**Tecnologias da HTML5**

As tecnologias do HTML5 são divididas oficialmente em oito áreas, descritas a seguir.

**Semântica**

O foco principal da HTML5 é dar sentido ao conteúdo de um site ou de um Web App. Com tecnologias como Microdados e Microformatos, a HTML5 permite que sejam identificados nomes, locais, preferências, empresas em um texto e que essas informações sejam interligadas. Para isso, faz uso de uma rica coleção de tags, dando significado semântico a textos que antes eram apenas conjuntos de palavras sem sentido para as máquinas.

Além de tags semânticas para o conteúdo do texto, a parte de formulários ganhou novas entradas de dados que estão melhores associadas ao seu real objetivo. Há tipos de entrada de dados específicos para e-mail, números, calendários, cores e muito mais.

**Offline e armazenamento**

Graças a tecnologias como HTML5 App Cache, Local Storage, Indexed DB e as novas especificações de API de arquivos, os Web Apps e sites podem ser mais rápidos e continuar trabalhando mesmo se não houver conexão com a Internet.

**Acesso ao dispositivo**

Com a utilização da API de Geolocalização, as aplicações Web podem acessar recursos de dispositivos e melhorar a experiência do usuário: desde áudio/vídeo até microfones e câmeras.

**Conectividade**

Com mais eficiência em conectividade, que é necessária nas aplicações em tempo real, como chat, temos, também, velocidade maior para jogos e melhor comunicação. O envio de dados entre cliente e servidor também se torna muito mais rápido e eficiente com a utilização de Web Sockets e Server-Sent Events.

Em Ajax, temos uma requisição cliente/servidor de forma assíncrona. A novidade com Server-Sent Events é que podemos inverter esse processo: o servidor é que envia eventos para o cliente. Isso é ideal para jogos com vários jogadores simultâneos.

AJAX (Asynchronous JavaScript and XML) é uma abordagem de desenvolvimento web que permite que as páginas da web atualizem partes específicas do conteúdo de forma assíncrona, ou seja, sem a necessidade de recarregar toda a página.

Em resumo, o AJAX é uma técnica que permite atualizações assíncronas de conteúdo em páginas da web, melhorando a interatividade e a responsividade do usuário ao permitir que partes específicas da página sejam atualizadas sem recarregar a página inteira.

No contexto de desenvolvimento web e programação, "assíncrono" refere-se a operações ou processos que ocorrem de forma independente, sem um padrão rígido de sincronização. Em outras palavras, as operações assíncronas não precisam esperar uma pela outra para começar ou terminar.

Quando se fala em programação assíncrona, isso geralmente significa que uma parte do código pode continuar executando enquanto outra parte está aguardando algum tipo de resposta ou evento. Isso é particularmente útil em situações em que há operações que podem levar algum tempo para serem concluídas, como chamadas de rede, acesso a bancos de dados ou animações complexas.

**Latência** refere-se ao atraso ou tempo que leva para um pacote de dados viajar de um ponto a outro em uma rede. É o tempo que decorre desde o momento em que um dado é enviado até o momento em que ele é recebido no destino.

A latência é medida em milissegundos (ms) e é um fator importante que afeta a velocidade de comunicação e a resposta em sistemas de rede.

**Multimídia**

Áudio e vídeo são os principais recursos de multimídia para integração com suas aplicações Web e sites.

**Gráficos, 3D e efeitos**

Com a utilização de SVG, Canvas, WebGL e recursos CSS3 3D, é possível produzir experiências visuais incríveis para o usuário, sem a necessidade de plug-ins, ou seja, nativamente no navegador.

**Desempenho e integração**

Utilizando tecnologias como Web Workers e XMLHttpRequest 2, é possível criar suas aplicações Web e conteúdo dinâmico (Ajax) de forma mais rápida, além da grande variedade de técnicas que permite o aumento de desempenho.

**CSS3**

CSS3 é uma das maiores revoluções e, embora seja independente da HTML5, é entendida como uma de suas variantes em termos de novas tecnologias de padrão aberto para estilização e efeitos de um documento formatado. Com ela, você cria suas aplicações Web sem sacrificar a estrutura semântica ou o desempenho. São disponibilizadas novas formas de estilização de fontes, de alinhamento de layout, de seletores avançados, animação e transformação.

**Aula 3 - Conhecendo a estrutura HTML 5**

**Tags e atributos do HTML5**

Tag é o termo atribuído aos códigos utilizados em HTML. As tags são identificadas por estarem entre o sinal menor que (<) e maior que (>), conforme demonstram os seguintes exemplos: <html></html>; <body></body>.

A estrutura de um documento HTML é formada basicamente por tags, as quais podem ou não possuir atributos. É preciso ter em mente que as tags e os atributos que compõem um documento HTML5 devem estar de acordo com algumas regras, a fim de que sua estrutura esteja adequada.

Dentre essas regras, destacamos a necessidade de que um elemento vazio, assim como os

outros elementos, tenha uma tag de abertura e outra de fechamento, por exemplo, <title></

title>. Além disso, os valores referentes aos atributos devem ser colocados entre aspas duplas, até mesmo os valores que são compostos por números.

As tags são hierárquicas, logo, dentro de um par de tags podemos ter outras tags que atuam como elementos filhos das tags principais. É o que ocorre, por exemplo, com a tag <title></title> que é filha da tag <head></head>, que por sua vez é filha do elemento ou tag raiz <html></html>.

<html>

<head>

<title>HTML5 Fundamentos</title>

</head>

<body>

</body>

</html>

**Estrutura do código HTML5**

A seguir, temos um modelo que demonstra a estrutura correta de um código HTML5.



Podemos verificar que este exemplo demonstra a estrutura correta de um código HTML, pois todas as regras mencionadas anteriormente foram obedecidas: o valor do atributo foi colocado entre aspas, todas as tags foram declaradas em letra minúscula (caixa baixa) e todas as tags abertas também foram fechadas.

**Tipo de documento (DOCTYPE)**

DOCTYPE é um tipo de declaração que deve obrigatoriamente constar em um código HTML, sendo incluída antes do elemento raiz deste código. Determina ao navegador qual o tipo de documento ele deverá interpretar.

**Elemento raiz (html)**

O elemento raiz de um código HTML deve ser a tag<html>. Tal elemento pode, ainda, possuir um atributo lang responsável por determinar qual o idioma do documento em questão, e ainda permitir que os navegadores sugiram a melhor tradução quando algum usuário de um país cujo idioma seja diferente do documento acessar o site. Para o português do Brasil, utilizamos lang=”pt-br”.

O atributo lang pode ser utilizado também dentro de textos no documento com a intenção de determinar o idioma que está sendo utilizado em um determinado local. Por exemplo: <span lang=”es”> Bienvenido a HTML5</ span>.

**Cabeçalho (head)**

O cabeçalho, representado pelo elemento <head>, contém informações genéricas a respeito do conteúdo do documento e da forma que será exibido. Essas informações genéricas também são chamadas de metadados, os quais representam informações capazes de descrever outros dados. Portanto, podemos dizer que a função da tag **<head>** é determinar o cabeçalho de uma página.

Todos os elementos que compõem esta tag não são exibidos pelo browser. Dentre esses elementos, podemos ter metatags, folhas de estilo, scripts, entre outros. Dentro da tag **<head>**, deve ser inserida a tag **<title>**, cuja função é determinar o título da página.

Se desejarmos carregar estilos para formatação de conteúdo na própria página, utilizamos o elemento ou tag **<style></style>**. Se os estilos forem carregados externamente, utilizamos o elemento **<link>.**

Quando precisamos adicionar programação de script ao documento, de forma que essa seja carregada antes do corpo do site, podemos adicionar o elemento **<script></script>** como elementos filhos de <head></head>.

**Corpo da página (body)**

O corpo de uma página HTML5 é determinado pela tag <body>, cujos elementos têm a função de determinar tudo o que será apresentado pelo browser de forma gráfica. Observe o exemplo a seguir:

*<body>*

*Conhecendo a estrutura do código HTML5*

*</body>*

**Metatags**

Definimos metatags como sendo códigos referentes a informações e a indicações. Assim como as folhas de estilo e os scripts, as metatags representam um dos elementos a serem colocados no cabeçalho da página (tag <head>).

O elemento utilizado para realizar a declaração de uma metatag é o <meta />, cuja função é

fornecer dados capazes de descrever informações presentes no corpo da página. Normalmente, essas informações referem-se à descrição do conteúdo e a palavras-chave.

**Meta name**

O elemento <meta name /> costuma referir-se às informações relacionadas ao autor da página, conforme demonstra o exemplo a seguir:

*<meta name=”author” content=”autor\_da\_página” />*

Podemos observar que “author” é um atributo que permite determinar o nome do desenvolvedor da página.

**Palavras-chave e descrições**

Para compreender a importância das palavras-chave e das descrições, é preciso ter em mente que as buscas dinâmicas realizadas por meio dos motores de busca são auxiliadas, entre outras regras, por metatags.

A fim de realizar uma pesquisa, os motores de busca utilizam regras que verificam os cabeçalhos e o conteúdo das páginas na rede, para encontrar as palavras-chave que estão presentes nas metatags e que são representadas pelo atributo keywords. Esses aplicativos são chamados de robots.

Assim que a palavra-chave é encontrada, todas as descrições contendo o atributo description que foram encontradas pelas metatags são exibidas em uma página de resultado. Embora as metatags possam conter várias palavras-chave, as quais devem ser separadas por vírgulas, a descrição requer mais objetividade, visto que um texto de descrição bastante extensa pode não ser visualizado por inteiro, uma vez que o motor de busca limita a quantidade de caracteres a ser exibida:

*<meta name=”keywords” content=”HTML5, CSS3, Semântica, Canvas” />*

*<meta name=”description” content=”Nossa empresa esta situada no melhor...” />*

Os mecanismos que encontram as palavras-chave (keywords) apresentam a descrição do site, a qual está presente no atributo description. Quando trabalhamos com a Internet, temos não apenas as ferramentas de busca, mas também os catálogos, que são sites destinados à realização de pesquisas e cujo banco de dados pode ser alimentado, desde que seja feito um cadastramento para isso.

**Meta charset**

Quando precisamos tornar nosso site disponível para os principais idiomas, precisamos determinar o encoding que o site utilizará. Existe um encoding que permite utilizar cerca de

um milhão de caracteres: chama-se utf-8. Para determinar o encoding utilizado no documento, temos o elemento meta com o atributo charset <meta charset/>:

*<meta charset=”utf-8”>*

Exemplo de Código:

~~~~html

<!-- <!DOCTYPE html> Declaração que define o tipo de documento como HTML5.-->

<!DOCTYPE html>

<!--<html> A tag raiz que envolve todo o conteúdo HTML-->

<!--Atributo lang responsável por determinar qual o idioma do documento-->

<html lang="pt-br">

<!-- <head> A seção onde informações do documento são definidas, como

título, metadados e links para arquivos externos (CSS, JavaScript, etc.)-->

<head>

<!--<title> Define o título da página, que é exibido

na guia do navegador ou na barra de título.-->

<title>Aula 03 - HTML 5</title>

<!-- <meta> Usado para definir metadados sobre o documento,

como codificação de caracteres, autor, descrição, etc.-->

<meta charset="utf-8" />

<meta name="author" content="Alex Sousa" />

<meta name="keywords" content="tags,css,html5" />

<meta name="description" content="Aulas de HTML5 com CSS" />

</head>

<!--<body> A seção onde o conteúdo visível

da página é colocado, como texto, imagens, links, etc.-->

<body style="font-family: Calibri;">

<!--<h1> Tags de cabeçalho, usadas para criar títulos

com diferentes níveis de importância.-->

<h1>Aulas de HTML5 CSS</h1>

<!--<hr /> Linha horizontal.-->

<hr />

<!-- <p> Usada para parágrafos de texto.-->

<p style=" font-family: Calibri; font-size: 16px;">Aqui eu escrevo o texto da minha página web.</p>

<!-- <button> Botões-->

<div align="center" style=background-color:black><button

style="color: black; font-size: 16px;font: bolder; font-family: Calibri; background-color: darkgrey;height: 50px; width: 100px;">Botão</button>

</div>

</body>

</html>

~~~~

**Aula 4 - Formatando um Documento HTML 5**

**Entendendo a semântica de um documento**

Quando criamos um documento HTML5, seja ele um site ou um aplicativo Web (Web App),

precisamos conhecer a estrutura dos elementos principais de um documento HTML5 bem

formatado.

Em sua nova versão, o HTML permite utilizarmos elementos que possuem significado tanto para os desenvolvedores, quanto para os motores de busca. Isso é chamado de semântica.

menu de navegação, é possível utilizar o elemento **<nav>** criado especificamente como uma seção que contém um menu de navegação:



**Elementos de um documento HTML5**

Em um documento HTML5, todos os elementos começam com a definição do tipo do documento. Isso é possível por meio da tag ***<!doctype HTML>***, seguida pela abertura da tag ***<html>***.

Após essa definição, temos a criação do cabeçalho do documento por meio da tag ***<head>***, que irá definir o que deve ser carregado antes da página ser exibida ao usuário.

Definido o cabeçalho e seu título, o corpo do documento é criado por meio da tag ***<body>***, formando a estrutura básica a seguir:



Dentro da tag **body**, colocamos o conteúdo de nosso site, que será dividido por tags que

representam seu significado.

Veja, a seguir, uma lista com os principais elementos do HTML5 e seus respectivos significados:

**• html**

Este é o elemento raiz do documento HTML, ficando logo no início. Dentro dele, ficam todos

os elementos filhos.

**• head**

Este é o primeiro elemento filho após o elemento raiz; é o cabeçalho do documento, que deve ser carregado antes da página ser exibida ao usuário. Dentro dele, utilizamos a tag **<meta charset=”utf-8” />**.

**• link**

Elemento responsável por criar uma referência para um arquivo externo, possui um atributo **rel** (relation) para explicar o motivo dessa referência. O valor mais comum para **rel** é Stylesheet, referindo-se a um arquivo externo que irá dar forma e organização ao documento atual, formatando cor, fontes, formas e todo o documento.

**• body**

O início do corpo do documento.

**• header**

O cabeçalho do documento. Quando o elemento header for filho de outro elemento dentro

de um documento, por exemplo, dentro de uma section, ele será o cabeçalho deste bloco

específico. Logo, em um documento podemos ter vários blocos section e cada um ter seu

header independente.

**• main**

O conteúdo principal da página é alocado dentro da tag main. É recomendável utilizar o atributo **role=”main”**, uma vez que nem todos os navegadores implementaram esse novo

elemento. Informações não essenciais podem ser deixadas de fora.

**• section**

Uma seção com conteúdos diversos no documento, por exemplo, a introdução ou o título. Um layout pode ser dividido em várias sections, desde que possuam conteúdo que reflitam algum sentido semântico.

**• nav**

Uma seção contendo um menu de navegação, formado por vínculos que ligam um documento ou página a outra página.

**• article**

Quando é necessário armazenar o texto principal do documento, por exemplo, um artigo de

uma revista, jornal, ou blog, ele deve ficar dentro do elemento article, que pode possuir um cabeçalho com o título principal e subtítulo. É recomendável a criação de apenas um elemento article por documento.

**• aside**

Uma seção que contém conteúdo relacionado aos elementos em sua volta. É comumente

utilizada para barras laterais, tais como publicidade, podendo conter outros elementos filhos

como nav, representando um menu de opções.

**• h1- h6 e hgroup**

Estes elementos permitem criar o título principal do documento utilizando h1 e os subtítulos

hierárquicos de h2 até h6. Quando necessário, é possível agrupar esses títulos dentro de um grupo representado pelo elemento hgroup.

**• footer**

O rodapé do documento ou de uma section pode conter outros elementos filhos, tais como

nav, tornando-o assim um menu de navegação no rodapé.

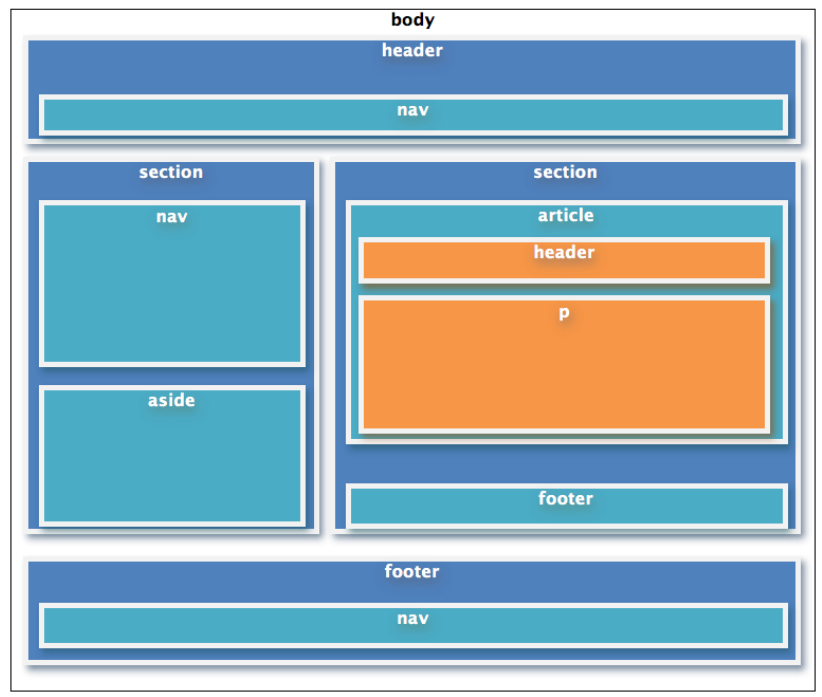
**• p**

Elemento utilizado para parágrafo.

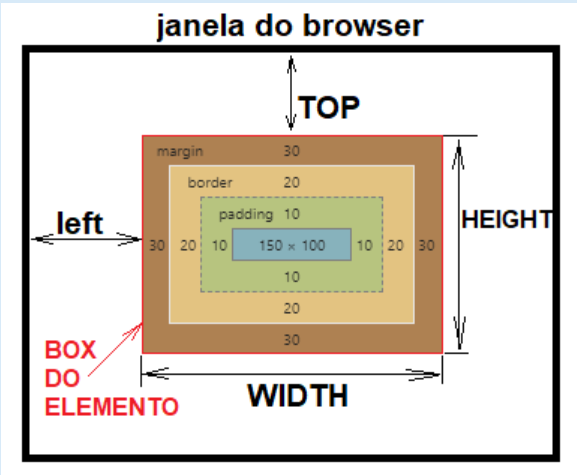
**• div**

Uma seção do site que não possui significado semântico.

Na imagem a seguir, temos a estrutura de um documento, simples e resumida, para mostrar como são estruturados os elementos anteriormente citados. Essa imagem resumida é criada quando o design de interface está sendo desenvolvido e a esse recurso damos o nome de wireframe. Observe:



**Posição dos Elementos**



Para criar esta estrutura, o procedimento é o seguinte:

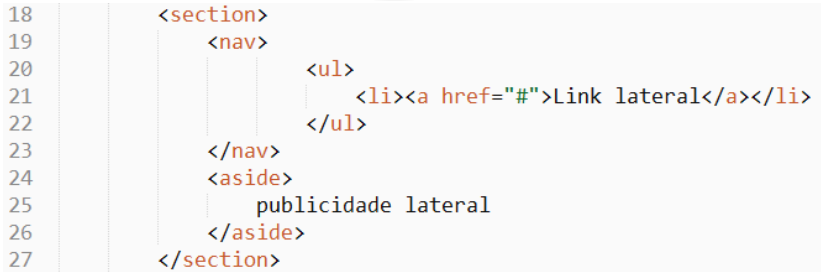


O primeiro passo é criar o doctype, o elemento raiz, o cabeçalho do documento, a codificação para acentuação, o título do site, finalizar o cabeçalho e iniciar o corpo do documento:

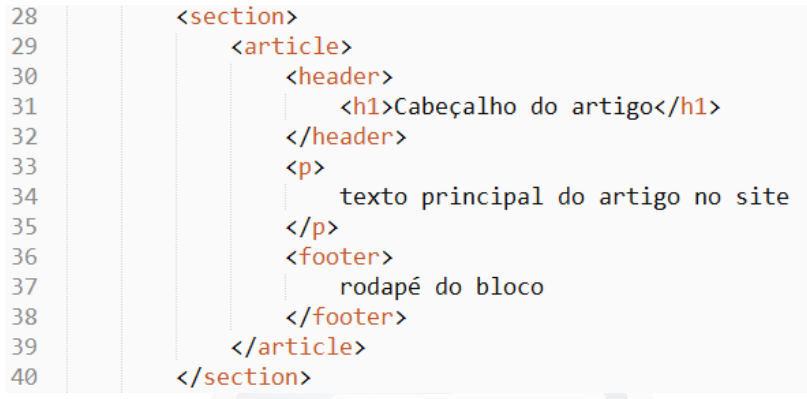


Agora, criamos o cabeçalho do documento, com o elemento <header>. Dentro do dele, colocamos o elemento de título principal <h1>, também o elemento de navegação <nav>,

com uma lista <ul>:



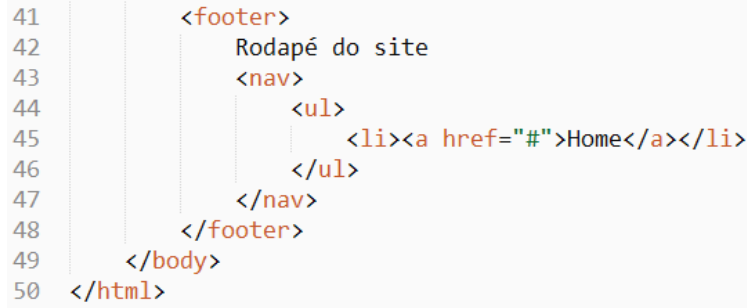
Uma seção para os itens laterais é criada, com um menu de navegação utilizando o elemento <nav>. Dentro desta seção, é criada uma lista com o elemento <ul> e um elemento <aside> com informações relacionadas ao conteúdo é adicionado dentro da section; este elemento pode representar, por exemplo, uma publicidade:



Um elemento <section> é adicionado para o conteúdo principal e, dentro deste elemento, criamos um elemento <article> para o texto principal desta página, com um cabeçalho, um

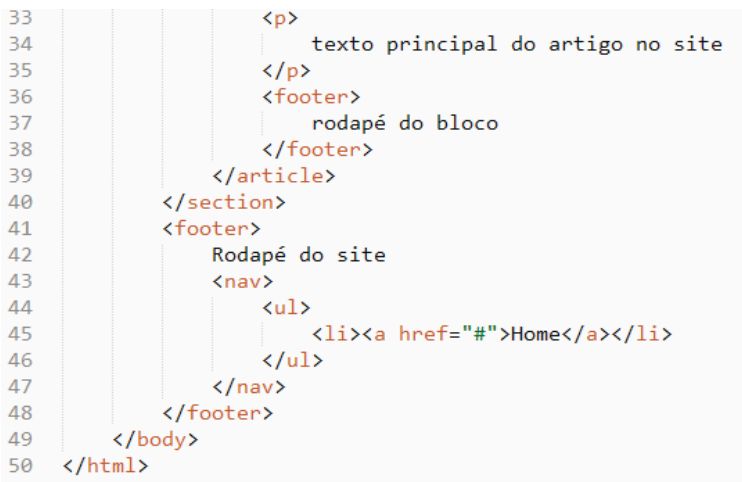
parágrafo e um rodapé.

Quando um elemento <article> é criado, os elementos filhos devem fazer referência direta ao texto. Por exemplo, dentro do elemento <article>, temos um <header> com um <h1> para o título do texto principal:



Na parte final do documento, é criado um elemento <footer>, que determina o rodapé do documento, com uma frase que pode compreender os direitos autorais e um menu de navegação com o elemento <nav>, que compreende alguns links importantes do documento:





Com o documento completo, a importância dos elementos semânticos do HTML5 é reforçada, cada parte da página possui um elemento que dá sentido ao conteúdo e organiza as informações tanto para o usuário quanto para os mecanismos de busca.