



# PROGRAMAÇÃO PARA WEB

---

Andrea Konzen - Aula 02

# Professores

## **ANDREA KONZEN**

Professora Convidada

Formada em Ciência da Computação, com mestrado em Ciência da Computação na área de Inteligência Artificial pela (PUCRS) e doutorado em Informática na Educação na área de Inteligência Artificial com aplicação em Sistemas Educacionais pela (UFRGS). Além disso, possui pós-doutorado na área de Machine Learning em projetos voltados para exploração autônoma com múltiplos robôs. Atua como professora adjunta na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) e como coordenadora de projetos na área de Inteligência Artificial na Foreducation EdTech. É secretária regional da Sociedade Brasileira de Computação do Estado (SBC), pesquisadora de projetos na área de Inteligência Artificial com ênfase em Sistemas Inteligentes e Aprendizagem de Máquina aplicados em áreas como Saúde, Educação e Agricultura, com o propósito de reverter benefícios concretos e significativos para a sociedade.

## **LUIS FERNANDO PLANELLA GONZALEZ**

Professor PUCRS

Doutor Ciências da Computação (PUCRS, 2018). Desenvolvedor e arquiteto Java com experiência profissional desde 1999, certificado pela Sun como programador e desenvolvedor de componentes web na plataforma Java. Entusiasta de software livre.

# *Ementa da disciplina*

Estudo do desenvolvimento de aplicações com HTML, CSS e JavaScript.  
Estudo sobre Document Object Model (DOM). Utilização de forms em aplicações WEB.  
Desenvolvimento de aplicações responsivas e acessíveis.

# 4 – Desenvolvimento para Web

## Projeto para Web

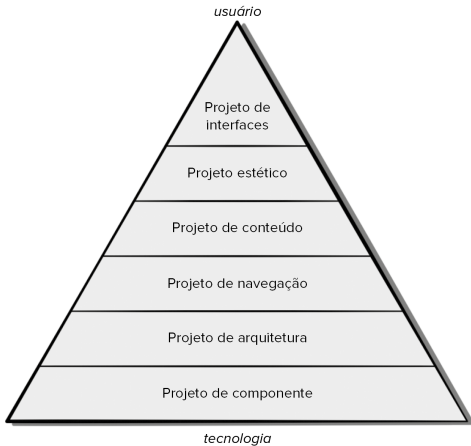
Permite a criação de um modelo que pode ser avaliado em termos de qualidade e aperfeiçoado antes de o conteúdo e o código serem gerados, os testes serem realizados e os usuários envolverem-se em grande número.

## **Projeto para Web**

### **Projetos grandes requerem:**

engenheiros Web, designers gráficos, desenvolvedores de conteúdo, programadores, especialistas em banco de dados, arquitetos de informação, engenheiros de rede, especialistas em segurança e testadores.

# Projeto para Web



## Projeto de interfaces

Os objetivos de uma interface para WebApp:

- (1) estabelecer uma janela para o conteúdo e a funcionalidade fornecidos pela interface;
- (2) guiar o usuário com uma série de interações com a WebApp; e
- (3) organizar as opções de navegação e conteúdo disponíveis para o usuário.



# Projeto de interfaces

## Questões a considerar

- interface sólida - devemos primeiro usar o projeto visual -  
abrange várias características, mas deve enfatizar o layout e a  
forma dos mecanismos de navegação
- interação com o usuário - podemos fazer uso de uma metáfora  
apropriada que permita ao usuário ter um entendimento intuitivo  
da interface

## Projeto de interfaces

- opções de navegação - podemos selecionar menus de navegação posicionados de modo padrão, ícones gráficos representados de uma maneira que permita ao usuário reconhecer que o ícone é um elemento de navegação

## **Projeto estético (ou de layout)**

É o esforço artístico que complementa os aspectos técnicos do projeto de WebApps.

O layout das páginas é um aspecto do projeto estético que pode afetar a utilidade e a usabilidade.

## Projeto estético (ou de layout)

### Diretrizes gerais:

#### **Não tenha medo de espaços abertos**

O congestionamento visual resultante torna difícil para o usuário identificar as informações ou os recursos necessários e cria um caos visual desagradável.

#### **Enfatize o conteúdo**

O uso de página Web típico deve ter 80% de conteúdo e o espaço restante dedicado à navegação e a outros recursos.

## Projeto estético (ou de layout)

Diretrizes gerais:

**Organize os elementos do layout da parte superior esquerda para a inferior direita**

Se os elementos do layout tiverem prioridades específicas, os de alta prioridade devem ser colocados na parte superior esquerda do espaço da página.

**Agrupe a navegação, o conteúdo e as funções geograficamente dentro da página**

Definir padrões

## Projeto estético (ou de layout)

Diretrizes gerais:

**Não estenda seu espaço com a barra de rolagem**

Reduzir o conteúdo da página Web ou apresentar o conteúdo necessário em várias páginas.

**Considere a resolução e o tamanho da janela do navegador ao elaborar seu layout**

Padronizar de acordo com o dispositivo

## Projeto conteúdo

Aqui um objeto de conteúdo tem maior alinhamento com um objeto de dados para software tradicional.

Um objeto de conteúdo possui atributos que incluem informações específicas de conteúdo (normalmente definidas durante a modelagem de requisitos da WebApp) e atributos de implementação exclusivos, especificados como parte do projeto

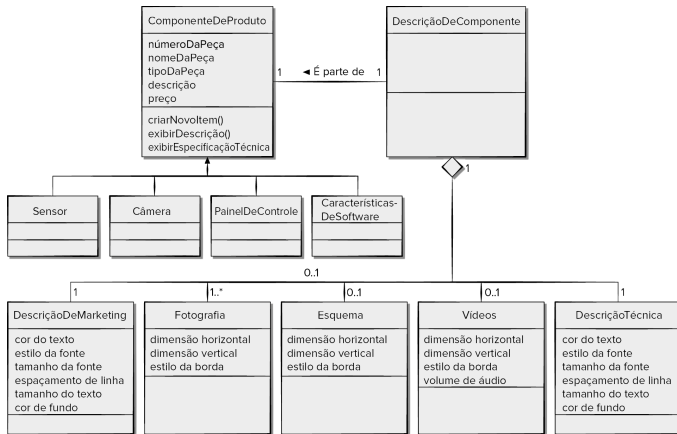
# Projeto conteúdo

## Exemplo:

uma classe de análise, `ComponenteDoProduto`, desenvolvida para o sistema de comércio eletrônico do CasaSegura. O atributo da classe de análise, descrição, é representado como uma classe de projeto chamada `DescriçãoDeComponente`, composta por cinco objetos de conteúdo: `DescriçãoMarketing`, `Fotografia`, `DescriçãoTécnica`, `Esquema` e `Vídeos`. As informações contidas no objeto de conteúdo são indicadas na forma de atributos. Por exemplo, `Fotografia` (uma `imagem.jpg`) possui os atributos `dimensão horizontal`, `dimensão vertical` e `estilo da borda`.



# Projeto conteúdo



## **Projeto de arquitetura**

Ligado aos objetivos estabelecidos, ao conteúdo a ser apresentado, aos usuários que visitarão a página e à filosofia de navegação estabelecida.

# Projeto de arquitetura

Identificar:

a **arquitetura do conteúdo** (maneira pela qual objetos de conteúdo são estruturados para apresentação e navegação) e,

a **arquitetura da aplicação** (maneira pela qual a aplicação é estruturada para administrar a interação com o usuário, tratar tarefas de processamento interno, navegação efetiva e apresentação de conteúdo).

## Projeto de arquitetura

As aplicações devem ser construídas usando-se camadas em que diferentes preocupações são levadas em conta.

Os dados da aplicação devem ser separados do conteúdo da página (nós de navegação) e, os conteúdos devem estar claramente separados dos aspectos da interface (páginas).

*Manter a interface, a aplicação e a navegação separadas simplifica a implementação e aumenta a reutilização.*

## **Projeto de arquitetura**

### **A arquitetura Modelo-Visão-Controlador (MVC)**

Modelo popular de arquitetura para WebApps que separa a interface do usuário da funcionalidade e do conteúdo de informações.

## Projeto de arquitetura

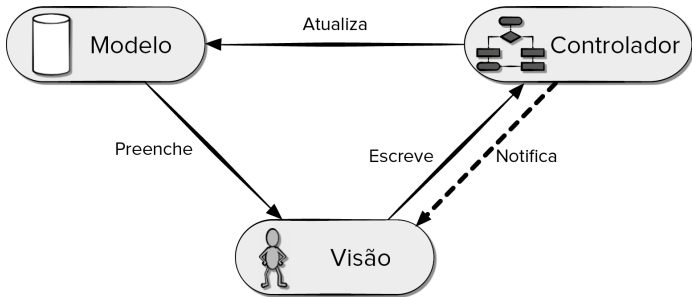
**Modelo:** contém todo o conteúdo e a lógica de processamento específicos à aplicação, inclusive todos os objetos de conteúdo.

**Visão:** contém todas as funções específicas à interface e possibilita a apresentação do conteúdo e lógica de processamento.

**Controlador:** gerencia o acesso ao modelo e à visão e coordena o fluxo de dados entre eles.

# Projeto de arquitetura

as solicitações ou os dados do usuário são manipulados pelo controlador. O controlador também seleciona o objeto de visão aplicável, de acordo com a solicitação do usuário. É transmitida uma solicitação de comportamento ao modelo, que implementa a funcionalidade ou recupera o conteúdo necessário para atender à solicitação. O objeto-modelo pode acessar dados armazenados em um banco de dados corporativo, como parte de um repositório de dados local ou de um conjunto de arquivos independentes. Os dados desenvolvidos pelo modelo devem ser formatados e organizados pelo objeto de visão apropriado e transmitidos do servidor de aplicações de volta para o navegador instalado no cliente para exibição na máquina do usuário.



## Projeto de navegação

Definir os percursos de navegação que permitirão aos usuários acessarem o conteúdo e as funções da WebApp.

Para tanto, identificamos a semântica de navegação para diferentes usuários do site e definimos a mecânica (sintaxe) para obter a navegação.



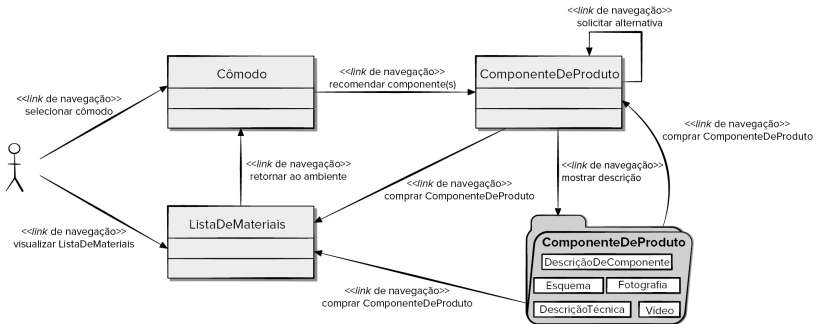
## Projeto de navegação

A base são as **unidades semânticas de navegação (NSUs)**

“um conjunto de informações e estruturas de navegação relacionadas que colaboram no cumprimento de um subconjunto de requisitos de usuário relacionados”

Uma NSU descreve os requisitos de navegação para cada caso de uso - mostra como um ator se movimenta pelos objetos de conteúdo ou funções da WebApp.

# Projeto de navegação



## Projeto de navegação

Podemos criar uma NSU para cada caso de uso associado ao papel de cada usuário.

Durante os estágios iniciais do projeto de navegação, a arquitetura do conteúdo da WebApp é avaliada para determinar um ou mais WoNs para cada caso de uso.

À medida que o projeto prossegue, sua tarefa é definir a mecânica de navegação

## **Projeto em nível de componentes**

Para alcançar um conjunto de capacidades, temos de projetar e construir componentes de programa que sejam idênticos em sua forma aos componentes para software tradicional

# Projeto em nível de componentes

Vejam algumas delas:

- (1) executam processamento localizado para gerar recursos de navegação e conteúdo de forma dinâmica;
- (2) fornecem recursos de cálculo ou processamento de dados apropriados para a área de negócios do aplicativo;
- (3) fornecem sofisticadas consultas e acesso a bancos de dados; e
- (4) estabelecem interfaces de dados com sistemas corporativos externos.

*Temos de projetar e construir componentes de programa que sejam idênticos em sua forma aos componentes para software tradicional.*

## Projeto em nível de componentes

Para fins de economia de custos, você pode projetar componentes de forma que possam ser utilizados sem modificações em diversas plataformas móveis diferentes.

O ambiente de implementação, as linguagens de programação e os padrões de projeto, estruturas e software podem variar um pouco, mas a estratégia de projeto geral permanece a mesma.

# Exemplo prático

## Projeto de mobilidade para desenvolvimento Webapps

### 1.Interfaces:

O projeto é um aplicativo de delivery de comida que deve ser projetado para ser intuitivo e fácil de usar para o cliente. As interfaces devem ser responsivas e adaptáveis a diferentes tamanhos de tela de dispositivos móveis.

Haverá interfaces diferentes para clientes e restaurantes parceiros, com diferentes níveis de acesso e funcionalidades.

## Exemplo prático

### 2. Layout:

O layout deve ser atraente e envolvente para os usuários, incluindo elementos de design modernos e consistentes (ícones, botões de navegação, fontes claras e agradáveis e cores harmoniosas).

A interface do cliente deve ter uma página inicial com as ofertas do dia, uma página para selecionar o restaurante, uma página de menu com imagens dos pratos, uma página de carrinho para revisar o pedido e uma página de pagamento.



## Exemplo prático

### 3. Conteúdo:

O conteúdo do aplicativo deve ser preciso, completo e atualizado.

As informações sobre os restaurantes devem incluir menus, horários de funcionamento, preços, opções de pagamento e opiniões de outros usuários.

As descrições dos pratos devem incluir ingredientes, preços e imagens.

## Exemplo prático

### 4. Navegação:

A navegação do aplicativo deve ser fácil de usar e previsível.

Os botões de navegação devem ser consistentes em todas as páginas do aplicativo.

Deve haver um sistema de busca eficiente para permitir que os clientes encontrem rapidamente os restaurantes e pratos desejados. O aplicativo deve ter uma opção de "histórico de pedidos".

## Exemplo prático

### 5. Arquitetura:

Será desenvolvido com base na arquitetura de três camadas: apresentação, lógica de negócios e banco de dados.

A camada de apresentação cuidará da interface do usuário, a camada de lógica de negócios cuidará de aspectos como a validação de formulários e a seleção de restaurantes e pratos e, a camada do banco de dados cuidará do armazenamento de informações de clientes, restaurantes e pedidos.

## Exemplo prático

### 6. Componentes:

O aplicativo será dividido em vários componentes, como carrinho de compras, sistema de pagamento, sistema de busca, sistema de avaliação, gerenciamento de pedidos, etc.

Cada componente será desenvolvido de forma independente e, em seguida, integrado ao aplicativo como um todo.

***Pronto! Projeto concluído!***

A partir disso você tem um conjunto de informações relevantes para desenvolver o projeto da sua aplicação web.

**Lembre-se que o projeto é essencial para a fase de desenvolvimento do seu sistema web.**

## **Referência utilizada em todos os materiais até agora:**

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 9 ed. Porto Alegre: AMGH, 2021.

# 5 - Conceitos e aplicação de HTML

## Recursos adicionados no HTML5

- oferece suporte nativo para áudio e vídeo.
- suporta imagens em formatos SVG, Canvas e outros elementos gráficos vetoriais gráficos.
- utiliza o banco de dados SQL na web.
- JavaScript e a interface do navegador rodam em processos separados.



## Principais recursos do HTML5

- **Elemento canvas:** permite desenhar gráficos em uma página web.

O canvas é uma área na página que se tem controle total aos pixels desta região, podendo desenhar diversos elementos gráficos.

*tags* <canvas> </canvas>

Na primeira *tag* pode-se incluir diversos atributos.

- **Elementos de áudio e vídeo:** no HTML5 foram criadas tags específicas para inclusão de áudios e vídeos, sem a necessidade do uso de plug-ins externos.

*tags* <audio> </audio>

*tags* <video> </video>

Na primeira *tag* pode-se incluir diversos atributos.

# Sintaxe do HTML5

No HTML5 temos 3 termos básicos: elemento, etiqueta (tag) e atributo.

## Elementos (elements):

refere-se ao identificadores que descrevem os objetos, como texto, imagens, multimídia, etc.. Incluindo a estrutura e o conteúdo.

### Exemplo:

`<p> </p>` : para designar um parágrafo

`<a> </a>` : para designar um link

# Sintaxe do HTML5

## **Etiqueta (tags):**

é mais usual utilizar o termo tag.

Os elementos em HTML são muitas vezes formados por um conjunto de tags.

As tags não são visíveis no navegador (página web).

Existem dois tipos de tags – tag dupla e tag simples.

**Tag dupla:** é um elemento que vem com os sinais < e >

O fechamento desta tag é dado pelo sinal de “menor que” precedido de uma barra “/” precedido do label e finalizado com o sinal de “maior que”.

`<h1> PÓS PUCRS On-line </h1>`

`<h1>` h1 é uma tag de formatação de cabeçalho (header).

Tudo que estiver compreendido entre a tag de abertura `<h1>` e a tag de fechamento `</h1>`, está sujeita a função da tag.

Podem ser colocadas diversas tags na mesma linha:

`<h1><b><i> PUCRS On-line </i></b></h1>`

**Tag simples:** a tag simples é mais comum quando deseja-se inserir um elemento em uma determinada área da página web.

A tag simples não tem abertura e fechamento, apenas representa-se desta forma: `<img src />`

Exemplo: ``

ou

``

## Atributo:

são propriedades usadas para adicionar instruções adicionais a um determinado elemento.

Estes atributos devem ser colocados na tag de abertura.

Cada atributo normalmente tem parâmetros que devem ser colocados entre aspas duplas.

## Exemplo:

```
<button id="btao_login" class="css_btao"> Clique Aqui </button>
```

**id** (identificador)  
é um atributo e  
“btao\_login” é o  
parâmetro.

**class** (identificador)  
é um atributo e  
“css\_btao” é o  
parâmetro.

# Regras básicas do HTML5

Regras básicas:

**1)** os elementos e atributos podem ser escritos em letras minúsculas ou maiúsculas.

**Exemplo:**

*<button> ou <BUTTON>*

## Regras básicas do HTML5

2) as tags duplas devem ter a tag de abertura e fechamento.

**Exemplo:**

Código HTML	Apresentação no browser	Código HTML	Apresentação no browser
<pre>&lt;ul&gt;     &lt;li&gt; item 1: &lt;/li&gt;     &lt;li&gt; item 2: &lt;/li&gt; &lt;/ul&gt;</pre>	<ul style="list-style-type: none"><li>• item 1:</li><li>• item 2:</li></ul>	<pre>&lt;ol&gt;     &lt;li&gt; item 1: &lt;/li&gt;     &lt;li&gt; item 2: &lt;/li&gt; &lt;/ol&gt;</pre>	<ol style="list-style-type: none"><li>1. item 1:</li><li>2. item 2:</li></ol>

**tag ul** (unordered list – lista desordenada) cria uma lista de itens sem uma numeração. E a tag li (list item – lista de itens) define a lista do item.

**tag ol** (ordered list – lista ordenada) cria uma lista de itens numerada em ordem.



3) tags simples recomenda-se o uso da barra antes do sinal > de fechamento da tag.

### Exemplo:

**, recomenda-se **

4) as tags devem seguir a ordem de abertura e fechamento, sem a sobreposição de tags.

**Exemplo:**

***<p><b><i>PUCRS On-line</i></b></p>***

A primeira tag de abertura será a última a fechar.

**5)** todos os atributos de tags devem ser colocados entre aspas duplas.

**Exemplo:**

***<button id="bt\_ok" class="css\_bt">OK</button>***

**6)** Elementos pais podem ter mais de um elemento filho, porém elementos filhos não podem ter elementos pais.

**Exemplo:**

**Incorreto, pois `<div>` é um elemento filho e não comporta um elemento pai (`body`)**

```
<div>  
  <body>  
  
  </body>  
</div>
```

**Correto, pois o elemento pai é o `<body>` e o elemento filho é `<div>`. O elemento pai pode ter n filhos.**

```
<body>  
  <div>  
  
  </div>  
</body>
```

## Estrutura de um arquivo HTML

`<!DOCTYPE html>`

`<html>`

`<head>`

`</head>`

`<body>`

`</body>`

`</html>`

# Estrutura básica de um documento HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>

</head>
<body>
  Olá Mundo!!!
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
</head>
<body>
  Olá Mundo!!!
</body>
</html>
```

*Se não identificar os caracteres com acento, pode-se usar o **padrão UTF-8**, codificação em 8 bits.*

*Essa configuração é feita no HEAD*

*<meta charset="UTF-8">*

*Portanto, em meta charset indica qual é o conjunto de caracteres que será usado na página.*

*Comentários: <!-- comentário -->*

**<!DOCTYPE html>** – A declaração !DOCTYPE informa ao navegador a versão do HTML que está sendo utilizada no documento.

No HTML5, basta incluir !DOCTYPE html, e assim o navegador já saberá que se trata de um documento na versão HTML5;

**<html></html>** – Elemento básico da estrutura do HTML.

A tag html tem dois elementos filhos: head e body.

**<head></head>** – Delimita o cabeçalho do documento.

Não possui nenhum valor visível, porém é capaz de transmitir ao navegador diversas informações muito úteis e essenciais a uma boa apresentação do seu documento HTML;

**<title></title>** – Define o título da sua página, o nome que aparecerá na aba, janela ou guia.

**<meta/>** – Permite inserir metadados ao seu documento, por exemplo, a informação `charset="UTF-8"`, que garante a compatibilidade do código com os caracteres de padrão latino americano;

**<body></body>** – A tag que representa o corpo do documento.



# Estruturação do conteúdo na página web

## Elemento <div>

separa a página web em divisões ou seções. Cada divisão permite agrupar elementos como texto, imagens, formulários, etc.. Além disso, quando utilizar CSS, poderá criar um estilo para cada divisão.

Este elemento é dupla tag **<div> </div>**.

### Atributos:

Atributo	Valor	Descrição
<b>class</b>	<b>Nome da classe</b>	Define uma classe para o elemento.
<b>id</b>	<b>Id</b>	Define um identificador (id) para o elemento.
<b>lang</b>	<b>Linguagem preferencial</b>	Define a linguagem que será usada. No caso do português, usa-se lang="pt-br".
<b>tittle</b>	<b>Texto</b>	Informação adicional ao elemento.
<b>dir</b>	<b>Ltr ou rtl</b>	Define o sentido de leitura, da esquerda para direita (ltr-left to right) ou da direita para a esquerda (rtl – right to left).
<b>accesskey</b>	<b>character</b>	Define uma tecla de atalho para acessar este elemento.

# Exemplo

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="UTF-8"/>
```

```
<title>POS PUC On-line</title>
```

```
</head>
```

```
<body>
```

```
<div id="banner">
```

```

```

```
</div>
```

```
<div id="menu">
```

```

```

```
</div>
```

```
</body>
```

```
</html>
```

Seção 1 ou divisão 1

Seção 2 ou divisão 2

## Elemento <span>

semelhante ao div, porém é uma forma de atribuir atributos a um pequeno grupo de elementos.

A tag span é usada normalmente para customizar uma determinada parte de um texto.

*<p>Esta disciplina usa **<span class="css\_texto">** HTML, CSS, Javascript **</span>** para desenvolvimento.</p>*

### Tag **<header>**

header (cabeçalho) contém normalmente logo, slogan, banner de uma página web.

Esta tag é dupla **<header> </header>**

### Tag **<footer>**

o footer (rodapé) geralmente está localizado no fim da página web.  
Contém informações avisos legais e/ou autoria e/ou links.

Dupla tag **<footer></footer>**.

### Tag **<nav>**

a tag nav contém os links principais do site.

A vantagem de agrupar nesta tag é poder criar uma customização (CSS – estilo) para este grupo apenas de elementos.

É uma tag dupla **<nav></nav>**.

## Tag <section>

é usada para agrupar diferentes tipos de elementos do mesmo tema. Geralmente é usado com o elemento header.

É dupla <section> </section>.

Tag <aside>: a tag aside é dupla <aside> </aside>.

Geralmente com informações adicionais (complementares) e fica localizado ao lado do texto.

## Tag <article>

É dupla <article> </article>.

Usada quando deseja-se encapsular uma parte do conteúdo da página.

### Tunisie

Portugal news Tunisia

La Tunisie (arabe: تونس), en forme longue la **République tunisienne** (arabe:جمهورية تونس de Jamahiriya al Fihriyya), est un pays d'Afrique du Nord.

Elle est bordée au nord et à l'est par la mer Méditerranée, à l'ouest par l'Algérie avec 965 kilomètres de frontière commune et au sud-est par la Libye avec 439 kilomètres de frontière. Sa capitale Tunis est située dans le nord-est du pays, au bord du golfe de Tunis. Plus de 90 % de la superficie du territoire est occupée par le désert du Sahara, le reste étant constitué de régions montagneuses et de plaines fertiles, héritées de la civilisation carthagénoise qui atteignit son apogée au I<sup>er</sup> siècle av. J.-C., avant de devenir le « grenier à blé » de l'Empire romain.

Longtemps appelée **Régence de Tunis**, notamment sous la domination ottomane, la Tunisie passe sous protectorat français le 12 mai 1881 avec la signature du traité du Bardo. Avec l'établissement de l'indépendance, le 20 mars 1956, le pays s'achemine, au début, vers le statut d'une monarchie constitutionnelle avant pour succéder à monarque Lamine Bey<sup>[1]</sup>, élu souverain et devient bey, régnant de la dynastie des Hachémites<sup>[2]</sup>. Avec la proclamation de la république, le 25 juillet 1957, c'est le leader nationaliste Habib Bourguiba qui devient le premier président de la République tunisienne et moderne le pays. Toutefois, en 1987, au terme de trente ans à la tête du pays dont la fin est marquée par le cinquantenaire et la mort de Bourguiba, le Premier ministre Zine el-Abidine Ben Ali lui succède, mais poursuit dès lors les principes républicains de « transparence » lors des élections électorales. Après 23 ans d'une présidence autoritaire et problématique, caractérisée par l'impopularité de la **Constitution**<sup>[3]</sup><sup>[4]</sup>, Ben Ali est chassé le 14 janvier 2011 par une révolution populaire. Trouvant refuge en Arabie saoudite, puis finalement à Doha<sup>[5]</sup><sup>[6]</sup>, il fut tué, avec son épouse Leila Ben Ali, lors d'un attentat d'extrême violence.

Longtemps sous tutelle française de la Communauté internationale, la Tunisie fut également partie de la Ligue arabe, de l'Union africaine et de la Communauté des États arabes saoudiens.



## Outas Tag's muito utilizadas

**Parágrafos e textos:** tags que servem para estruturar o texto em uma página web

**Tag <p>:** cria um novo parágrafo.

### Exemplo:

<p> Esta é uma aula sobre estrutura de texto. </p>

<p> Cada tag <p> cria um novo parágrafo. </p>

### Resultado:

Esta é uma aula sobre estrutura de texto.

Cada tag <p> cria um novo parágrafo.

## Outas Tag's muito utilizadas

### Tag <blockquote>

cria um novo bloco de texto com uma tabulação.

#### Exemplo:

<p> Esta aula é sobre tags de texto: <blockquote> Existem várias tags!</blockquote> </p>

#### Resultado:

Esta aula é sobre tags de texto:  
Existem várias tags!

## Outas Tag's muito utilizadas

### Tag `<strong>`

coloca uma sentença em negrito.

A tag `<b>` faz o mesmo, mas aconselha-se usar a tag `<strong>` no HTML5.

### Exemplo:

`<p>Aula de <strong>HTML5</strong>: uso da tag strong!</p>`

### Resultado:

Aula de **HTML5**: uso da tag strong!



## Outas Tag's muito utilizadas

### Tag <em>

coloca uma sentença em itálico.

A tag <i> faz o mesmo, mas aconselha-se usar a tag <em> no HTML5.

### Exemplo:

<p>Aula sobre desenvolvimento de <em>webpage</em> em HTML5!</p>

### Resultado:

Aula sobre desenvolvimento de *webpage* em HTML5!

## Outas Tag's muito utilizadas

### Tag <small>

coloca o texto em um tamanho menor que os demais.

### Exemplo:

```
<p>Aula sobre HTML5.</p>
```

```
<p>Nesta aula aprendemos sobre o uso da tag small!</p>
```

### Resultado:

Aula sobre HTML5.

Nesta aula aprendemos sobre o uso da tag small!

## Outas Tag's muito utilizadas

### Tag <abbr>

usada quando se deseja colocar o significado de uma sigla quando o mouse passa por cima da sigla.

### Exemplo:

```
<p>Aula sobre <abbr title="HyperText Markup  
Language">HTML5</abbr></p>
```

### Resultado:

Aula sobre HTML5!

## Outas Tag's muito utilizadas

### Tag <var>

usada para reforçar que uma parte do texto define uma variável.

### Exemplo:

<p>O resultado da equação <var>x</var>+<var>y</var> é:</p>

### Resultado:

O resultado da equação  $x+y$  é:

## Outas Tag's muito utilizadas

### Tag <kbd>

keyboard (kbd) mostra o termo em fonte monotype, ou seja, dá mais ênfase que se trata de uma tecla do teclado que deve ser pressionada.

### Exemplo:

<p>Para abrir o prompt de comando, digite em pesquisar <kbd>cmd</p> e depois Enter.</p>

### Resultado:

Para abrir o prompt de comando, digite em pesquisar cmd e depois Enter.

## Outas Tag's muito utilizadas

### Tag <sub>

coloca o texto em subscrito.

### Exemplo:

<p>Tomar água (H<sub>2</sub>O) todos os dias é muito importante!</p>

### Resultado:

Tomar água (H<sub>2</sub>O) todos os dias é muito importante!

## Outas Tag's muito utilizadas

### Tag <sub>

coloca o texto em subscrito.

### Tag <sup>

coloca o texto em sobrescrito.

## Outas Tag's muito utilizadas

### Tag <time>

usada para colocar a hora de forma destacada.

#### Exemplo:

<p>A aula começa as <time> 10:30</time> na segunda-feira. </p>

#### Resultado:

A aula começa as 10:30 na segunda-feira.

*Esta tag tem um atributo que é o datetime. O datetime permite que os navegadores adicionem lembretes sobre um determinado evento.*

#### Exemplo:

<p>A voo sai as <time datetime="2023-05-10 20:00"> de Porto Alegre.</time></p>



# Listas para estruturar o texto

## Listas com marcadores

algumas tags que servem para elaborar listas com marcadores. Estas listas podem ser ordenadas ou não ordenadas. Em conjunto a estas tags, utiliza-se a tag <li> (list item).

### Tag <ol>

define uma lista ordenada (ordered list – ol).

#### Exemplo:

```
<p> Tecnologias:</p>
<ol>
    <li> HTML5</li>
    <li> CSS3</li>
</ol>
```

#### Resultado:

Tecnologias:  
1. HTML5  
2. CSS3

# Listas para estruturar o texto

A tag <ol> tem alguns atributos que modificam a forma de numerar.

Se não for especificado nada, a numeração segue a numeração decimal, do contrário poderá assumir outros tipos.

<p>Exemplo:</p> <pre>&lt;ol type="A"&gt;   &lt;li&gt; HTML5&lt;/li&gt;   &lt;li&gt; CSS3&lt;/li&gt;   &lt;li&gt; Javascript&lt;/li&gt; &lt;/ol&gt;</pre>	<p>Exemplo:</p> <pre>&lt;ol type="i"&gt;   &lt;li&gt; HTML5&lt;/li&gt;   &lt;li&gt; CSS3&lt;/li&gt;   &lt;li&gt; Javascript&lt;/li&gt; &lt;/ol&gt;</pre>	<p>Exemplo:</p> <pre>&lt;ol type="I"&gt;   &lt;li&gt; HTML5&lt;/li&gt;   &lt;li&gt; CSS3&lt;/li&gt;   &lt;li&gt; Javascript&lt;/li&gt; &lt;/ol&gt;</pre>
<p>Resultado:</p> <ul style="list-style-type: none"><li>A. HTML5</li><li>B. CSS3</li><li>C. Javascript</li></ul>	<p>Resultado:</p> <ul style="list-style-type: none"><li>i. HTML5</li><li>ii. CSS3</li><li>iii. Javascript</li></ul>	<p>Resultado:</p> <ul style="list-style-type: none"><li>I. HTML5</li><li>II. CSS3</li><li>III. Javascript</li></ul>

# Listas para estruturar o texto

## Tag <ul>

define uma lista não-ordenada (unordered list – ul).

### Exemplo:

```
<p> Nesta disciplina iremos aprender:</p>
<ul>
    <li> HTML5</li>
    <li> CSS3</li>
    <li> Javascript</li>
</ul>
```

### Resultado:

Nesta disciplina iremos aprender:

- HTML5
- CSS3
- Javascript

# Listas para estruturar o texto

## Tag <dl> , <dd> e <dt>

conjunto de tags são usadas quando se precisa organizar uma definição em formato de lista.

**<dl> definition list**

**<dt> definition term**

**<dd> definition description.**

**Exemplo:**

**<dl>**

**<dt>HTML5:</dt>**

**<dd>É uma Linguagem de Marcação de Hiper Texto, ou seja, é a forma utilizada para a construção de páginas web.</dd>**

**<dd> O arquivo HTML pode ser escrito em qualquer editor de texto simples e ser salvo com a extensão htm ou html. </dd>**

**</dl>**

**Resultado:**

**HTML5:**

É uma Linguagem de Marcação de Hiper Texto, ou seja, é a forma utilizada para a construção de páginas web.  
O arquivo HTML pode ser escrito em qualquer editor de texto simples e ser salvo com a extensão htm ou html.

# Hiperlinks

## Hiperlink no arquivo HTML5 (página externa e interna)

Utiliza-se a tag `<a></a>`.

Esta tag aceita alguns atributos.

**Link absoluto:** contém um endereço completo para acessar.

**Exemplo:** `<a href="https://www.pucrs.br"> PUCRS </a>`

*Neste caso a palavra PUCRS ficará sublinhada de azul e quando clicada levará para a página da PUCRS.*

**Link relativo:** contém um endereço de uma página que faz parte deste website.

**Exemplo:** `<p>No HTML5 os <a href="hiperlink.html"> hiperlinks </a> funcionam como uma conexão.</p>`

*Neste caso a palavra hiperlinks ficará sublinhada de azul e quando clicada levará para a página hiperlink.html.*

# Hiperlinks

Com o path onde está a página

**Exemplo:**

<p>O HTML5 utiliza o recurso de <a href="nome\_pasta/nome\_pasta/hiper.html"> hiperlinks.</a> </p>

**Hiperlink no arquivo HTML5 para um ponto na mesma página web**

Criar um identificador (id) num determinado ponto da página.

Geralmente utiliza-se um header (título) e insere um id.

Para fazer o link com este ponto na página, utiliza-se como o símbolo (#) seguido do identificador (id).

**Exemplo:**

<p><h1 id="html5"> Introdução ao HTML5</h1>

<p>Como visto na <a href="#html5">introdução ao HTML5</a>, HTML5 é uma Linguagem de Marcação de Hiper Texto.</p>

# Hiperlinks

## Hiperlink no arquivo HTML5 para um ponto em outra página web

Criar um identificador (id) num determinado ponto da página.  
Geralmente utiliza-se um header (título) e insere um id.

### Exemplo:

**Nome do arquivo 1:** **pagina\_introducao.html**

```
<p><h1 id="html5"> Introdução ao HTML5</h1>
```

**Nome do arquivo 2:** pagina\_principal.html

```
<p>Como visto na <a href="pagina_introdução.html#html5">introdução ao HTML5</a>, HTML5  
é uma Linguagem de Marcação de Hiper Texto.</p>
```

# Hiperlinks

## Hiperlink para um endereço de e-mail

No atributo *href* utiliza-se o valor *mailto*

### Exemplo:

```
<a href="mailto:fulano@gmail.com">Entre em contato por e-mail!</a>
```



# Hiperlinks

## Hiperlink para o download de um arquivo

No atributo *href* coloca-se o nome do arquivo.

### Exemplo:

```
<a href="livro.pdf">Faça o download do livro!</a>
```

## Inserir um descritivo quando o “mouse” passar pelo hiperlink de download do arquivo

No atributo *href* coloca-se o nome do arquivo.

### Exemplo:

```
<a href="livro.pdf" tittle="Livro sobre HTML/CSS/Javascript">Faça o download do livro!</a>
```

# Imagens, sons e vídeos

## Imagem

Para inserir uma imagem que está armazenada em uma pasta ou link, utiliza-se a tag `<img>`.

A tag `<img>` tem alguns atributos:

Atributo	Valor	Descrição
src	URL	O endereço onde está hospedada (salva) a imagem
width	Inteiro positivo	O valor pode ser expresso em pixels ou percentualmente em relação a largura da tela.
height	Inteiro positivo	O valor pode ser expresso em pixels ou percentualmente em relação a altura da tela.
usemap	Caracteres	É usado para definir áreas da imagem clicável.
alt	Caracteres	Texto que aparecerá na página, caso a imagem não exista.
title	Caracteres	Texto que aparece quando o mouse passa por cima da imagem.

# Imagens, sons e vídeos

## Imagem

O tag `img` é usado com imagens e o atributo `src` informa o nome do arquivo, o caminho ou até um link externo a página web.

```

```

```

```

```

```

**Importante:** A imagem deve ser salva na pasta

# Áudio

Para inserir um áudio que está armazenado em uma pasta ou link, utiliza-se a tag `<audio></audio>`

## Exemplo:

```
<p><h1>Música de Abertura</h1></p>  
<audio src="abertura.mp3" controls></audio>
```

As tag `<audio>` e `<video>` tem alguns atributos específicos:

Atributo	Valor	Descrição
src	URL ou nome do arquivo	Este é um atributo obrigatório. Onde é informado o nome do arquivo e seu caminho.
controls	controls	Habilita a presença dos botões (play, pause, som) do controle visual do player.
autoplay	autoplay	Estes atributos não estão habilitados na maioria dos navegadores, justamente para evitar que saia tocando um áudio/vídeo indesejado.
preload	auto	
loop	loop	Este atributo funciona da seguinte forma, o usuário clica em play e depois o áudio ficará em loop, do contrário tocará uma vez só.

## Vídeo

Para inserir um vídeo que está armazenado em uma pasta ou link, utiliza-se a tag `<video></video>`

### Exemplo:

```
<p><h1>Vídeo Tema da Vitória!</h1></p>  
<video src="tema_vitoria.mp4" controls></video>
```

Os atributos para a tag `<audio>`, também valem para a tag `<video>`. Existe um atributo específico para a tag `<video>`, tamanho do vídeo.

Atributo	Valor	Descrição
width e height	Um valor inteiro e positivo	Tamanho da tela na página web.

# Formulários

# Formulários

O uso de Formulários é a maneira de o usuário enviar dados para um servidor web para processamento.

**O formulário (form) usa uma tag específica, chamada `<form>....</form>`.**

**Esta tag informa ao browser que é a definição de uma tabela.**

# Tags

`<fieldset>`

agrupar campos de entrada de acordo com o seu tema.

`<legend>`

colocar uma legenda no formulário.

`<label>`

associar um rótulo (label) a um elemento do formulário.

Informações pessoais

Informe o nome  Informe o e-mail



# Tags

```
<form action="...(1)..." method="POST">
  <fieldset>
    <legend>Informações pessoais</legend>
    <label>Informe o nome</label>
    <input type="text" name="nome"/>
    <label>Informe o e-mail</label>
    <input type="text" name="email"/>
    <button type="submit">Envia</button>
  </fieldset>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

## Teste de envio de dados do formulário


```
<form action="...(1)..." method="POST">  
  <fieldset>  
    <legend>Informações pessoais</legend>  
    <label>Informe o nome</label>  
    <input type="text" name="nome"/>  
    <label>Informe o e-mail</label>  
    <input type="text" name="email"/>  
    <button type="submit">Envia</button>  
  </fieldset>  
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.


# Envio de dados do formulário

Opção 1: criar um arquivo PHP que recebe estas informações.

Opção 2: usar um *formsubmit*, por exemplo, *formsubmit.com*

 **FORMSUBMIT**

EMAIL LINK Beta

DOCS DEMO HELP SUPPORT  SPONSOR

## Easy form endpoints for your HTML forms

Connect your form to our form endpoint and we'll email you the submissions. No PHP, Javascript or any backend code required.

EXAMPLE FORM / CONTACT FORM

```
<form action="https://formsubmit.co/your@email.com" method="POST">
  <input type="text" name="name" required>
  <input type="email" name="email" required>
  <button type="submit">Send</button>
</form>
```

Form backend platform for designers and developers.

# Elementos de um formulário

## tag <INPUT>

uma das tags mais utilizadas em formulários HTML e considerada a mais importante.

Representa todos os campos de entrada de dados de um formulário.

A tag <input> tem alguns atributos, o mais importante é o *type*.  
Este atributo define o tipo de campo de entrada:

type = "text"

type = "email"

type = "submit"

# Componentes do atributo type

Type	Descrição	Criado no HTML5?
Button	Define um botão. Pode utilizar também a tag <button> ou informa o type button	-----
Ccheckbox	Define um caixa de checagem <b>Qual pet você tem?</b> <input type="checkbox"/> cachorro <input type="checkbox"/> gato	-----
Date	Define um campo de data <b>Data da busca do pet para banho e tosa?</b> <input type="text" value="dd/mm/aaaa"/>	SIM
Datetime-local	Define um campo de data e horário <b>Data da consulta:</b> <input type="text" value="dd/mm/aaaa -- : --"/>	SIM
Email	Define um campo de e-mail	SIM
File	Define um campo para upload de arquivos	-----
number	Define um campo de intervalo de número <b>Escolha uma opção:</b> <input type="text" value="1"/>	SIM
Password	Define um campo de senha (mascarando-a) <b>Digite sua Senha</b> => <input type="password" value="....."/>	-----
Radio	Define um campo de opções <b>Qual é a idade do seu pet?</b> <input type="radio"/> até 1 ano <input type="radio"/> 1 a 5 anos	-----
Range	Define um campo de intervalo <input type="range"/>	SIM
Reset	Apaga o que estava escrito em um campo do formulário.	-----
Submit	Define um botão de envio de dados do formulário	-----
Tel	Define um campo de telefone	SIM

# Elementos de um formulário

## BUTTON

é uma forma de usar um botão no formulário sem utilizar a tag `<button>`.

```
<form action="....(1)...." method="POST">
```

```
  <fieldset>
```

```
    <legend>Escolha uma das opções abaixo</legend>
```

```
    <label>Envio dos dados</label>
```

```
    <input type="button" value="envia"/>
```

```
    <label>Cancela todos os dados do formulário</label>
```

```
    <input type="button" value="cancel"/>
```

```
  </fieldset>
```

```
</form>
```

(1) O atributo `action` define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

Escolha uma das opções abaixo

Envio dos dados  Cancela todos os dados do formulário

# Elementos de um formulário

## CHECKBOX

é uma forma de seleção de opções muito usado em formulários.

```
<form action="....(1)...." method="POST">
<fieldset>
  <legend>Qual é o pet que você tem?</legend>
  <input type="checkbox" name="cachorro" value="cachorro"/>
  <label>Cachorro</label>
  <input type="checkbox" name="gato" value="gato"/>
  <label>Gato</label>
</fieldset>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

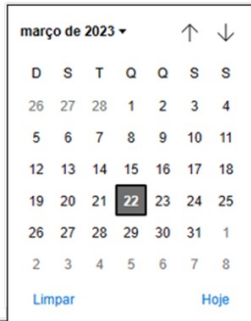
# Elementos de um formulário

## DATE

define um campo de data no formulário.

```
<form action=".....(1)....." method="POST">
<fieldset>
  <label>Escolha uma data do banho e tosa:</label>
  <input type="date" name="databanhotosa"/>
</fieldset>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.



março de 2023 ▼ ↑ ↓

D	S	T	Q	Q	S	S
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Limpar Hoje

Escolha uma data do banho e tosa: dd/mm/aaaa



# Elementos de um formulário

## E-MAIL E PASSWORD

define um campo de entrada de e-mail e senha e deixa mascarada a senha digitada. Recurso muito importante em formulários.

```
<form action=".....(1)....." method="POST">
```

```
<fieldset>
```

```
<label>Email:</label>
```

```
<input type="email" name="email"><br><br>
```

```
<label>Senha:</label>
```

```
<input type="password" name="senha" minlength="8"><br><br>
```

```
<input type="submit">
```

```
</fieldset>
```

```
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.



Email:

Senha:

# Elementos de um formulário

## FILE

define um campo para fazer upload (subida de arquivos) num formulário.

```
<form action=".....(1).....">  
  <label>Selecione o arquivo:</label>  
  <input type="file" name="arquivo"><br><br>  
  <input type="submit">  
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

```
<form action=".....(1).....">  
  <label>Selecione os arquivos:</label>  
  <input type="file" name="arquivos" multiple><br><br>  
  <input type="submit">  
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

# Atributo type e outros componentes

## RADIO

é uma forma de seleção de opções, diferentemente do checkbox, no radiobutton apenas uma opção pode ser escolhida. Muito utilizado em formulários.

```
<form action=".....(1)....." method="POST">
```

```
<fieldset>
```

```
<label>Escolha o seu pet:</label>
```

```
<br/>
```

```
<label>Cachorro</label>
```

```
<input type="radio" value="cachorro"/>
```

```
<label>Gato</label>
```

```
<input type="radio" value="gato"/>
```

```
</fieldset>
```

```
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

# Atributo type e outros componentes

## RESET

limpa/apaga os dados de um campo do formulário.

```
<form action=".....(1)....." method="POST">
```

```
<fieldset>
```

```
  <legend>Qual é o pet que você tem?</legend>
```

```
  <label>Nome do pet:</label>
```

```
  <input type="text"/>
```

```
  <input type="reset" value="Reset"/>
```

```
</fieldset>
```

```
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

# Atributo type e outros componentes

## SUBMIT

define um botão para submissão dos dados do formulário.

```
<form action=".....(1)....." method="POST">
<fieldset>
  <label>Envio dos dados do formulário</label>
  <input type="submit" value="submit"/>
</fieldset>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

# Atributo type e outros componentes

## TEL

define um campo para telefone no formulário.

```
<form action=".....(1)....." method="POST">
```

```
<fieldset>
```

```
<label>Email:</label>
```

```
<input type="email" name="email"><br><br>
```

```
<label>Senha:</label>
```

```
<input type="password" name="senha" minlength="8"><br><br>
```

```
<label>Informe o seu telefone celular:</label>
```

```
<input type="tel" name="fone" placeholder="dddXXXXXXXX" pattern="[0-9]{10}" required><br><br>
```

```
<input type="submit">
```

```
</fieldset>
```

```
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

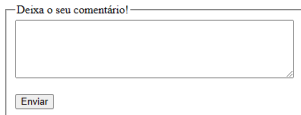
# Outras tags usadas com formulários

## TEXTAREA TAG

Este não é um atributo do input, mas é uma forma de reservar uma área para escrita de comentário do usuário. Muito comum em formulários para observações, opiniões, elogios, etc..

```
<form action=".....(1)....." method="POST">
<fieldset>
  <legend>Deixa o seu comentário!</legend>
  <textarea name="comentario" cols="50" rows="5"></textarea><br><br>
  <input type="submit">
</fieldset>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.



Deixa o seu comentário!

Enviar

# Outras tags usadas com formulários

## SELECT TAG

Esta tag é muito comum em formulários. Permite a seleção de itens em uma lista de opções.

```
<form action="....(1)...." method="POST">
  <p>Selecione o setor que você deseja atendimento:
  <select name="setor">
    <option value="racoes">Rações</option>
    <option value="medicamentos">Medicamentos</option>
    <option value="acessorios">Acessórios</option>
    <option value="consultas">Consulta</option>
    <option value="outros">Outros</option>
  </select>
  <input type="submit">
</p>
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.



# Outras tags usadas com formulários

## OUTPUT TAG

Esta tag é usada para mostrar o resultado de alguma operação.

```
<form oninput="x.value=parseFloat(a.value)*parseFloat(b.value)">
```

```
  <p>Digite o número de horas trabalhadas: <input type="text" id="a"></p>
```

```
  <p>O valor da hora é: <input type="text" id="b"></p>
```

```
  <p>O valor a receber dia 5 é R$ <output name="x"></output></p>
```

```
</form>
```

Digite o número de horas trabalhadas:

O valor da hora é:

O valor a receber dia 5 é R\$ 4140

# ATRIBUTOS DO INPUT ADICIONADOS NO HTML5

## ATRIBUTO DO INPUT – PLACEHOLDER

O placeholder consiste numa descrição do que deve ser digitado no campo de entrada.

```
<label>Informe o nome</label>
```

```
<input type="text" name="nome" placeholder="Digite o nome"/>
```

```
<label>Informe o e-mail</label>
```

```
<input type="text" name="email" placeholder="Digite o e-mail"/>
```

Informações pessoais

Informe o nome

Informe o e-mail

# ATRIBUTOS DO INPUT ADICIONADOS NO HTML5


## ATRIBUTO DO INPUT – REQUIRED

Existem situações onde tem campos de entrada que exigem obrigatoriamente o preenchimento. Nestes casos utiliza-se o atributo *required*.

```
<input type="text" name="email" placeholder="Digite o e-mail" required/>
```

Informações pessoais

Informe o nome	<input type="text" value="Anderson"/>	Informe o e-mail	<input type="text" value="Digite o e-mail"/>	<input type="button" value="Envia"/>
----------------	---------------------------------------	------------------	--	--------------------------------------

 Preencha este campo.

# ATRIBUTOS DO INPUT ADICIONADOS NO HTML5

## ATRIBUTO DO INPUT - PATTERN

O atributo *pattern* corresponde ao padrão que deve ser digitado num campo de entrada.  
Exemplo: só é aceito números; é aceito número e letras.

```
<form action="....(1)...." method="POST">
```

```
<fieldset>
```

```
<legend>Faça o login no sistema</legend>
```

```
<input type="text" name="login" pattern="[A-Za-z]{2,}" placeholder="Digite o login" required/>
```


```
<input type="submit" value="Submit"/>
```

```
</fieldset>
```

```
</form>
```

(1) O atributo action define o local/arquivo (através de uma URL) ao qual serão enviados todos os dados coletados do formulário.

Login

 Faça a correspondência com o formato solicitado.

***Mais conhecimentos sobre HTML você vai ver na aula 3***

***Referência utilizada:***

REBAH, Hassen Ben; BOUKTHIR, Hafedh; CHÉDEBOIS, Antoine. Website design and development with HTML5 and CSS3. London: ISTE Press, 2021.

# 6 – Conceitos e aplicação de CSS

# CSS - Introdução

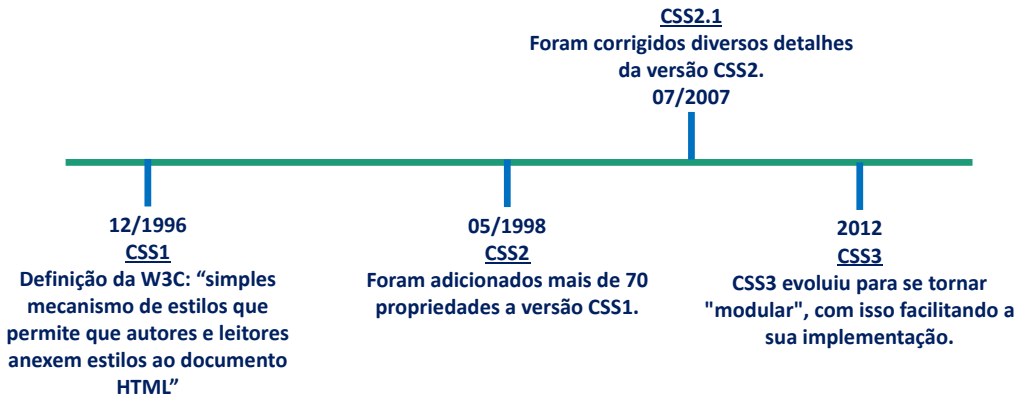
**CSS - Cascading Style Sheets** - é uma linguagem de estilo usada para definir a aparência e o layout de documentos HTML e XML.

É usado para controlar a apresentação visual de páginas da web, permitindo aplicar estilos, cores, fontes, espaçamentos e outras propriedades visuais a elementos HTML.

O CSS é essencial para o design e desenvolvimento de sites modernos e responsivos.

# CSS

Versões desde 1996:





## Sintaxe do CSS

- CSS não é sentive case, ou seja, maiúscula ou minúscula não faz diferença. Exceto para identificadores (id).
- Espaços e linhas em branco não interferem no código CSS.
- Identificadores (nome, identificador e classes) não podem começar com número, mas pode conter no seu nome letras de A-Z, a-z, 0-9, bem como hífen (-) e undeline (\_).
- Comentários são feitos usando /\* no começo do comentário e \*/ no final do comentário.

# Estrutura do CSS

O princípio de operação do CSS é baseado em declarações:

**selector { property : value }**

- selector: o elemento da página web o qual se deseja aplicar o estilo.
- property e value: o grupo de regras que define o estilo.

Exemplo:

```
p
{
    color : green;
    font-family : "Arial"
}
```

*Elementos para um parágrafo, É possível incluir mais de um elemento da página web para receber o mesmo estilo, separando por vírgula. Exemplo: p, div { color: green }*

## Estrutura do CSS

O CSS pode ser declarado em arquivo separado ou dentro do próprio arquivo HTML.

Deve ser inserido entre <head> e </head>.

Além disso, deve se utilizar uma tag específica, chamada <style>....</style>

Se for arquivo separado, salve com a extensão .css

Para incluí-lo no arquivo html, use a tag link.

```
<head>  
  <link href="estilos.css" rel = "stylesheet">  
  <meta charset="UTF-8">  
  <title>Página de Estilos</title>  
</head>
```

## Estrutura do CSS

Outra forma de incluir o arquivo com o estilo css:

usando @import entre as tags <style> e </style>.

```
<head>  
  
  <meta charset="UTF-8">  
  <title>Página de Estilos</title>  
  
  <style>  
    @import "estilos.css"  
  </style>  
  
</head>
```

# Estrutura do CSS

Outra possibilidade é incluir diretamente na tag.

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Página de Estilos</title>
```

```
</head>
```

```
<body>
```

```
  <h1 style= "color: red">Título com um estilo de cor </h1>
```

```
</body>
```

# Propriedade Display

O CSS oferece várias propriedades de **display** para controlar como um elemento é exibido na página.

Cada valor da propriedade de display determina o tipo de layout do elemento, como ele é posicionado na página e como ele interage com outros elementos ao redor dele.

Os principais valores da propriedade de display são **block**, **inline**, **flex** e **grid**, entre outros.

# Propriedade Display

## **block;**

Os elementos são exibidos em uma linha separada, ocupando todo o espaço horizontal disponível.

Usados para criar seções distintas de conteúdo na página.

*Exemplo de uso: para elementos como <div>, <p>, <ul>, <ol>, etc.*

## **inline;**

Os elementos são exibidos em linha, um ao lado do outro, dentro do fluxo normal do texto.

*Exemplo de uso: para elementos como <span>, <a>, <img>, etc.*

# Propriedade Display

## **flex;**

Os elementos são posicionados como um único elemento flexível dentro de um container. Permite definir o tamanho, a ordem e a direção dos itens dentro do container, tornando-os mais fáceis de controlar e posicionar.

*Exemplo de uso: criar layouts responsivos.*

## **grid;**

Os elementos com esta propriedade são posicionados em um sistema de grade. Permite definir a estrutura da grade, as áreas de células, as linhas e colunas, e controlar como os itens são posicionados dentro da grade.

*Exemplo de uso: criar layouts complexos com várias áreas e tamanhos diferentes.*



# Propriedade Position

A propriedade **position** permite controlar como um elemento é posicionado na página em relação a outros elementos.

Existem cinco valores principais da propriedade position:  
**static, relative, absolute, fixed e sticky.**

# Propriedade Position

## **static;**

Este é o valor padrão para a propriedade de position.

Os elementos são posicionados no fluxo normal do documento e de acordo com o fluxo da página. Eles não são afetados pelas propriedades **top, right, bottom ou left**.

## **relative;**

Os elementos são posicionados em relação à sua posição original no fluxo normal do documento, assim os valores **top, right, bottom e left** são aplicados em relação a essa posição original, não em relação a outros elementos na página.

# Propriedade Position

## **absolute;**

Os elementos são posicionados em relação ao elemento pai mais próximo que tenha uma posição definida.

O elemento é removido do fluxo normal do documento e não ocupa espaço em seu local original, permitindo que outros elementos se posicionem em relação a ele.

## **fixed;**

Os elementos são posicionados em relação à janela do navegador, permanecendo na mesma posição, mesmo quando o usuário rolar a página.

O elemento é removido do fluxo normal do documento e não ocupa espaço em seu local original, permitindo que outros elementos se posicionem em relação a ele.

# Propriedade Position

## **sticky;**

Os elementos são posicionados como relative até que o usuário role a página a uma determinada posição, onde ele se torna fixed. O elemento permanecerá na mesma posição em relação à janela do navegador até que o usuário role a página para uma posição diferente.

Quando um elemento é posicionado como sticky, ele ainda ocupa espaço em seu local original no fluxo do documento, mas é deslocado de sua posição original quando se torna fixed.

# Apresentação da página web

Pode ser uma impressão, um projetor multimídia ou um navegador em dispositivos móveis.

Desta forma, pode-se criar um estilo para cada tipo de forma de apresentação.

Para isso utiliza-se @media <nome>

Exemplo:

@media screen		@media print
{		{
.....	ou	.....
}		}

# Apresentação da página web

Outra forma de declarar arquivos de estilos para cada tipo de display:

```
<head>  
  <meta charset="UTF-8">  
  <title>Página de Estilos</title>  
  <link href="estilo_pc.css" media="screen" rel="stylesheet">  
  <link href="estilo_print.css" media="print" rel="stylesheet">  
</head>  
  
<body>  
  
  <h1 style="color: red">Título com um estilo de cor </h1>  
  
</body>
```

# Apresentação da página web

Dependendo do tamanho da tela pode-se desejar mudar algumas propriedades do estilo aplicado.

```
<head>
  <meta charset="UTF-8">
  <title>CSS Animado</title>
  <style>
    @media screen and (max-width: 1024px)
    {
      body
      {
        background-color: blue;
        color: white
      }
    }
    @media screen and (max-width: 640px)
    {
      body
      {
        background-color: cornsilk;
        color: white
      }
    }
  </style>
</head>
```

Se a tela tiver a largura máxima de 1024 pixels, o corpo do html será mostrado com cor de fundo azul e cor das letra em branco.

Se a tela tiver a largura máxima de 640 pixels, o corpo do html será mostrado com cor de fundo cyan e cor das letra em branco.

# Apresentação da página web

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>CSS Animado</title>
    <style>
      @media screen and (max-width: 1024px)
      {
        body
        {
          background-color: blue;
          color: white;
        }
      }

      @media screen and (max-width: 640px)
      {
        body
        {
          background-color: cornsilk;
          color: blue;
        }
      }
    </style>
  </head>
```

```
<body>
  <h1>Teste do CSS para telas de tamanhos diferentes.</h1>
  <h2>Neste caso em telas até 640
    pixels a cor de fundo é
    "cornsilk"</h2>
  <h2>e as letras são em tom
    azul.</h2>
  <h2>Entretanto, telas entre 640
    e 1024 pixels, a cor de
    fundo é azul</h2>
  <h2>e as letras são brancas.</h2>
  <h2>E acima de 1024 pixels a tela
    fica branca com as letras
    pretas.</h2>
</body>
</html>
```



# Ordem de estilos adicionados a uma página web

Num sistema web composto por diversas páginas, também teremos diversos arquivos CSS para tornar a página mais atrativa. Como saber a ordem que os estilos serão aplicados?

**Exemplo:**

arquivo → estilo1.css	arquivo → estilo2.css
<pre>p {     color: blue; }</pre>	<pre>p {     color: green; }</pre>

```
<html>
  <head>
    <title>Teste do CSS</title>
    <style>
      @import "estilo1.css";
      @import "estilo2.css";
    </style>
  </head>
```

```
    <body>
      <p>Que cor o texto ficará?</p>
    </body>
  </html>
```

**Resultado:**

Que cor o texto ficará?

## Para criar uma prioridade, então pode-se usar um identificador (id – identification).

```
<html>
<head>
  <title>Teste do CSS</title>
  <style>
    div#texto p
    {
      color: blue;
    }

    p
    {
      color: red;
    }
  </style>
</head>
```

tag + # + id + tag  
Neste caso o estilo só será aplicado ao div que tem o id=texto.

```
<body>
  <div id="texto">
    <p>Que cor o texto ficará?</p>
  </div>
  <p>E neste que não tem div e id?</p>
</body>
</html>
```




Resultado:

Que cor o texto ficará?

E neste que não tem div e id?

# Seletores em CSS

O seletor (selector) é como uma classe é identificada.

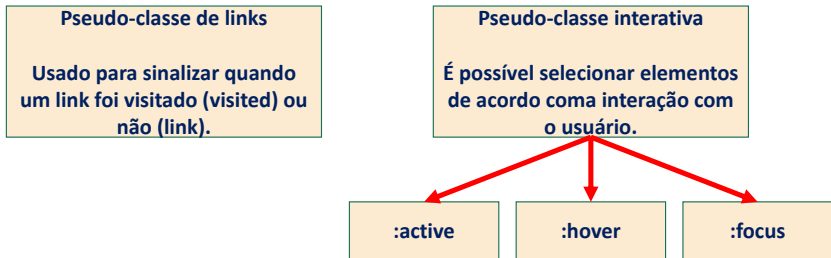
Seletor (selector)  **.class\_texto**  
    {  
        **color: blue;**  
        **font-size: 14px;**  valor (value)  
    }  
      
Propriedade (property)

Seletor	Exemplo do uso	Exemplo da descrição
#id	#texto	Seleciona o elemento pelo id="texto"
.class	.css_texto	Seleciona todos os elementos da class="css_texto"
<i>elemento.class</i>	p.texto	Seleciona somente os elementos <p> com a classe texto (class="css_texto")
* (asterisco)	*	Seleciona todos os elementos
<i>elemento</i>	p	Seleciona todos elementos <p>
<i>elemento, elemento,...</i>	div, p	Seleciona todos elementos<div> e todos os elementos <p>

# Pseudo-Classes e Pseudo-elementos

# Pseudo-classes

Todas as formas de seletores vistos até agora atendem boa parte das aplicações, porém existem outros seletores que tornam a página mais dinâmica.



# Pseudo-classe de links

## Exemplo:

```
<style>
  a:link
  {
    color: blue;
  }
  a:visited
  {
    color: gray;
  }
</style>
```

```
<body>
  <a href="http://www.google.com">Google</a>
</body>
```

## **Resultado:**

Primeira vez que a página é carregada e não clicado no link

Google <antes de ser clicado no link>

Google <após clicar no link Google....ficando em tom cinza(gray)>

# Pseudo-classe interavita

Segundo a W3C (World Wide Web Consortium) temos três pseudo-classe interativas:

**pseudo-classe :hover**

corresponde quando o usuário mostra um elemento mas não o ativa.

*Exemplo: essa pseudoclasse é aplicada pelo navegador se o cursor estiver passando sobre a caixa que limita o elemento*

## Pseudo-classe interavita

**pseudoclasse :active**

corresponde quando um elemento está sendo ativado pelo usuário.

*Exemplo: durante o tempo entre o usuário pressionando o botão do mouse e liberá-lo;*



## Pseudo-classe interavita

pseudoclasse :focus

corresponde quando um elemento tem foco.

*Exemplo: inclui eventos de todos os formulários de entrada*

# Pseudo-classes

Um exemplo de uso de pseudo-classe -> **:hover**

```
<html>
  <head>
    <style>
      img:hover
      {
        border : dotted;
      }
    </style>
    <title>Uso de pseudo-classe</title>
  </head>

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



Quando o mouse se posiciona sobre a imagem...aparece o pontilhado contornando a imagem.

# Pseudo-classes

## Um exemplo de uso de pseudo-classe -> :focus

```
<html>
  <head>
    <style>
      input:focus
      {
        color: red;
        background-color: lightpink;
      }
    </style>
    <title>Uso de pseudo-classe</title>
  </head>
  <body>
    <h3>
      <label>Nome do proprietário:</label>
      <input type="text" name="nome" placeholder="Digite o nome">
    </h3>
  </body>
</html>
```



Nome do proprietário:

Quando o usuário clica o mouse no campo input, a cor de fundo muda para lightpink e quando digita o nome, as letras ficam na cor vermelha.

***Mais conhecimentos sobre CSS você vai ver na aula 3***

***Referência utilizada:***

REBAH, Hassen Ben; BOUKTHIR, Hafedh; CHÉDEBOIS, Antoine. Website design and development with HTML5 and CSS3. London: ISTE Press, 2021.

# 7 - Conceitos e aplicação de JavaScript

# Histórico

O JavaScript foi originalmente desenvolvido sob o nome de **Mocha**, posteriormente teve seu nome modificado para **LiveScript** e, por fim, **JavaScript**.

# O que é javascript

**JavaScript** é uma linguagem de script.

Seu código deve ser executado dentro de um interpretador.

O JavaScript, para ser interpretado, deverá ser executado dentro de um navegador (browser).

# Uma linguagem cliente

É Linguagem *client side* (age no lado do cliente)

Vantagem: diminui seu processamento no lado do serviço,  
diminuindo a escala vertical desses servidores.



# Uma linguagem cliente

- É muito utilizada do lado do servidor através de ambientes como o node.js.
- Foi concebida para ser uma linguagem com OO baseada em protótipos, tipagem fraca e dinâmica e funções de primeira classe.
- Possui suporte à programação funcional e apresenta recursos como *fechamentos* e *funções* de alta ordem comumente indisponíveis em linguagens populares como Java e C++.

# Documentação

Documentação completa do JavaScript é encontrada no site do Mozilla: <https://developer.mozilla.org/en/docs/JavaScript>

# Principais Padrões

**A Linguagem Núcleo - ECMAScript** (implementação e regularização das regras da linguagem JavaScript);

Padrão mantido por ECMA International - Associação Industrial de padronização de tecnologias da Informação e Comunicação;

**DOM** (Document Object Model):

Define a Interface da Linguagem com o Browser;

Padrão mantido por W3C;

# Sintaxe

A maior parte de sua sintaxe vem do Java, mas também é influenciado por Awk, Perl e Python.

- Instruções são chamadas de declaração e são separadas por um ponto e vírgula (;).
- Espaços, tabulação e uma nova linha são chamados de espaços em branco.

# Sintaxe

- O código fonte dos scripts são lidos da esquerda para a direita e são convertidos em uma sequência de elementos de entrada como símbolos, caracteres de controle, terminadores de linha, comentários ou espaço em branco.
- A sintaxe dos comentários em JavaScript é semelhante como em C++ e em muitas outras linguagens.

# Iniciando

Para inserir códigos JavaScript

Tag HTML apropriada:

```
<script>  
</script>
```

*O script pode ser incluído a partir de uma nova página na seção*  
*<body>*

# Iniciando

Da mesma forma como nos arquivos CSS, podemos deixar funções e comandos JavaScript em arquivos externos com extensão .JS

Importando:

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

# Classe Document

## Algumas Propriedades :

title – Define ou Retorna o Título da Página;

URL – Retorna o URL completo da página;

## Alguns Métodos:

write() – Escreve texto no documento;

writeln() – Escreve uma linha de texto no documento;



# Eventos

É possível disparar scripts a partir de diversos tipos de eventos;

## **Exemplo - clique em um botão:**

Tag: <button>Clique Aqui! </button>

Atributos:

type="button";

onclick="alert('Bem vindo!')"

**<button type = "button" onclick="alert ('Bem vindo')"> Clique Aqui!  
</button>**

# Declarações de variáveis

- Com a palavra chave var.

Exemplo: `var x = 42`

- Por simples adição de valor.

Exemplo, `x = 42` (declara uma variável global)

- Com a palavra chave let.

Exemplo: `let y = 13` (variável local de escopo de bloco)

# Declarações de variáveis

JavaScript é uma linguagem de tipagem dinâmica e fraca:

- Não é necessário declarar o tipo de uma variável;
- Todas as variáveis são objetos (referência);
- Números são todos reais de 64bits;
- A variável irá “alterar” o seu tipo de dado conforme os valores forem atribuídos:

# Declarações de variáveis

Exemplos de Tipo de dado dinâmico:

**var x;** // x é indefinido

**x = 5;** // x é um número

**x = "John";** //x é uma string

**x = true;** // x é um valor lógico

**x = null;** // x é indefinido

# Entradas

## JavaScript:

```
function Soma()  
{  
  var e1 = parseInt(document.getElementById("v1").value);  
  var e2 = parseInt(document.getElementById("v2").value);  
  document.getElementById("res").innerHTML = "Resposta: " + (e1 + e2);  
}
```

*Conversão de texto para inteiro*

## HTML:

```
<input type= "text" id="v1"><br>  
<input type= "text" id="v2">  
<p id="res">Resposta: </p>  
<button type= "button" onclick="Soma()">Soma</button>
```

# Arrays

Arrays são desenvolvidos através de um construtor e possuem tamanho dinâmico

```
var nomes = new Array();  
//var nomes = [];  
nomes[0] = "João";  
nomes[1] = "Maria";  
nomes.push("Joana");
```

# Objetos

Objetos possuem **atributos** e **métodos**:

– Assim como em outras linguagens orientadas a objetos, separados por ponto;

– Para criar um objeto carro:

```
var carro={placa:"ABC-1234" , ano:2023}
```

– Para usar os atributos:

```
carro.ano = 2022;
```

```
document.write(carro.placa);
```

– Para adicionar novo atributo:

```
carro.cor = "branco";
```

# Classes

Javascript é orientada a objetos por prototipação, não possui exatamente o conceito de classes; É possível construir uma função que gera novas variáveis e, assim, simular o comportamento de uma classe;



# Classes

```
function Carro(ano, placa) { //Construtor
  this.ano = ano; //Atributo
  this.placa = placa;
  var nCor = Math.random() * Carro.cores.length;
  this.cor = Carro.cores[Math.floor(nCor)];
  this.alterarAno = function(novoAno) { //Método
    this.ano = novoAno;
  };
}

Carro.cores = ["Azul", "Branco", "Vermelho"]; //Atributo estático
Carro.adicionarCor = function (novaCor) { //Método estático
  Carro.cores.push(novaCor);
};
```

# **Estruturas**

**Seleção** – igual em Java

**Repetição** – igual em Java

# Strings

Uma string é um conjunto de caracteres delimitados por aspas simples (') ou aspas duplas (").

Exemplo:

```
let mensagem = 'Olá, mundo!';
```

# Strings

## Interpolação de strings:

inserir expressões JavaScript dentro de uma string.

Cria strings complexas que contêm **valores de variáveis, resultados de expressões e chamadas de funções.**

## Sintaxe:

``${expr}``

# Strings

## Exemplo:

```
let nome = 'Maria';  
let idade = 40;
```

Usando interpolação para criar uma mensagem que inclui os valores dessas variáveis:

```
let mensagem = `Meu nome é ${nome} e eu tenho ${idade} anos.`;
```

**Resultado:** 'Meu nome é Maria e eu tenho 40 anos.'

# Strings

Outro exemplo:

```
let preco = 20;
```

```
let quantidade = 5;
```

```
let mensagem = `O preço total é ${preco * quantidade}.`;
```

A interpolação de strings é uma forma poderosa e flexível de criar strings dinamicamente em JavaScript.

# Módulos

Módulos em JavaScript são uma forma de **encapsular código em arquivos separados**, tornando mais fácil a organização e reutilização de código em aplicações maiores.

Com módulos, podemos criar bibliotecas reutilizáveis, que podem ser usadas em várias partes de uma aplicação, ou separar o código de uma aplicação em arquivos menores e mais fáceis de gerenciar.

# Módulos

Usa-se a palavra-chave **import** para importar funções, objetos e variáveis de outros arquivos no código, e a palavra-chave **export** para exportar funções, objetos e variáveis de um arquivo para serem usados em outros arquivos.



# Módulos

## Exemplo:

temos dois arquivos, um chamado meumodulo.js e outro chamado main.js.  
O arquivo meumodulo.js pode conter o seguinte código:

```
export function soma(a, b) {  
  return a + b;  
}  
  
export const pi = 3.1415;  
  
export default function mensagem() {  
  console.log('Olá, mundo!');  
}
```

O arquivo meumodulo.js exporta:  
a função soma(), a constante pi e a  
função padrão mensagem().

Podemos importar esses valores em nosso arquivo  
main.js, usando a seguinte sintaxe:

```
import { soma, pi } from './meumodulo.js';  
import minhaMensagem from './meumodulo.js';  
  
console.log(soma(2, 3)); // output: 5  
console.log(pi); // output: 3.1415  
minhaMensagem(); // output: 'Olá, mundo!'
```

Foram importadas as funções soma() e pi de meumodulo.js, e a  
palavra-chave default serviu para importar a função padrão  
minhaMensagem() de mymodule.js

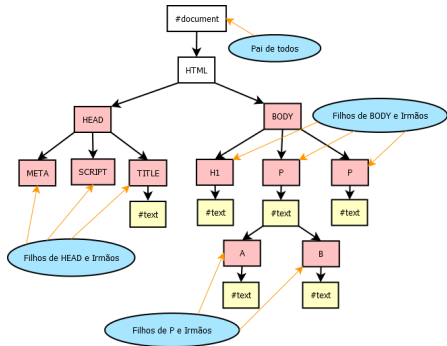
# Manipulação do DOM

**Manipulação do DOM (Document Object Model)** é uma das tarefas mais comuns no JavaScript quando se trabalha em páginas web.

O DOM é uma representação em árvore do conteúdo HTML, e é com ele que interagimos para **adicionar, remover ou modificar** elementos da página.

# Manipulação do DOM - árvore

Existem elementos pai (**parent**), filhos (**childs**) e irmãos (**siblings**). Estes elementos são caracterizados na forma como estão na árvore:



# Manipulação do DOM

**Acesso ao DOM:** usando o objeto `document`, que representa o documento HTML.

Pode-se usar métodos e propriedades desse objeto para acessar e manipular elementos da página.

## Exemplos:

**`document.getElementById('id')`:** retorna o elemento com o ID especificado.

**`document.getElementsByTagName('tag')`:** retorna um array com todos os elementos com a tag especificada.

**`document.createElement('tag')`:** cria um novo elemento com a tag especificada.

**`element.appendChild(child)`:** adiciona um novo filho ao elemento especificado.

**`element.innerHTML`:** define o conteúdo HTML do elemento especificado.

# Manipulação do DOM

## Exemplos:

**Modificar elemento de uma página** obtendo uma referência ao elemento com o ID meu-elemento e modificar seu conteúdo usando a propriedade innerHTML.

```
// Obtém o elemento pelo ID
```

```
const elemento = document.getElementById('meu-elemento');
```

```
// Modifica o conteúdo do elemento
```

```
elemento.innerHTML = 'Novo conteúdo do elemento';
```

# Tratamento de exceções

Para:

- Erros de sintaxe;
- Recursos inexistentes (diferentes browsers);
- Entrada de dados errada;
- Etc.

**Delimitar área que será verificada:**

```
try {  
    //Código passivo de erro  
}
```

**Capturar um eventual erro:**

```
catch (erro) {  
    //Tratamento para eventual erro capturado  
}
```

# Comunicação com Servidor

Existem várias formas de se comunicar com o servidor em JavaScript.

Então, vamos ver o uso de **fetch** e **XMLHttpRequest**.

## **Fetch API:**

É uma API do JavaScript para buscar recursos da rede.

Baseada em Promises, e permite que você faça requisições HTTP para o servidor e trate as respostas em um formato fácil de usar.

# Comunicação com Servidor

## Exemplo:

a seguinte função usa a **Fetch API** para buscar os dados de um usuário em uma API

```
function buscarUsuario(id) {  
  return fetch('https://api.com/usuarios/${id}`')  
    .then((resposta) => {  
      if (!resposta.ok) {  
        throw new Error('Não foi possível buscar o usuário.');      }  
      return resposta.json();  
    });  
}
```

a função `fetch()` faz a requisição HTTP para a API e retorna uma Promise. Depois, usa o método `then()` para tratar a resposta da API, convertendo a resposta para JSON e lançando um erro se a resposta não for bem-sucedida.



# Comunicação com Servidor

## **XMLHttpRequest:**

É uma API antiga do JavaScript para fazer requisições HTTP.

Baseada em callbacks e é menos fácil de usar do que a Fetch API, mas ainda é amplamente usada em muitos projetos.

***Mais conhecimentos sobre JavaScript você vai ver na aula 3***

***Referência utilizada:***

FRISBIE, Matt. Professional JavaScript for web developers. Indianapolis: Wrox, 2020.

**A partir de agora, aproveite para  
colocar em prática todo  
conteúdo/conhecimento  
adquirido nas aulas 1 e 2**

**PUCRS** online  **uol** edtech.