

Web Academy

Fundamentos de Programação Front-end



Manoel Limeira de Lima Júnior



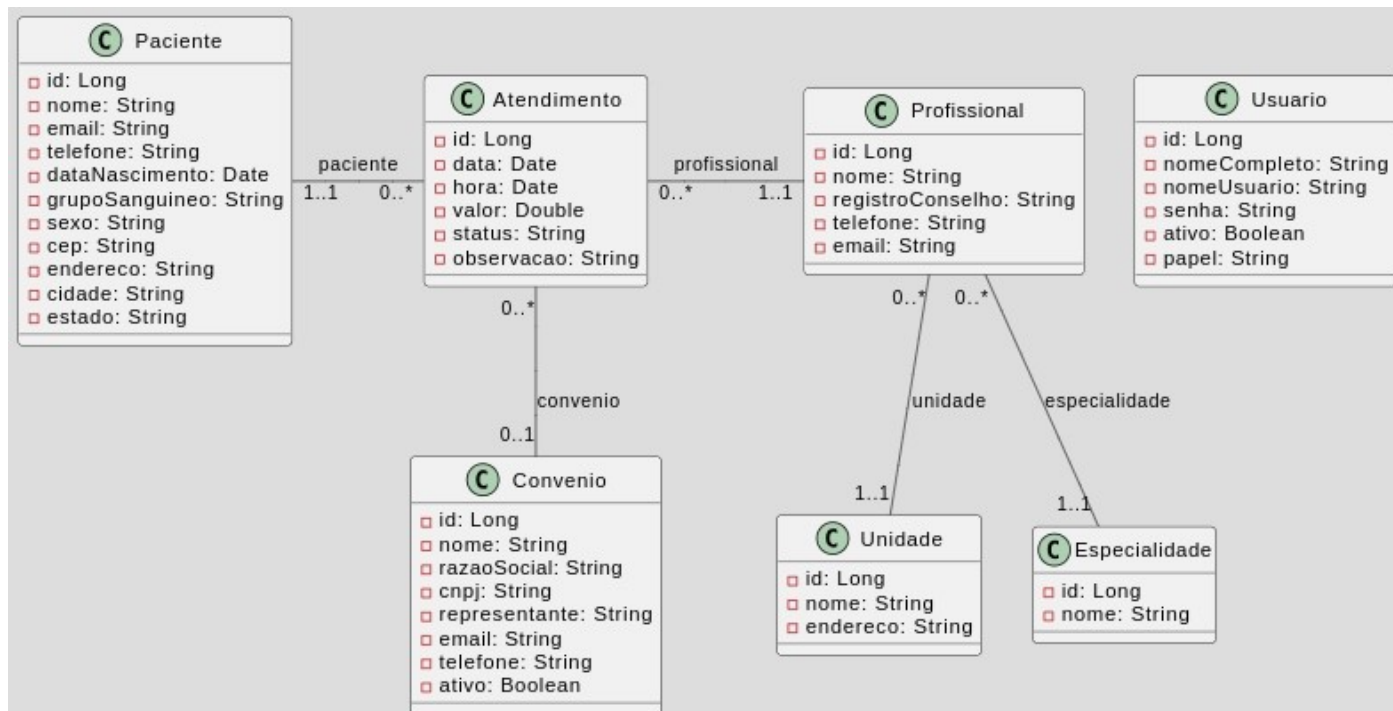
Web Academy



Apresentação

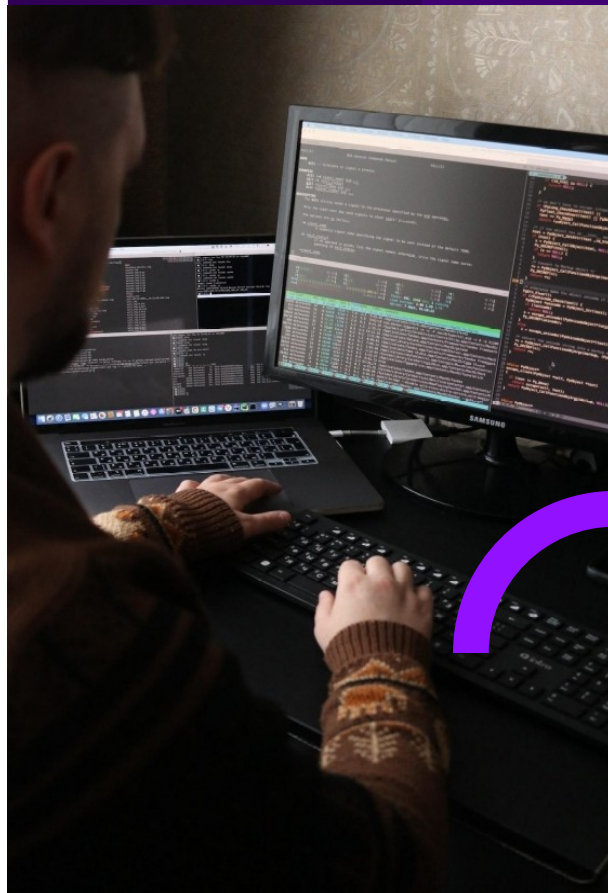
SGCM – Sistema de Gerenciamento de Consultas Médicas

- Documentação: <https://github.com/webacademyufac/sgcmdocs>
 - Diagrama de classes



Ementa

1. Fundamentos de **HMTL**.
2. Padrões e recomendações da **W3C**, semântica e acessibilidade.
3. Estilização de páginas HTML com **CSS**.
4. Técnicas de design responsivo.
5. Tipos de dados, funções, objetos, arrays e manipulação de eventos em **JavaScript**.
6. Manipulação de **DOM** (Document Object Model).
7. **JSON** (JavaScript Object Notation).
8. Requisições assíncronas.



Objetivos

- **Geral**

- Capacitar o aluno na utilização de **procedimentos e técnicas básicas** de desenvolvimento de aplicações para a WEB, com **ênfase nos fundamentos** de tecnologias voltadas ao desenvolvimento **front-end**.

- **Específicos:**

- Apresentar os principais conceitos de linguagens, protocolos e ferramentas que dão suporte ao funcionamento da Web;
- Compreender a importância dos padrões Web na produção de códigos válidos, semanticamente corretos e acessíveis;
- Capacitar o aluno no emprego correto dos recursos disponíveis nas tecnologias HTML, CSS e JavaScript, para construção de aplicações Web, separando conteúdo, apresentação e interatividade.

Conteúdo programático

Introdução

O lado cliente (front-end) e o lado servidor (back-end);
O protocolo HTTP, HTML e a Web;
Evolução do HTML;
Tecnologias de front-end; Padrões web, acessibilidade e design responsivo.

HTML

Introdução ao HTML;
Estrutura de um documento HTML;
Principais elementos (tags).

CSS

Introdução ao CSS;
Bordas e margens (box model); Sintaxe e seletores;
Herança; Aplicação de CSS: cores, medidas, textos e layout.

JavaScript

Introdução ao JavaScript; Sintaxe; Principais tipos de dados; Objetos e Arrays; Formas de utilização; Eventos; DOM; JSON; Requisições assíncronas (AJAX).

Bibliografia



Web

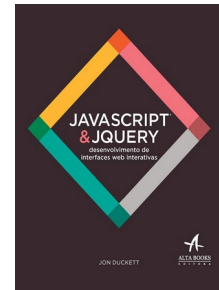


HTML e CSS: projete e construa websites.

Jon Duckett

1a Edição – 2016

Editora Alta Books



JavaScript e JQuery: desenvolvimento de interfaces web interativas.

Jon Duckett

1a Edição – 2016

Editora Alta Books

Academy

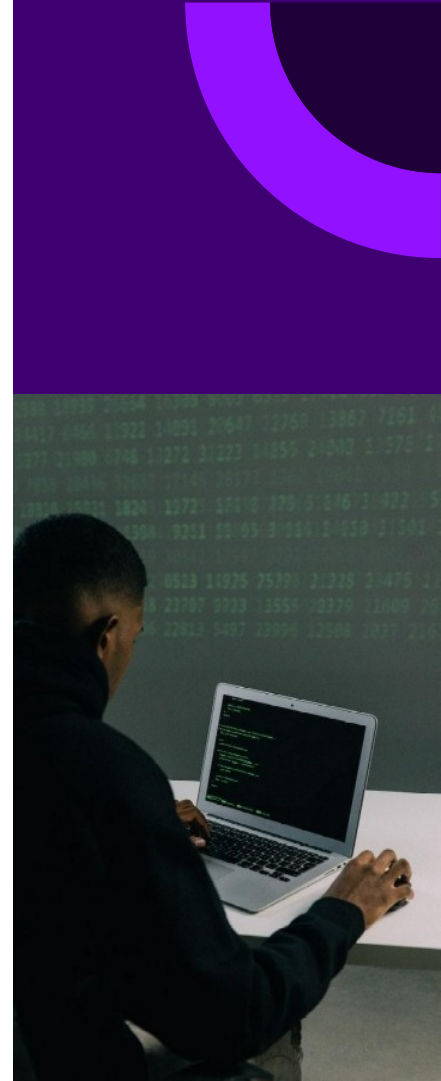
Sites de referência

- **MDN Web Docs: Aprendendo desenvolvimento web.**
 - <https://developer.mozilla.org/pt-BR/docs/Learn>
- **W3Schools Online Web Tutorials.**
 - <https://www.w3schools.com/>
- **W3C Standards.**
 - <https://www.w3.org/standards/>



Ferramentas

- **Visual Studio Code**
 - <https://code.visualstudio.com/Download>
- **Live Server (Extensão do VS Code)**
 - <https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>
- **Git**
 - <https://git-scm.com/downloads>
- **Chrome Developer Tools (F12)**





Web Academy



Introdução

O protocolo HTTP, HTML e a Web (WWW)

- Nos anos **1980**, **Tim Berners-Lee**, Físico do CERN, trabalhava no projeto Enquire, que tinha como um dos objetivos criar o que ficou conhecido como **hipertexto**.
- O Hipertexto relaciona textos, imagens, sons, vídeos e qualquer tipo de conteúdo multimídia.
- Tim também criou o **HTTP** (*HyperText Transfer Protocol*), os **URLs** (*Uniform Resource Locators*), que são a base da web até hoje, o primeiro navegador **Web** (*WorldWideWeb*, mais tarde renomeado Nexus) e o primeiro servidor web.



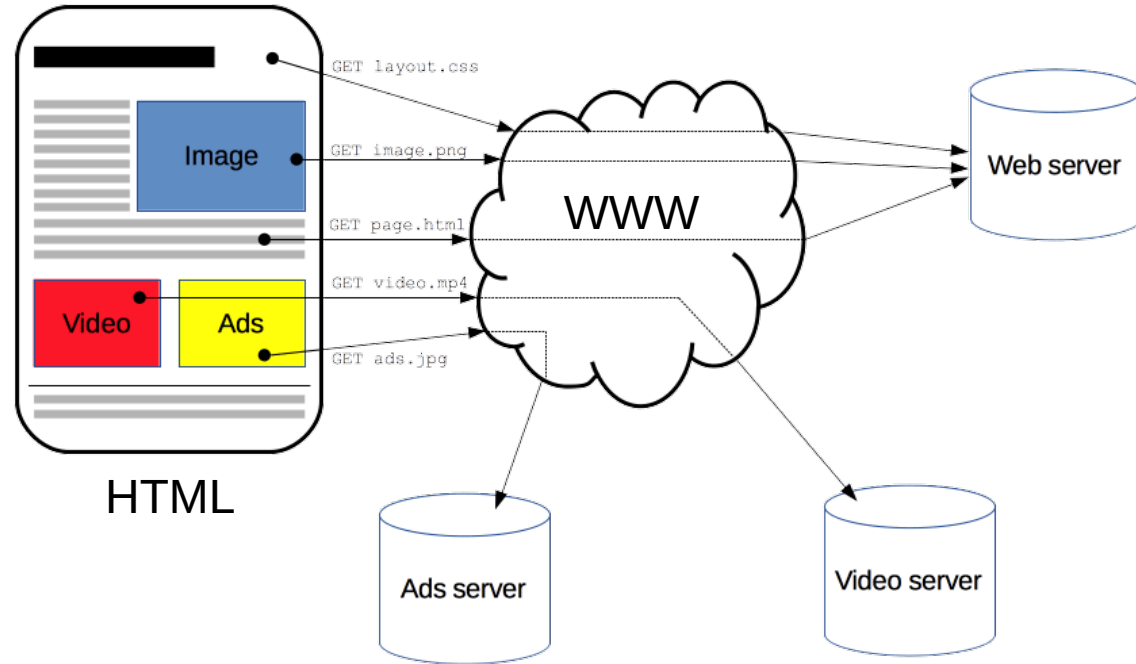


O protocolo HTTP, HTML e a Web (WWW)

- Com base no **TCP/IP** (*Transmission Control Protocol/Internet Protocol*) surgiu a ideia de transmitir o conteúdo hipertexto pela rede.
- Para isso foi criado o protocolo **HTTP** (*Hypertext Transfer Protocol*).
- Uma linguagem para criação de conteúdo hipertexto foi criada, o **HTML** (*HyperText Markup Language*).
- E além disso foi criado o conceito **WWW** (*World Wide Web*) que engloba todos os serviços de conteúdo multimídia baseados no protocolo HTTP.

O protocolo HTTP, HTML e a Web (WWW)

- HTTP é um **protocolo cliente-servidor** que permite a obtenção de recursos, como documentos HTML.
- Clientes e servidores se comunicam trocando mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de requisições ou (**requests**) e as mensagens enviadas pelo servidor são chamadas de respostas (**responses**).



O protocolo HTTP, HTML e a Web (WWW)

- Primeiro site criado com a linguagem HTML pra funcionar sob o protocolo HTTP:
 - <http://info.cern.ch/>
- O número exato de sites muda a cada segundo, estima-se 1,2 bilhão de sites na internet em fevereiro de 2025, cerca de 17% estão ativos
 - <https://siteefy.com/how-many-websites-are-there/>

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#), [Policy](#), November's [W3 news](#), [Frequently Asked Questions](#).

[What's out there?](#)

Pointers to the world's online information, [subjects](#), [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#), X11 [Viola](#), [NeXTStep](#), [Servers](#), [Tools](#), [Mail robot](#), [Library](#).)

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#), etc.

Evolução da linguagem HTML

Ano	Versão	Evolução Histórica
1991	HTML 1.0	Primeira versão publicada no CERN por Tim Berners-Lee.
1995	HTML 2.0	Desenvolvimento (formulários, imagens embutidas e tabelas) se deu em colaboração com várias empresas e fabricantes de navegadores.
1997	HTML 3.2	Sob responsabilidade do W3C, trazendo padronização e incluindo suporte para CSS e applets de Java.
1999	HTML 4.01	A HTML 4.01 publicada pelo W3C foi uma revisão da HTML 3.2, incluindo novos recursos como suporte para frames, scripts do lado do cliente (JavaScript) e melhorias na acessibilidade.
2000	XHTML	O W3C criou a linguagem XHTML versão 1.0 (baseada na tecnologia XML) a partir da HTML versão 4 e propôs acabar com a linguagem HTML.
2004		Discutiu-se a evolução HTML 5, proposta apresentada pela Mozilla e Opera, e foi rejeitada pela W3C que havia optado por evoluir a linguagem XHTML.

Evolução da linguagem HTML (cont.)

Ano	Versão	Evolução Histórica
2007	WHATWG XHTML 2.0	Contrários a decisão do W3C, Firefox, Opera e Safari criaram o WHATWG. A W3C ainda lançou, em 2007, o XHTML 2.0, mas o WHATWG já tinha uma proposta de nova versão do HTML e o W3C aceitou a proposta.
2014	HTML 5	Inclusão de novos elementos e atributos de mídia e formulário, semântica aprimorada e APIs JavaScript para interação com a web moderna. Projetada para funcionar em vários dispositivos.
2016	HTML 5.1	Novos recursos e refinamentos, como o elemento <picture> para seleção de imagem responsiva e o elemento <dialog> para criar caixas de diálogo modais.
2017	HTML 5.2	Novos recursos como o elemento <main> para identificar o conteúdo principal de uma página e o elemento <details> para criar caixas de detalhes que podem ser expandidas ou recolhidas pelo usuário
2020	HTML 5.3	Novos elementos e recursos, como o elemento <slot> para ajudar na construção de componentes da web reutilizáveis

História e evolução da Web

- **Web 1.0 (1991 - 2004)**

- Sites estáticos, conteúdo fixo, HTML básico (*hiperlinks*), design semelhante ao de páginas impressas. Vídeos eram raros. Os internautas apenas consumiam o que estava disponível. Exceções: envios de e-mail e formulários de cadastro. Exemplos: sites de universidades, órgãos governamentais e empresas.

- **Web 2.0 (2004 - ...)**

- Sites dinâmicos, interação do usuário, a produção de conteúdo ganhou espaço possibilitando a inserção em páginas de blogs e redes sociais. Popularização da tecnologia AJAX – *Asynchronous Javascript and XML* (2005) e o uso dos estilos em CSS – *Cascading Style Sheet*. Exemplos: Wikipedia, YouTube, Facebook.

- **Web 3.0 (2014 - ...)**

- Automação, Web semântica, maior personalização e descentralização por meio de *Blockchain* e algoritmos para melhorar a eficiência de atividades com o uso de *Machine Learning*.

- **Web 4.0 (atualmente)**

- Cuidados com políticas de privacidade, Internet das coisas (IoT), forte tendência na integração de serviços com uso de Inteligência Artificial

Fonte: <https://br.hubspot.com/blog/marketing/evolucao-web>

Evolução da linguagem HTML

- O projeto do **HTML 5** com o apoio do W3C teve início em 2008.
- A nova versão trazia pela primeira vez a separação total entre **semântica**, **estilo** e **interatividade**.



Tecnologias relacionadas a sistemas web

- A estrutura de uma página web é baseada atualmente em 3 tecnologias principais. Além do **HTML**, são elas:
 - **CSS** (*Cascading Style Sheets*): linguagem que define o layout de documentos HTML;
 - **JavaScript**: linguagem de programação que roda no lado cliente (navegador).



Wayback Machine

- Banco de dados digital mantido pelo Internet Archive com bilhões de páginas de internet;
- Permite visualizar versões antigas de páginas web;
- <https://archive.org/web/>



Wayback Machine

- Globo.com em 2000



Wayback Machine

- UFAC em 2000

The screenshot displays the homepage of the Universidade Federal do Acre (UFAC) in 2000. The layout is organized into several sections:

- Header:** Features the UFAC crest on the left and a banner with the text "Por uma Universidade Verdaderamente Amazônica" and "Universidade Federal do Acre" in the center. The banner includes two small images of campus buildings.
- Left Sidebar (Dark Blue):** Contains a vertical list of links: "A Universidade", "Reitoria", "Pró-Reitorias", "Departamentos/ Cursos", "Biblioteca", "Colégio de Aplicação", "Rede UFACNet", and "Informações".
- Main Content Area:**
 - Editais & Concursos (Yellow):** Includes a "NOVO!" alert and a notice about a new list of books for the 2000/2001 vestibular, with a link to "Confira aqui!".
 - Links Úteis (Green):** Lists "Página Inicial dos Links", "Órgãos do Governo", "Universidades Brasileiras", and "Busca na Web".
 - Novidades (Orange):** Announces "Mestrado e Doutorado NOVO!" and provides details about the selection process for the 2000/2001 academic year, including a link to "Confira!!".
 - Downloads (Pink):** Features a logo for "cadastro de pesquisadores" and a link to the "Diretório do Grupo de Pesquisas no Brasil - versão 4.0 /CNPq".
- Right Sidebar (Blue):** Titled "Veja Mais >>>", it contains links to "Calendário Acadêmico", "Email's Úteis", "Projetos na UFAC", "Fotos da UFAC", and "Eventos Nacionais".
- Footer:** Includes the "PRODOC" logo (Produção de Docentes), the "POP-AC" logo, the "Bne" logo (Banco Nacional de Empresas) with the website "WWW.BNE.COM.BR", the "Fundação BIOMA" logo, the "Linha Direta On line" logo, and the "Educação Santa Margarida" logo. Contact information for UFAC is provided, including the address "BR 364 Km 04 Cep: 69915-900 Bairro Distrito Industrial, Caixa Postal 500 - Rio Branco - Acre", phone numbers "PABX: (0xx68) 229-2244" and "Design by webmaster@ufac.br", and a copyright notice "(c) 2000 - Todos os direitos reservados."

O que são os padrões web?

- Os **padrões web** (*web standards*) são amplamente discutidos e empregados por desenvolvedores e pessoas envolvidas com o desenvolvimento de aplicações para web.
- São **recomendações** (e não normas!) destinadas a orientar os desenvolvedores para o uso de boas práticas de construção de páginas web que tornam o conteúdo acessível para todos.
- São essenciais para a construção de uma **web aberta, acessível e interoperável**, garantindo que todos possam desfrutar de uma experiência de qualidade.

O que são os padrões web?

- Apesar de existirem **órgãos normatizadores**, como o **ISO** (International Organization for Standardization) e **ECMA** (European Computer Manufacturers Association), normalmente quando discutimos padrões web nos referimos aos padrões do **W3C** (World Wide Web Consortium).
- Uma **recomendação** do W3C é uma especificação ou um conjunto de diretrizes que passou por discussão e foi estabelecido um consenso, passando a ser indicado seu amplo emprego.

Padrões Web

- O trabalho do W3C é abrangente e alcança diversas tecnologias.
- Essa abrangência pode ser agrupada em três segmentos:
 - Código válido;
 - Código semanticamente correto;
 - Separação entre **conteúdo** (HTML), **apresentação** (CSS) e **interatividade** (JavaScript).



Benefícios na adoção de padrões web

- Melhor indexação pelos mecanismos de busca;
- Renderização mais rápida;
- Garantia de funcionamento completo da página;
- Páginas com melhor aspecto de apresentação;
- Comportamento uniforme entre diferentes navegadores de internet.

Acessibilidade na Web

- **Acessibilidade** significa permitir que o maior número de pessoas possam usar a web, independente da sua limitação.
- **Restrições no acesso a web** é um problema que afeta muitas pessoas que possuem algum tipo de necessidade especial.
- Ainda existem muitas páginas com barreiras de acessibilidade que dificultam ou mesmo tornam impossível o acesso.



Acessibilidade na Web: exemplos de barreiras

- Imagens que não possuem texto alternativo.
- Formulários que não podem ser navegados em uma sequência lógica ou que não estão rotulados.
- Páginas com tamanhos de fontes absoluta, que não podem ser aumentadas ou reduzidas facilmente.
- Páginas que, devido ao layout inconsistente, são difíceis de navegar quando ampliadas por causa da perda do conteúdo adjacente.



Padrões web e acessibilidade

- Os padrões web representam o básico para uma página web acessível.
- É também importante acrescentar aos padrões web as técnicas de acessibilidade associadas ao **WCAG** (*Web Content Accessibility Guidelines*) e suas recomendações.
- As diretrizes WCAG abrangem um vasto conjunto de recomendações que têm como objetivo **tornar o conteúdo Web mais acessível**.

Design responsivo

- O conceito de Design Responsivo surgiu quando em 2010 um desenvolvedor chamado Ethan Marcotte, criou o artigo Responsive Web Design, para o blog A List Apart.
- Um conjunto de práticas que permite criar uma aplicação web com **conteúdo acessível**, otimizando a experiência do usuário, respeitando as limitações, **independente do dispositivo que está sendo utilizado**.
- Não se trata de criar uma versão para cada tipo de dispositivo.



Design responsivo

- Vantagens:
 - Rapidez nas manutenções, pois não precisa atualizar em versões diferentes do site;
 - Melhor posicionamento nos motores de busca em relação a não responsivos;
 - Consistência na experiência do usuário porque os visitantes recebem a mesma experiência mesmo acessando dispositivos.
- Desvantagens:
 - Otimização para celulares, a velocidade de carregamento é menor em comparação a outros dispositivos;
 - Dispositivos menores podem causar dificuldades para projetar navegações mais complexas;
 - O tempo gasto em um projeto usando Design Responsivo é maior do que a execução de um site para monitores 'desktop'.



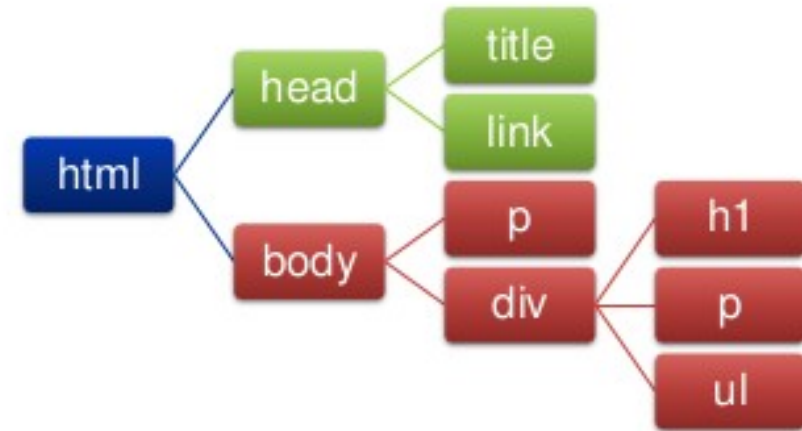
Web Academy



HTML (HyperText Markup Language)

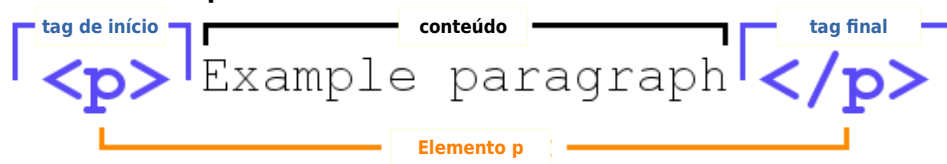
Introdução a HTML

- HTML é uma linguagem interpretada pelo navegador para exibir conteúdo.
- Nossa referência é o HTML 5.
- O documento HTML é composto por elementos hierarquicamente organizados.



Tags (elementos)

- Para inserir um elemento em um documento HTML, utilizamos **tags** correspondentes a esse elemento.



- As tags não diferenciam maiúsculas de minúsculas: `<BODY>` significa o mesmo que `<body>`
- O W3C recomenda letras minúsculas e exige letras minúsculas para tipos de documentos mais rígidos, como XHTML.

Exemplos de tags

```
<html> </html>
<head> </head>
<script> </script>
<title> </title>
<table> </table>
<body> </body>
<p> </p>
<h1> </h1>
<br>
```


Tags (elementos)

- Alguns elementos HTML são classificados como **normal elements**, são abertos com uma tag e fechados com uma barra seguida da mesma tag.
- Exemplo:
- Há também os chamados **void elements**, que não possuem conteúdo, sendo abertos e fechados com apenas uma tag.
- Opcionalmente podem conter uma barra no final da tag.
- Exemplo:

```
<h1>WEB ACADEMY</h1>
```

```

```

Estrutura básica de uma página HTML

- Um documento HTML válido precisa obrigatoriamente seguir uma estrutura básica.
- O primeiro elemento não é um tag, mas sim uma instrução que indica para o navegador a versão HTML.
- Para ver o código HTML da página digite **CTRL + U** ou com o botão direito e selecione “Exibir código-fonte da página”.
- Com o botão direito em um elemento (ou em uma área em branco) é possível **"Inspeccionar"** para ver como os elementos são compostos.
- O **Markup Validation Service** examina e fornece um relatório para informar o que há de errado com seu HTML (<https://validator.w3.org/>).

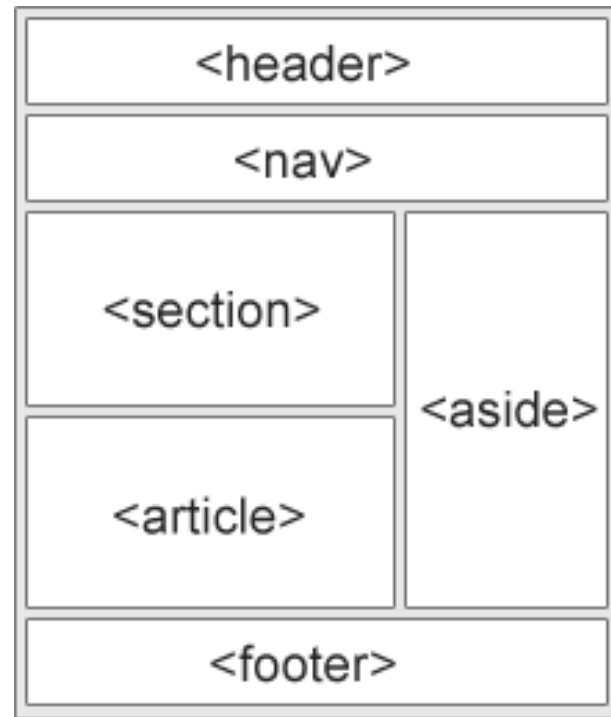
```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="utf-8"/>
  <title>Título</title>
</head>
<body>
  <p>Conteúdo</p>
</body>
</html>
```

Doctype

- Para cada tipo de documento existe uma instrução doctype específica.
- Exemplos:
 - HTML 5: `<!DOCTYPE html>`
 - HTML 4.01 Strict: `<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
 - XHTML 1.0 Strict: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`

Estrutura e Layout

- **<header>** Define o cabeçalho da página ou seção.
- **<nav>** contém a principal funcionalidade de navegação da página.
- **<section>** define uma seção que agrupa um conteúdo.
- **<article>** conteúdo relacionado que faz sentido por si só.
- **<main>** Define a seção principal da página.
- **<aside>** Define o conteúdo lateral que não está diretamente relacionado ao conteúdo principal.
- **<footer>** Define rodapé da página ou seção.



Títulos do conteúdo

- Quando se quer indicar que um texto é um título deve-se utilizar as tags de título (heading).
- São tags de conteúdo que vão de **<h1>** até **<h6>**, sendo **<h1>** o título principal e mais importante, e **<h6>** o título de menor relevância.

```
<h1>Título</h1>
```

```
<h2>Título</h2>
```

```
<h3>Título</h3>
```

```
<h4>Título</h4>
```

```
<h5>Título</h5>
```

```
<h6>Título</h6>
```

Títulos do conteúdo

- A ordem de importância, além de influenciar no tamanho padrão de exibição do texto, tem **impacto nas ferramentas que processam HTML**, como as ferramentas de indexação de conteúdo para buscas (Google, Bing, etc).
- Além disso, os navegadores especiais para acessibilidade também interpretam o conteúdo dessas tags de maneira a **diferenciar seu conteúdo e facilitar a navegação do usuário** pelo documento.

Parágrafos

- Para exibir qualquer texto em uma página, é recomendado que ele esteja dentro de uma tag filha da tag **<body>**, sendo a marcação mais indicada para textos comuns a tag de parágrafo: **<p>**.

- Exemplo:

```
<p>Primeiro parágrafo.</p>
```

```
<p>Segundo parágrafo.</p>
```

- Os navegadores ajustam os textos dos parágrafos à largura do elemento pai, inserindo as quebras de linha necessárias automaticamente.

Caracteres especiais

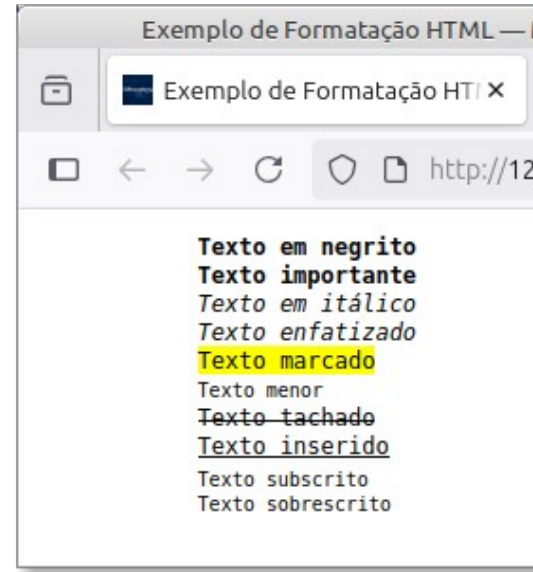
- Em HTML, os caracteres `<`, `>`, `"`, `'` e `&` são especiais. Eles fazem parte da própria sintaxe HTML. Para incluir esses caracteres especiais deve-se inserir um E comercial (`&`) seguido da referência com um ponto e vírgula (`;`) no final.

Caractere	Literal
<code><</code>	<code>&lt;</code>
<code>></code>	<code>&gt;</code>
<code>"</code>	<code>&quot;</code>
<code>'</code>	<code>&apos;</code>
<code>&</code>	<code>&amp;</code>

Formatação e Comentários

- Os elementos de formatação foram projetados para exibir tipos especiais de texto.
- Para escrever um comentário HTML, coloque-o nos marcadores especiais `<!--` e `-->`.
- Os comentários não são exibidos pelo navegador, mas ajudam a documentar o código HTML.

```
<pre>
<b>Texto em negrito</b>
<strong>Texto importante</strong>
<i>Texto em itálico</i>
<em>Texto enfatizado</em>
<mark>Texto marcado</mark>
<small>Texto menor</small>
<del>Texto tachado</del>
<ins>Texto inserido</ins>
<sub>Texto subscrito</sub>
<sup>Texto sobrescrito</sup>
<!-- Comentários -->
</pre>
```



Elementos genéricos (de agrupamento)

- **<div>** e **** são elementos genéricos que não representam um conteúdo específico, mas são úteis para agrupar conteúdos (ou elementos) que compartilham atributos de estilo.
- Devem ser utilizados apenas quando não existirem outros elementos para representar o conteúdo.
- Diferença: **<div>** é um elemento de nível de bloco (agrupar blocos) e **** de nível de linha (agrupar texto).

```
<!DOCTYPE html>
<html>
<body>
  <h1>Título</h1>
  <div>
    <h2>Exemplo</h2>
    <p>Dentro do elemento DIV</p>
  </div>
  <p>Parágrafo fora do elemento DIV
```

que contém um elemento

```
<span>SPAN</span>.
```

```
</p>
</body>
</html>
```

Listas

- Para criar listas em HTML são utilizadas as tags:
 - **** cria listas não ordenadas;
 - **** cria listas ordenadas;
 - **** cria itens nas listas.
- As listas podem ser aninhadas (lista dentro da lista)
- O atributo **type** das tags de criação define o tipo de marcador do item. Possíveis valores não ordenados: “disc”, “square”, “circle” e ordenados: “1”, “A”, “a”, “I”, “i”

```
<h4>Lista não
ordenada:</h4>
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ul>
<h4>Lista ordenada:</h4>
<ol>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ol>
```

Lista não ordenada:

- Item A
- Item B
- Item C

Lista ordenada:

1. Item A
2. Item B
3. Item C

Imagens

- A tag **** insere uma imagem e possui dois atributos obrigatórios:
 - **src**: indica o URL (*Uniform Resource Locator*), ou seja, o caminho do arquivo.
 - **alt**: define um texto alternativo caso a imagem não seja carregada.
- Em HTML 5:
 - **<figure>**: especifica conteúdo como ilustrações, diagramas, fotos, etc.
 - **<figcaption>**: define uma legenda.
- Atributos de tamanho:
 - **width** e **height** especificam a largura e a altura da imagem (em pixels). Cuidado com a proporcionalidade.

URL absoluto: imagem hospedada em outro site.

src="www.site.com/images/foto.jpg"

URL relativo: imagem hospedada no próprio site. Se o URL começar com uma barra, será relativo ao domínio.

src="/images/foto.jpg".

```
<figure>
```

```
  
```

```
  <figcaption>
```

```
    Legenda da foto.
```

```
  </figcaption>
```

```
</figure>
```


Links

- As ligações (âncoras) entre páginas (hiperlinks ou simplesmente link) são definidas pela tag **<a>**
- O atributo **href** (referência de hipertexto) especifica o URL da página de destino.
- Os links podem ser criados sobre conteúdo de texto simples ou vários outros tipos de elementos HTML, como imagens, títulos, etc.
- O atributo **target** especifica onde abrir a página:
 - **_self**: abre na mesma janela/aba
 - **_blank**: abre em uma nova janela/aba

```
<a
href="http://www.ufac.br">UFAC</a>
<a href="http://csi.ufac.br"
target="_blank">
  <h1>SI-UFAC</h1>
</a>
<a href="http://webacademy.ufac.br"
">
  <figure>
    
  </figure>
</a>
```

Tabelas

- Uma tabela é definida não apenas por uma tag, mas pode ter até 10 tags diferentes
- Três elementos básicos:
 - **<table>**, **<tr>** e **<td>**
- Objetivo: apresentar dados tabulares, comparativos, etc. (não para posicionar elementos na página)

```
<table>  
  <tr> <td></td> <td></td> <td></td> </tr>  
  <tr> <td></td> <td></td> <td></td> </tr>  
  <tr> <td></td> <td></td> <td></td> </tr>  
  <tr> <td></td> <td></td> <td></td> </tr>  
</table>
```

Tabelas

Tag	Descrição
<table>	Define uma tabela
<tr>	Insere uma linha na tabela
<td>	Insere uma célula dentro de um elemento <tr>
<th>	Insere uma célula (cabeçalho) dentro de um elemento <tr>
<caption>	Atribui um título ou descrição para a tabela
<colgroup>	Especifica um grupo de colunas para formatação
<col>	Define propriedades da coluna para cada elemento dentro do <colgroup>
<thead>	Define o cabeçalho da tabela
<tbody>	Define o corpo (conteúdo principal) da tabela
<tfoot>	Define o rodapé da tabela

```

<table>
  <caption>Alunos</caption>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Nota</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Aluno A</td>
      <td>9.0</td>
    </tr>
    <tr>
      <td>Aluno B</td>
      <td>4.5</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2">Quantidade de alunos: 2</td>
    </tr>
  </tfoot>
</table>

```

Tabelas (Exemplo)

Alunos	
Nome	Nota
Aluno A	9.0
Aluno B	4.5
Quantidade de alunos: 2	

Formulários

- Um formulário serve para enviar informações
- A tag **<form>** define, dentre outras coisas, que a página irá processar as informações
- Os tipos de campos são definidos pela tag **<input>**, e suas identificações pela tag **<label>**
- O atributo **name** identifica o campo no formulário, o **type** define o tipo do campo
- Lista de tipos de input:
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#input_types

O atributo **action** define a ação a ser executada quando o formulário for enviado.

```
<form action="/action_page.jsp">
  <label for="nome">Nome:</label>
  <input type="text" name="nome" id="nome"
placeholder="Digite seu primeiro nome">
  <br>
  <label for="sobrenome">Sobrenome:</label>
  <input type="text" name="sobrenome"
id="sobrenome" placeholder="Digite seu
sobrenome">
  <br>
  <input type="submit" value="Enviar">
</form>
```

Nome:

Sobrenome:



Web Academy



CSS (Cascading Style Sheets)

Introdução ao CSS

- As **folhas de estilo em cascata** (Cascading Style Sheets – CSS) descrevem a apresentação de um documento HTML, isto é, são regras que especificam como os elementos devem ser exibidos.
- Foram criadas para preencher uma lacuna deixada pelo HTML: nunca houve a intenção de adicionar **tags de formatação**.
- Adicionar formatação para cada elemento ou página é trabalhoso e tira o foco do objetivo principal do HTML: **descrever e organizar o conteúdo**.
- Exemplo do que pode ser realizado com CSS:
www.csszengarden.com

Introdução ao CSS

Versões antigas do HTML
(vários atributos para cada tag)

Utilização recomendada
(atributo style)

```
<body bgcolor="blue"> <body style="background-color: blue">
```

- As declarações CSS acima produzem o mesmo efeito.
- A principal diferença é que CSS permite outras formas de organizar as declarações que tratam da formatação do documento.
- Com o botão direito também é possível "**Inspecionar**" para ver como os elementos são compostos.

Formas de aplicação do CSS

- Há **3 formas de aplicar CSS em documentos HTML** (em ordem de prioridade):
 1. Aplicando um estilo diferente para cada elemento HTML por meio do atributo *style* (**inline**);
 2. Aplicando um **estilo interno** para um determinado documento HTML;
 3. Utilizando um **arquivo externo** é possível mudar a apresentação (estilo) de toda aplicação ou site com um único arquivo.

Formas de aplicação do CSS

<!--Inline-->

```
<!DOCTYPE html>
<html>
<body>
  <h1 style="color: blue">
    WEB ACADEMY
  </h1>
  <p style="color: red">
    Curso de HTML
  </p>
</body>
</html>
```

<!--Interno-->

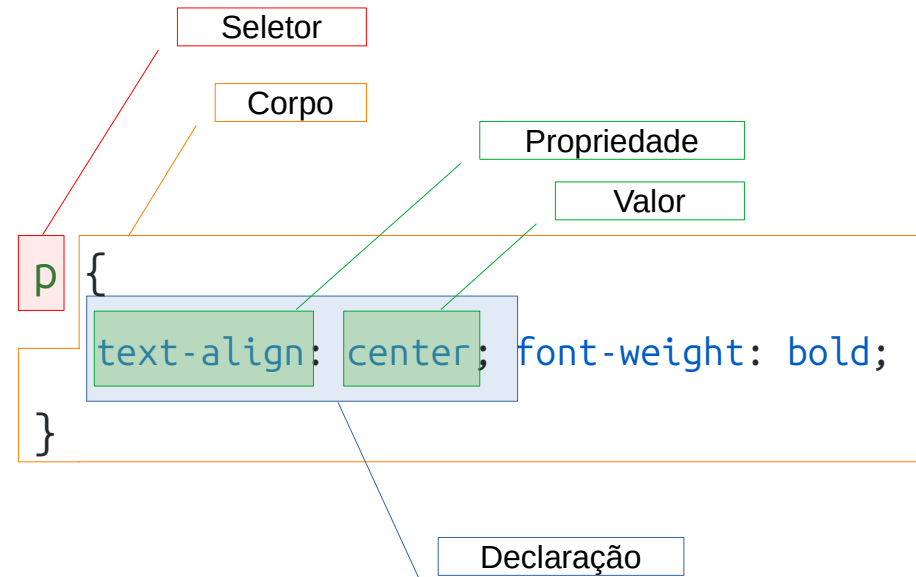
```
<!DOCTYPE html>
<html>
<head>
  <style>
    body{background-color: grey;}
  </style>
</head>
<body>
  <h1>WEB ACADEMY</h1>
  <p>Curso de HTML</p>
</body>
</html>
```

<!--Externo-->

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
        type="text/css"
        href="estilo.css">
</head>
<body>
  <h1>WEB ACADEMY</h1>
  <p>Curso de HTML</p>
</body>
</html>
```

Sintaxe

- As declarações (regras) **CSS** possuem uma **sintaxe muito simples**;
- Consiste na **propriedade** seguida do seu **valor**, separados pelo sinal de dois pontos (":");
- Para separar várias propriedades usamos o ponto-e-vírgula.



Seletores

- O seletor identifica para quais elementos uma declaração CSS será aplicada.
- Podem ser de cinco tipos (básicos):
 - **Universal**: seleciona todos os elementos;
 - **Elemento (tipo)**: seleciona elementos com base no nome do elemento (tag);
 - **Classe**: seleciona elementos com um atributo de classe específico;
 - **ID**: usa o atributo ID para selecionar um elemento específico;
 - **Atributo**: seleciona elementos com base no valor de um atributo específico.
- Mais seletores:
 - https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/Selectors

Seletores

```
<!DOCTYPE html>
<html>
<head>
  <title>Seletores</title>
</head>
<body>
  <a href="http://www.google.com">Google</a>
  <a href="http://www.ufac.br">UFAC</a>
  <a href="http://csi.ufac.br">SI-UFAC</a>
  <a href="http://webacademy.ufac.br">Web
Academy</a>
</body>
</html>
```

```
* {
  /* Universal */
  color: red;
}
a {
  /* Elemento (tag) */
  color: red;
}
.link_red {
  /* Classe */
  color: red;
}
#link_red {
  /* ID */
  color: red;
}
a[href="http://www.google.com"] {
  /* Atributo */
  color: red;
}
```


Agrupamento de seletores

- É possível agrupar seletores, separados por vírgula, aplicando a mesma formatação para vários tipos de elementos.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p, h1, h2 {
      text-align: center;
      color: red;
    }
  </style>
</head>
<body>
  <h1>WEB ACADEMY</h1>
  <p>Curso de CSS</p>
  <h2>Introdução</h2>
  <p>Seletores</p>
  <p>Agrupamento de Seletores</p>
</body>
</html>
```

Combinação de seletores

- É possível combinar múltiplos seletores:
 - Aplicando **um ponto (.)** entre o elemento HTML e a classe;
 - Aplicando um **espaço em branco** entre os seletores (elementos descendentes);
 - Aplicando o **operador soma (+)** entre os seletores (vizinho mais próximo);
 - Aplicando o **operador (>)** entre os seletores (filho direto);
 - Aplicando o **operador (~)** entre os elementos (vizinhos posteriores)

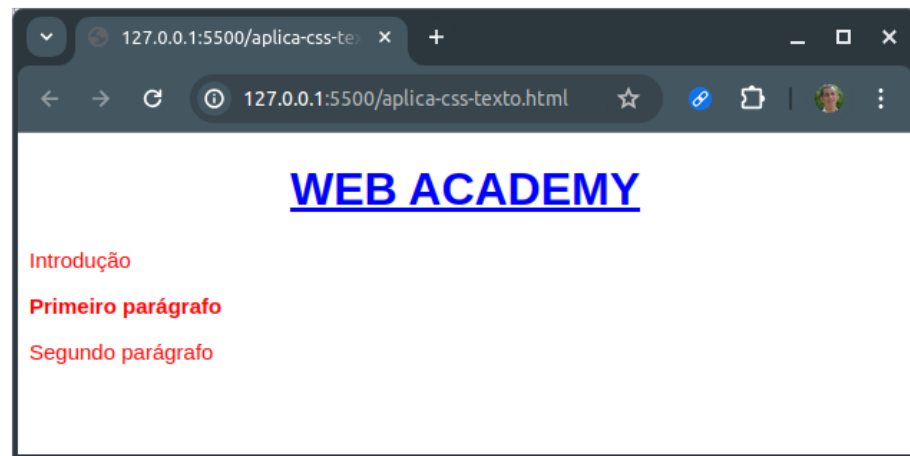
```
<!DOCTYPE html>
<html>
<head>
  <style>
    p.p1 {font-weight: bold;}
    p span {font-style: italic;}
    p+h2 {font-size: 72px;}
  </style>
</head>
<body>
  <h1>WEB ACADEMY</h1>
  <p>Curso de CSS</p>
  <h2>Introdução</h2>
  <p class="p1">Seletores</p>
  <p>Agrupamento de
  <span>Seletores</span></p>
</body>
</html>
```

Aplicação do CSS: Textos

- Há muitas propriedades para formatação de texto em CSS, dentre as quais destacam-se:
 - **color**: define a cor do texto;
 - **text-align**: define o alinhamento (*left*, *right*, *center*, *justify*);
 - **text-decoration**: adiciona traços sob (*underline*), sobre (*overline*) ou no meio (*line-through*) do texto;
 - **font-family**: define a fonte utilizada no texto;
 - **font-size**: tamanho da fonte;
 - **font-weight**: define a espessura da fonte (*normal*, *bold*).

Aplicação do CSS: Textos

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      color: blue;
      font-family: Arial, Helvetica, sans-serif;
    }
    h1 {
      font-size: 6em;
      text-align: center;
      text-decoration: underline;
    }
    p {color: red; font-size: 15px; }
    p.negrito {font-weight: bold; }
  </style>
</head>
<body>
  <h1>WEB ACADEMY</h1>
  <p>Introdução</p>
  <p class="negrito">Primeiro parágrafo</p>
  <p>Segundo parágrafo</p>
</body>
</html>
```



Aplicação do CSS: Medidas

- Para definir um tamanho ou uma distância, devemos utilizar as unidades de medida específicas do CSS.
- Podemos classificar essas unidades em **absolutas** e **relativas**:
 - Absolutas: **cm** (centímetro), **mm** (milímetro), **px** (pixel);
 - Relativas: a mais utilizada é **em**, que representa a medida proporcional do elemento onde é aplicada (em relação ao tamanho da fonte do elemento pai).

Cálculo da medida "em"

- Para calcular a medida **em**, é necessário multiplicar o valor da medida pelo tamanho da fonte do elemento pai.

```
<div class="pai">
  <p class="filho">
    Este é um parágrafo.
  </p>
</div>
```

```
.pai {
  font-size: 16px;
}
.filho {
  font-size: 1.2em;
}
```

- O tamanho da fonte do elemento filho será **1,2 vezes o tamanho da fonte do elemento pai**, que é 16 pixels. Portanto, o tamanho da fonte do elemento filho será 19,2 pixels (16 x 1,2).

Aplicação do CSS: Cores

- As cores são especificadas usando nomes de cores predefinidos ou valores **RGB**, **HEX**, **HSL**, **RGBA**, **HSLA**;
- Existem 140 nomes de cores definidos:
 - https://www.w3schools.com/colors/colors_names.asp
- **RGBA** e **HSLA** permitem controlar o canal **alfa** (opacidade) para definir o nível de transparência.

```
<h1 style="background-color: tomato;">  
  red  
</h1>  
<h1 style="background-color: rgb(255,99,71);">  
  rgb(255,99,71)  
</h1>  
<h1 style="background-color: #ff6347;">  
  #ff6347  
</h1>  
<h1 style="background-color: hsl(9,100%,64);">  
  hsl(9,100%,64%)  
</h1>  
<h1 style="background-color: rgba(255,99,71,0.5);">  
  rgba(255,99,71,0.5)  
</h1>  
<h1 style="background-color:  
  hsla(9,100%,64%,0.5);">  
  hsla(9,100%,64%,0.5)  
</h1>
```


Pseudo-classes e pseudo-elementos

- Uma **pseudo-classe** permite aos seletores **especificar estados** de um elemento.
- Sintaxe:
- Um **pseudo-elemento** permite aos seletores **especificar uma parte** de um elemento
- Sintaxe:

```
/* Altera a cor da fonte quando  
o cursor passar sobre o link */
```

```
a:hover {  
  color: #003366;  
}
```

```
/* Adiciona “:” após o label do  
formulário */
```

```
form label::after {  
  content: “:”;  
}
```

Cascata, Herança e Especificidade

Cascata

A ordem das regras tem importância no sentido que, dado dois elementos de mesma especificidade, a **última regra é a que será aplicada**.

```
h1 {color: red;}  
h1 {color: blue;}
```

Herança

Propriedade CSS dos elementos pais são **herdados por seus elementos filhos**.

Exemplo: a cor do texto definida para o elemento <body> será a mesma para os elementos internos que não tem cor definida. Não é aplicável para algumas propriedades (Ex: background).

Especificidade

Em CSS, o **seletor mais específico prevalece**.

O peso é definido por valores atribuídos pela quantidade de ID, Classe e Tipo, onde ID é o seletor mais específico e Tipo o menos específico.

```
#principal {color: red;}  
.principal {color: blue;}  
h1 {color: green;}
```

Cálculo de especificidade

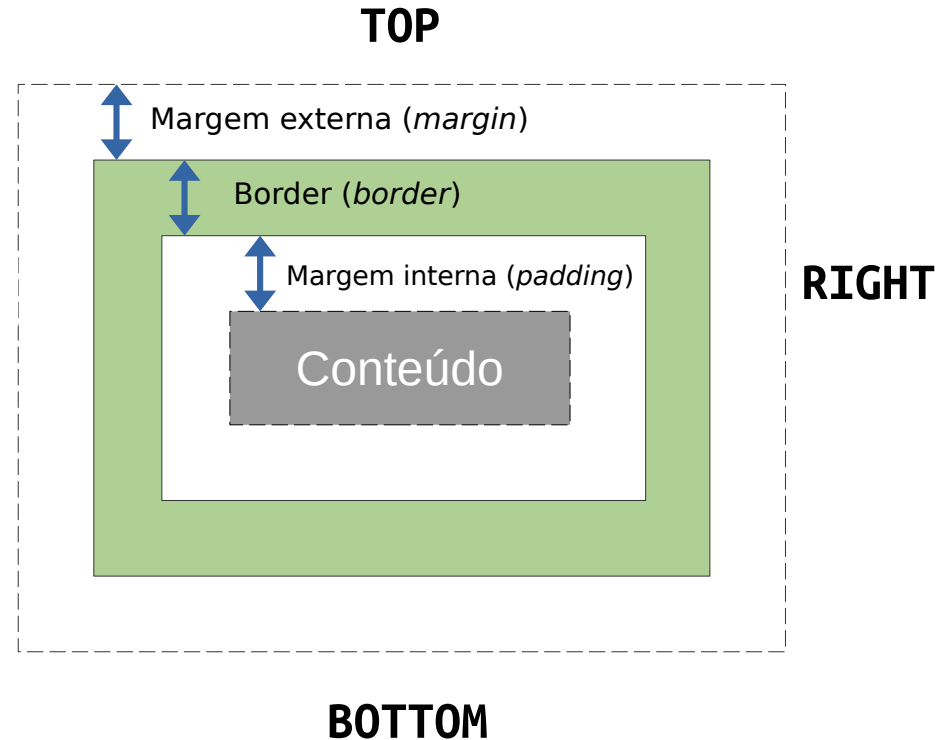
- Para calcular a especificidade em CSS, é necessário atribuir um valor a cada tipo de seletor:
 - Universal (*): 0
 - Elemento ou pseudo-elemento: **0-0-1**
 - Classe, pseudo-classe ou atributo: **0-1-0**
 - ID: **1-0-0**
 - Estilo inline (atributo style): **1-0-0-0**

```
a[href="http://www.google.com"]
{
  /* 0-1-1 */
  color: black;
}
p a { /* 0-0-2 */
  color: orange;
}
p a#link_id { /* 1-0-2 */
  color: grey;
}
p a.link_class { /* 0-1-2 */
  color: navy;
}
```

Box Model (bordas e margens)

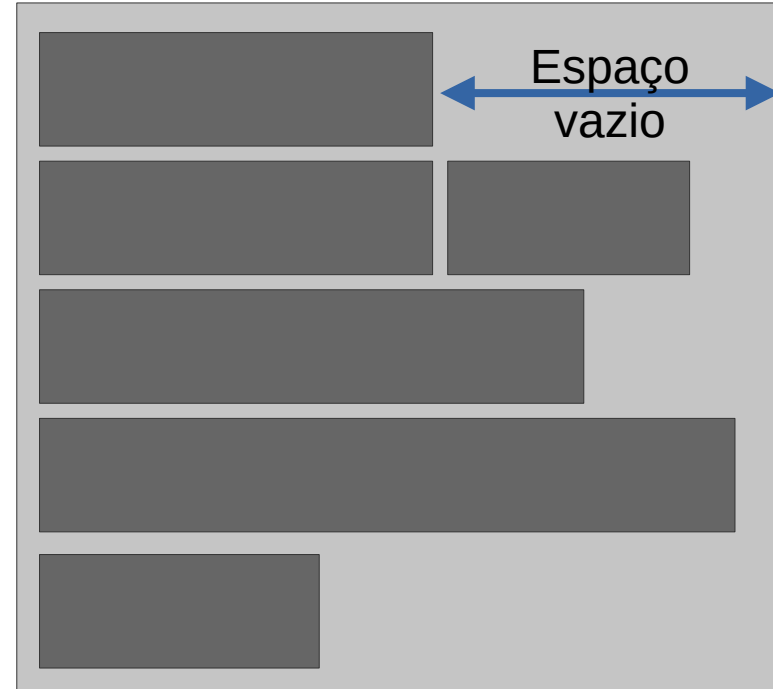
- Todo elemento HTML está contido em um **box**;
- O box é composto por **conteúdo**, **margem interna** (padding), **borda** (border) e **margem externa** (margin);
- Dois tipos principais:
 - **block-level**: ocupam todo o espaço horizontal, provocando quebras de linha;
 - **inline-level**: ocupam somente o espaço necessário para o seu conteúdo.
- Propriedade box-sizing:
 - content-box: limita pelo conteúdo
 - border-box: limita pela borda

LEFT



Layout: fluxo normal

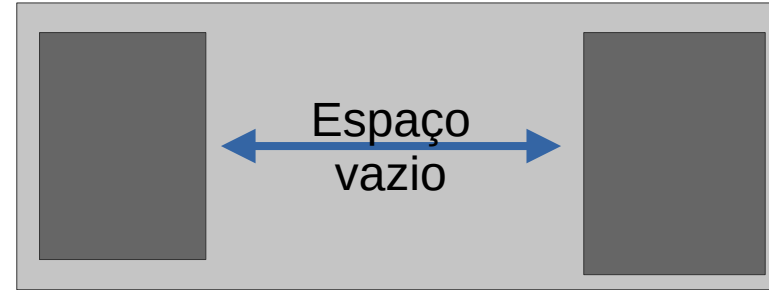
- Os **elementos** nas páginas da Web **se dispõem de acordo com o fluxo normal**, onde os elementos a nível de bloco são dispostos um abaixo do outro, e os elementos em nível de linha são mostrados lado a lado.



Layout: flex e flexbox

- Método de layout **unidimensional** para **dispor itens em linhas ou colunas**, sendo que **os itens são flexíveis** para preencher espaço adicional e encolhem para caber em espaços menores.

Linha



Coluna



Layout: grid

- Sistema de layout **bidimensional** que permite **dispor o conteúdo em linhas e colunas**, além de possuir muitos recursos que simplificam a criação de layouts complexos.



Media Queries

- São úteis quando se deseja modificar a página dependendo das características do dispositivo, como o **tamanho da tela** ou se o conteúdo será exibido em **mídia impressa**.

```
@media screen and (max-width: 640px)
{
  nav li a {
    width: 100%;
  }
}
```

```
@media print {
  table th {
    color:
    black;
  }
}
```



Web Academy



JavaScript



Introdução ao JavaScript

- Considerando as 3 principais tecnologias do lado cliente (front-end), **JavaScript** complementa **HTML** e **CSS** com recursos de uma **linguagem de programação**.
- O foco HTML é o **conteúdo**, o foco CSS é a **apresentação**, o restante fica por conta de JavaScript, sobretudo os aspectos relacionados a **interatividade**.
- Desta forma, o foco do JavaScript é permitir que as páginas sejam dinâmicas, tornando-as mais interativas.
- Baseado na especificação **ECMAScript** (ECMA-262).

Sintaxe

```
// Declaração de variáveis
var x = 5;
var y = 6;
// Função
function soma() {
    if (x > 0) {
        return x + y;
    }
}

// Chamada da função
soma();
// Função anônima (arrow function)
const somar = () => { return x + y };
somar();
```

```
(param1, param2, ..., paramN) => { statements }
(param1, param2, ..., paramN) => expression
// equivalente a: => { return expression; }
```

// Parênteses são opcionais quando só há um nome de parâmetro:

```
(singleParam) => { statements }
singleParam => { statements }
```

// A lista de parâmetros para uma função sem parâmetros deve ser escrita com um par de parênteses.

```
() => { statements }
```

Variáveis: *var*, *let* e *const*

- Originalmente, JavaScript suportava apenas ***var***, mas seu funcionamento pode ser bastante **confuso**:
 - Permite que variáveis com mesmo nome possam ser declaradas.
 - Uma variável pode ser declarada depois de ser inicializada!
 - Escopo local declarada dentro de funções e global fora das funções.
- ***let*** funciona de com escopo de bloco e **não possui os mesmos problemas**.
- ***const*** permite declarar constantes, isto é, variáveis que **não podem alterar seu valor após inicialização**.

```
// Declaração de variável
// após ser inicializada
numero = 1;
var numero;

// Declaração de variável
// com mesmo nome
var numero = 2;
var numero = 3;
```

Variáveis: *var*, *let* e *const*

```
// Declaração de variável global e local
// Testar com a opção let
function exemplo() {
  var x = 0;
  if (true) {
    var x = 10;
    var y = 20;
    console.log("Dentro do bloco: x =", x, "y =", y); // Saída: 10 e 20
  }
  console.log("Fora do bloco: x =", x); // Saída: 10 (x acessível por ser var)
  console.log(y); // Saída: 20 ou ReferenceError com let
}

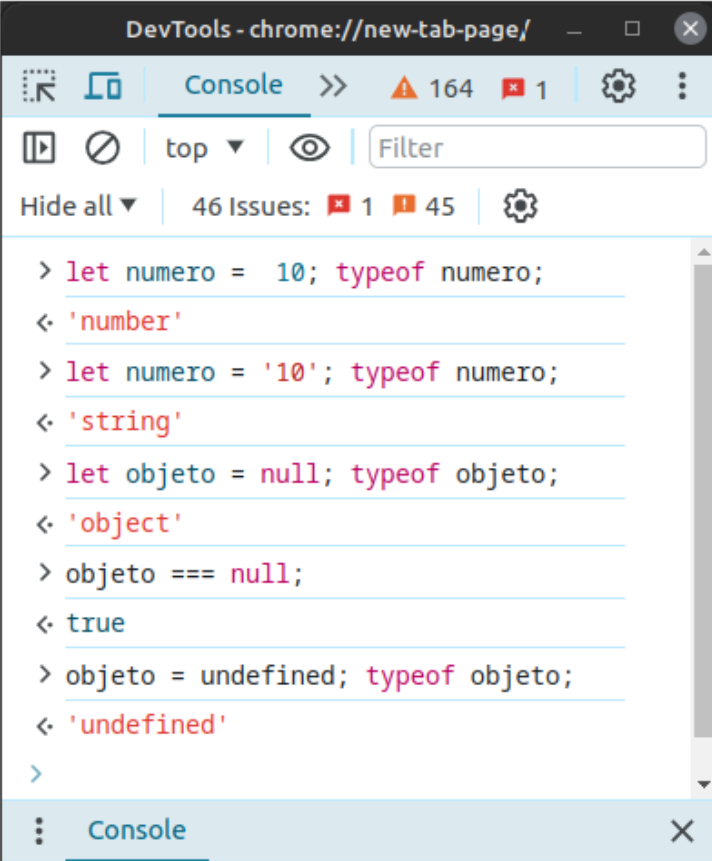
exemplo();
```

Variáveis: *var*, *let* e *const*

- Declarações **var** tem escopo global ou função, **let** e **const** têm escopo de bloco.
- Variáveis **var** podem ser atualizadas e declaradas novamente dentro de seu escopo, **let** podem ser atualizadas, mas não podem ser declaradas novamente e **const** não podem ser atualizadas nem declaradas novamente.
- Todas passam por **hoisting** (mecanismo que move as declarações de variáveis e de funções para o topo de seu escopo antes da execução).
- Enquanto **var** e **let** podem ser declaradas sem ser inicializadas, **const** precisa da inicialização durante a declaração.
- Sempre declare variáveis: **const** se o valor e se o tipo (Arrays e Objetos) não devem ser alterados, **let** se não puder usar **const** e **var** se precisar oferecer suporte a navegadores antigos.

Tipos de dados

- JavaScript possui tipagem **fraca** (aceita operações implícitas entre tipos diferentes) e **dinâmica** (não exige o tipo na declaração).
- Tipos primitivos:** string, number, bigint, boolean, symbol, null, undefined e object.
 - undefined:** variável não teve valor atribuído.
 - null:** ausência intencional de valor.
- == e ===** são operadores de comparação. O **==** (igualdade solta) realiza uma comparação de igualdade após a conversão de tipo, enquanto **===** (igualdade estrita) verifica a igualdade sem conversão de tipo, exigindo que ambos: valores e tipos sejam iguais para retornar true.



The screenshot shows the Chrome DevTools Console with the following code and output:

```
> let numero = 10; typeof numero;
< 'number'

> let numero = '10'; typeof numero;
< 'string'

> let objeto = null; typeof objeto;
< 'object'

> objeto === null;
< true

> objeto = undefined; typeof objeto;
< 'undefined'

>
```

Estruturas de controle (if, else)

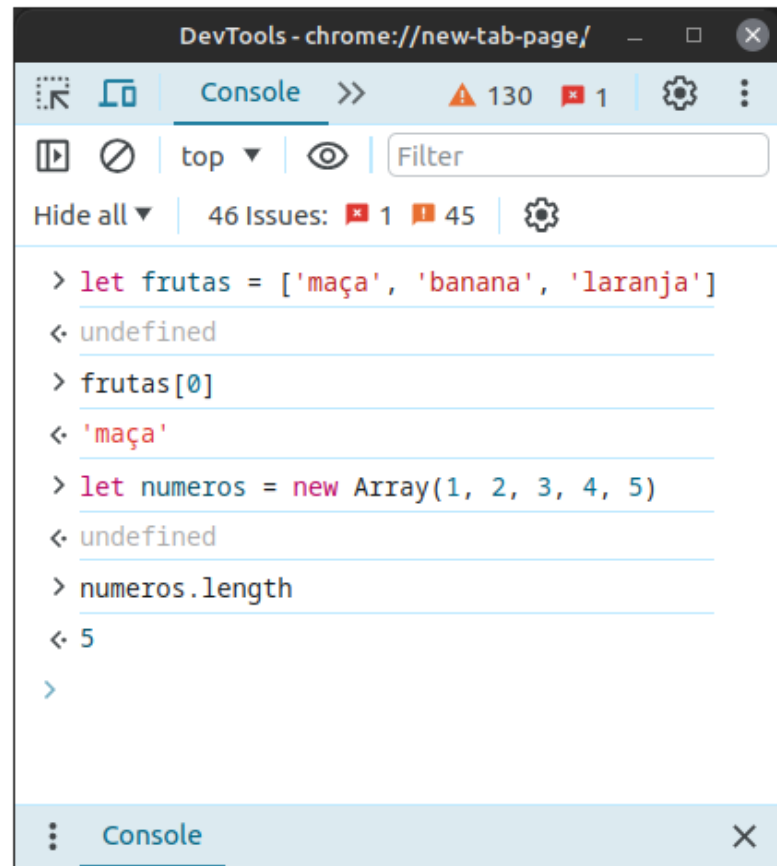
- Uma expressão (premissa), ou simplesmente condição, pode ser avaliada como verdadeira (true) ou falsa (false)
- Múltiplas condicionais **if ... else** podem ser aninhados quando necessário.
- Múltiplas instruções pode ser executadas em bloco ({ ... })
- Os valores primitivos boolean: **true** e **false** são diferentes dos valores **true** e **false** do objeto Boolean.
- Qualquer valor que não for undefined, null, 0, NaN, ou uma string vazia (""), e qualquer objeto, incluindo um objeto Boolean cujo valor é false, é avaliado como true quando passado por uma condicional

```
if (condição1)
    instrução1
else if (condição2)
    instrução2
else if (condição3)
    instrução3
...
else
    instruçãoN
```

```
if (condição) {
    instrução1
} else {
    instrução2
}
```

Arrays

- Estruturas de dados que armazenam uma coleção de elementos (iteráveis), que podem ser de qualquer tipo.
- Os elementos de um array são acessados pelo seu índice, começando do zero.
- Lista de métodos e propriedades:
 - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array



Arrays: formas de iteração

- Existem 3 formas principais de iterar sobre arrays em JavaScript:
 - Usando o loop **for**
 - Usando o loop **for...of**
 - Usando o método **forEach**

- O método **forEach** é **menos eficiente**, porque implica em chamadas de função para cada elemento do array, mas na maioria das vezes não é significativo.

```
let frutas = ['maçã', 'banana', 'laranja'];
for (let i = 0; i < frutas.length; i++) {
  console.log(frutas[i]);
}
for (const fruta of frutas) {
  console.log(fruta);
}
frutas.forEach((fruta, indice, array) => {
  console.log(fruta);
});
```

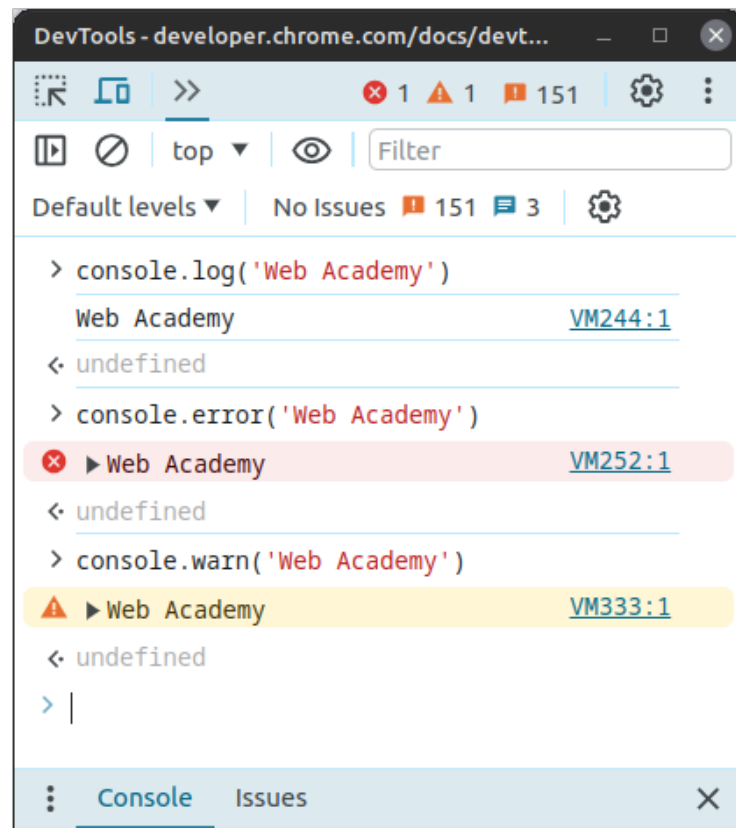
Objetos

- Um objeto em JavaScript consiste em um **conjunto de propriedades** (variáveis e funções).
- As propriedades do objeto podem ser acessadas usando **notação de colchetes ou de ponto**.
- A palavra reservada **this** faz referência ao objeto atual.

```
//let estado = new Object(); // Cria um objeto vazio
//let estado = {}; // Forma alternativa de criar um objeto
let estado = {
  nome: "Acre",
  populacao: 906876,
  capital: {
    nome: "Rio Branco",
    populacao: 413418
  },
  estados_limitrofes: ["Amazonas", "Rondônia"],
  indicadores: function () {
    alert("Indicadores do estado de " + this.nome + ":"
      + "\n- Expectativa de vida (2015): 73,6 anos"
      + "\n- IDH (2017): 0,719");
  }
}
// Notação de colchetes
estado["capital"]["nome"];
estado["indicadores"]();
// Notação de ponto
estado.capital.nome;
estado.indicadores();
```

O objeto console

- **Ferramenta de depuração** disponível nos navegadores que permite exibir informações, mensagens de erro, avisos e outros tipos de dados no console do navegador. Métodos mais comuns:
 - **console.log()**
 - **console.error()**
 - **console.warn()**
- Os três métodos exibem mensagens, mas em formatos diferentes.



Formas de utilizar JavaScript

- Formas de inserir código JavaScript em documentos HTML:
 - Por meio da tag **script** com o código JavaScript no corpo do documento HTML (**interno**);
 - Também utilizando a tag **script** é possível carregar um arquivo **externo** com o código JavaScript;
 - Ou ainda por meio dos **eventos**, utilizando atributos específicos de tags HTML:
 - https://www.w3schools.com/tags/ref_eventattributes.asp

Formas de utilizar JavaScript

- Tag script **externo** com o código JavaScript;
- Tag **script** com o código no corpo do documento HTML (**interno**);
- **Eventos** utilizando atributos específicos de tags HTML.

```
<!DOCTYPE html>
<head>
  <script src="script.js" defer></script>
</head>
<body>
  <script>
    alert('Olá mundo!');
  </script>
  <button type="button"
    onclick="alert('Olá mundo!')">
    Clique aqui
  </button>
</body>
</html>
```

Atrasa a execução do script

Eventos

- Os eventos são **ações** atribuídas a um determinado **elemento da página web** (imagem, botão, parágrafo, etc.), que podem ser capturadas e permitem que o sistema apresente uma resposta para o usuário.
- Formas de usar eventos em páginas web:
 - Utilizando atributos **inline** (não é uma boa prática).

```
<button type="button"
onclick="alert('Olá mundo!')">
  Clique aqui
</button>
```

Retorna o primeiro elemento no documento que corresponde ao seletor CSS.

- Alterando as **propriedades do objeto**;

```
let botao =
document.querySelector('button');
botao.onclick = () => {
  alert('O botão foi clicado!');
}
```

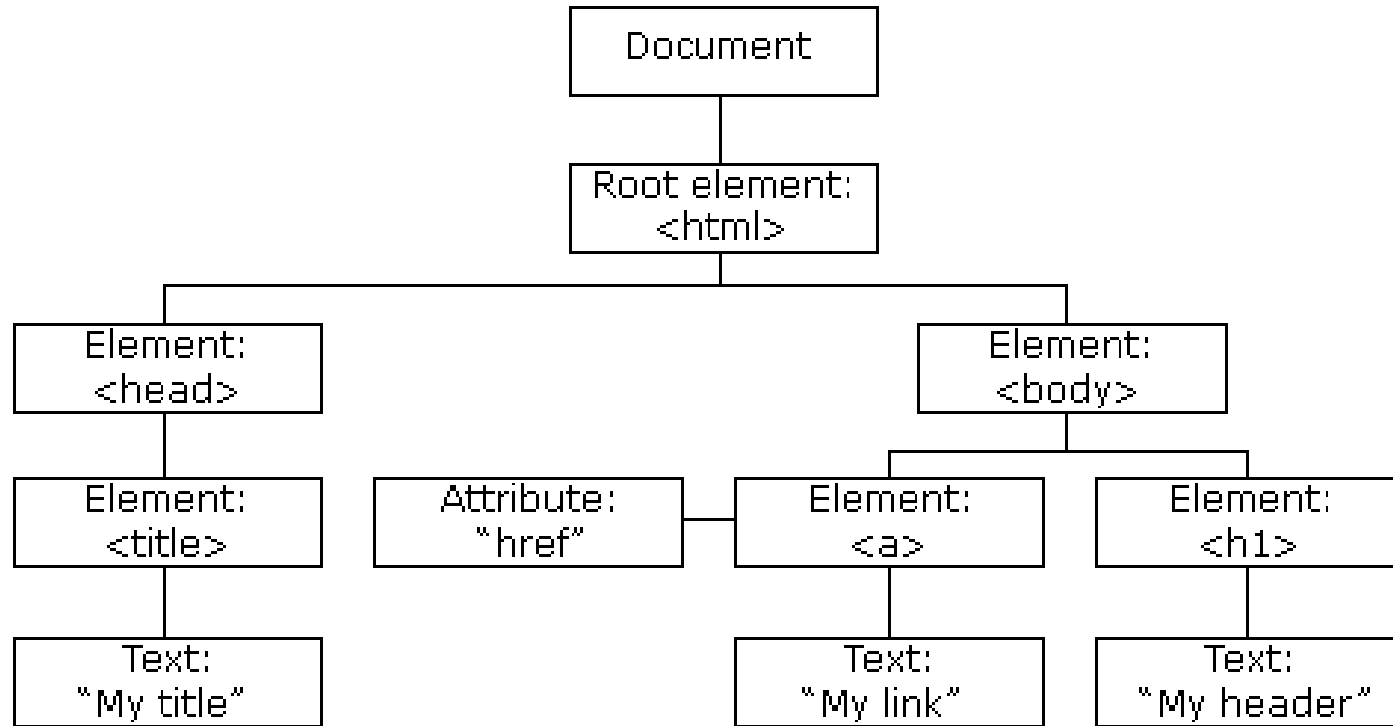
- Por meio de **manipuladores de eventos**: permite maior controle, incluindo **atribuição de múltiplos eventos** (recomendado!).

```
let botao = document.querySelector('button');
function alertaA() { alert('Mensagem A') };
function alertaB() { alert('Mensagem B') };
botao.addEventListener('click', alertaA);
botao.addEventListener('click', alertaB);
```

DOM (Document Object Model)

- O **DOM** permite acessar, alterar, inserir e remover elementos em documento HTML, utilizando chamadas JavaScript.
- **Obedece a hierarquia dos elementos HTML**, que podem ser representados como uma árvore de objetos.
- Utilizando DOM é possível modificar elementos e atributos HTML, além das propriedades CSS.

DOM (Document Object Model)



DOM (Document Object Model)

Alterando
conteúdo e cor da
fonte do elemento
h1

```
<!DOCTYPE html>
<html>
<body>
  <h1>Título</h1>
  <button type="button">Clique aqui</button>
  <script>
    let botao =
document.querySelector('button');
    botao.addEventListener('click', () => {
      let titulo =
document.querySelector('h1');
      titulo.innerHTML = 'WEB ACADEMY';
      titulo.style.color = 'red';
    });
  </script>
</body>
</html>
```

DOM (Document Object Model)

- Outros métodos podem ser consultados em:
 - https://www.w3schools.com/js/js_htmlDOM.asp
 - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

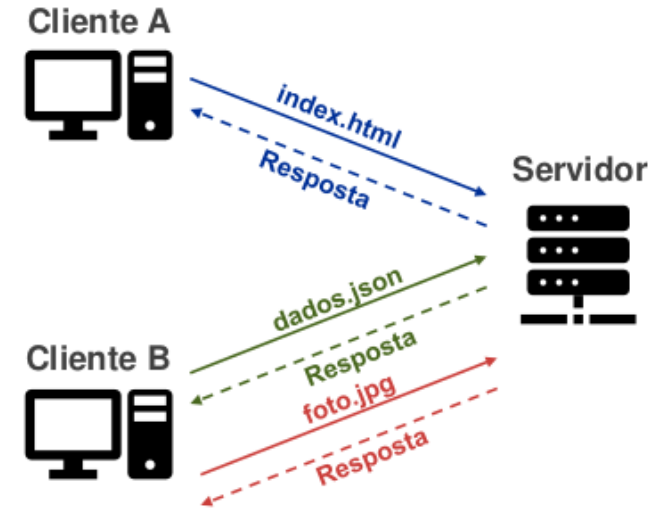
JSON

- JSON (JavaScript Object Notation) é um formato de arquivo leve, baseado em texto, auto descritivo, para **armazenamento e transmissão de dados**.
- É um tipo de objeto JavaScript, isto é, um conjunto de pares chave e valor (apenas propriedades, sem métodos).

```
[
  {
    "nome": "Acre",
    "capital": "Rio Branco",
    "regiao": "Norte",
    "populacao": 906876,
    "estados_limitrofes": [
      "Amazonas", "Rondônia"
    ]
  },
  {
    "nome": "Rondônia",
    "capital": "Porto Velho",
    "regiao": "Norte",
    "populacao": 1796460,
    "estados_limitrofes": [
      "Acre", "Amazonas", "Mato Grosso"
    ]
  }
]
```


Requisições assíncronas

- **Aplicações web funcionam através de requisições HTTP**, dentro de uma arquitetura que pode ser definida genericamente como cliente/servidor.
- Neste sentido, as requisições HTTP podem ser de dois tipos:
 - **Síncrona**: quando o processo que fez a requisição fica bloqueado até receber uma resposta do servidor;
 - **Assíncrona**: onde podem ser enviadas várias requisições em paralelo, em cada uma delas aguarda sua respectiva resposta, isto é, não há sincronismo entre as requisições.



Requisições assíncronas

- Requisições assíncronas representam a base de um conceito muito popular que surgiu em meados dos anos 2000 como uma “nova abordagem para aplicações web” denominada **AJAX** (Asynchronous JavaScript + XML).
- AJAX envolve várias tecnologias (XHTML, CSS, DOM, XML, JavaScript), mas depende sobretudo do componente **XMLHttpRequest** (XHR).
- Especificação:
<https://xhr.spec.whatwg.org/>

```
<button type="button">Carregar texto</button>
<p id="texto">0 texto será carregado aqui</p>
<script>
  let xhr = new XMLHttpRequest();
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    let texto = document.querySelector("#texto");
    xhr.open("GET", "http://exemplo.com/exemplo.txt");
    xhr.addEventListener('readystatechange', function(){
      if (xhr.readyState == 4 && xhr.status == 200){
        texto.innerHTML = xhr.responseText;
      }
    });
    xhr.send();
  });
</script>
```

Requisições assíncronas com Promises

- **Promises** representam o resultado de uma operação assíncrona que pode ser concluída no futuro. Pode estar em um de três estados:
 - **Pendente** (pending): operação ainda não foi concluída.
 - **Realizada** (fulfilled): operação foi concluída.
 - **Rejeitada** (rejected): operação assíncrona falhou.
- **Fetch API** é um exemplo de Promise, permitindo realizar requisições assíncronas com o uso dos métodos **then** e **catch**, sendo uma alternativa ao **XMLHttpRequest**, com sintaxe mais simples.

```
new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('Sucesso!');
    }, 5000);
}).then((resultado) => {
    console.log(resultado);
}).catch((erro) => {
    console.error(erro);
});
```

Requisições assíncronas com Promises

```
<button type="button">Carregar texto</button>
<p id="texto">0 texto será carregado aqui</p>
<script>
  let xhr = new XMLHttpRequest();
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    let texto = document.querySelector("#texto");
    xhr.open("GET", "http://exemplo.com/exemplo.txt");
    xhr.addEventListener('readystatechange', function () {
      if (xhr.readyState == 4 && xhr.status == 200) {
        texto.innerHTML = xhr.responseText;
      }
    });
    xhr.send();
  });
</script>
```

```
<button type="button">Carregar texto</button>
<p id="texto">0 texto será carregado aqui</p>
<script>
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    let texto =
    document.querySelector("#texto");
    let url =
    "http://exemplo.com/exemplo.txt";
    fetch(url).then(resposta => {
      texto.innerHTML = resposta;
    });
  });
</script>
```

Requisições assíncronas com Promises

```
<button type="button">Carregar texto</button>
<p id="texto">0 texto será carregado aqui</p>
<script>
  let xhr = new XMLHttpRequest();
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    let texto = document.querySelector("#texto");
    xhr.open("GET", "http://exemplo.com/exemplo.txt");
    xhr.addEventListener('readystatechange', function () {
      if (xhr.readyState == 4 && xhr.status == 200) {
        texto.innerHTML = xhr.responseText;
      }
    });
    xhr.send();
  });
</script>
```

```
<button type="button">Carregar texto</button>
<p id="texto">0 texto será carregado aqui</p>
<script>
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    let texto =
    document.querySelector("#texto");
    let url =
    "http://exemplo.com/exemplo.txt";
    fetch(url).then(resposta => {
      texto.innerHTML = resposta;
    });
  });
</script>
```

Referências

- DUCKETT, Jon. **HTML e CSS**: projete e construa websites. 1. ed. Alta Books, 2016. 512 p.
- DUCKETT, Jon. **Javascript e JQuery**: desenvolvimento de interfaces web interativas. 1. ed. Alta Books, 2016. 640 p.
- MOZILLA (ed.). **MDN Web Docs**: Aprendendo desenvolvimento web. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn>.
- W3SCHOOLS (ed.). **W3Schools Online Web Tutorials**. 2023. Disponível em: <https://www.w3schools.com/>.



Web Academy



Obrigado!