



# WEB ACADEMY

## Fundamentos de Programação Front-end

Daniel Augusto Nunes da Silva

# **Apresentação**



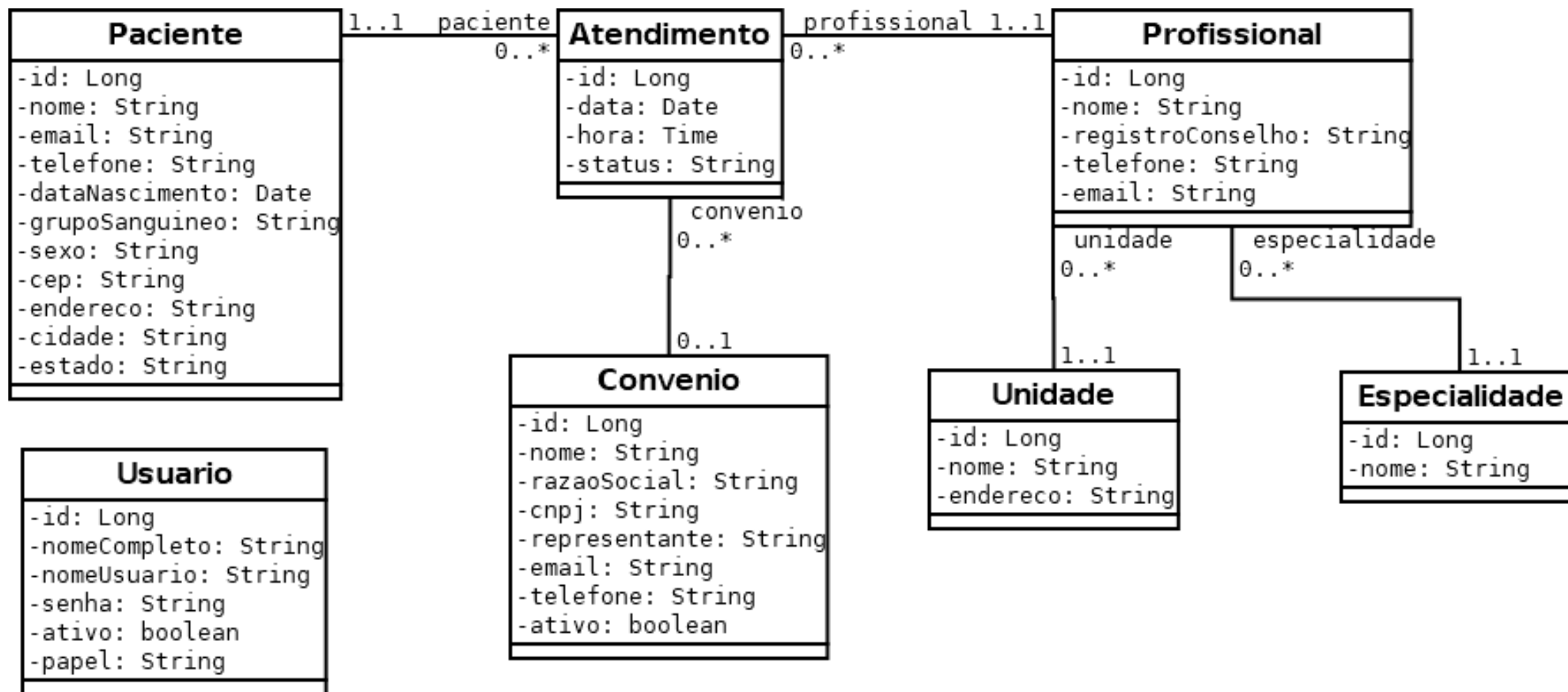
# Ementa

- Fundamentos de **HMTL**. Padrões e recomendações da **W3C**, semântica e acessibilidade. Estilização de páginas HTML com **CSS**. Técnicas de design responsivo. Tipos de dados, funções, objetos, *arrays* e manipulação de eventos em **JavaScript**. Manipulação de **DOM** (Document Object Model). **JSON** (JavaScript Object Notation). 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.

# SGCM – Diagrama de Classes



# 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 e layout de documentos 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



**HTML e CSS: projete e construa websites.**

Jon Duckett

1ª Edição – 2016

Editora Alta Books

ISBN 9788576089391



**JavaScript e JQuery: desenvolvimento de interfaces web interativas.**

Jon Duckett

1ª Edição – 2016

Editora Alta Books

ISBN 9781118871652

# 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 (Ctrl+Shift+I)**

# Contato

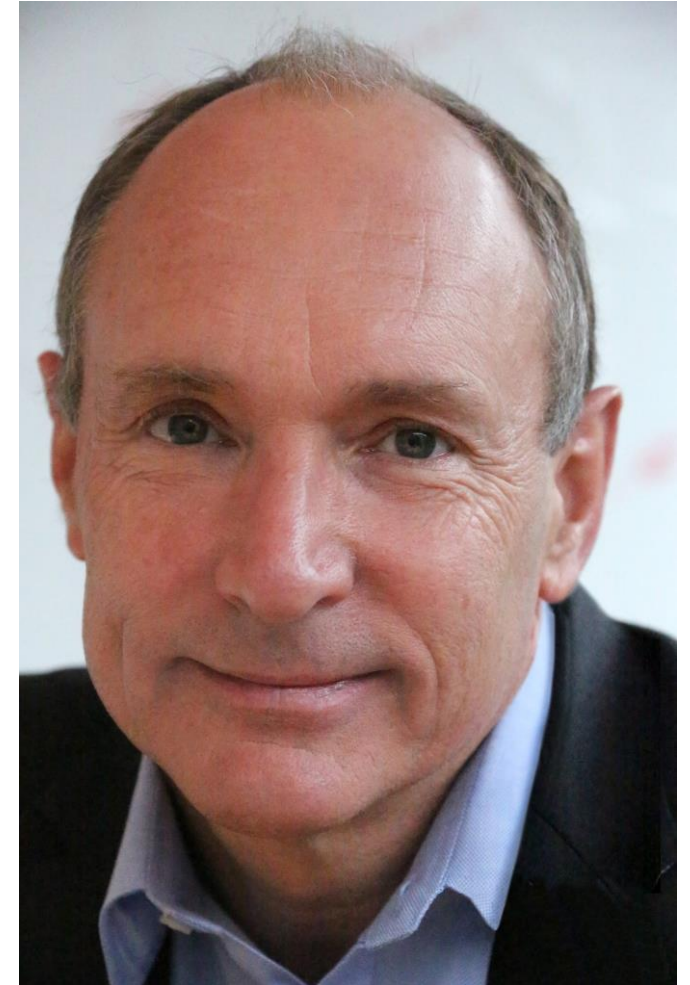


<https://linkme.bio/danielnsilva/>

# Introdução

# O protocolo HTTP, HTML e a Web

- 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 e qualquer tipo de conteúdo multimídia.



# O protocolo HTTP, HTML e a Web

- Com base no protocolo **TCP/IP** surgiu a ideia de transmitir o conteúdo hipertexto pela rede.
- Para isso foi criado o protocolo **HTTP** (protocolo de transferência de hipertextos).
- Foi necessária ainda criar uma linguagem para criação de conteúdo hipertexto, o **HTML**.
- E além disso foi criado o conceito **World Wide Web** (www) que engloba todos os serviços de conteúdo multimídia baseados no protocolo **HTTP**.





# O protocolo HTTP, HTML e a Web

- Primeiro site criado com a linguagem HTML pra funcionar sob o protocolo HTTP:
  - <http://info.cern.ch/hypertext/WWW/TheProject.html>

## 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 do HTML

- **1990:** HTML versão 1;
- **1995:** HTML versão 2, cujo desenvolvimento se deu em colaboração com várias empresas, incluindo fabricantes de navegadores (Problema: **falta de padrão**);
- **1995:** HTML versão 3, sob responsabilidade do **W3C** (World Wide Web Consortium), trazendo **padronização**.
- **1997:** HTML versão 4, que se tornou muito popular e foi bastante utilizada por anos;
- **2004:** O W3C propõe acabar com a linguagem HTML e substituí-lo por uma linguagem baseada na tecnologia XML, surgindo assim o **XHTML** versão 1.

# Evolução do HTML

- Contrários a decisão do W3C, Firefox, Opera e Safari criaram o **WHATWG** para continuar o desenvolvimento do HTML;
- A W3C ainda lançou, em 2007, o XHTML 2.0, mas o WHATWG já tinha uma proposta de nova versão do HTML;
- O W3C aceitou a proposta e desistiu do XHTML.



# Evolução do HTML

- Tem início em 2008 o projeto do **HTML 5** com o apoio da W3C;
- 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**: linguagem que define o layout de documentos HTML;
  - **JavaScript**: linguagem de programação que roda no lado cliente (navegador).

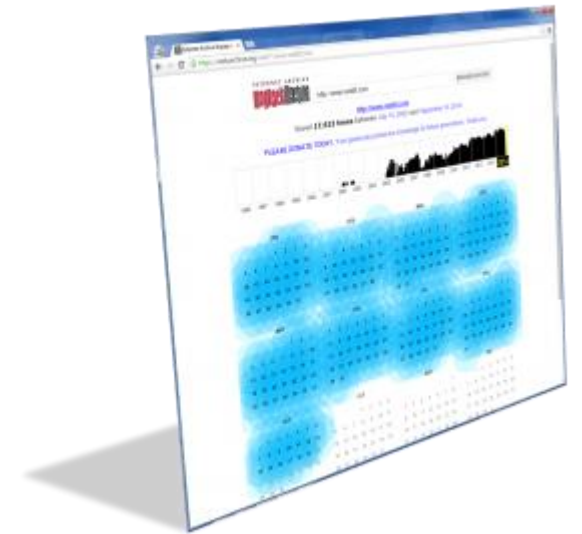




# Conhecem o Wayback Machine?

- 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/>

# WayBackMachine



## Site do UOL em 1996



# Globo.com em 2000

The screenshot shows the Globo.com website interface from 2000. At the top, a blue header bar contains the date and time (Segunda-feira, 25 de Fevereiro de 2019, 13:55), navigation links (Chat, E-mail, Compras), and financial indicators (Ibovespa 14.581,52 -2,08%, Dólar Com 1,81 0,16%). Below this is a search bar and a dropdown menu for 'no Portal'. The main content area is divided into several sections: a left sidebar with a menu of categories (Busca, Chat, Cinema, Compras, Diversão e Arte, Economia, Esportes, Música, Notícias, Saúde, Seguros, Televisão, Tempo), a central area with a large video player and two news articles (Pete Townshend vende raridades em leilão virtual, Malan descarta dólar como moeda única do Mercosul), and a right sidebar with a chat window (GloboChat), a video player (Globo.com Video), and a sports results section (RESULTADOS ESPORTIVOS). The bottom of the page features a yellow bar with 'Conexões GLOBO' and a login section for 'Nome do Usuário' and 'Senha'.

Segunda-feira, 25 de Fevereiro de 2019 13:55

INDICADORES FINANCEIROS

Ibovespa 14.581,52 -2,08%

Dólar Com 1,81 0,16%

Chat | E-mail | Compras

Busca no Portal Canais

Globo.com

SUL AMERICA SEGUROS

por muito menos do que você imagina.

1º Brasileiro de Histórias do Futebol

Busca

Chat

Cinema

Compras

Diversão e Arte

Economia

Esportes

Música

Notícias

Saúde

Seguros

Televisão

Tempo

Conexões GLOBO

Publicações

Rádios

Televisão

O G L O B O

MÚSICA

ECONOMIA

Pete Townshend vende raridades em leilão virtual

Compositor venderá algumas de suas guitarras históricas, além de cartas e discos de ouro do The Who, através da Internet.

Malan descarta dólar como moeda única do Mercosul

Ministro das Relações Exteriores da Argentina discorda, mas ressalta que dolarização abrupta não seria conveniente.

MAIS DESTAQUES

Globo.com Video

GloboChat

Escolha o apelido:

Escolha a sala:

Salas

Entrar na sala

Jornal Nacional

Jornal Nacional exhibe provas de envolvimento de brasileiros no narcotráfico internacional

elefante

Clique Aqui

RESULTADOS ESPORTIVOS

FUTEBOL

Corinthians (BRA) 3

Rosario Central (ARG) 2

Nome do Usuário

Senha

OK!

# UFAC em 2000



A Universidade  
Reitoria  
Pró-Reitorias  
Departamentos/  
Cursos

Cursos de Mestrado e  
Especialização

Biblioteca  
Colégio de Aplicação  
Informações

Melhor Visualizado com  
800x600

**Fc** 466814  
Desde 23.06.2000  
Atualizado em 04 / 12 / 2000.



Por uma Universidade  
Verdadeiramente Amazônica

Universidade Federal do Acre

**un**

Editais & Concursos

**NOVO!** **Atenção Candidatos ao Vestibular 2001.** Informações sobre o Concurso [aqui](#).

**NOVO!** **Outros Editais**

Novidades

**GED 2000 NOVO!**  
Atenção Professores. Estão disponíveis os formulários da Gratificação de Estimulo à Docência. [Confira](#).

Links Úteis

Página Inicial dos Links  
Órgãos do Governo  
Universidades Brasileiras  
Busca na Web

Downloads

Diretório do Grupo de Pesquisas no Brasil - versão 4.0 /CNPq



[GABARITO DO VESTIBULAR](#)

**Veja Mais »»»**

- » **Calendário Acadêmico**
- » **Email's Úteis**
- » **Projetos na Ufac**
- » **Fotos da Ufac**
- » **Eventos**

**PRODOC**  
Produção de Docentes

**POP-AC**



**CANDIDATOS**  
**15'000 VAGAS**  
**EMPRESAS**  
**65'000**  
**Currículos**  
**A CUSTO ZERO !**



Segunda-Feira, 4 de Dezembro de 2000. Boa Tarde! 13:44:00



**Fundação BIOMA**  
**SETEM - Física do Clima**

Universidade Federal do Acre - UFAC  
BR 364 Km 04 Cep: 69915-900 Bairro Distrito Industrial  
Caixa Postal 500 Rio Branco - Acre  
PABX: (0xx68) 229-2244  
Design by [webmaster@ufac.br](mailto:webmaster@ufac.br)  
(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.



# O que são os padrões web?

- Apesar de existirem **órgãos normatizadores**, como o **ISO Standards** e **ECMA**, normalmente quando discutimos padrões web nos referimos aos padrões do **W3C**.
- 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;
- Compatibilidade futura;
- Garantia de funcionamento completo da página;
- Páginas com melhor aspecto de apresentação;
- Comportamento uniforme entre diferentes navegadores de internet.

# Acessibilidade na web

- **Acessibilidade** na web significa permitir que o maior número de pessoas possível pode 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 que possuem barreiras de acessibilidade que dificultam ou mesmo tornam impossível o acesso.

# Exemplos de barreiras ao acessar o conteúdo

- 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.
- Textos apresentados como imagens, porque não quebram as linhas quando ampliadas.



# 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** e suas recomendações.
- As Diretrizes de Acessibilidade para Conteúdo Web (WCAG) abrangem um vasto conjunto de recomendações que **têm como objetivo tornar o conteúdo Web mais acessível.**

# Design responsivo

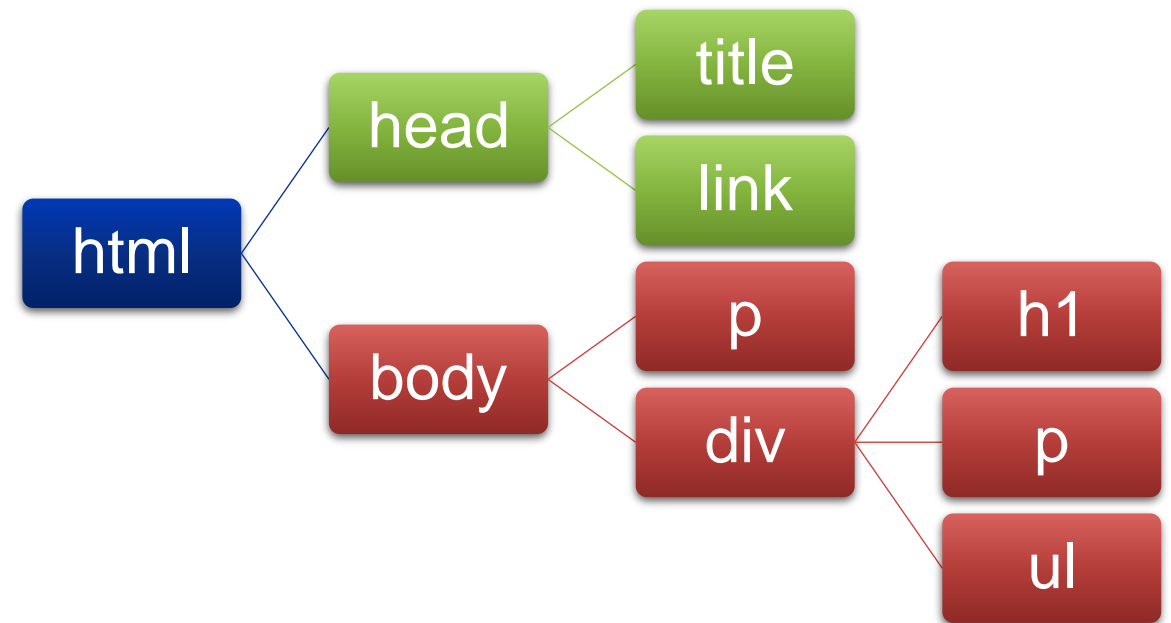
- **Design responsivo** é um conceito que permite que uma aplicação web forneça um **conteúdo acessível** e uma **experiência de usuário otimizada**, respeitadas as limitações, **independente do dispositivo** que está sendo utilizado.
- Não se trata de criar uma versão para cada tipo de dispositivo.



# HTML

# Introdução ao HTML

- O **HTML** é uma linguagem interpretada pelo navegador para a 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 são definidas usando a sintaxe: **<nomedatag>**

## Exemplos de tags

- |                      |                    |
|----------------------|--------------------|
| ▪ <html> </html>     | ▪ <h1> </h1>       |
| ▪ <head> </head>     | ▪ <p> </p>         |
| ▪ <meta>             | ▪ <a> </a>         |
| ▪ <script> </script> | ▪ <img>            |
| ▪ <title> </title>   | ▪ <table> </table> |
| ▪ <body> </body>     | ▪ <br>             |

# Tags (elementos)

- Alguns elementos HTML são classificados como **normal elements**, que são abertos com uma tag e fechados com outra tag.
- Exemplo:

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

- Há também os chamados **void elements**, que não possuem conteúdo, sendo abertos e fechados com apenas uma tag.
- Exemplo:

```

```



# Estrutura 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 do HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <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">**

# Títulos

- 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>`

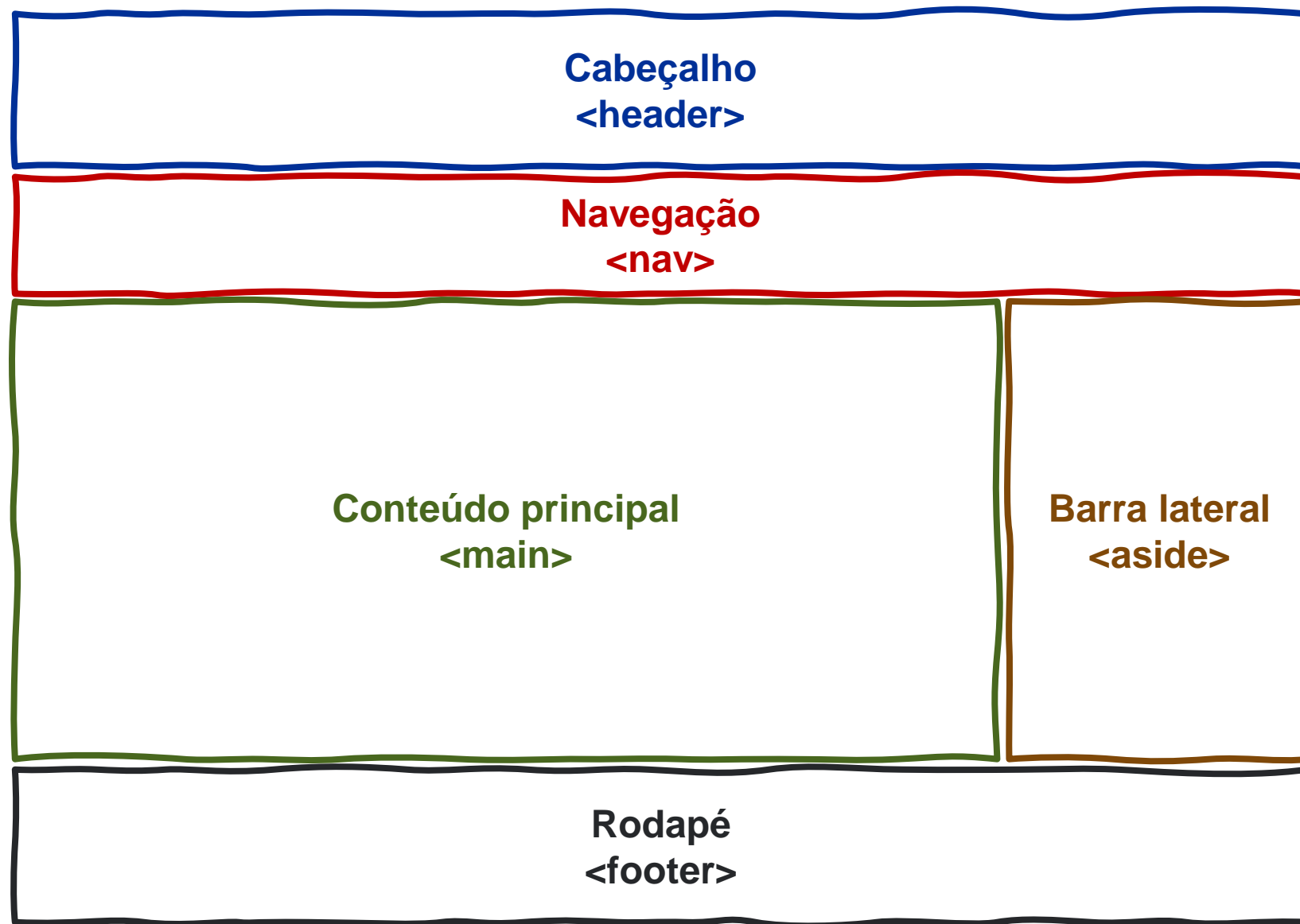
# Títulos

- 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.

# Estrutura e layout



# Elementos genéricos

- **<div>** e **<span>** são elementos genéricos que não representam nenhum 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** e **<span>** de **nível de linha**.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Título</h1>
    <div>
      <h2>Exemplo</h2>
      <p>
        Parágrafo 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:
  - **<ul>** cria listas não ordenadas;
  - **<ol>** cria listas ordenadas;
  - **<li>** cria itens para as listas.

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

## **Lista não ordenada:**

- Item A
- Item B
- Item C

## **Lista ordenada:**

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

# Imagens

- A tag **<img>** insere uma imagem, e possui dois atributos obrigatórios:
  - **src**: indica a URL para o arquivo;
  - **alt**: define um texto alternativo caso a imagem não seja carregada.
- No HTML 5:
  - **<figure>**: especifica conteúdo como ilustrações, diagramas, fotos, etc.;
  - **<figcaption>**: define uma legenda.

```
<figure>
    
    <figcaption>
        Legenda da foto.
    </figcaption>
</figure>
```

# Links

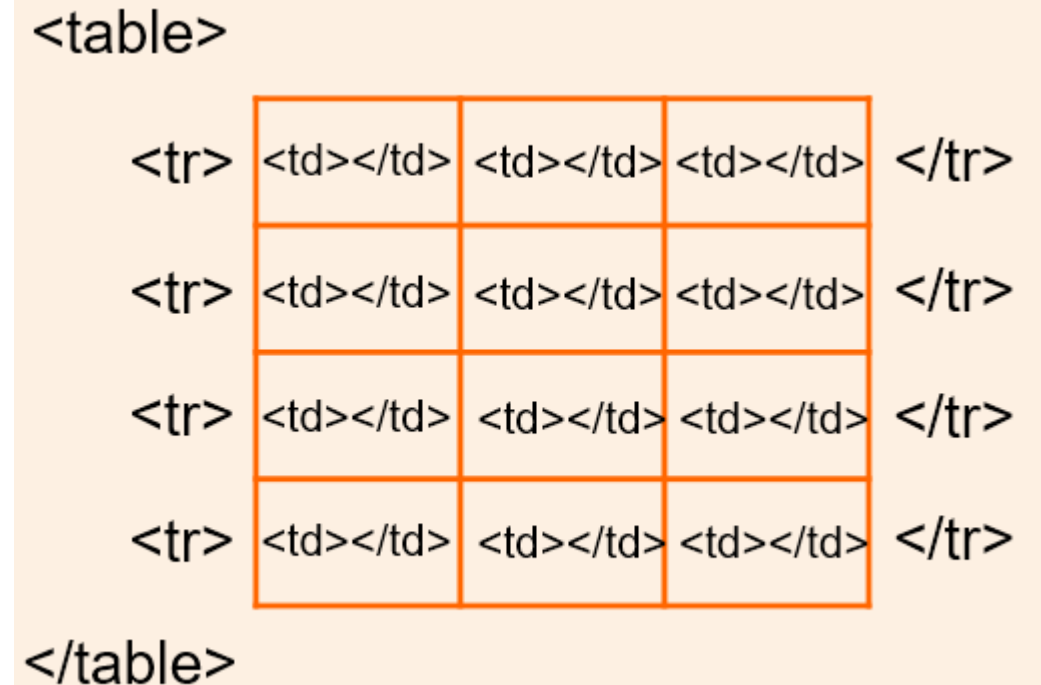
- As ligações entre páginas (hiperlinks ou simplesmente link) são definidas pela tag `<a>`;
- O atributo principal **href** especifica a URL da página de destino;
- O links podem ser criados sobre conteúdo de texto simples ou vários outros tipos de elementos HTML, como imagens, títulos, etc.

```
<a href="http://webacademy.ufac.br">  
    WEB ACADEMY  
</a>  
  
<a href="http://webacademy.ufac.br">  
    <figure>  
          
        <figcaption>  
            WEB ACADEMY  
        </figcaption>  
    </figure>  
</a>
```

# Tabelas

- Uma tabela é definida não apenas por uma tag, mas por 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
<code>&lt;table&gt;</code>	Define uma tabela
<code>&lt;tr&gt;</code>	Insere uma linha na tabela
<code>&lt;td&gt;</code>	Insere uma célula dentro de um elemento <code>&lt;tr&gt;</code>
<code>&lt;th&gt;</code>	Insere uma célula (cabeçalho) dentro de um elemento <code>&lt;tr&gt;</code>
<code>&lt;caption&gt;</code>	Atribui um título ou descrição para a tabela
<code>&lt;colgroup&gt;</code>	Especifica um grupo de colunas para formatação
<code>&lt;col&gt;</code>	Define propriedades da coluna para cada elemento dentro do <code>&lt;colgroup&gt;</code>
<code>&lt;thead&gt;</code>	Define o cabeçalho da tabela
<code>&lt;tbody&gt;</code>	Define o corpo (conteúdo principal) da tabela
<code>&lt;tfoot&gt;</code>	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>

```

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

# Formulários

- Um formulário serve basicamente para enviar informações;
- A tag **<form>** define, dentre outras coisas, a página que 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, 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](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#input_types)

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

Nome:

Sobrenome:



**CSS**

# Introdução ao CSS

- As **folhas de estilo em cascata** (*Cascading Style Sheets* – **CSS**) descrevem a apresentação de um documento HTML, isto é, como os elementos devem ser exibidos;
- Foi criado para preencher uma lacuna deixado 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**.

# Introdução ao CSS

Versões antigas do HTML  
(vários atributos para cada tag)

```
<body bgcolor="blue">
```

Utilização recomendada  
(atributo style)

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

- As declarações CSS acima produzem o mesmo efeito;
- A principal diferença é que o CSS permite outras formas de organizar as declarações que tratam da formatação do documento.

# 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;
  3. Ou utilizando um **arquivo externo** é possível mudar a apresentação (estilo) de toda aplicação ou site com um único arquivo.

# Formas de aplicação do CSS

## Externo

```
<!DOCTYPE html>
<html>
  <head>
    <link
      rel="stylesheet"
      type="text/css"
      href="estilo.css">
  </head>
  <body>
    <h1>Isto é um título</h1>
    <p>Isto é um parágrafo.</p>
  </body>
</html>
```

## Interno

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        background-color: red;
      }
    </style>
  </head>
  <body>
    <h1>Isto é um título</h1>
    <p>Isto é um parágrafo.</p>
  </body>
</html>
```

## Inline

```
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color:blue">
      Isto é um título
    </h1>
    <p>
      Isto é um parágrafo.
    </p>
  </body>
</html>
```

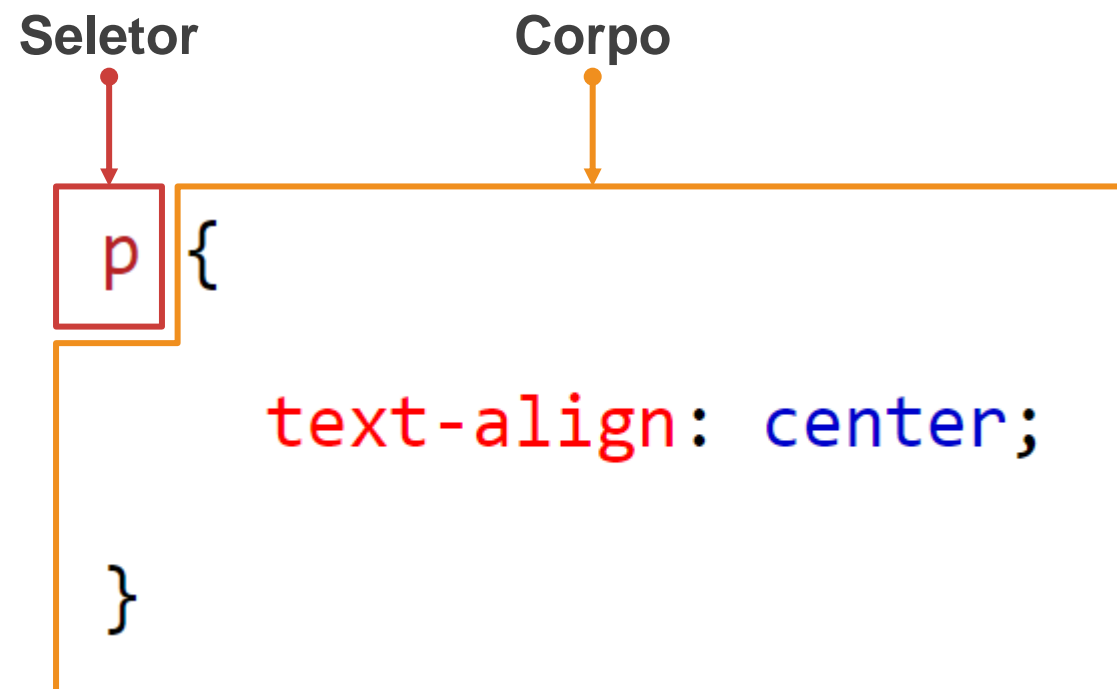
# Sintaxe

- As declarações **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.

```
p {  
    text-align: center;  
}
```

# Sintaxe

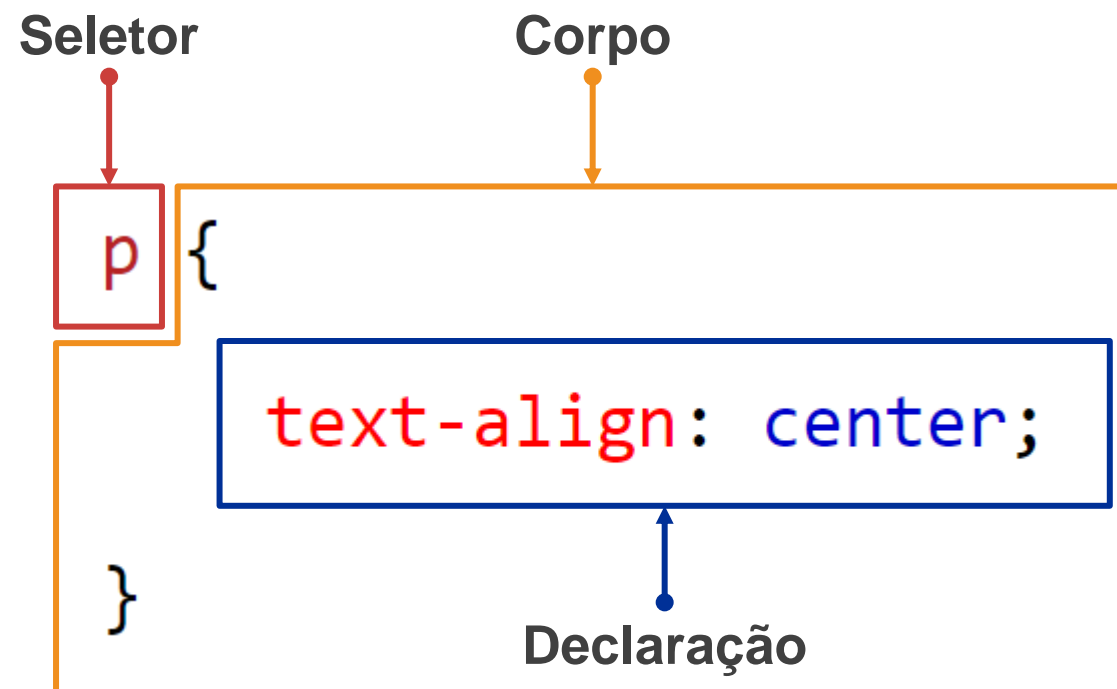
- As declarações **CSS** possuem uma **sintaxe muito simples**;
- Consiste na **propriedade** seguida do seu **valor**, separados pelo sinal de dois pontos (":");
- Para separar várias propriedades usamos o ponto-e-vírgula.





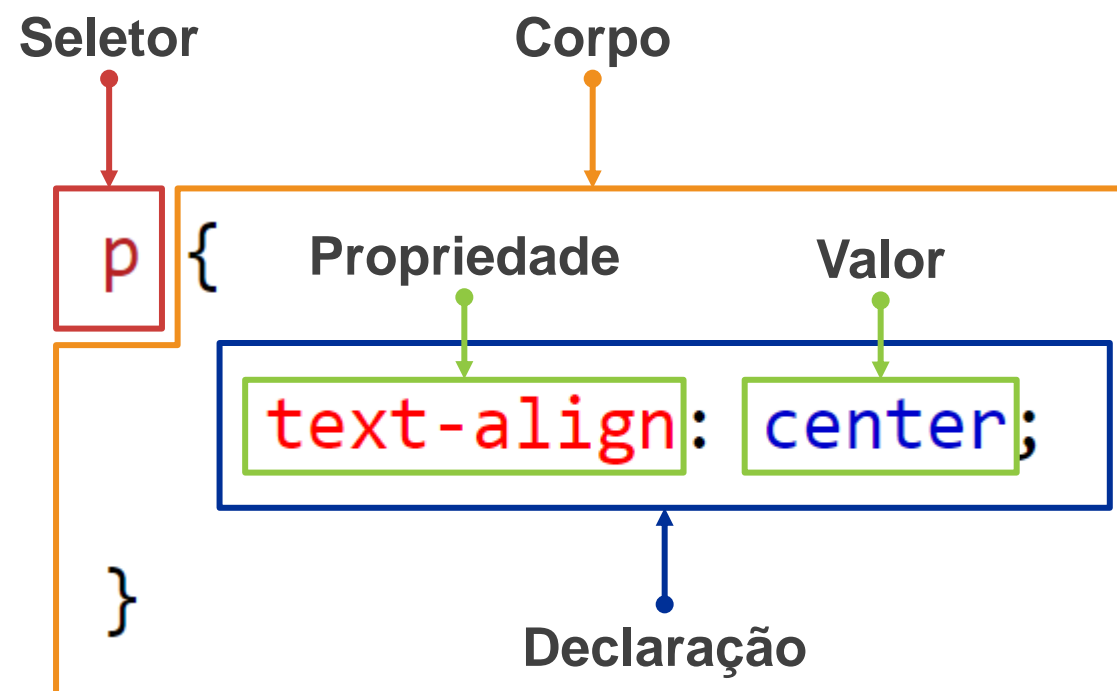
# Sintaxe

- As declarações **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.



# Sintaxe

- As declarações **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):
  - Seletor **universal**: seleciona todos os elementos;
  - Seletor de **elemento (tipo)**: seleciona elementos com base no nome do elemento;
  - Seletor de **classe**: seleciona elementos com um atributo de classe específico;
  - Seletor de **ID**: usa o atributo ID para selecionar um elemento específico;
  - Seletor de **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](https://developer.mozilla.org/pt-BR/docs/Learn/CSS/Building_blocks/Selectors)

# Seletores

```
<!DOCTYPE html>
<html>
  <head>
    <title>Web Academy</title>
  </head>
  <body>
    <a href="http://www.google.com"
      id="link_1"
      class="link">
      Web Academy
    </a>
  </body>
</html>
```

```
* { /* Universal */
  color: red;
}

a { /* Elemento */
  color: red;
}

.link { /* Classe */
  color: red;
}


#link_1 { /* ID */
  color: red;
}

a[href="http://www.google.com"] { /* Atributo */
  color: red;
}
```

# Agrupamento de seletores

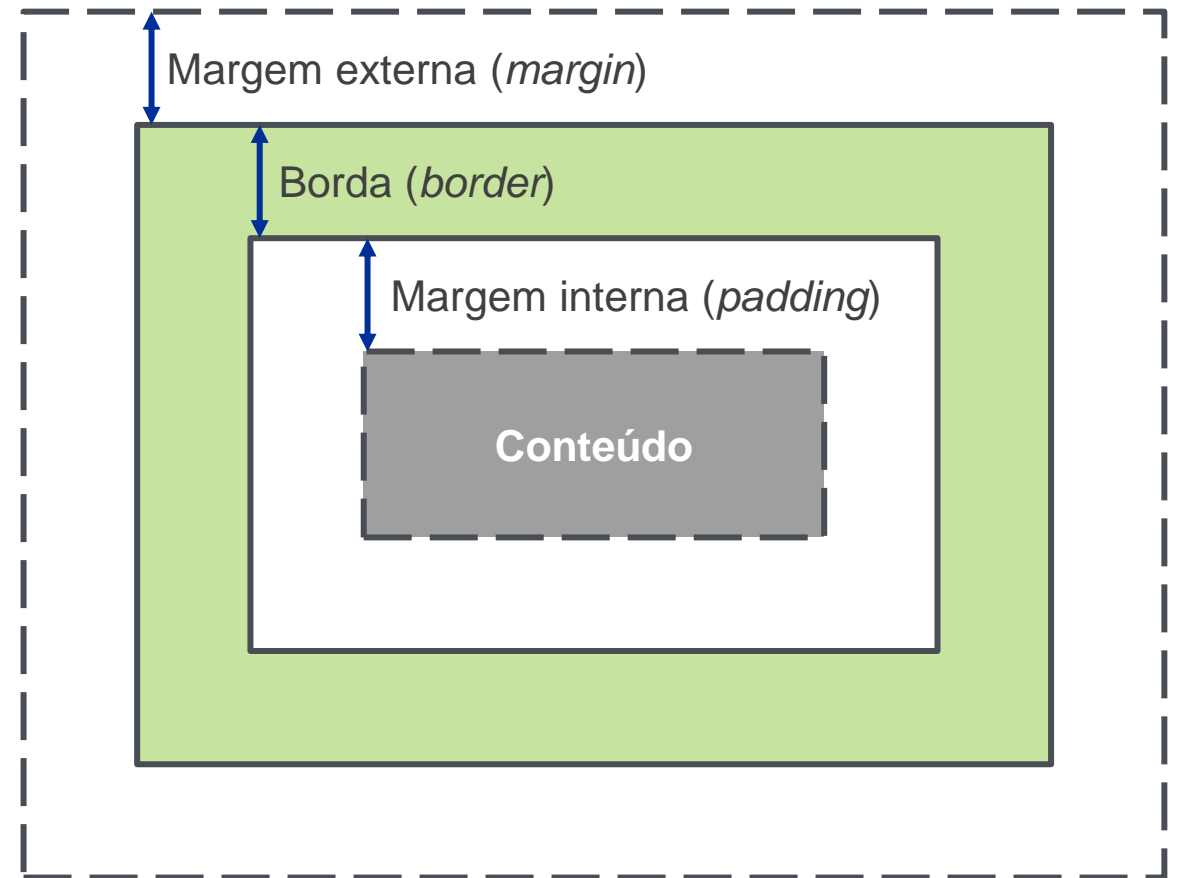
- É possível agrupar seletores, separados por vírgula, aplicando a mesma formatação para vários tipos de elementos.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p, h1, h2 {
        text-align: center;
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Isto é um título</h1>
    <h2>Isto é um subtítulo</h2>
    <p>Isto é um parágrafo</p>
    <p id="p1">Outro parágrafo.</p>
    <p class="p1">Terceiro parágrafo.</p>
  </body>
</html>
```



# 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.



# Box Model

block-level

**Justiça referenda zona de amortecimento e outras normas para proteger floresta urbana em São Paulo**

**Ampla articulação desencadeada por ação civil pública do MPF possibilitou elaboração de normas para uso sustentável da Mata de Santa Genebra em Campinas (SP)**

A 3ª Turma do Tribunal Regional Federal (TRF3) referendou, por unanimidade, um conjunto de normas para proteger a Mata de Santa Genebra, localizada na divisa dos municípios de Campinas (SP) e Paulínia (SP). A decisão, de 20 de fevereiro, foi disponibilizada no diário eletrônico de 27 de fevereiro. Remanescente da Mata Atlântica, com 252 hectares de área, essa floresta urbana foi declarada Área de Relevante Interesse Ecológico (Arie) em 1985 e sua proteção e uso sustentável foram viabilizados por meio de ampla articulação do **Ministério Público Federal** (MPF).

[Leia mais...](#)

inline-level

# Aplicação do CSS: Textos

- Há muitas propriedades para formatação de texto em CSS, dentre as quais destaca-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: 2em;
        text-align: center;
        text-decoration: underline;
      }
      p { color: red; font-size: 15px; }
      p.negrito { font-weight: bold; }
    </style>
  </head>
  <body>
    <h1>Isto é um título</h1>
    <p>Isto é um parágrafo.</p>
    <p class="negrito">Outro parágrafo.</p>
    <p>Terceiro parágrafo.</p>
  </body>
</html>
```

Isto é um título

Isto é um parágrafo.

**Outro parágrafo.**

Terceiro parágrafo.

# 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).

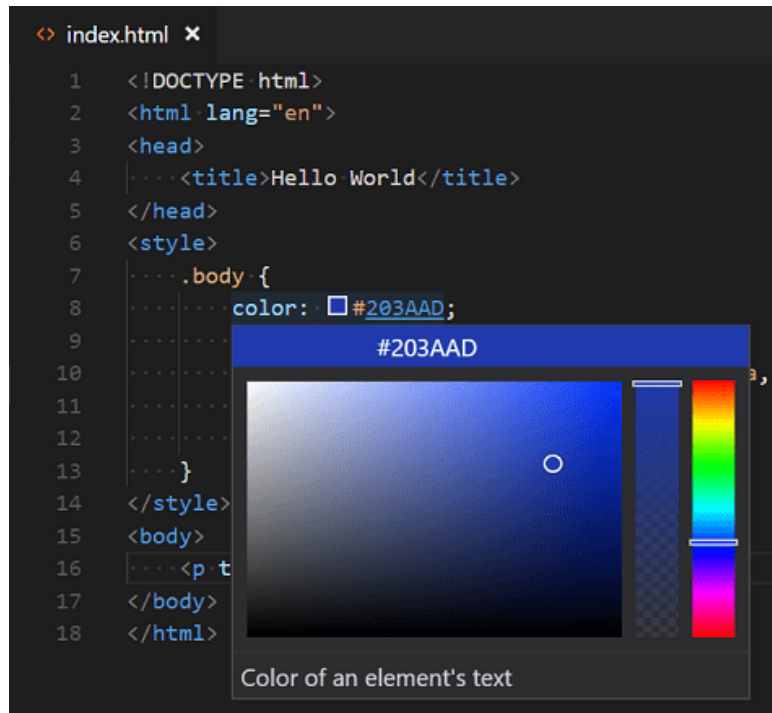
# Aplicação do CSS: Cores

- 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](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>
```

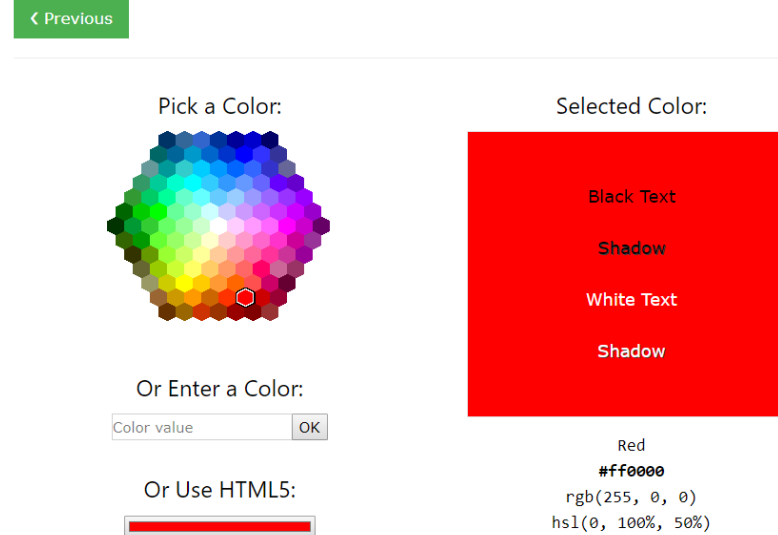
# Aplicação do CSS: Seletor de cores

## VS Code



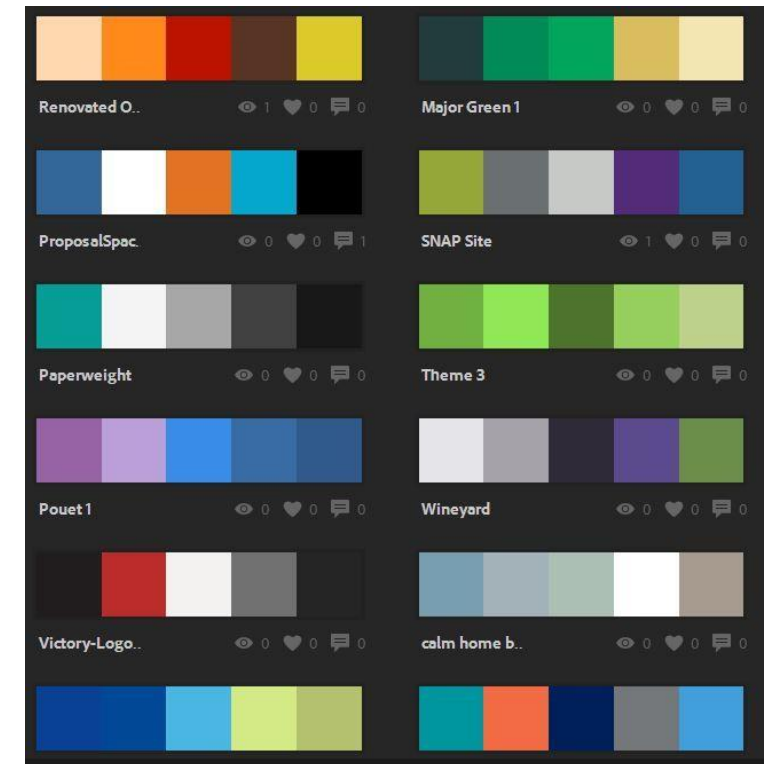
## W3Schools

### HTML Color Picker



[https://www.w3schools.com/colors/colors\\_picker.asp](https://www.w3schools.com/colors/colors_picker.asp)

## Adobe Color Wheel



<https://color.adobe.com/pt/create/color-wheel/>

# Pseudo-classes e pseudo-elementos

- Uma **pseudo-classe** permite aos seletores **especificar estados** de um elemento.

- Sintaxe:

**seletor:pseudo-classe**

```
/* Altera a cor da fonte quando  
   o cursor passa sobre o link */  
a:hover {  
    color: #003366;  
}
```

- Um **pseudo-elemento** permite aos seletores **especificar uma parte** de um elemento.

- Sintaxe:

**seletor::pseudo-elemento**

```
/* 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 demais elementos internos que não tem cor definida. **Não é aplicável para algumas propriedades.**

## Especificidade

- Em CSS **o seletor mais específico prevalece.** O peso é definido por valores atribuídos pela quantidade de ID, Classe e Tipo, onde o 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: 1
  - Classe, pseudo-classe ou atributo: 10
  - ID: 100
- Estilo inline (atributo *style*): 1000

```
* { /* 0-0-0 */  
  color: red;  
}  
  
a { /* 0-0-1 */  
  color: blue;  
}  
  
.link { /* 0-1-0 */  
  color: green;  
}  
  
#link_1 { /* 1-0-0 */  
  color: yellow;  
}
```



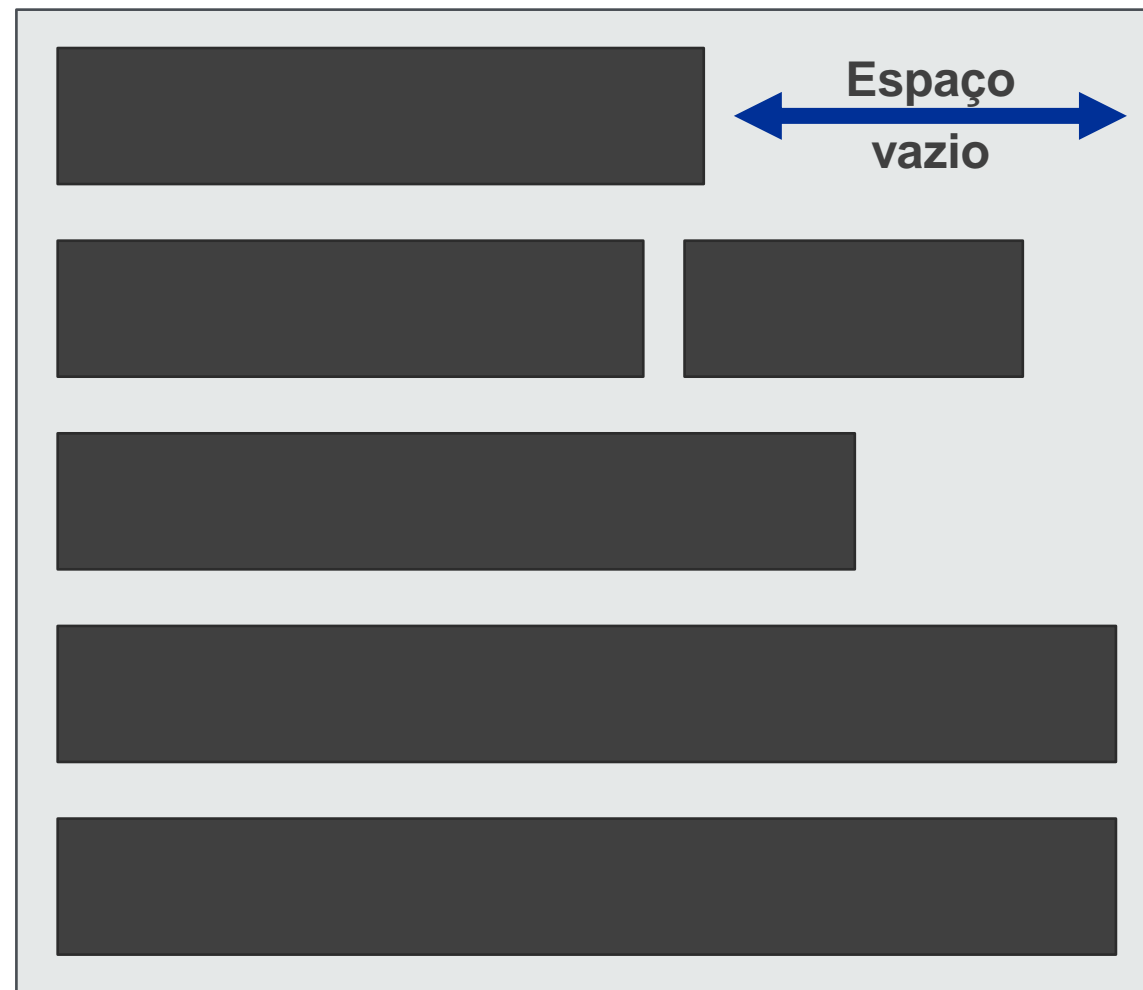
# Cálculo de especificidade

- Para calcular a especificidade em CSS, é necessário atribuir um valor a cada tipo de seletor:
  - Universal (\*): 0
  - Elemento ou pseudo-elemento: 1
  - Classe, pseudo-classe ou atributo: 10
  - ID: 100
- Estilo inline (atributo *style*): 1000

```
a[href="http://www.google.com"] { /* 0-1-1 */  
    color: black;  
}  
  
p a { /* 0-0-2 */  
    color: orange;  
}  
  
p a#link_1 { /* 1-0-2 */  
    color: grey;  
}  
  
p a.link { /* 0-1-2 */  
    color: navy;  
}
```

# 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 acima do outro, e os elementos em nível de linha são mostrados lado a lado.



# Layout: 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;  
    }  
  
}
```

# JavaScript

# Introdução ao JavaScript

- Considerando as 3 principais tecnologias do lado cliente (front-end), **JavaScript** complementa o **HTML** e **CSS** com recursos de uma **linguagem de programação**;
- Sendo o foco do HTML o **conteúdo** e o foco do CSS a **apresentação**, o restante fica por conta do 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();
```



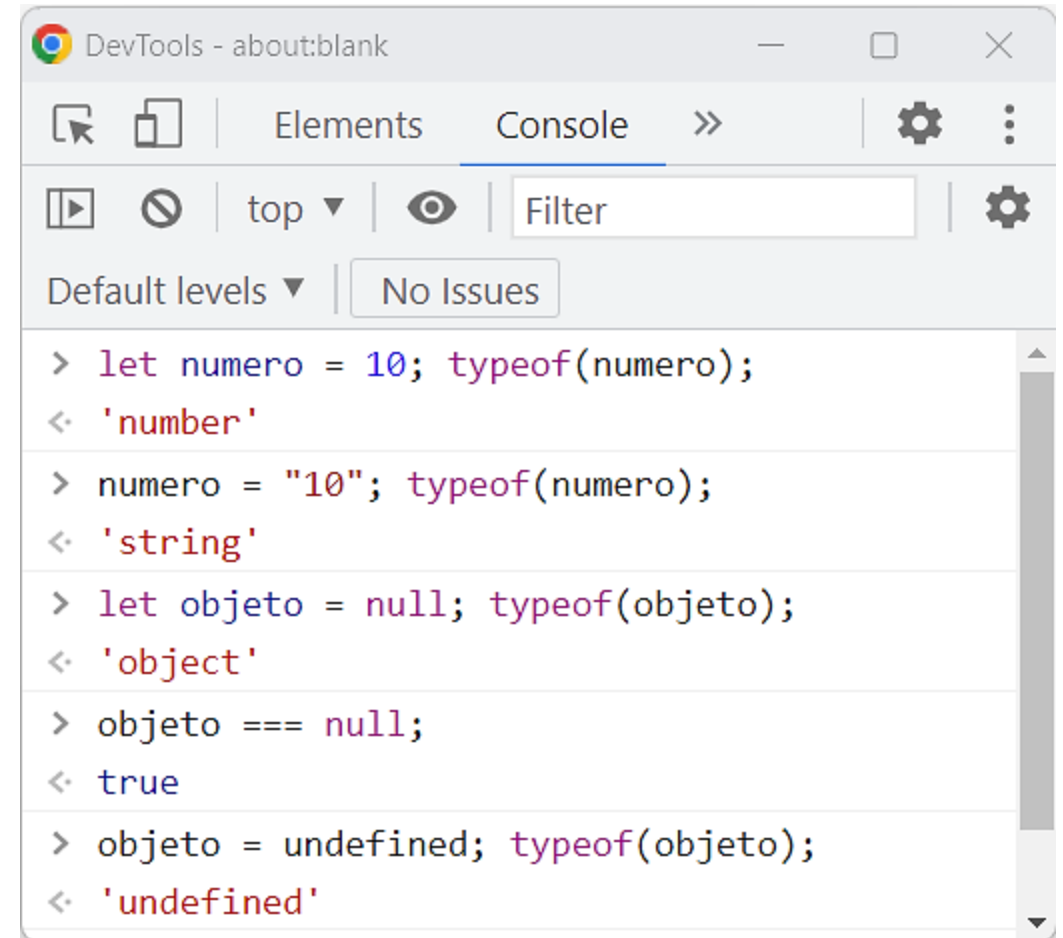
# *var, let e const*

- 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!**
- **let** funciona de forma diferente, 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;
```

# Tipos de dados

- JavaScript possui tipagem **fraca** e **dinâmica**.
- **Tipos primitivos:** String, Number, BigInt, Boolean, Symbol, Null, Undefined e Object.
  - **undefined:** variável não teve valor atribuído.
  - **null:** ausência intencional de valor.



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

```
> let numero = 10; typeof(numero);  
< 'number'  
  
> numero = "10"; typeof(numero);  
< 'string'  
  
> let objeto = null; typeof(objeto);  
< 'object'  
  
> objeto === null;  
< true  
  
> objeto = undefined; typeof(objeto);  
< 'undefined'
```

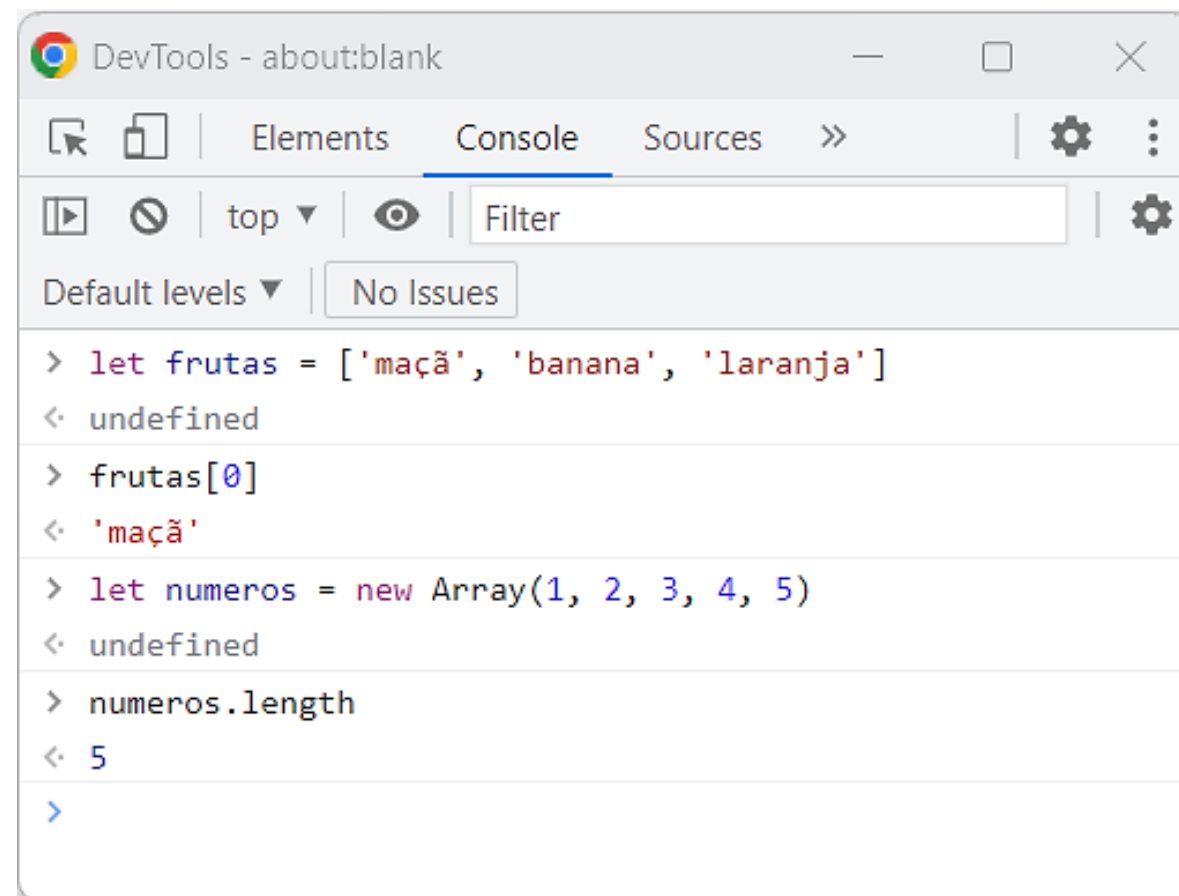
# Objetos

- 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();
```

# 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](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array)

A screenshot of the Chrome DevTools console window. The title bar says "DevTools - about:blank". The "Console" tab is selected. The console shows a series of commands and their results: 1. Command: `let frutas = ['maçã', 'banana', 'laranja'];` Result: `undefined`. 2. Command: `frutas[0];` Result: `'maçã'`. 3. Command: `let numeros = new Array(1, 2, 3, 4, 5);` Result: `undefined`. 4. Command: `numeros.length` Result: `5`. The console interface includes a filter input, a "No Issues" button, and a "Default levels" dropdown.

```
> let frutas = ['maçã', 'banana', 'laranja']
< undefined

> frutas[0]
< 'maçã'

> let numeros = new Array(1, 2, 3, 4, 5)
< undefined

> numeros.length
< 5

>
```

# 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'];

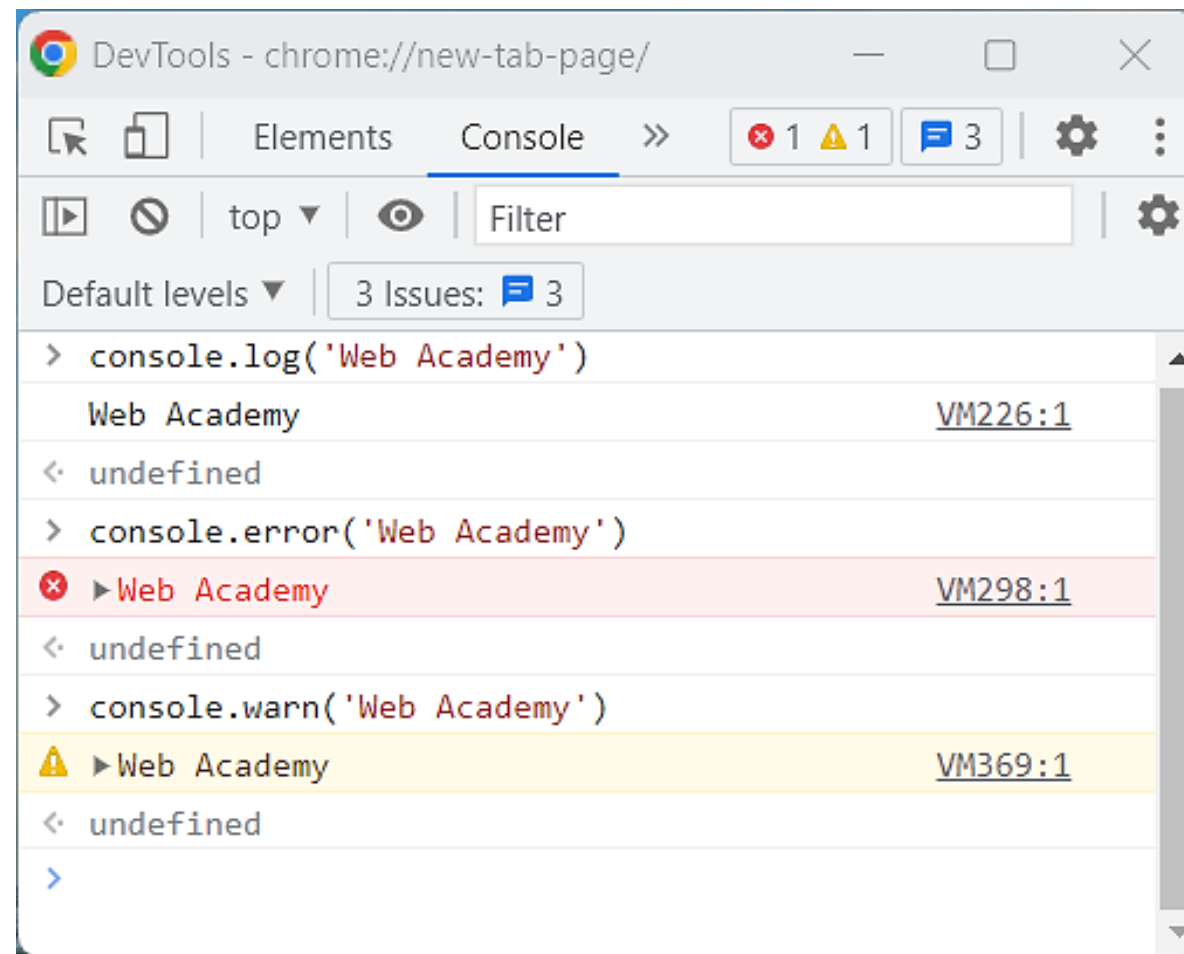
for (let i = 0; i < frutas.length; i++) {
  console.log(frutas[i]);
}

for (const fruta of frutas) {
  console.log(fruta);
}

frutas.forEach((fruta, indice, array) => {
  console.log(fruta);
});
```

# 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 utilização do 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](https://www.w3schools.com/tags/ref_eventattributes.asp)

# Formas de utilização do JavaScript

```
<!DOCTYPE html>
```

```
<head>
```

**Arquivo externo**

```
<script src="script.js" defer></script>
```

```
</head>
```

```
<body>
```

```
<script>
```

```
    alert('Olá mundo!');
```

```
</script>
```

```
<button type="button"
```

**Evento**

```
    onclick="alert('Olá mundo!')">
```

Clique aqui

```
</button>
```

```
</body>
```

```
</html>
```

Atrasa a execução do script



**Código interno**

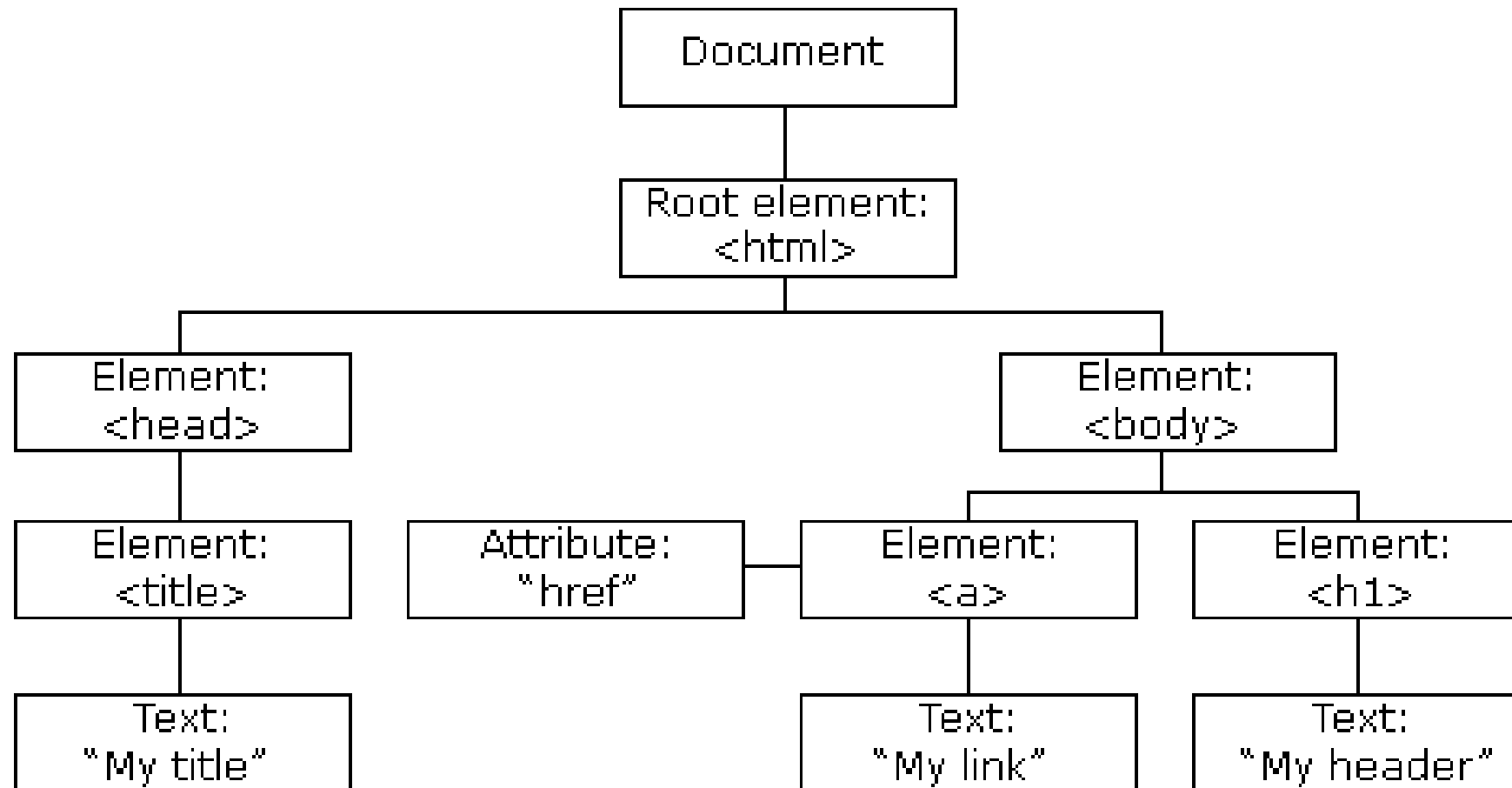




# DOM (Document Object Model)

- O **DOM** permite acessar, alterar, inserir e remover elementos em documento HTML, utilizando chamadas JavaScript;
- **Obedece a hierarquia do 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

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>
```

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

# DOM (Document Object Model)

- Outros métodos podem ser consultados em:
  - [https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp)
  - [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)

# 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>
```

- 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);
```

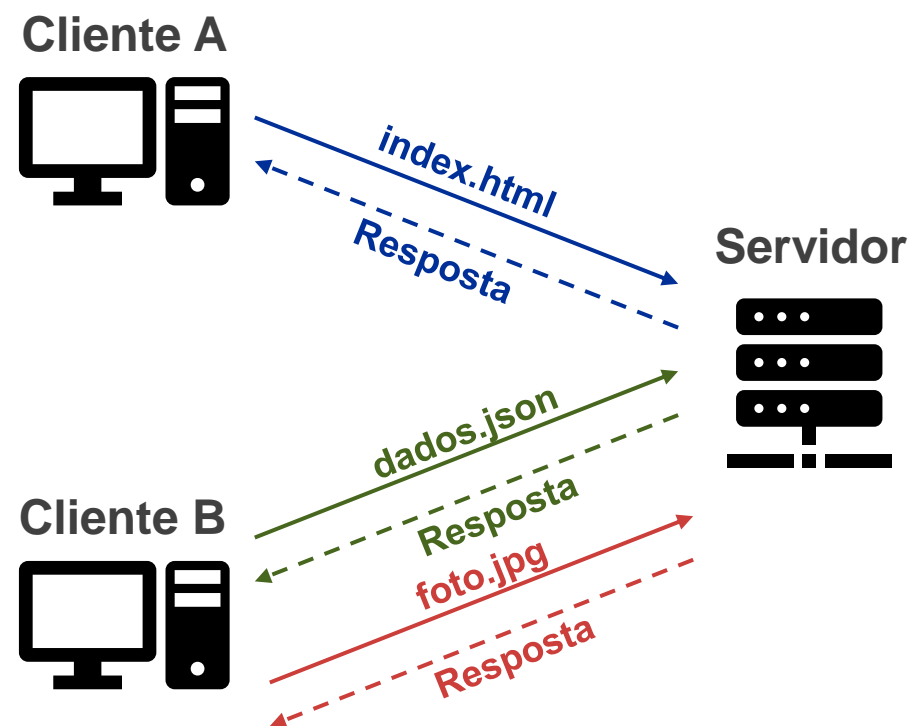
# JSON

- JSON (**J**ava**S**cript **O**bject **N**otation) 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” denominado **AJAX** (**A**synchronous **J**avaScript + **X**ML).
- 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">O 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>
```

**Fim!**



# Referências

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