

# DBStart

HTML e CSS

Instrutor: Júlio Pereira Machado ([julio.machado@pucrs.br](mailto:julio.machado@pucrs.br))



# Interface com o Usuário



# Tecnologias

- Múltiplos navegadores com diferentes *engines*
- HTML
- CSS
- JavaScript

# Design

- Necessidade de um processo de *design* adequado para uma boa experiência do usuário
- Conceitos essenciais:
  - Prototipação
    - <https://www.draw.io/>
    - <https://moqups.com/>
    - <https://marvelapp.com>
  - Usabilidade
  - Acessibilidade
    - <https://webaim.org/>
    - <https://developer.chrome.com/docs/lighthouse/>

# HTML



# Hypertext Markup Language (HTML)

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Basic HTML</title>
  </head>
  <body>
    <h1>This is a Basic Header</h1>
    <p>This is a basic paragraph.</p>
  </body>
</html>
```

- HTML é a linguagem utilizada para prover estrutura para páginas web
- HTML utiliza marcações (*tags*), tais como <p> e <h1> para realizar essa função
- Navegadores “leêm” arquivos HTML e produzem a página web visível baseado nas marcações utilizadas

# Versões do HTML

- Durante os anos 2000, HTML 4.01 foi o padrão utilizado para as páginas web
  - HTML 4.01 era limitado em suas possibilidades
- Uma forte demanda por uma experiência rica na web, incluindo áudio, vídeo e maior interatividade levou ao desenvolvimento de uma nova versão do HTML

# Versões do HTML

- Duas grandes organizações envolvidas na padronização do HTML (e outras tecnologias relacionadas)
  - World Wide Web Consortium (W3C)
  - Web Hypertext Application Technology Working Group (WHATWG)



# Recursos Específicos por Navegador

- Microsoft Edge
  - <https://developer.microsoft.com/microsoft-edge/>
- Google Chrome
  - <https://developer.chrome.com>
- Safari
  - <https://developer.apple.com/safari/>

# Edição On-Line

- CodePen
  - <https://codepen.io/>
- JSFiddle
  - <https://jsfiddle.net/>
- Liveweave
  - <http://liveweave.com/>

# IDEs Online

- Codesandbox
  - <https://codesandbox.io/>
- Stackblitz
  - <https://stackblitz.com/>
- Replit
  - <https://replit.com/>

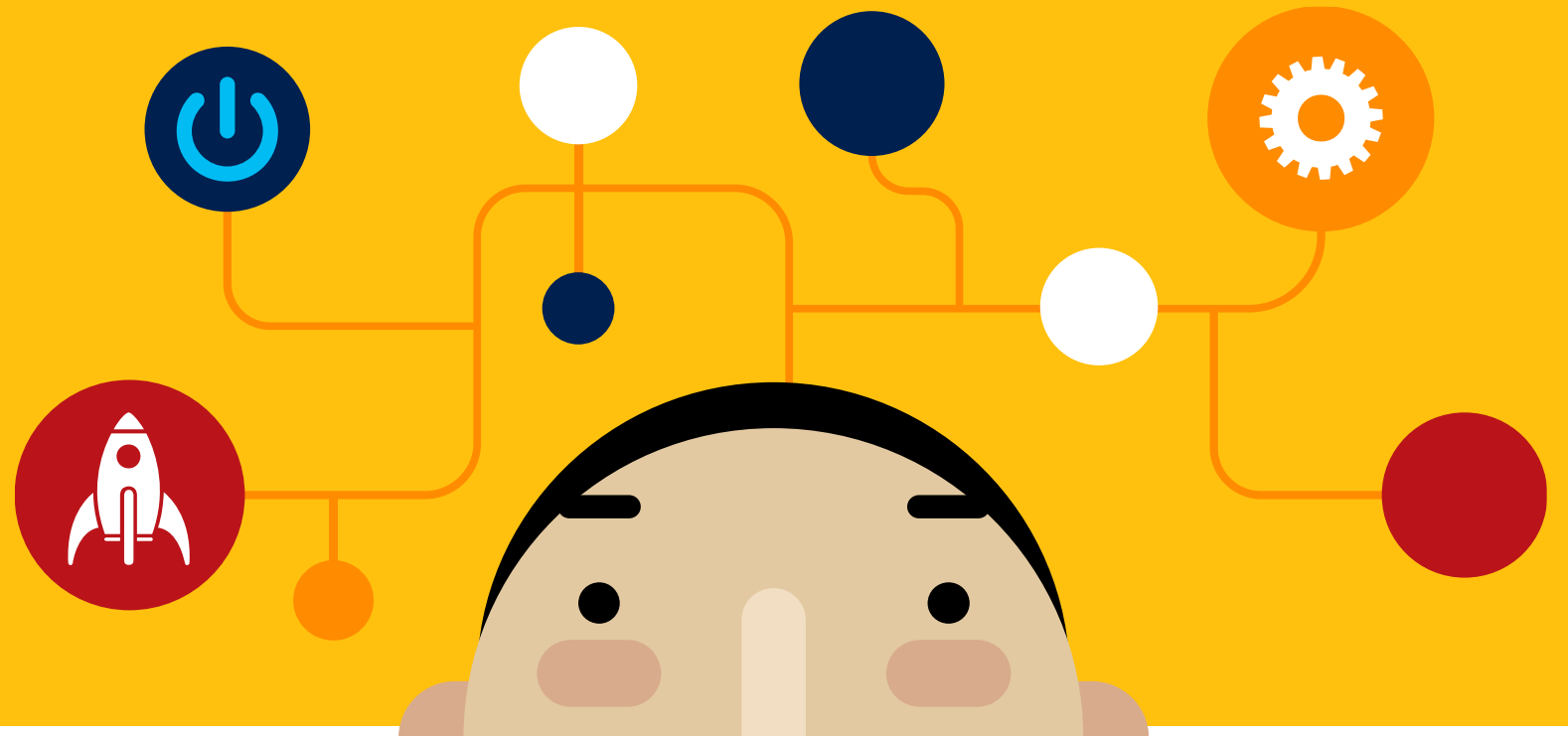
# Implantação

- Vercel
  - <https://vercel.com/>
- Netlify
  - <https://www.netlify.com/>
- Render
  - <https://render.com/>
- GitHub Pages
  - <https://pages.github.com/>

# Saiba Mais

Acesse os tutoriais mantidos pela Google em <https://web.dev/>

Acesse os tutoriais mantidos pela Mozilla em <https://developer.mozilla.org/>



# Marcações Básicas e Estrutura da Página



# HTML Tags

tag de abertura

```
<!DOCTYPE html>
```

```
<html lang="en" >
```

```
<head>
```

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

```
<title>Basic HTML</title>
```

```
</head>
```

```
<body>
```

```
<h1>This is a basic header</h1>
```

```
<p>This is a basic paragraph.</p>
```

```
</body>
```

```
</html>
```

tag de fechamento

- Uma tag é uma **palavra-chave** circundada por símbolos < e >
- Grande parte das tags vem em pares, com uma tag de abertura e uma de fechamento
  - Tags de fechamento são idênticas, mas incluem o símbolo / antes da palavra-chave
- Um par de tags ou uma tag vazia é chamado de **elemento**

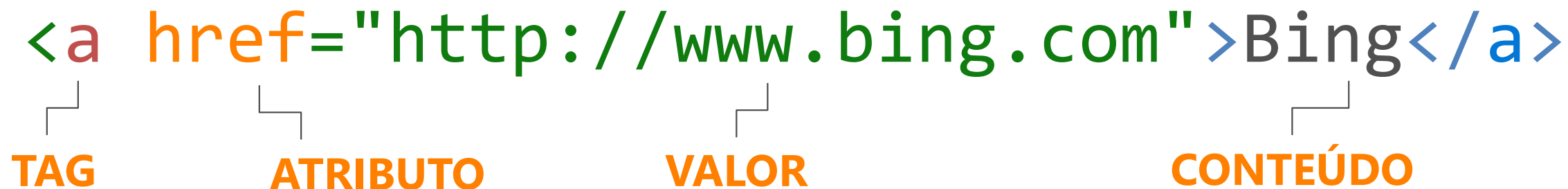
# HTML Tags Comuns

TAG	PROPÓSITO
<html>	Identifica a página como sendo um documento HTML
<head>	Contém informações utilizadas pelo navegador, como referências para código JavaScript e estilos CSS
<title>	Título do documento
<body>	Abrange o conteúdo da página web
<p>	Parágrafos
<a href="URL">	Links
<h1>	Tópico de nível 1
<img>	Imagens



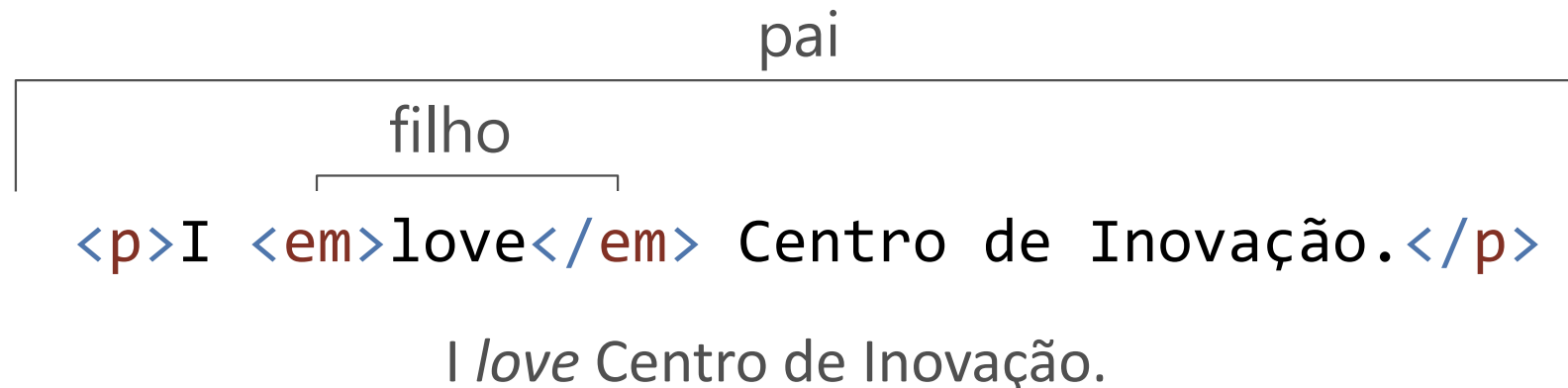
# Utilizando Atributos

- Elementos são utilizados em combinação com **atributos** para descrever informações
  - Em outras palavras, atributos podem ser utilizados para prover informações adicionais para um elemento que uma tag sozinha não consegue
- Cada elemento tem um conjunto específico de atributos suportados
  - HTML5 inclui alguns **atributos globais**, que podem ser utilizados com qualquer elemento
- Atributos são adicionados a tags utilizando a seguinte sintaxe:

`<a href="http://www.bing.com">Bing</a>`  
TAG                      ATRIBUTO                      VALOR                      CONTEÚDO

# Elementos Aninhados

- Criar páginas envolve combinar elementos, atributos e conteúdo
- Quando dois ou mais elementos se aplicam ao mesmo bloco de conteúdo, então eles devem estar aninhados
- **Aninhar** é o processo de colocar um elemento dentro de outro
  - O elemento mais externo é chamado de **pai**, enquanto o elemento mais interno é chamado de **filho**



# Caracteres Especiais no HTML

- Um caractere especial, tal como um sinal de percentagem ou símbolo de copyright, é conhecido como **entidade** em HTML
- Incluir entidades em uma página web requer uma codificação do caractere
- Cada caractere especial pode ser reproduzido utilizando seu nome de entidade OU um código numérico
  - Cada entidade inicia com e-comercial (&) e termina com ponto-e-vírgula (;)

CARACTERE ESPECIAL	DESCRIÇÃO	NOME DA ENTIDADE	CÓDIGO
©	Copyright	&copy;	&#169;
\$	sinal de Dólar	&dollar;	&#36;
%	Sinal de percento	&percnt;	&#37;
&	E-comercial	&amp;	&#38;

# Declaração DOCTYPE

- Uma **declaração doctype** é utilizada para ajudar um navegador determinar quais regras ele deve utilizar para renderizar a página web
- No HTML 4 a declaração requer uma referência a um Document Type Definition (DTD) e é mais complexa
- No HTML5 a declaração é mais simples

## HTML 4

```
<!DOCTYPE html PUBLIC  
"-//W3C//DTD XHTML 1.1//EN"  
"http://www.example.com/TR/xhtml11/DTD/  
xhtml11.dtd">
```

## HTML5

```
<!DOCTYPE html>
```

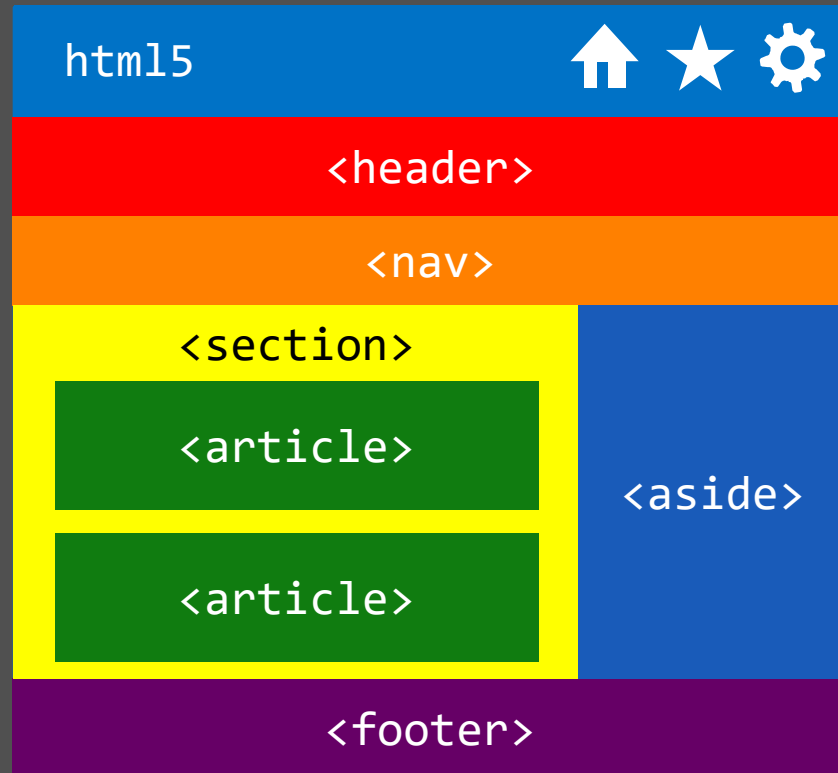
# Exemplo

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>HTML5 Elements in Action</title>
  </head>
  <body>
    <h1>This is a header</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

# HTML Semântico



# Organizando Conteúdo



- HTML5 introduziu novos elementos para organizar o conteúdo de páginas web
- As novas tags para organizar conteúdo são <header>, <section>, <footer>, etc
- Os nomes dessas tags são representativos de **marcações semânticas**

# Marcações Semânticas

- Marcações semânticas garantem que o nome de uma tag é representativo da função que o conteúdo marcado exerce
  - Por exemplo, a tag `<footer>` é utilizada para criar um rodapé para uma página web
- No HTML 4.01 e anterior, desenvolvedores tinham que utilizar a tag `<div>` para realizar diferentes funções
  - O significado da tag `<div>` vem de “divisão”
  - Evite utilizar esse tipo de solução caso exista um element semântico adequado!

## HTML5

```
<footer>Rodapé</footer>
```

## HTML 4.01

```
<div id="footer">Rodapé</div>
```

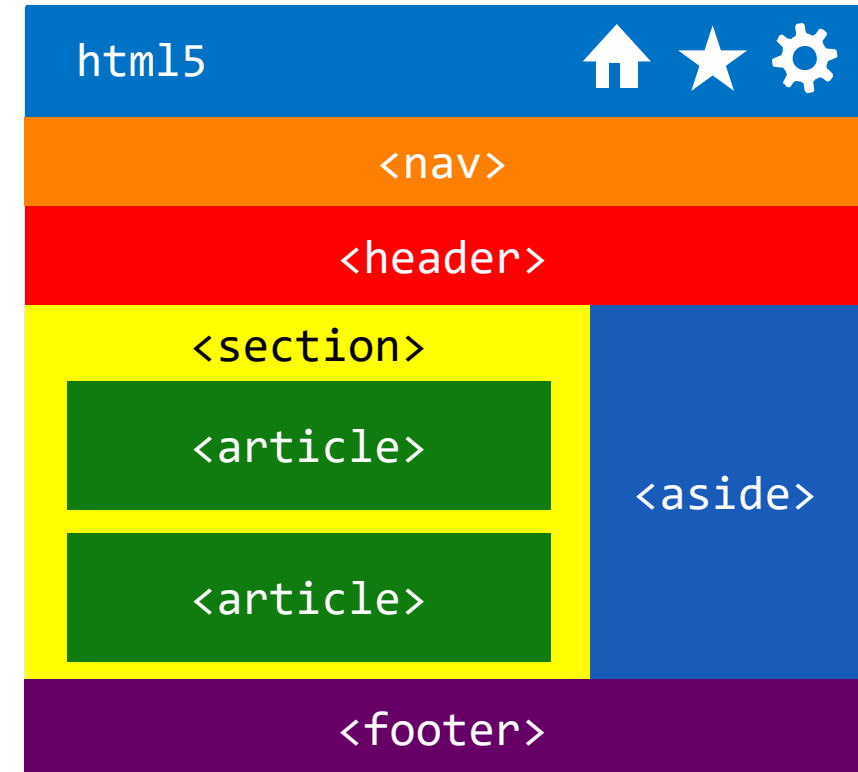
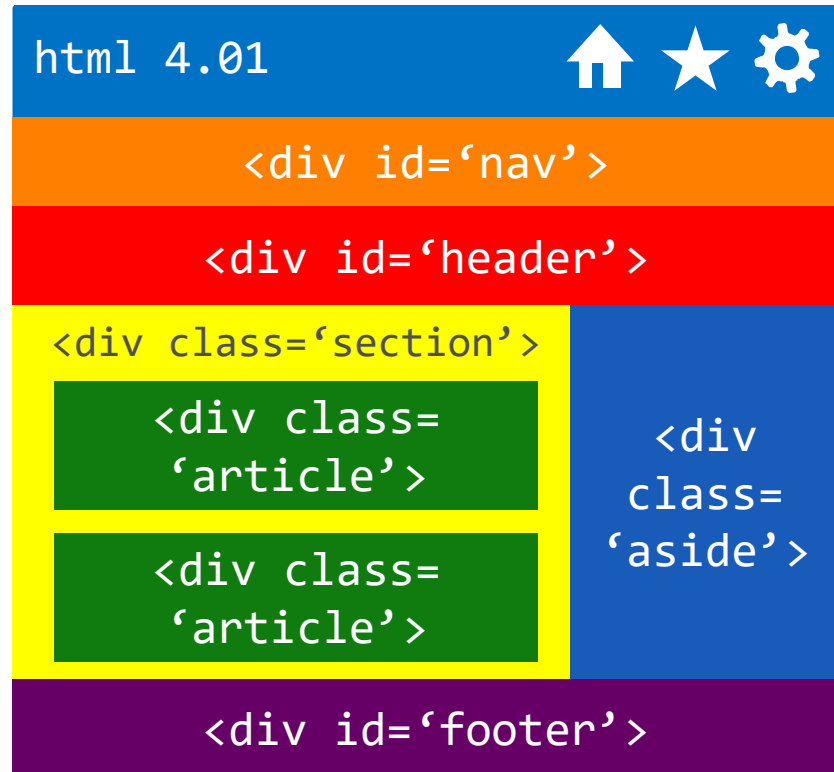


# Tag <div>

- Uso da tag <div> tipicamente requer o uso dos atributos **class** ou **id**
- **class** e **id** são atributos globais, que podem ser utilizados com qualquer elemento HTML
- Os atributos **class** e **id** podem receber qualquer valor definido pelo desenvolvedor
- **class** é utilizado para identificar um grupo de elementos
- **id** é utilizado para identificar elementos individuais

```
<div id="about">  
  <div id="about_stella">  
      
    <div id="slogan">Happy dogs are good dogs</div>  
  </div>  
</div>
```

# Marcação no HTML 4.01 vs. HTML5



# Tags Estruturais

TAG	DESCRIÇÃO
<address>	Define uma área para informações de contato para uma página ou seção
<article>	Define um artigo, tal como um artigo de revista ou jornal, post de blog, ou conteúdo similar
<aside>	Define conteúdo que é separado mas relacionado com conteúdo principal de uma página; similar a uma barra lateral de capítulos ou artigos de uma revista
<details>	Define detalhes adicionais pertinentes ao texto marcado; cria um componente iterativo que o usuário pode abrir ou fechar
<footer>	Define um rodapé para um documento ou seção; pode incluir informações como o autor da página, informações de contato, informações de copyright, etc
<header>	Define um cabeçalho para um documento ou seção; pode conter conteúdo introdutório ou links de navegação
<hgroup>	Agrupa cabeçalhos e subcabeçalhos (usando as tags <h1> a <h6>) gerando cabeçalhos multinível
<nav>	Define um bloco de links de navegação
<section>	Define uma seção em um documento, tal como capítulos, partes de uma tese, ou partes de uma página web cujo conteúdo é distinto de outro
<summary>	Define um cabeçalho visível para propósito de resumo; usuários podem clicar para abrir e fechar

# Elementos header e footer

O elemento **<header>** define o cabeçalho para uma página web, artigo ou documento

- Um cabeçalho tipicamente contém títulos, logotipos, ou fotos e pode ser a primeira coisa que o usuário enxerga ao visitar um site

O elemento **<footer>** define um rodapé para uma página web, artigo, ou documento e está tipicamente localizada ao final

- Um rodapé tipicamente possui informações sobre uma página, tais como copyright, links adicionais, etc

```
<article>
  <header>
    <h1>Learning HTML5</h1>
    <h2>Semantic Elements</h2>
  </header>
  <p>These HTML5 tags are great!</p>
  <footer>
    <p>Published: <time
      datetime="2015-06-
      20">June 20, 2015</time></p>
  </footer>
</article>
```

# Elemento `section`

- O elemento `<section>` define seções em um documento ou página web
- Para cada situação, uma determinada tag é mais adequada:

SITUAÇÃO	USE
Separar conteúdo que é independente de outro conteúdo na página	<code>article</code>
Planos para sindicalizar um bloco de conteúdo	<code>article</code>
Criar uma barra lateral	<code>aside</code>
Empacotar e posicionar múltiplas seções que não estão relacionados entre si	<code>div</code>

# Elemento `hgroup`

- O elemento `<hgroup>` é utilizado para agrupar cabeçalhos
- Permite organizar tags de cabeçalho, mas não impactam a forma que aparecem dentro da página

```
<section>
  <hgroup>
    <h1>HTML5</h1>
    <h3>Structuring a Web
      page</h3>
  </hgroup>
  <article>
    <p>With semantic tags,
      structuring a web page is
      easier than ever before!</p>
  </article>
</section>
```

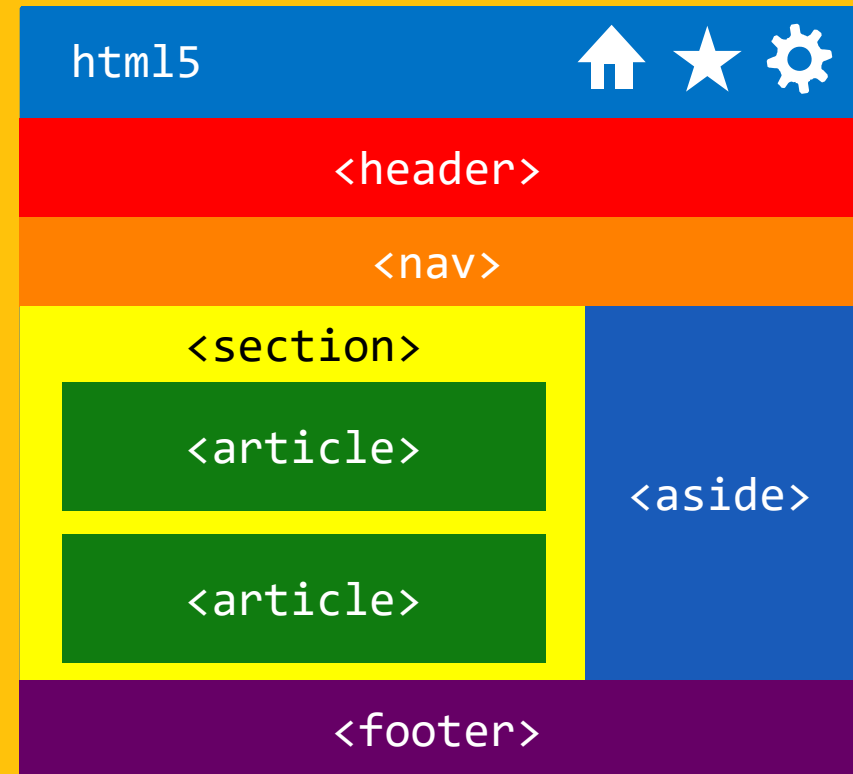
# Elemento **nav**

- O elemento **<nav>** é utilizado para organizar links de navegação entre páginas
- A tag **<nav>** não deve ser utilizada sobre cada link individual, mas como um agrupamento

```
<nav>  
  <a href="/html/">HTML</a>  
  <a href="/css/">CSS</a>  
  <a href="/javascript/">JavaScript</a>  
</nav>
```

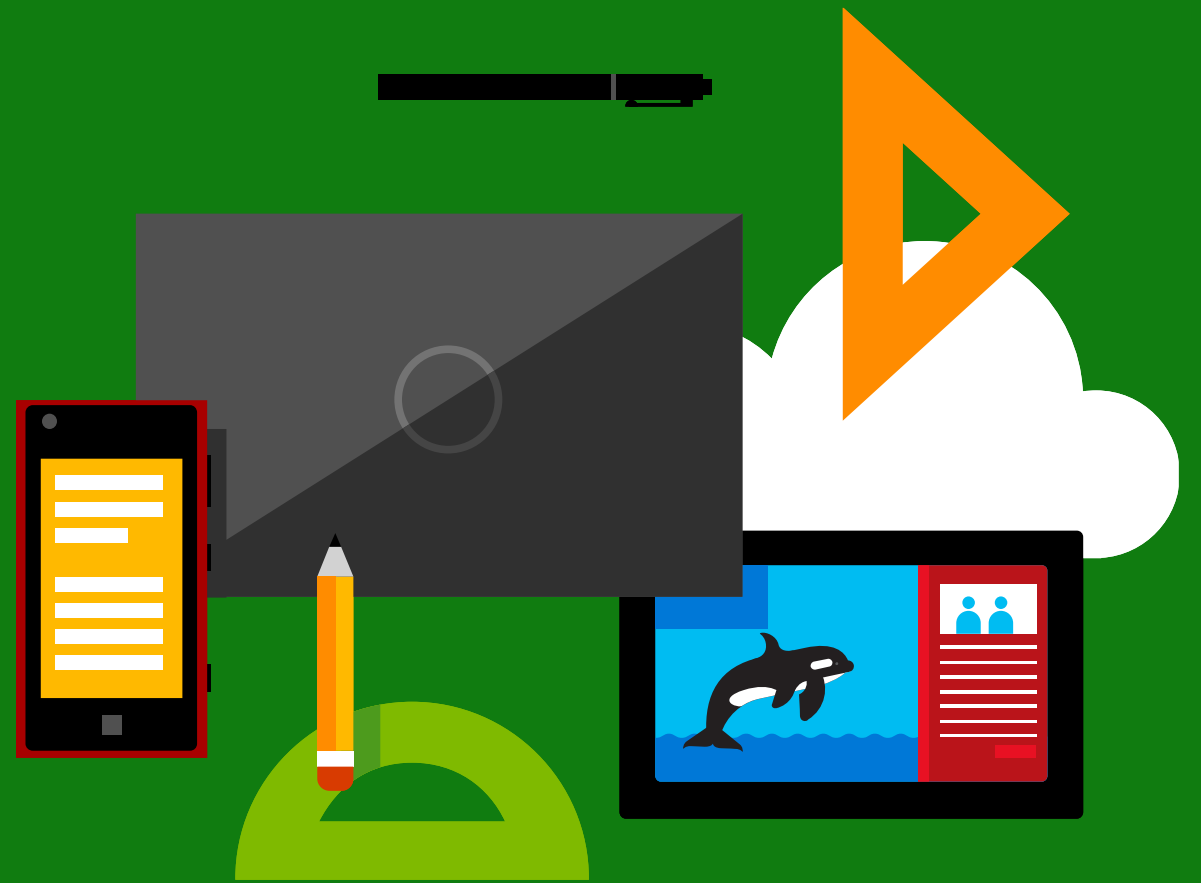
# Elementos `article` e `aside`

- O elemento `<article>` define um conteúdo independente e auto-contido
  - Um exemplo pode ser um artigo de notícia ou um post de blog
- O elemento `<aside>` é utilizado para definir um subconjunto do conteúdo geral de uma página
  - É importante mencionar que a tag `aside` não altera como o conteúdo aparece na página
  - Se o propósito é mudar o posicionamento de um elemento, deve ser utilizado CSS





# Elementos de Texto



# Elementos de Texto

ELEMENTO	FUNÇÃO NO HTML 4	FUNÇÃO NO HTML5
<b>	Enfatizar texto tornando-o destacado em bold	texto "estilisticamente deslocado"
<i>	Enfatizar texto tornando-o destacado em itálico	"voz alternativa ou humor"
<strong>	N/D	Indica texto de grande importância, enquanto enfatiza tornando-o destacado em bold
<em>	N/D	Indica texto enfático, enquanto enfatiza tornando-o destacado em itálico

# Exemplo

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>HTML Example</title>
</head>
<body>
  <h1>This is a sample page.</h1>
  <p>This page includes <em>some</em> <strong>nested
  elements</strong>.</p>
</body>
</html>
```

# Elementos de Texto em Desuso

- Com a adição de novos elementos, a padronização indica elementos ou atributos para remoção
- Elementos e atributos são removidos pois se tornam inúteis
- O processo é chamado de **depreciação**
- Elementos depreciados podem ainda ser renderizados em navegadores antigos, mas sugere-se que não sejam mais utilizados em navegadores mais novos

ELEMENTO DEPRECIADO	NOVO ELEMENTO
<acronym>	<abbr>
<applet>	<object>
<basefont>	Utilize CSS
<big>	Utilize CSS
<center>	Utilize CSS
<dir>	<ul>
<font>	Utilize CSS
<strike>	<del> ou utilize CSS

# Tabelas e Listas



# Criando Tabelas

- Tabelas consistem de colunas e linhas em uma grade
- Para criar uma tabela em HTML, utiliza-se uma combinação de tags:

TAG	DESCRIÇÃO
<code>&lt;table&gt;</code>	Cria uma tabela
<code>&lt;tr&gt;</code>	Cria uma linha
<code>&lt;th&gt;</code>	Cria um cabeçalho
<code>&lt;td&gt;</code>	Cria uma célula dentro de uma linha -> uma coluna
<code>&lt;colgroup&gt;</code>	Aplicar estilos a uma seleção de colunas
<code>&lt;thead&gt;</code>	Marca um grupo de cabeçalhos de linhas
<code>&lt;tfoot&gt;</code>	Marca um grupo de rodapés de linhas
<code>&lt;tbody&gt;</code>	Utilizado para formatar grupod de linhas
<code>&lt;caption&gt;</code>	Marca texto como legenda

# Exemplo

```
<caption>Number of hours worked on thesis.</caption>
<table border="1">
  <tr>
    <th>Month</th>
    <th>Hours</th>
  </tr>
  <tr>
    <td>April</td>
    <td>100</td>
  </tr>
  <tr>
    <td>June</td>
    <td>45</td>
  </tr>
  <tr>
    <td>July</td>
    <td>120</td>
  </tr>
</table>
```

# Criando Listas

- Existem dois tipos primários de listas no HTML5: ordenadas e não-ordenadas
- Listas ordenadas usam a tag **<ol>** e ordenam item na lista via números
- Listas não-ordenadas usam a tag **<ul>** e apresentam os itens em uma listagem de bolinas
- Itens são adicionados às listas via tag **<li>**

## LISTAS ORDENADAS

```
<h3>Favorite Foods</h3>
<ol>
  <li>Pizza</li>
  <li>Cake</li>
</ol>
```

## LISTAS NÃO-ORDENADAS

```
<h3>Seattle To-Do List</h3>
<ul>
  <li>Visit Space Needle</li>
  <li>Buy rain jacket</li>
</ul>
```



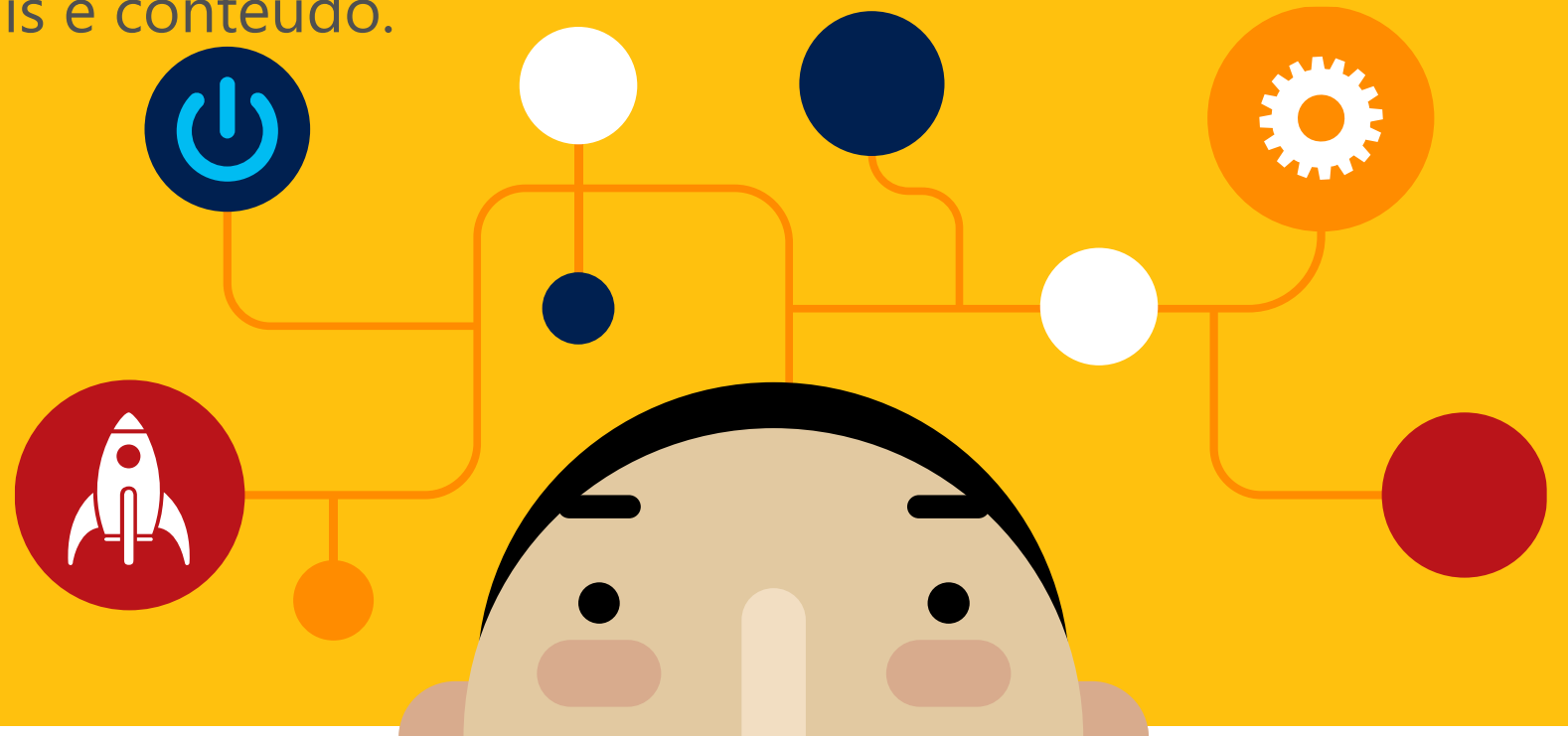
# Exemplo

```
<body>
  <h3>Favorite Foods</h3>
  <ol>
    <li>Pizza</li>
    <li>Cake</li>
  </ol>

  <h3>Seattle To-Do List</h3>
  <ul>
    <li>Visit Space Needle</li>
    <li>Buy rain jacket</li>
  </ul>
</body>
```

# Laboratório

Pense na estrutura e informações de um currículo. Crie um documento HTML com as informações essenciais do seu currículo. Não se preocupe com a aparência visual, apenas com as marcações estruturais e conteúdo.



# Elementos Gráficos



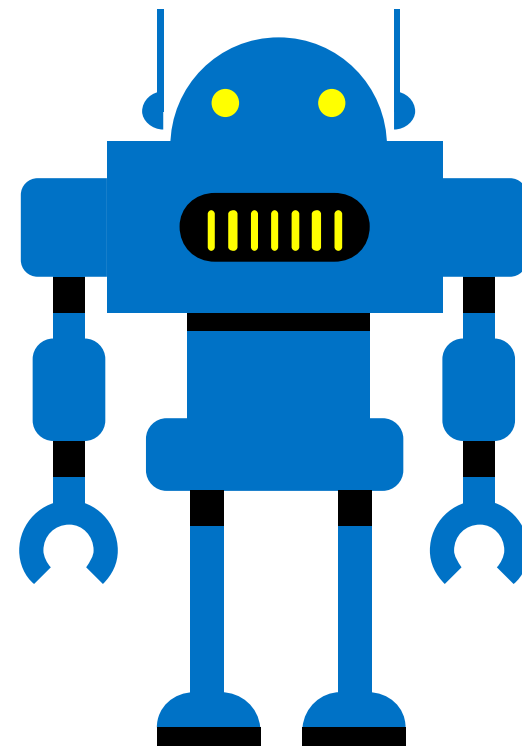
# Imagens e Gráficos no HTML

- Existem duas grandes categorias de imagens que podem ser utilizadas:
  - raster (bitmap)
  - vector
- **Imagens rasterizadas** são compostas de pixels, enquanto **imagens vetorizadas** são compostas de elementos geométricos diversos

Raster



Vector



# Raster vs. Vector

## Raster

- Fotografias são imagens rasterizadas
- Formatos incluem JPG, PNG, GIF, e BMP
- Imagens rasterizadas se tornam pixeladas quando aumentadas

## Vector

- Ilustrações digitais são imagens vetorizadas
- Imagens vetorizadas mantêm a qualidade quando aumentadas

# Elemento **img**

- Adicione imagens a uma página via tag **<img>**
  - NOTA: uma tag de fechamento não é necessária
- A tag **<img>** requer uso dos atributos **src** e **alt**
  - **src** significa fonte
  - **alt** significa alternativo
- **src** define o caminho para o arquivo de imagem
- **alt** define um texto acessível

# Atributos do elemento **img**

ATRIBUTO	VALOR	DESCRIÇÃO
src	URL	Especifica a localização da imagem
alt	Texto	Especifica texto acessível para a imagem
height	pixels	Especifica a altura da imagem
width	pixels	Especifica a largura da imagem
ismap	ismap	Especifica o uso de um mapa de imagem gerenciado pelo lado-servidor
usemap	#mapname	Especifica o uso de um mapa de imagem gerenciado pelo lado-cliente

# Elementos **figure** e **figcaption**

- O elemento **img** pode ser usado em combinação com dois elementos, **figure** e **figcaption**, para organizar imagens e prover títulos
- O elemento **figure** especifica o tipo de figura que está sendo adicionada, e pode também ser utilizado para agrupar imagens lado a lado
- O elemento **figcaption** pode ser utilizado para adicionar legendas antes ou após a imagem

```
<figure>  
  <figcaption>Who wouldn't want to take this dog for a walk?</figcaption>  
    
</figure>
```



# Canvas e SVG



# Elemento **canvas**

- O elemento canvas cria um container em branco para o desenho de gráficos
- É um elemento que pode ser utilizado para gerar gráficos via código em JavaScript
  - Desenhar no canvas é feito via Canvas API
- É bastante utilizado para criar animações e jogos 2D

```
<canvas id="myCanvas" width="500px" height="300px"></canvas>
```

gera um container em branco

# Exemplo

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Canvas Test</title>
  <script>
    function f1() {
      var canvas = document.getElementById("smlRectangle");
      context = canvas.getContext("2d");
      context.fillStyle = "rgb(0,0,255)";
      context.fillRect(10, 20, 200, 100);
    }
  </script>
</head>
<body onload = "f1();">
  <canvas id="smlRectangle" height="100" width="200 "></canvas>
</body>
</html>
```

# Alternativa para Navegadores Antigos

- O elemento canvas não funciona em navegadores antigos
- Para evitar problemas na renderização da página, deve-se adicionar conteúdo a ser renderizado no lugar do canvas
- O conteúdo pode ser imagem ou texto

```
<canvas id="sm1Rectangle2" height="100" width="200">
```

```
  
```

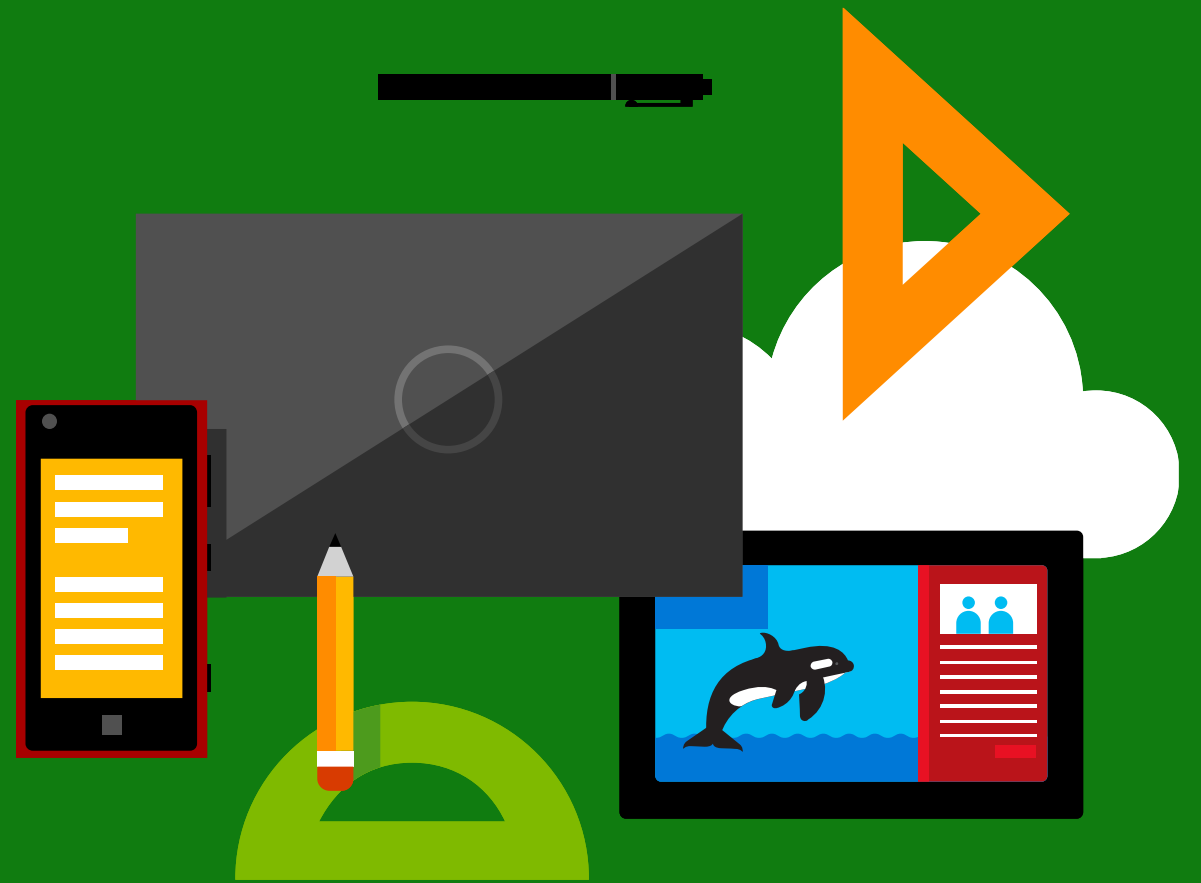
```
</canvas>
```

# Scalable Vector Graphics

- Scalable Vector Graphics (SVG) é uma linguagem para descrever gráficos vetoriais 2D em XML
- Com SVG, é possível prover instruções de desenho ao invés de um caminho para um arquivo
- Utiliza-se o elemento **svg**

```
<svg height="1000px" width="1000px">  
  <rect id="myRect" height="100px" width="100px" fill="blue"/>  
</svg>
```

# Mídia no HTML5



# Mídia no HTML5

- Multimedia é um componente essencial na experiência do usuário com páginas web
- Antes do HTML5, navegadores dependiam de plugins e tocadores de mídia para manipular áudio e vídeo
- Agora, navegadores que suportam HTML5 podem prove acesso à multimídia com as tags `<video>` e `<audio>`



# Vídeo

- Para embutir vídeo em um documento HTML utiliza-se o elemento **<video>**
- Use o atributo **src** para indicar o local do arquivo de vídeo
- Os atributos **height** e **width** ajudam a determinar a aparência do vídeo dentro da página web

```
<video src="cat_vid.mp4" height="300" width="400"></video>
```



# Vídeo e Atributos de Controle

Existe um conjunto de outros atributos que podem ser utilizados para controlar o vídeo

ATRIBUTO	DESCRIÇÃO
<b>poster</b>	Apresenta uma imagem estática enquanto o vídeo carrega
<b>autoplay</b>	Inicia o vídeo automaticamente ao carregar a página
<b>controls</b>	Apresenta controles na página para manipulação do vídeo
<b>loop</b>	Toca o vídeo em auto-repetição

HTML

```
<video  
  src="cat_vid.mp4  
  width="400"  
  height="300"  
  poster="meow.jpg"  
  autoplay  
  controls  
  loop>  
</video>
```

# Formatos de Vídeo

- Um conjunto de formatos são suportados pelos navegadores, incluindo MP4, H.264, OGG, e WebM
- Quando se especifica o tipo de vídeo, deve-se especificar o **codec**
  - Um codec é a tecnologia utilizada na compressão dos dados
- É uma boa prática utilizar a tag **<source>** em combinação com seu atributo **type**

## HTML

```
<video
  width="400"
  height="300"
  poster="meow.jpg"
  autoplay="autoplay"
  controls="controls"
  loop="loop">
  <source
    src="cat_vid.mp4"
    type="video/mp4"  />
</video>
```

# Compatibilidade de Navegadores

- Nem todo formato de vídeo é compatível com todos os navegadores
- O formato MP4 é o mais comumente utilizado
- Para garantir que um vídeo seja compatível, utiliza-se múltiplos formatos especificados no elemento <source>

## HTML

```
<video
  width="400"
  height="300"
  poster="meow.jpg"
  autoplay="autoplay"
  controls="controls"
  loop="loop">

  <source src="cat_vid.mp4"
  type="video/mp4" />

  <source src="cat_vid.ogg"
  type="video/ogg;
  codecs="theora, vorbis">
</video>
```

# Áudio

- O elemento para áudio do HTML5 tem funcionalidade semelhante ao elemento para vídeo
- Inclua a tag **<audio>** com o caminho para o arquivo de áudio
- Modifica-se o comportamento através dos atributos como:
  - autoplay
  - controls
  - loop

```
<audio src="myaudio.mp3" controls"></audio>
```

# Formatos de Áudio

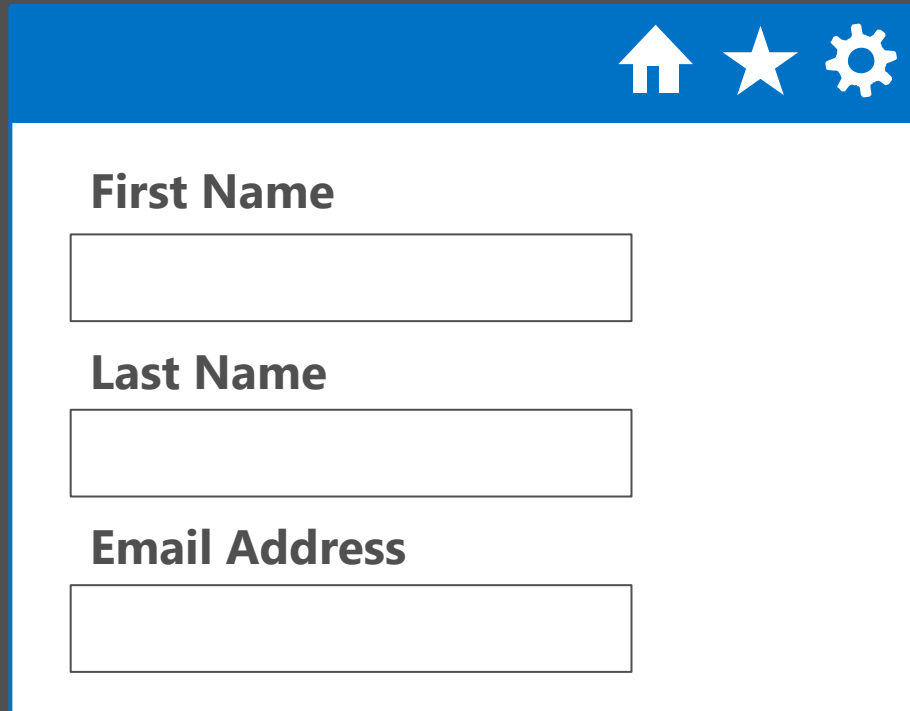
- Existem três formatos principais de áudio suportados pelos navegadores:
  - OGG
  - MP3
  - WAV
- Nem todo navegador suporta todo tipo de áudio
- Utiliza-se o atributo **source** para incluir múltiplos formatos

```
<audio controls>  
  <source src="myaudio.mp3" type="audio/mp3"/>  
  <source src="myaudio.ogg" type="audio/ogg"/>  
</audio>
```

# Entradas e Formulários



# Formulário Web



A web form interface with a blue header bar containing three icons: a home icon, a star icon, and a gear icon. The form contains three input fields, each with a label above it:

- First Name**
- Last Name**
- Email Address**

- Um **formulário web** é uma página que possui campos de entrada de dados do usuário
- **Entradas do formulário**, ou os dados providos pelos usuários, são enviados para o servidor onde são processados

# Criação de Formulários

- Cria-se um formulário utilizando o elemento **<form>**
- É prática comum identificar um formulário com o atributo id
- O elemento **<label>** apresenta um rótulo de texto para cada campo
- O elemento **<input>** é utilizado para ditar o tipo de campo de entrada

```
<form id="contact" method="post" action="">
  <label for="firstName">First Name</label>
  <input type="text" name="firstName" /><br/>
  <label for="lastName">Last Name</label>
  <input type="text" name="lastName" /><br/>
  <label for="email">Email</label>
  <input type="email" name="email" /><br/>
</form>
```



# Tipos de Input

INPUT TYPE	DESCRIÇÃO
text	Cria um campo de texto
password	Cria um campo de texto para senhas
submit	Cria um botão de submissão
radio	Cria um botão de rádio de seleção
checkbox	Cria uma caixa de seleção
date	Cria um campo para data
email	Cria um campo para e-mail
search	Cria um campo de busca

# Atributos e Valores para Input

- Existe um conjunto de atributos que podem ser utilizados com o elemento **<input>** para adicionar funcionalidades a um formulário
- Use o atributo `autofocus` para apontar o foco para um campo específico ao carregar uma página
- Use o atributo `required` quando um campo é obrigatório
- Use o atributo `placeholder` para adicionar um texto de marca d'água que ajuda o usuário com o conteúdo de um campo

## autofocus

```
<input type="text" name="firstName"
      autofocus="autofocus" />
```

## required

```
<input type="email" required />
```

## placeholder

```
<input type="text" name="firstName"
      placeholder="First Name" />
```

# Exemplo

```
<body>
  <h1>Form Demo!</h1>
  <form id="contact" method="post" action="">
    <label for="firstName">First Name</label>
    <input type="text" name="firstName" placeholder="First Name"/><br/>
    <label for="lastName">Last Name</label>
    <input type="text" name="lastName" placeholder="Last Name"/><br/>
    <label for="email">Email</label>
    <input type="email" name="email" placeholder="Email Address"/><br/>
    <label for="password">Password</label>
    <input type="password" name="password" placeholder="Password"/><br/>
    <input type="submit" name="submit" value="submit">
  </form>
</body>
```

# Validação de Formulários



# Validação de Formulários

- Validação é o processo de verificar se a informação obtida de um formulário está no formato adequado ou possui um valor adequado
- Questões comuns encontradas:
  - Campos vazios
  - Formatos de endereços de e-mail inválidos
  - Datas inválidas
  - Texto vs. números e vice-versa
- No HTML 4.01, validação obrigava o uso de código JavaScript, mas HTML5 oferece **validação automática**

Illustration of a web form with validation feedback:

- Header: Blue bar with icons for home, star, and settings.
- Form Fields:
  - Last Name**: Input field containing "Doe".
  - Email Address**: Empty input field.
- Submit Button: Grey button labeled "SUBMIT".
- Validation Message: A green callout box pointing to the empty email field with the text: "Please provide a valid email address."

# Validação no Lado-Cliente

First Name

Jane

Last Name

Doe

Email Address

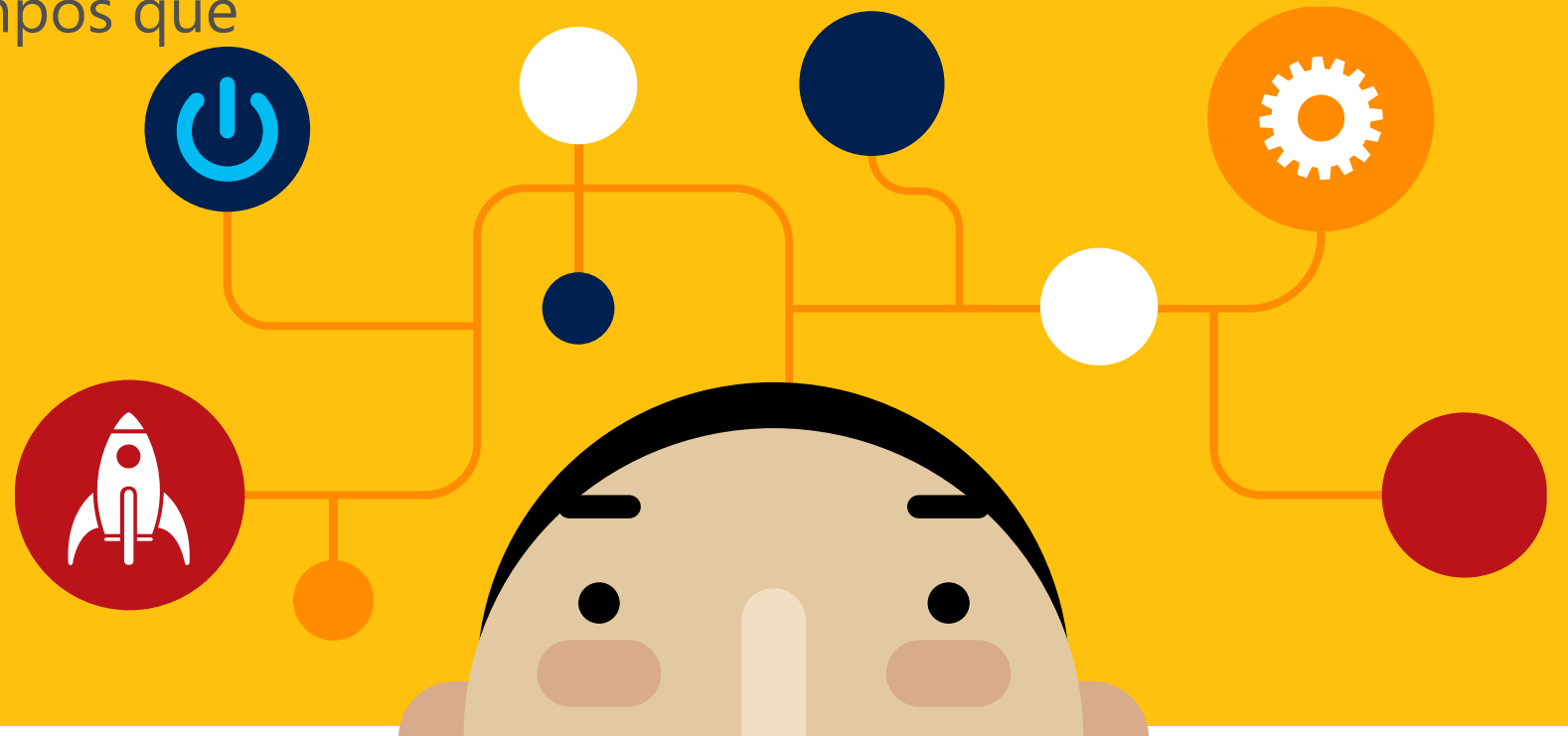
jane@live

Please provide a valid email address.

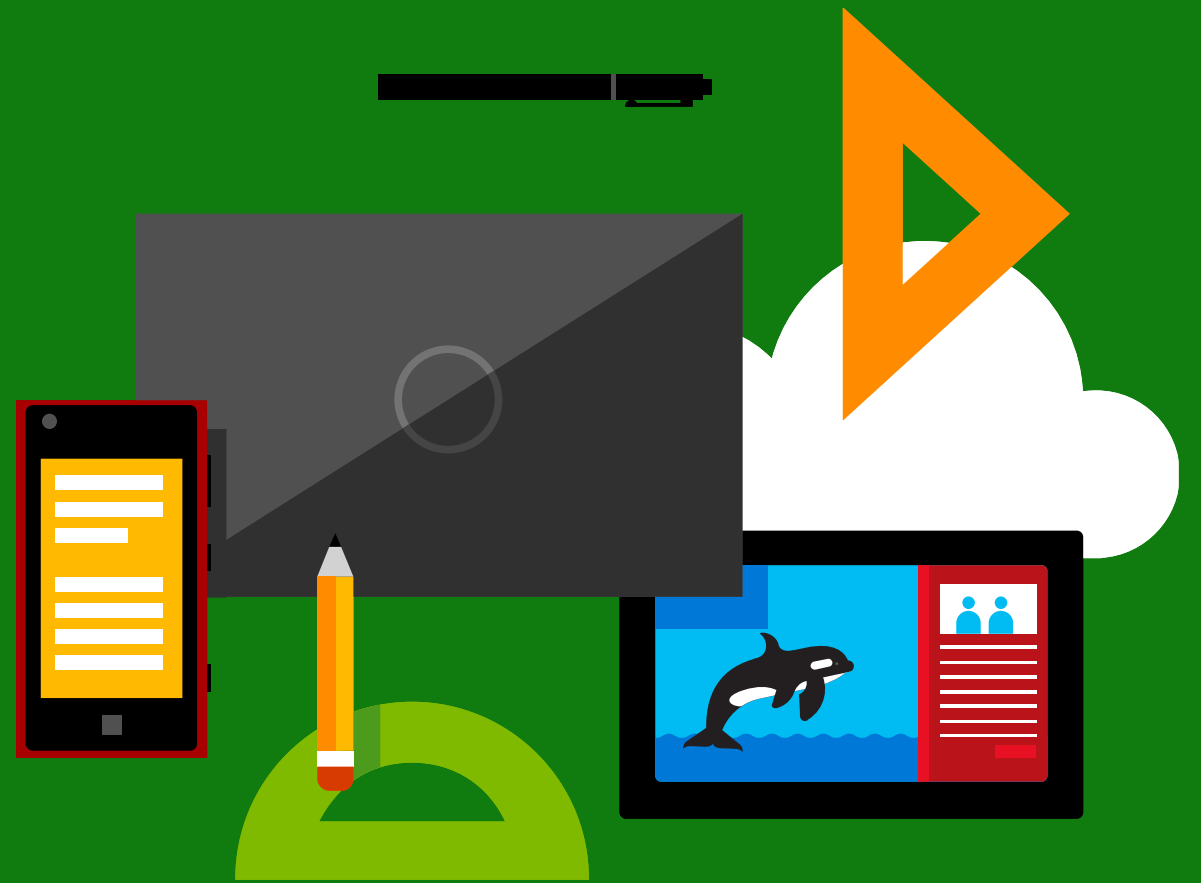
- Quando um navegador valida os dados providos por um usuário, chamamos de **validação no lado-cliente**
  - Se a validação é executado por um servidor, então chamamos de validação no lado-servidor
- Se um usuário informa um valor em formato incorreto em um campo, então o navegador instrui o usuário sobre o erro
  - O navegador determina se o dado é válido através dos atributos associado ao campo de entrada

# Laboratório

Pense nas informações necessárias para efetuar um cadastro pessoal em um sistema de compras on-line. Construa um formulário HTML para a coleta das informações do usuário. Procure utilizar campos que considere mais adequados.



# Cascading Style Sheets





# Cascading Style Sheets (CSS)

- **Cascading Style Sheets (CSS)** é uma linguagem que define como os elementos HTML são estilizados
  - Em outras palavras, HTML estrutura um documento, enquanto CSS o formata
- CSS pode ser aplicado a elementos HTML individuais (diretamente), ou pode ser mantido em um arquivo separado e aplicado através de referências
  - Arquivos que contêm CSSs são chamados de “folhas de estilos” e usam a extensão .css
- Ao contrário do HTML, CSS utiliza **regras** ao invés de tags

example.css

```
h1, h2, h3 {  
    font-family:  
    Arial;  
    color: yellow;  
}  
  
p {  
    font-family:  
    Arial;  
    font-size: 12px;  
}
```

# CSS3

- CSS3 é a versão do CSS que foi definida em conjunto com HTML5
- CSS3 é compatível com versões anteriores
- CSS3 apresenta um conjunto de novos efeitos, incluindo transformações 2D e 3D, bem como animações

# Prefixos Específicos

- Assim como HTML5, CSS3 pode não ser compatível com todos navegadores
- Novas propriedades são adicionadas o tempo todo, enquanto outras são modificadas
- Muitos navegadores oferecem propriedades alternativas
- O nome dessas propriedades deve aparecer com um prefixo específico
- Um prefixo específico é simplesmente uma palavra-chave circundada por traços

NAVEGADOR	PREFIXO
Internet Explorer, Edge (antigo)	-ms-
Edge	-webkit-
Firefox	-moz-
Opera (antigo)	-o-
Chrome	-webkit-
Safari	-webkit-

# Ligando CSS com HTML

- Existem vários modos de incorporar CSS ao HTML:

1. Estilo *inline*
2. Use do elemento `<style>` aninhado ao elemento `<head>`
3. Elemento `<link>` referenciando arquivo CSS separado

- Colocar CSS em um arquivo separado possui vantagens:

- Troca do estilo permite alterar um documento inteiro
- Times podem trabalhar em responsabilidades separadas: conteúdo e design visual

1 `<p style="color: red">`

2 

```
<style>
  h1 {
    font-family: 'Segoe
UI';
    color: #808080;
  }
</style>
```

3 

```
<link href="StyleSheet.css"
      rel="stylesheet"
      type="text/css">
```

# Como ligar HTML ao arquivo CSS?

- Arquivos CSS são ligados ao arquivo HTML via elemento **<link>**
- O atributo **href** aponta para o local do arquivo CSS
  - É extremamente importante que o nome e localização do arquivo estejam corretos ou nenhum estilo será aplicado
- O atributo **rel** deve ser configurado com "stylesheet", enquanto o atributo **type** deve ser configurado com "text/css"
  - O atributo type é opcional e pode ser entendido pelo contexto

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css">
```

# Ordem de Aplicação (“cascata”)

- Geralmente:

- Estilo in-line (dentro do elemento HTML)
- Folhas de estilo internas ou externas (indicadas no elemento head)
- Valor padrão do navegador

# Valores

Cada propriedade de um elemento HTML possui diferentes tipos de valores

Exemplos:

- [https://developer.mozilla.org/en-US/docs/Learn/CSS/Building\\_blocks/Values\\_and\\_units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units)

# Seletores e Declarações





# Seletores e Declarações

- Existem duas partes de uma regra CSS:
  - seletores
  - declarações
- Um **seletor** referencia o elemento HTML a ser estilizado
- Uma **declaração** é o estilo que se deseja aplicar ao elemento
  - declarações possuem duas partes: uma propriedade seguida por dois-pontos (:), um espaço, e um valor seguido por um ponto-e-vírgula (;)
  - declarações aparecem entre chaves {}



# Seletores e Declarações

- Elementos HTML podem ser referenciados por seletores de diversas maneiras, incluindo:
  - Nome da tag, tal como `p`, `h1`, `table`, etc.
  - Seletor de id, tal como `#navbar`, o qual inclui o símbolo de hashtag (`#`) como prefixo
  - Seletor de classe, como `.happy`, o qual inclui o ponto (`.`) como prefixo
- `id` e `class` são ambos atributos universais do HTML
  - `id` é utilizado para identificar elementos de forma única
  - `class` deve ser utilizado para categorizar elementos em grupos que serão estilizados de forma semelhante

nome da tag

```
h1 {  
  color: red;  
  font-family: sans-serif;  
  text-align: left;  
}
```

seletor id

```
#navbar {  
  background-color: green;  
}
```

seletor class

```
.happy {  
  font-size: 14px;  
}
```

# Seletores e Declarações

- Nome do elemento e class podem ser combinados no selector
- Lista de seletores podem ser informados

nome e classe

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

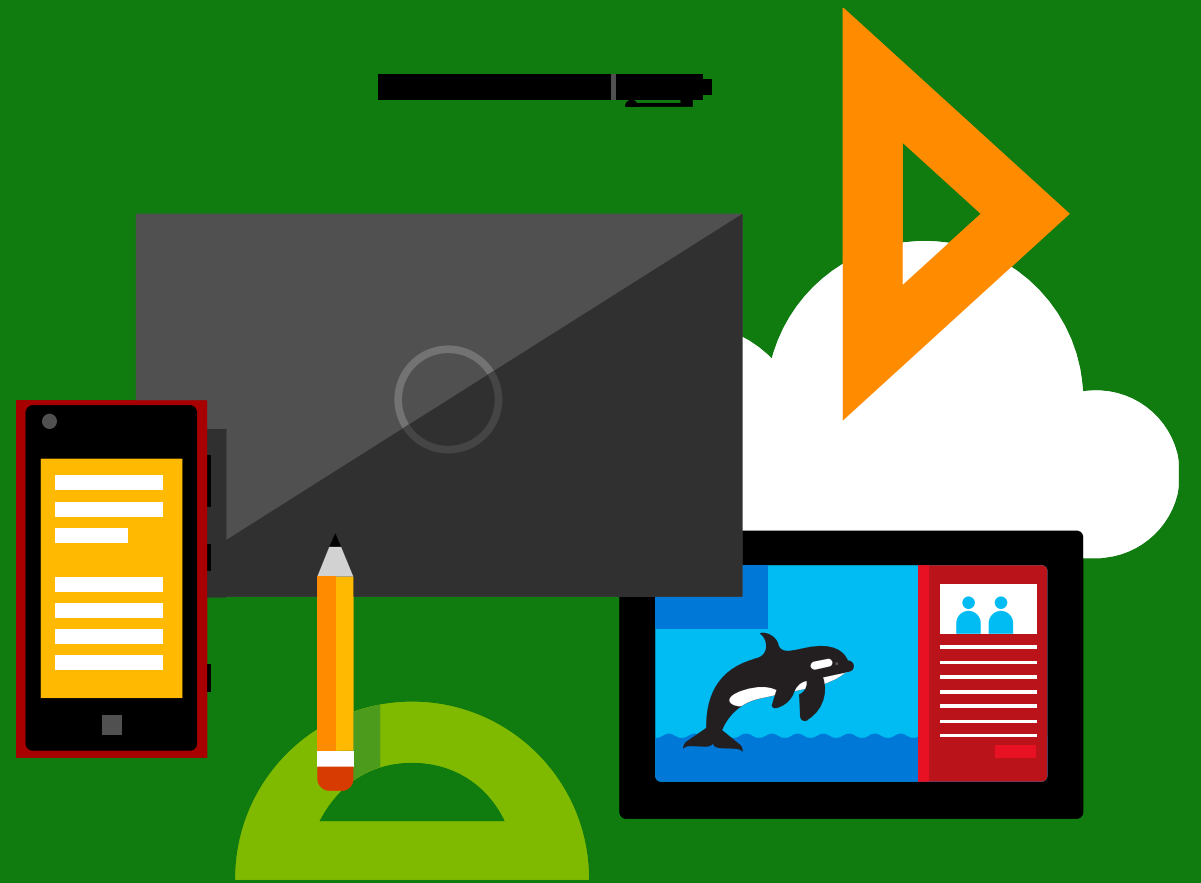
agrupamento

```
p,h1,h2 {  
    text-align: center;  
}
```

# Seletores e Declarações

- O conjunto de opções de seletores é bastante extensa e permite obter qualquer elemento da árvore DOM do HTML de formas alternativas
- Especialmente as pseudo-classes que permite selecionar elementos em função do estado atual e os pseudo-elementos que permitem selecionar partes de um elemento
- Consulte a lista em [https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp)

# Fontes e Famílias de Fontes



# Fontes e Famílias de Fontes

- Uma fonte é um conjunto de caracteres com um estilo e tamanho particulares
- O principal meio de especificar fontes em CSS é utilizando a propriedade **font-family**
- Os três tipos mais comuns de fontes são **serif**, **sans serif**, e **monospace**

Tipo de Fonte	Exemplo	Descrição
Serif	Times New Roman	possui decoração ao final de alguns caracteres
Sans Serif	Arial	não possui decoração ao final dos caracteres
Monospace	Courier	cada letra ocupa o mesmo espaço de tela

# Regra **@font-face**

- Anterior ao CSS3, desenvolvedores tinham que utilizar somente fontes disponíveis no dispositivo do usuário
  - Estas fontes são conhecidas como “web-safe”
- Essa característica era um grande limitador
- Regra **@font-face** do CSS3 permite que um desenvolvedor utilize qualquer fonte desde que ela esteja disponível no servidor web

```
@font-face {  
    font-family: "font-family-name";  
    src: url("http://website/fonts/fontfile")  
}
```

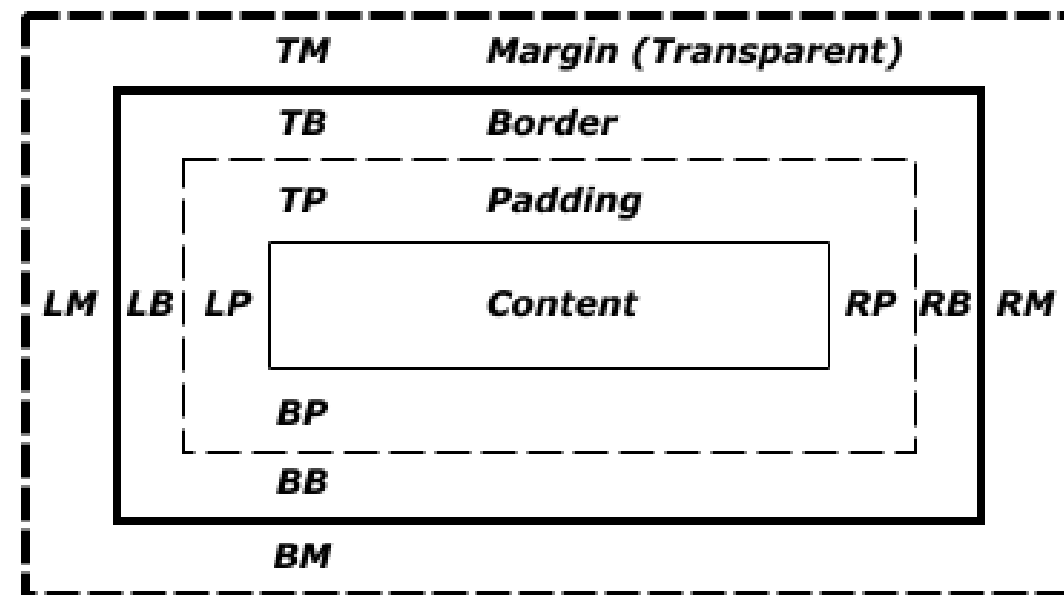
# CSS Box Model





# CSS Box Model

- O **CSS Box Model** define as regras de como um conteúdo é formatado dentro de uma página web
- Cada elemento do HTML está dentro de uma caixa com vários componentes, incluindo padding, border, e margin
  - **Padding** é o espaço entre o conteúdo e sua borda (é transparente)
  - **Border** é uma borda que envelopa a área representada pelo padding
  - **Margin** representa a margem que é o espaço entre a área da borda e outra caixa de outro elemento HTML



- Margin edge
- Border edge
- - - Padding edge
- Content edge

# Elementos de Bloco e Inline

- Com o CSS Box Model, existem duas categorias de elementos do HTML
  - block-level
  - inline
- **Block-level** sempre iniciam em nova linha e ocupam todo espaço horizontal
- **Inline** não iniciam uma nova linha e ocupam somente o espaço horizontal necessário

## BLOCK-LEVEL EXEMPLOS

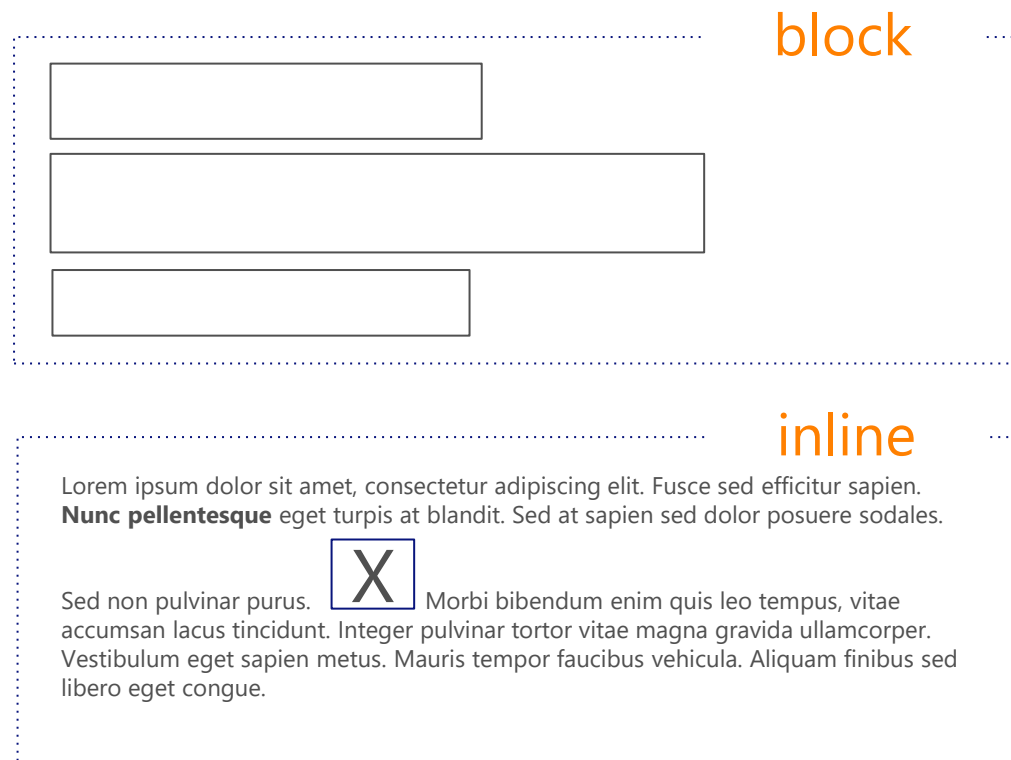
- `<div>`
- `<p>`
- `<ul>`, `<ol>`, `<dl>`
- `<h1>` - `<h6>`
- `<form>`

## INLINE EXEMPLOS

- `<img/>`
- `<a>`
- `<strong>`
- `<span>`
- `<input>`

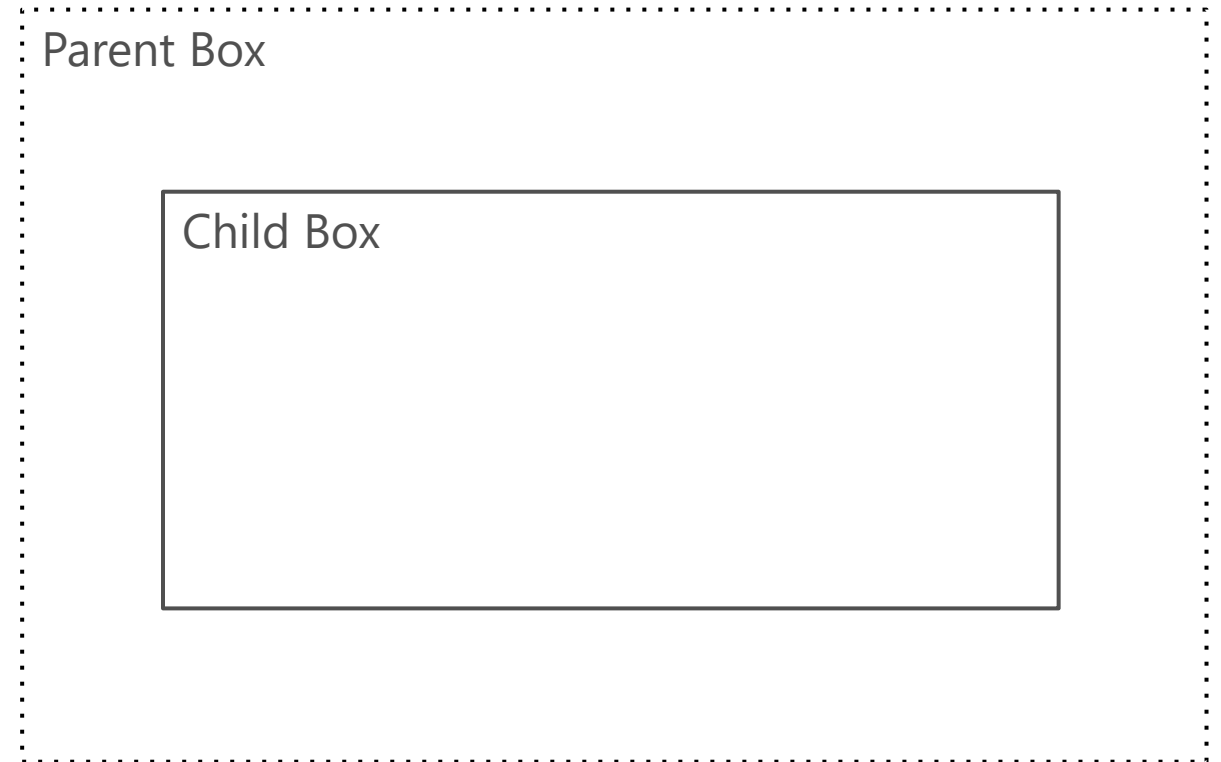
# Fluxo de Conteúdo

- **Flow** é um estilo de apresentação do HTML focado em preencher conteúdo em uma página horizontalmente em linhas de cima para baixo
- Existem dois tipos básicos de fluxo para a propriedade **display**:
  - **Block**
  - **Inline**
- No fluxo **block**, o conteúdo é colocado em sua própria linha, acima ou abaixo de outro conteúdo
  - Ele preenche toda linha da esquerda para direita
- No fluxo **inline**, o conteúdo é colocado na mesma linha de outros conteúdos, antes ou depois deles



# Relacionamento Pai/Filho

- Com o CSS Box Model, é possível para uma caixa conter outras caixas
  - A caixa mais externa é o pai
  - A caixa interna é o filho
- Uma caixa filha herda o estilo CSS da caixa pai, o que significa que um estilo aplicado a uma caixa pai será aplicado também ao filho
- Contudo, certas propriedades do CSS não suportam herança!



# Posicionando Elementos

- A propriedade **position** permite a colocação de um elemento em uma posição específica do documento
- Os valores utilizados são:

VALOR	DESCRIÇÃO
static	Posiciona o elemento dentro de seu fluxo normal
relative	Posiciona um elemento em relação a onde ele normalmente estaria no fluxo de conteúdo
absolute	Garante que o posicionamento do elemento não impacta os demais
fixed	Posiciona um elemento em relação à janela do navegador de maneira fixa
sticky	Mistura de static com fixed




- Exemplos:
  - [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Positioning](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Positioning)

# Float



# Elementos flutuantes

Float Example



## Float positioning

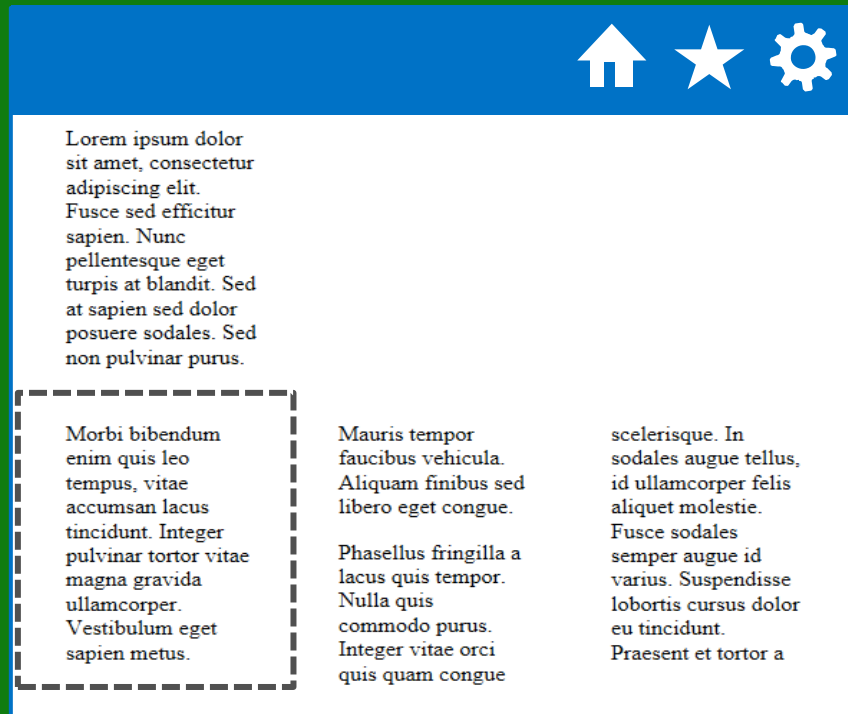
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce sed efficitur sapien. Nunc pellentesque eget turpis at blandit. Sed at sapien sed dolor posuere sodales. Sed non pulvinar purus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce sed efficitur sapien. Nunc pellentesque eget turpis at blandit. Sed at sapien sed dolor posuere sodales.

Lorem ipsum consectetur adipiscing sit amet felis. Integer ultrices viverra velit.

- Em adição à modificação do fluxo de uma página, HTML e CSS permitem que o desenvolvedor posicione elementos HTML de forma individual
- Exemplos:
  - [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Floats](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Floats)

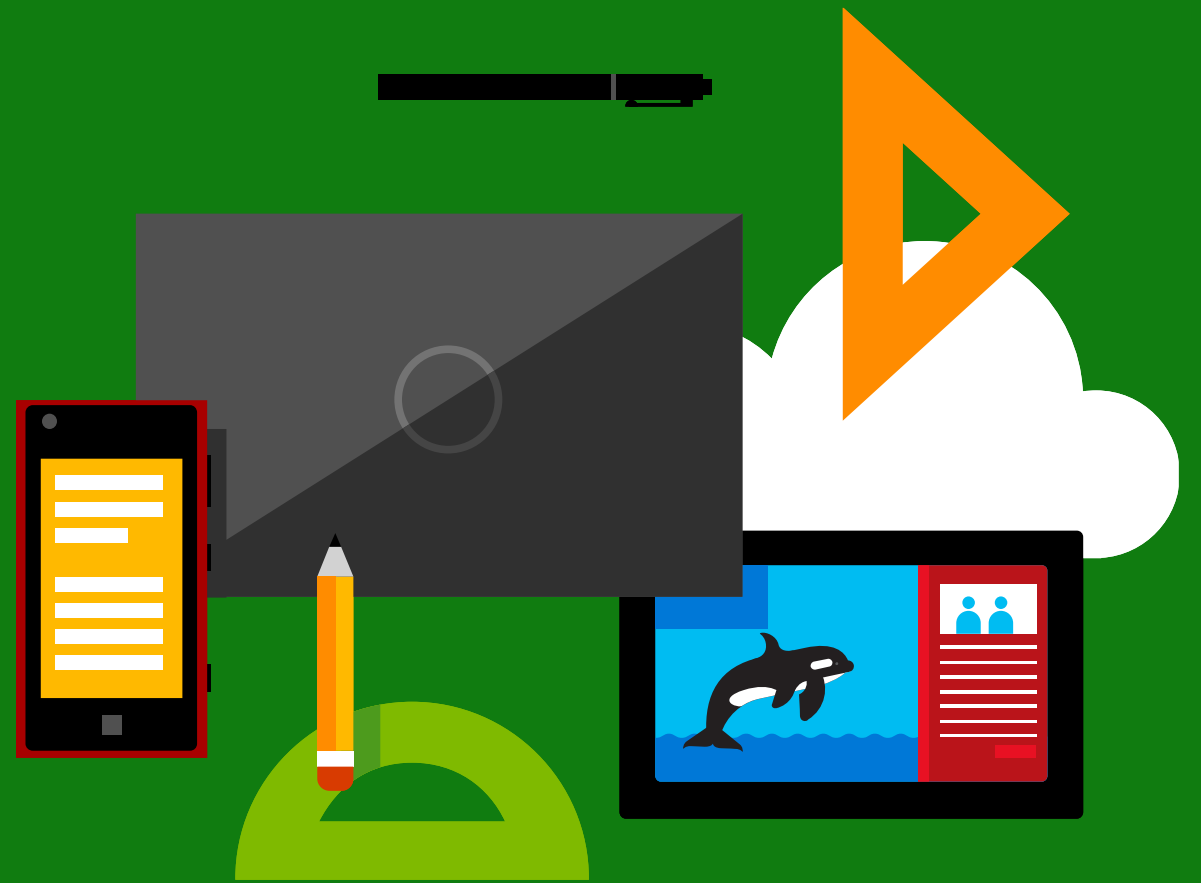
# Elementos flutuantes



- A propriedade **float** permite mover um elemento totalmente para a esquerda ou direita de uma página
  - Demais conteúdos irão se posicionar no entorno
- O valor de float é comumente configurado em `left` ou `right`
- Use a propriedade **clear** para prevenir que outros elementos flutuantes toquem o lado esquerdo ou direito de um elemento
- O valor de clear pode ser `left`, `right`, `both`, ou `none`
  - `left` deixa limpo o lado esquerdo
  - `right` deixa limpo o lado direito
  - `both` deixa limpo ambos lados
  - `none` permite que elementos se toquem

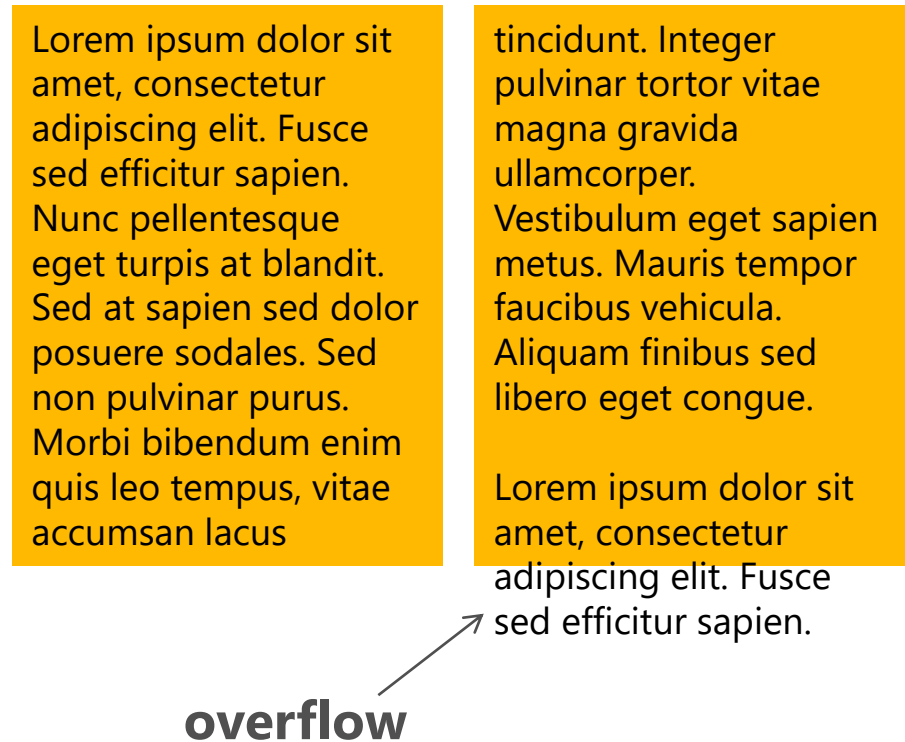


# Transbordo de Conteúdo



# Bounding Box

- Cada elemento HTML em uma página ocupa uma área retangular chamada de **bounding box**
- Com CSS, é possível modificar a altura ou largura do bounding box
- Se um elemento não cabe dentro dos limites do bounding box, então diz-se que o conteúdo transbordou - **overflow**
- É possível modificar como o conteúdo que transbordou é estilizado utilizando a propriedade `overflow` no CSS



# Scrolling Overflow

- Se o valor da propriedade **overflow** é definido como **scroll**, então todo o conteúdo do elemento permanece dentro dos limites do bounding box
- Isto permite que usuário utilizem uma barra de rolagem



# Overflow Visível e Overflow Escondido

- Com valor **visible**, conteúdo transbordado irá aparecer fora dos limites do bounding box e potencialmente sobrepor outros elementos
- Com valor **hidden**, conteúdo transbordado não irá aparecer fora dos limites do bounding box

## Visible Overflow

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce sed efficitur sapien. Nunc pellentesque eget turpis at blandit. Sed at sapien sed dolor posuere sodales. Sed non pulvinar purus. Morbi bibendum enim quis leo tempus, vitae accumsan lacus tincidunt. Integer pulvinar tortor vitae magna gravida ullamcorper. Vestibulum eget sapien metus. Mauris tempor faucibus vehicula. Aliquam finibus sed libero eget congue.

Phasellus fringilla a lacus quis tempor. Nulla quis commodo purus. Integer vitae orci quis quam congue scelerisque. In sodales augue tellus, id ullamcorper felis aliquet molestie. Fusce sodales semper augue id varius. Suspendisse lobortis cursus dolor eu tincidunt. Praesent et tortor a quam auctor tincidunt non ac odio. In varius, felis et molestie eleifend, ante ipsum ligula vel dui. Sed at nunc cursus ex vel ullamcorper.

## Hidden Overflow

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce sed efficitur sapien. Nunc pellentesque eget turpis at blandit. Sed at sapien sed dolor posuere sodales. Sed non pulvinar purus. Morbi bibendum enim quis leo tempus, vitae accumsan lacus tincidunt. Integer pulvinar tortor vitae magna gravida ullamcorper. Vestibulum eget sapien metus. Mauris tempor faucibus vehicula. Aliquam finibus sed libero eget congue.

Phasellus fringilla a lacus quis tempor. Nulla quis commodo purus. Integer vitae orci quis quam congue scelerisque. In

# Exemplo

```
<body>
  <h1>Overflow Demo</h1>
  <p>Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Fusce sed efficitur sapien.
Nunc pellentesque eget turpis
at blandit. Sed at sapien sed
dolor posuere sodales. Sed non
pulvinar purus. Morbi bibendum
enim quis leo tempus, vitae
accumsan lacus tincidunt.</p>
</body>
```

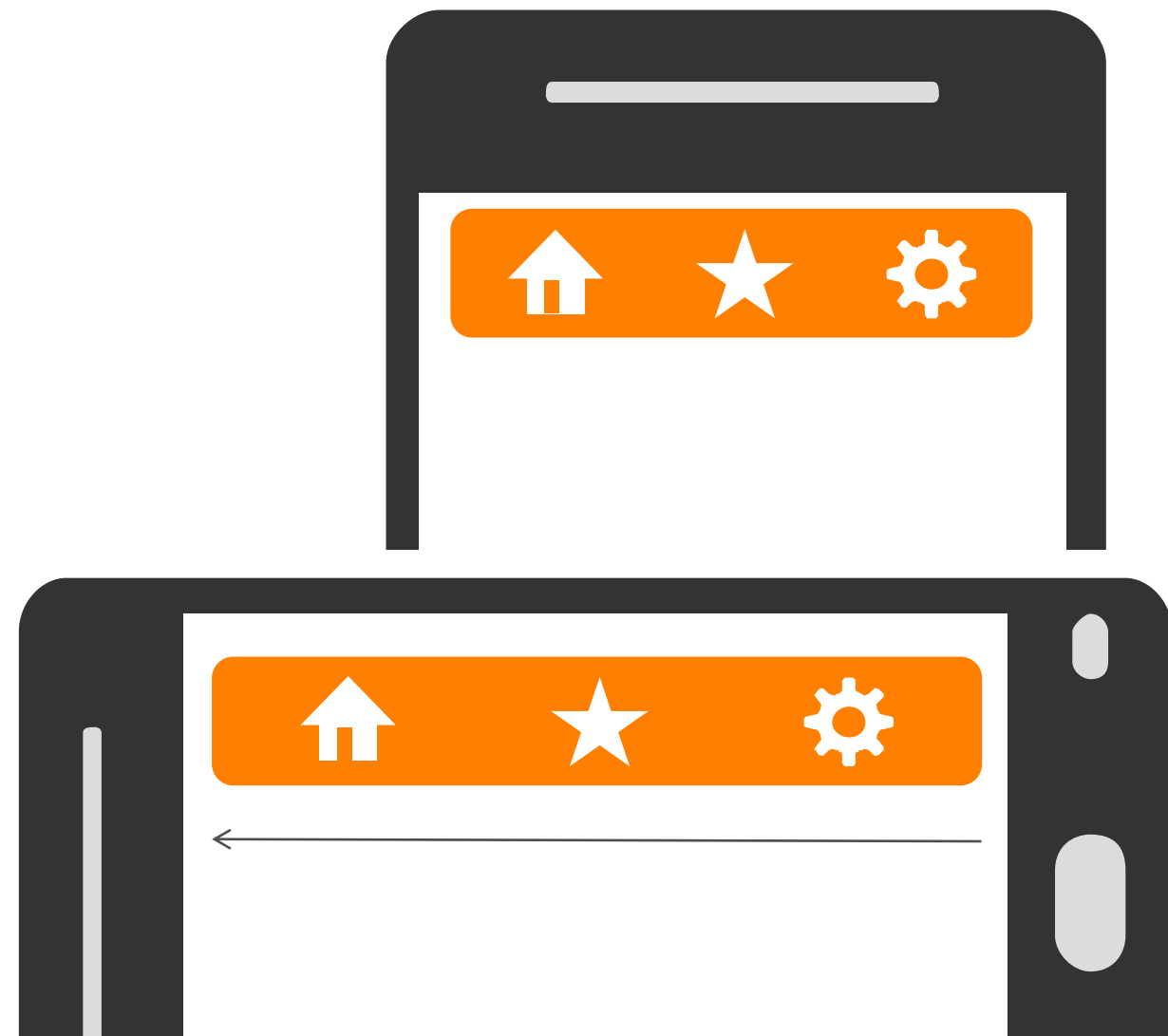
```
<style>
  p {
    height: 200px;
    width: 200px;
    background-color: #e1e1e1;
    overflow: scroll | visible | hidden;
  }
</style>
```

# Flexbox



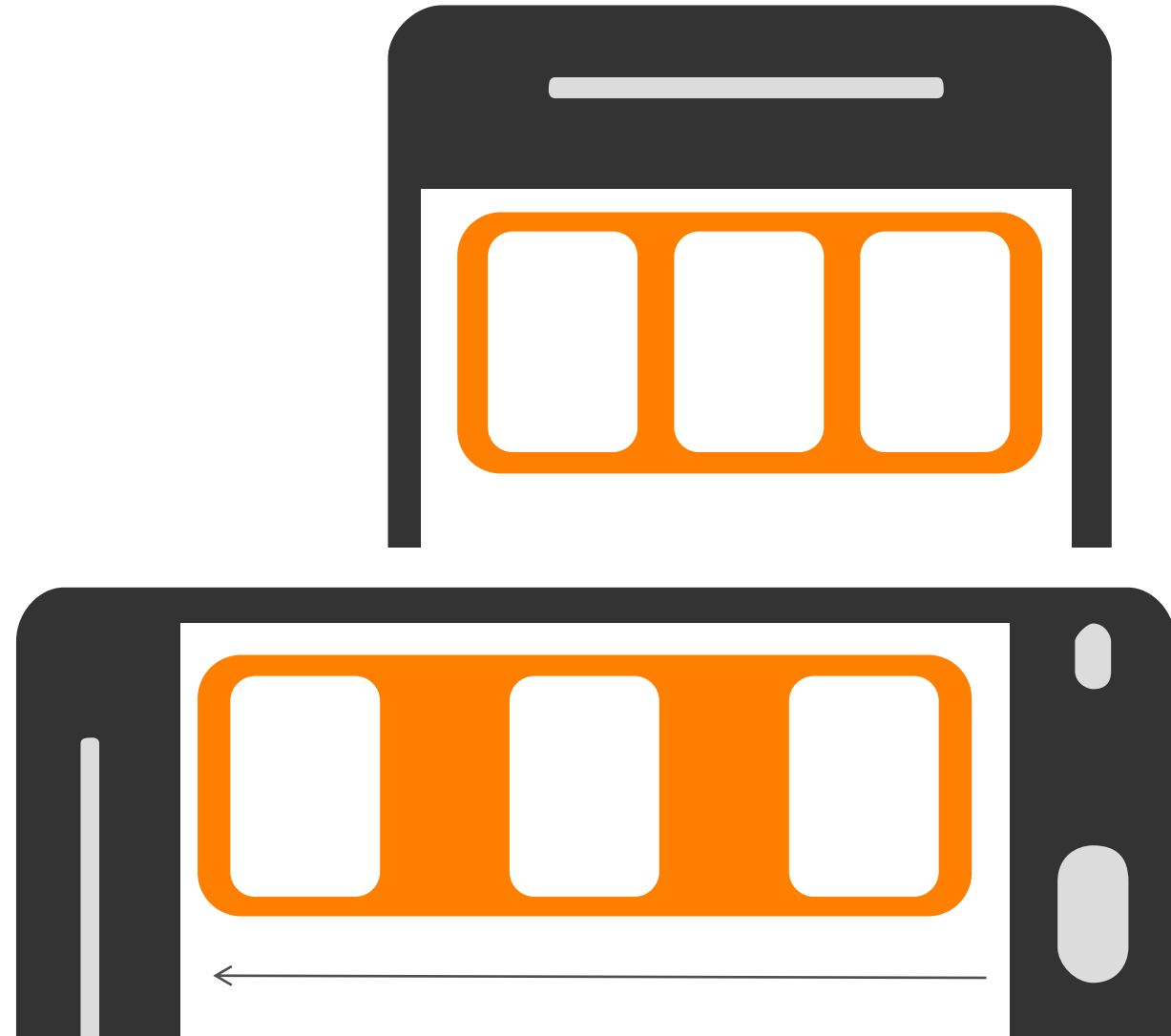
# CSS Flexbox Box Model

- CSS3 inclui o **Flexbox Box Model**, que é um modo de layout que provê uma maior flexibilidade quando o usuário altera o tamanho da janela do navegador
- Elementos, barras de navegação, formulários e imagens se redimensionarão e se reposicionarão automaticamente para ocupar o espaço disponível
- Utiliza-se o suporte de **media queries**
  - CSS utiliza a informação para ajustar o documento HTML



# Itens Flexbox

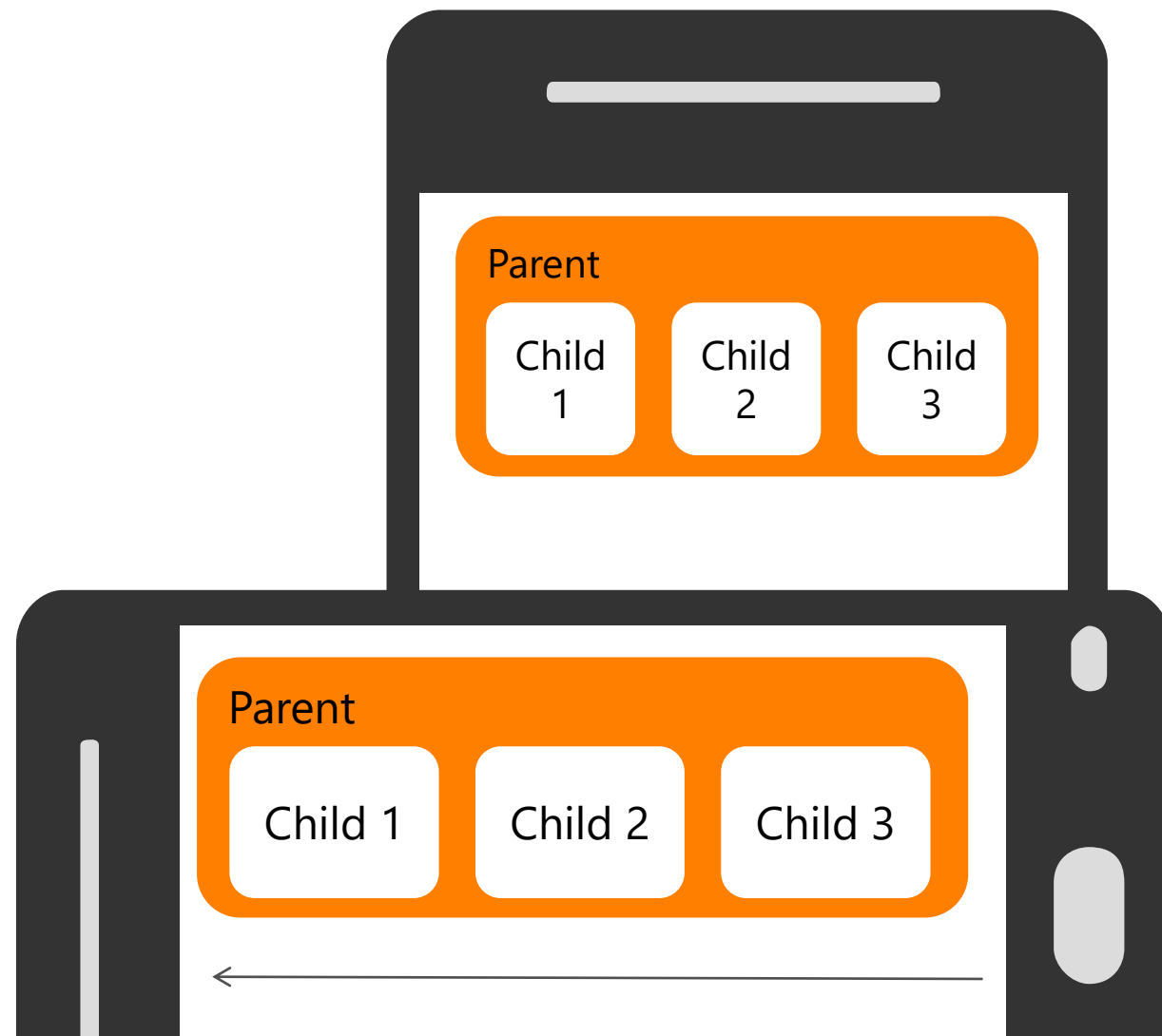
- Define-se que um elemento utiliza o modelo Flexbox através da propriedade **display**
- Dois valores: **flexbox** e **inline-flexbox**
  - O valor `flexbox` define a caixa como block-level
  - O valor `inline-flexbox` define a caixa como inline-level
- Exemplos:
  - [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Flexible\\_Box\\_Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout)





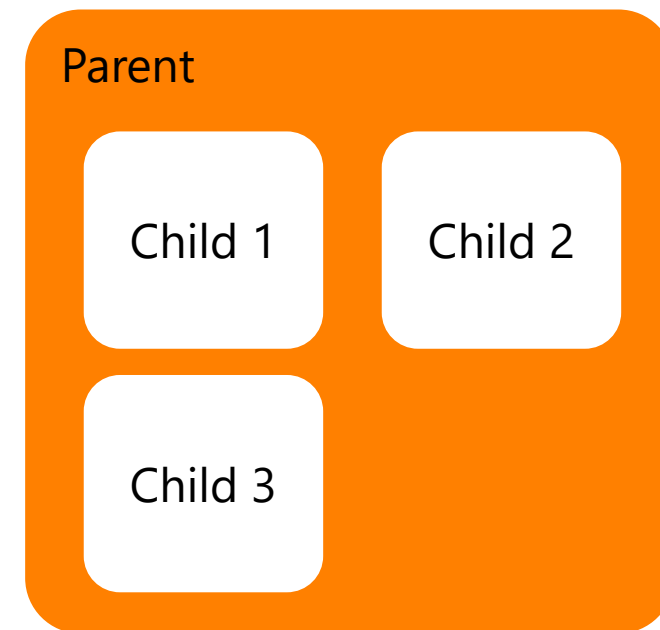
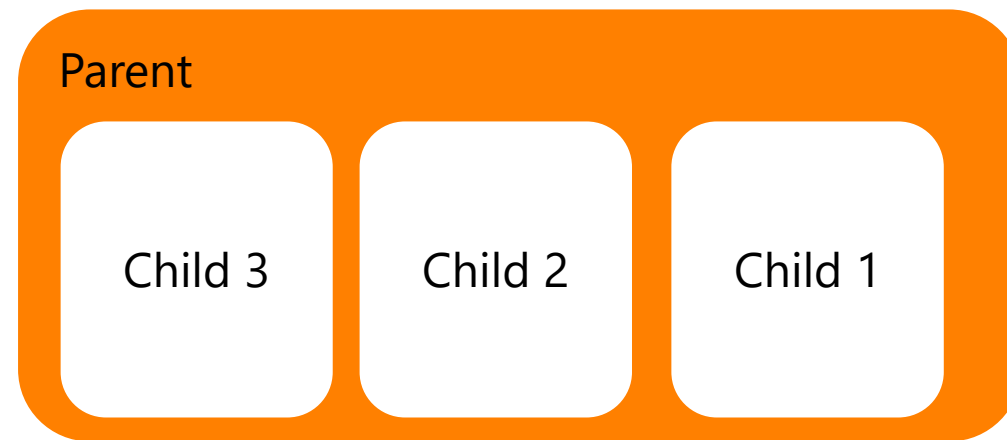
# Trabalhando com Flexboxes

- Flexboxes podem conter outras caixas filhas, que são referenciadas como **flexbox items**
- Com a propriedade **flex**, é possível tornar os itens flexíveis também
- Lembre-se que a propriedade **display** é utilizada para tornar as caixas pais flexíveis
- A propriedade **flex** também pode ser usada para escalonar proporcionalmente itens flexbox quando o flexbox aumenta ou diminui de tamanho



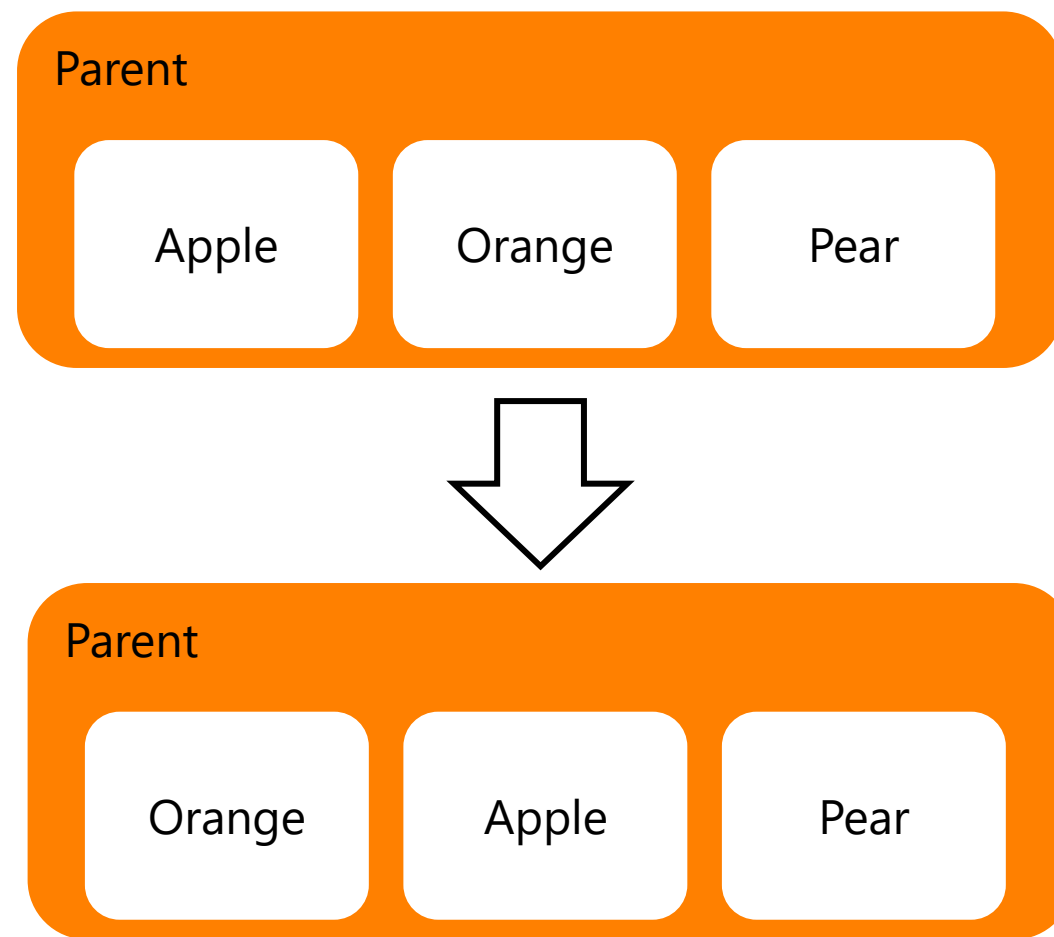
# Trocando a Direção dos Itens Filhos

- A propriedade **flex-direction** permite que se troque a direção das caixas filhas dentro de um flexbox
  - Valores `row`, `row-reverse`, `column`, e `column-reverse`
- A propriedade **flex-wrap** determina se as caixas filhas irão se deslocar para a próxima linha se a janela diminui de tamanho
  - Valores `nowrap`, `wrap`, e `wrap-reverse`
- A propriedade **flex-flow** define **flex-direction** e **flex-wrap** ao mesmo tempo

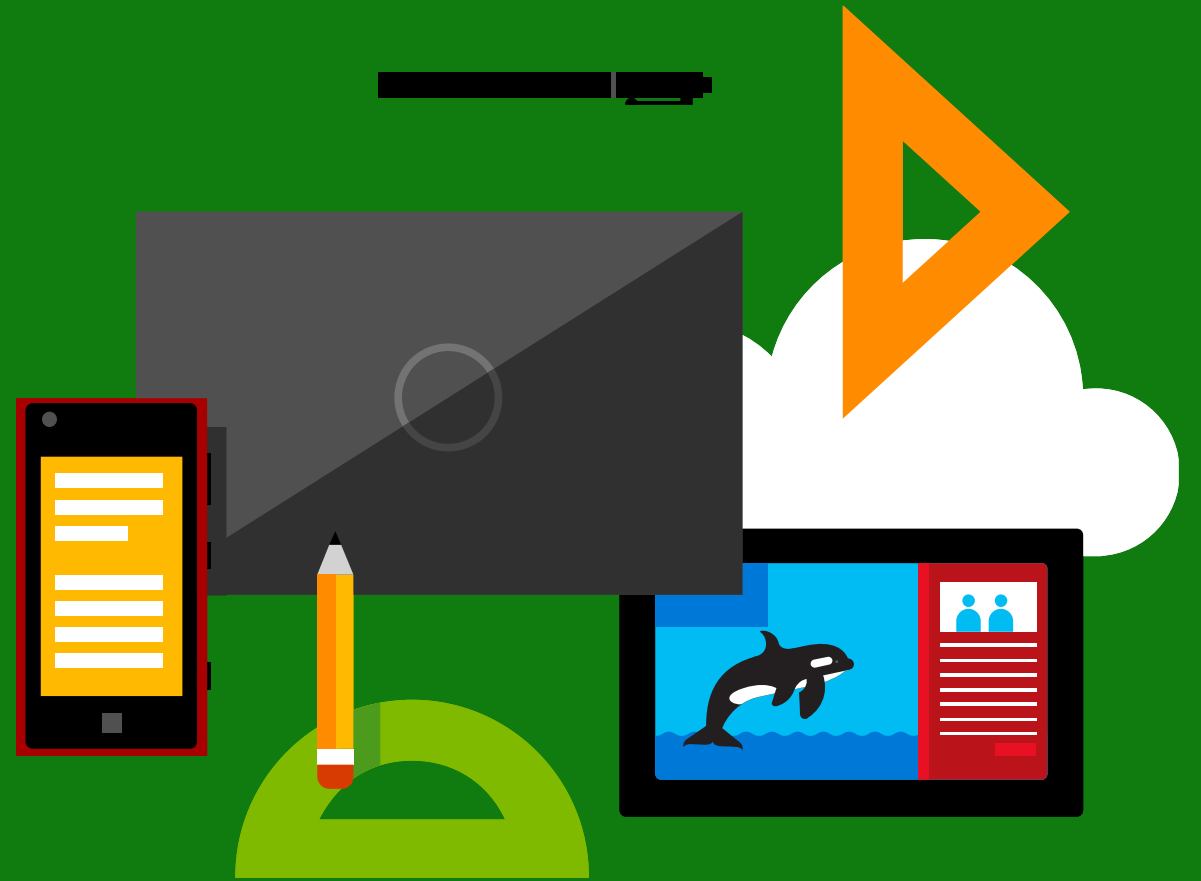


# Ordenando e Arranjando Conteúdo

- A propriedade **flex-order** permite ajustar a ordem do conteúdo dentro de um flexbox
- A propriedade agrupa caixas filhas a fim de controlar a ordem que aparecem em um layout
- O valor padrão é 0

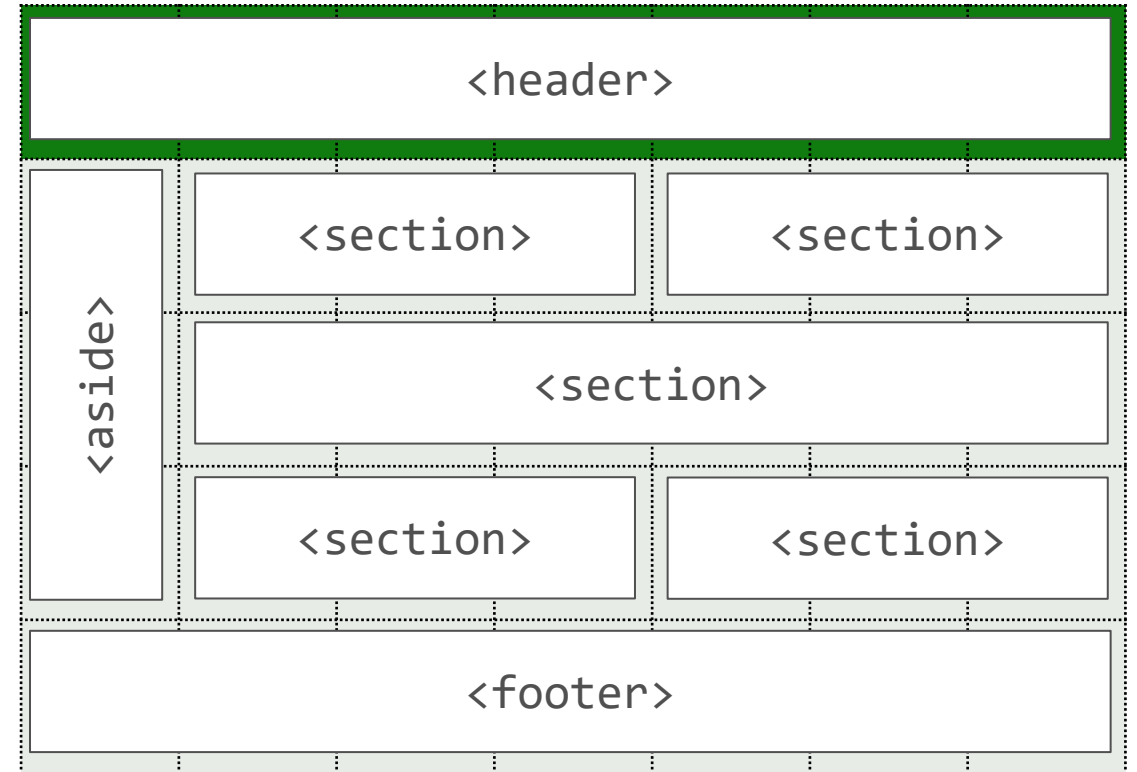


# Grid Layout



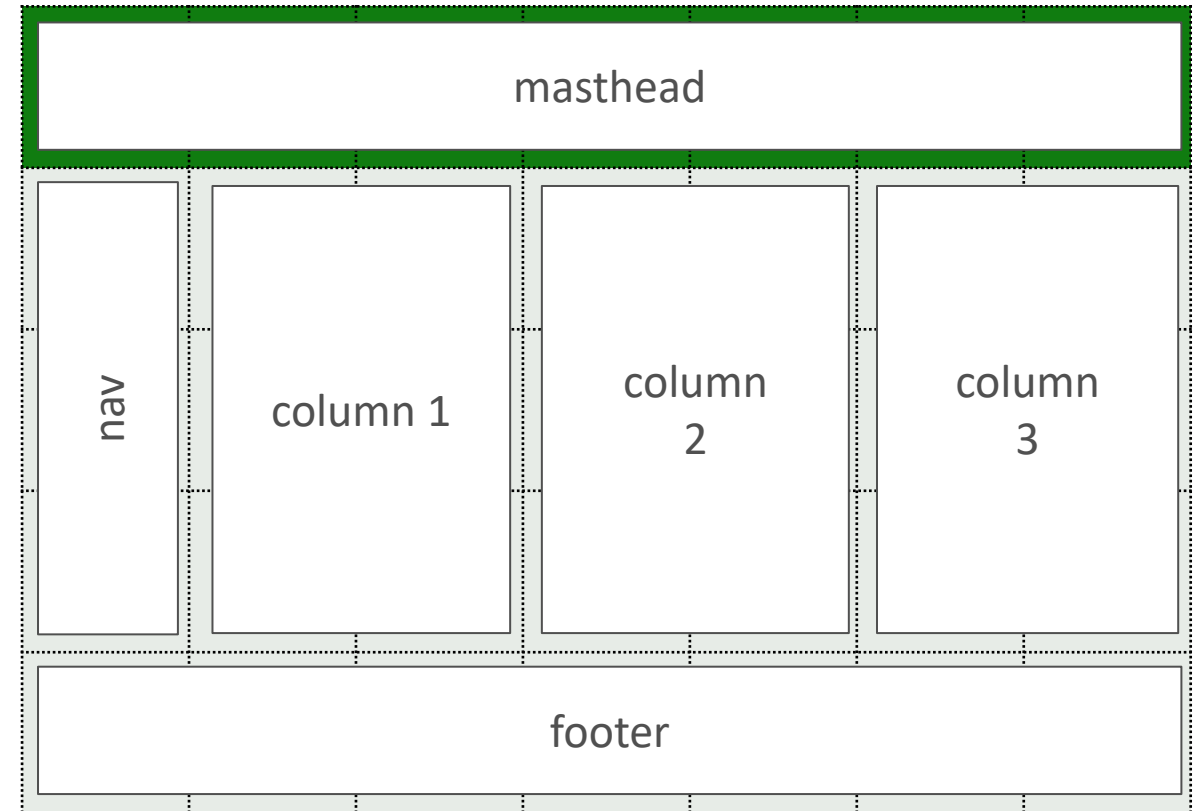
# Grid Layout Model

- Quando o Flexbox Box Model não for apropriado, pode-se utilizar o Grid Layout Model
- O Grid Layout Model utiliza CSS para estruturar conteúdo utilizando linhas e colunas
- Grids são extremamente flexíveis e uma alternativa fácil de organizar conteúdo sem recorrer a tabelas do HTML
- Exemplos:
  - [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Grids](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids)



# Grid Items

- Grid layouts são bastante similares a tabelas, pois são compostos de linhas e colunas
- O conteúdo dentro de grid layouts são também modulares, permitindo mover conteúdo de uma parte para outra
- Elementos filhos em um grid são chamados de grid items, e podem ser posicionados de acordo com grid tracks, grid lines, ou grid cells

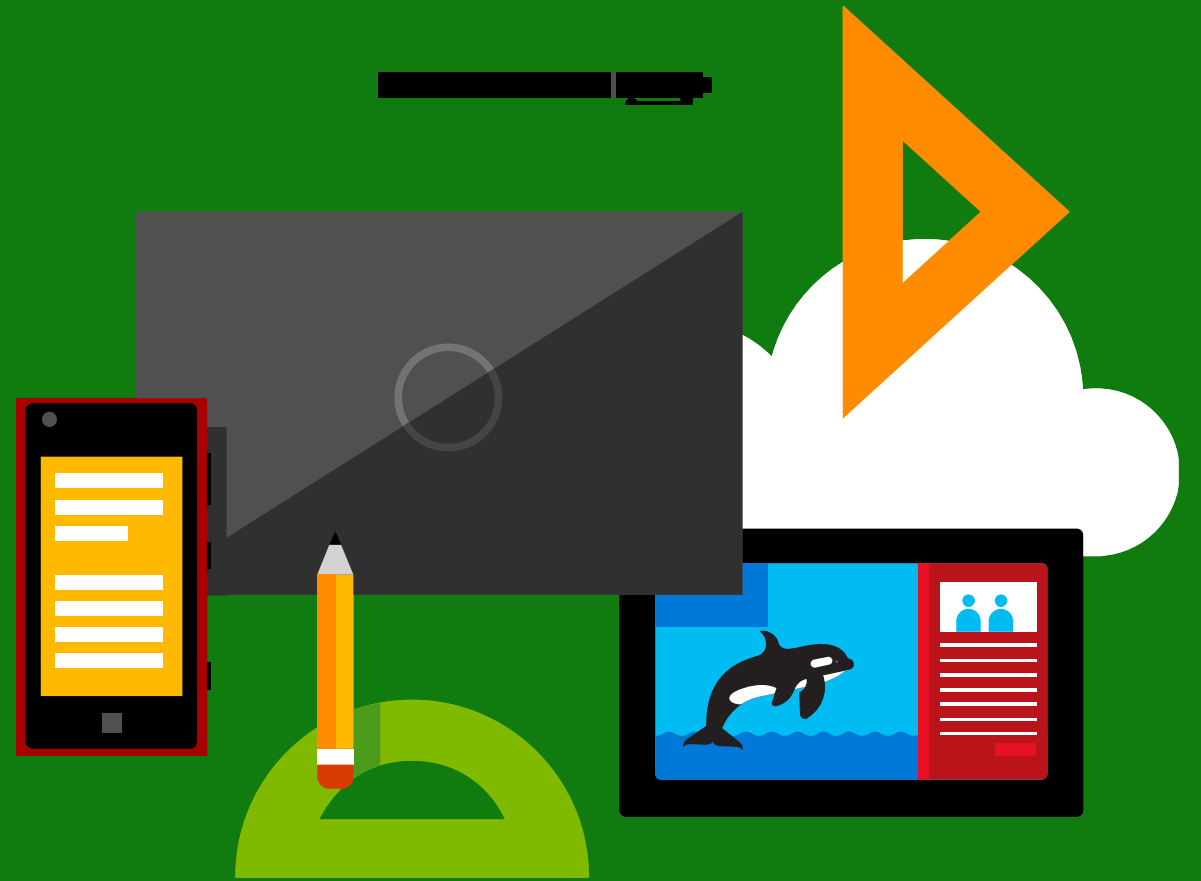


# Definindo o Grid Layout

- Defina um layout através da propriedade **display**, junto com os valores `grid` ou `inline-grid`

CONCEITO DE LAYOUT	DESCRIÇÃO
Grid tracks	As colunas e linhas de um grid
Grid lines	As linhas horizontais ou verticais que separam colunas e linhas
Grid cells	O espaço lógico onde o conteúdo é posicionado

# Código Legado



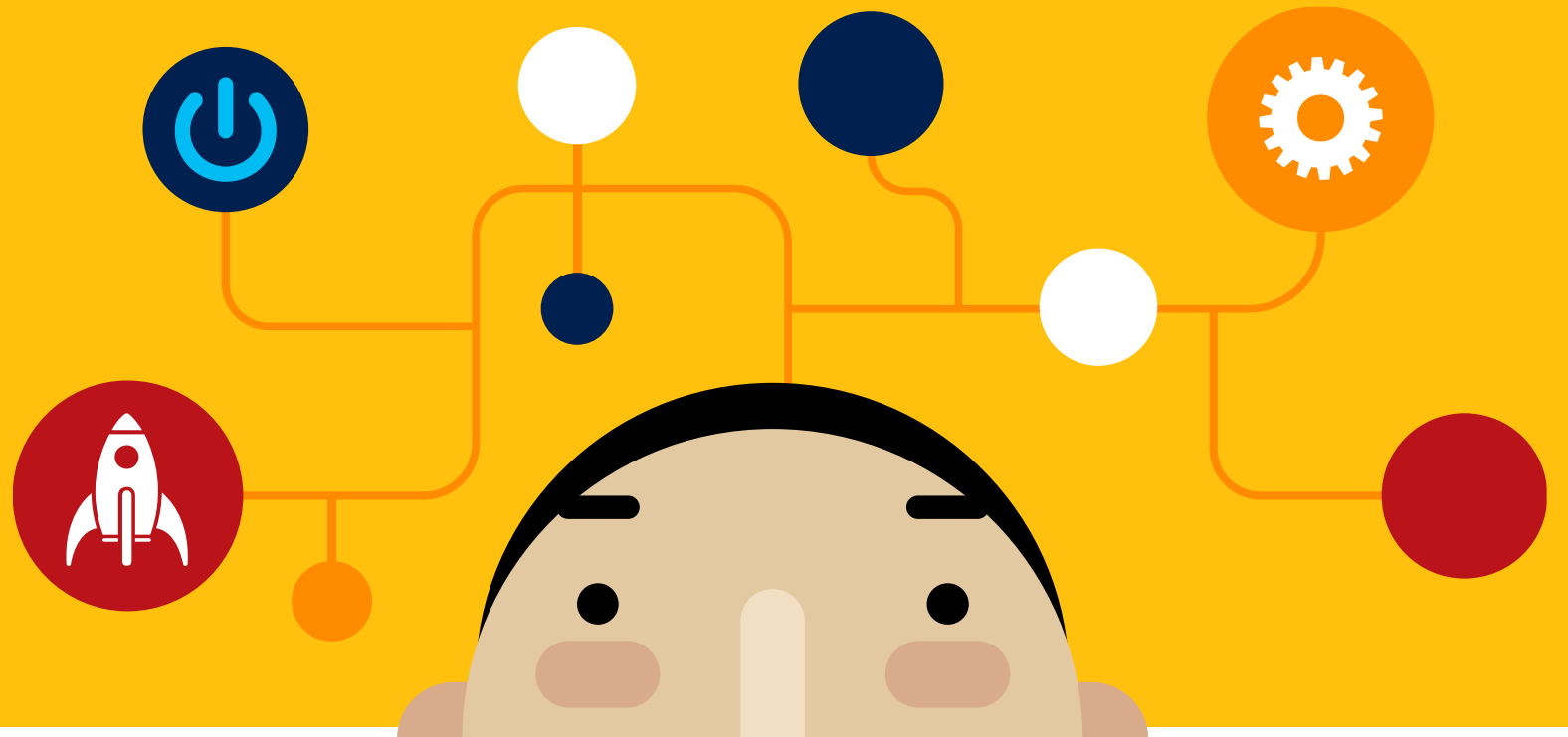


# Código Legado

- Cuidado com exemplos e folhas de estilo que NÃO utilizam os métodos mais atuais de posicionamento de elementos em uma página
- Evitar ao máximo usar “métodos legados”
  - Somente utilizar se os navegadores alvo não suportarem os novos padrões CSS
- Exemplo:
  - <https://github.com/mdn/learning-area/blob/master/css/styling-boxes/box-model-recap/css-tables-example.html>
- Saiba mais:
  - [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Legacy\\_Layout\\_Methods](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Legacy_Layout_Methods)

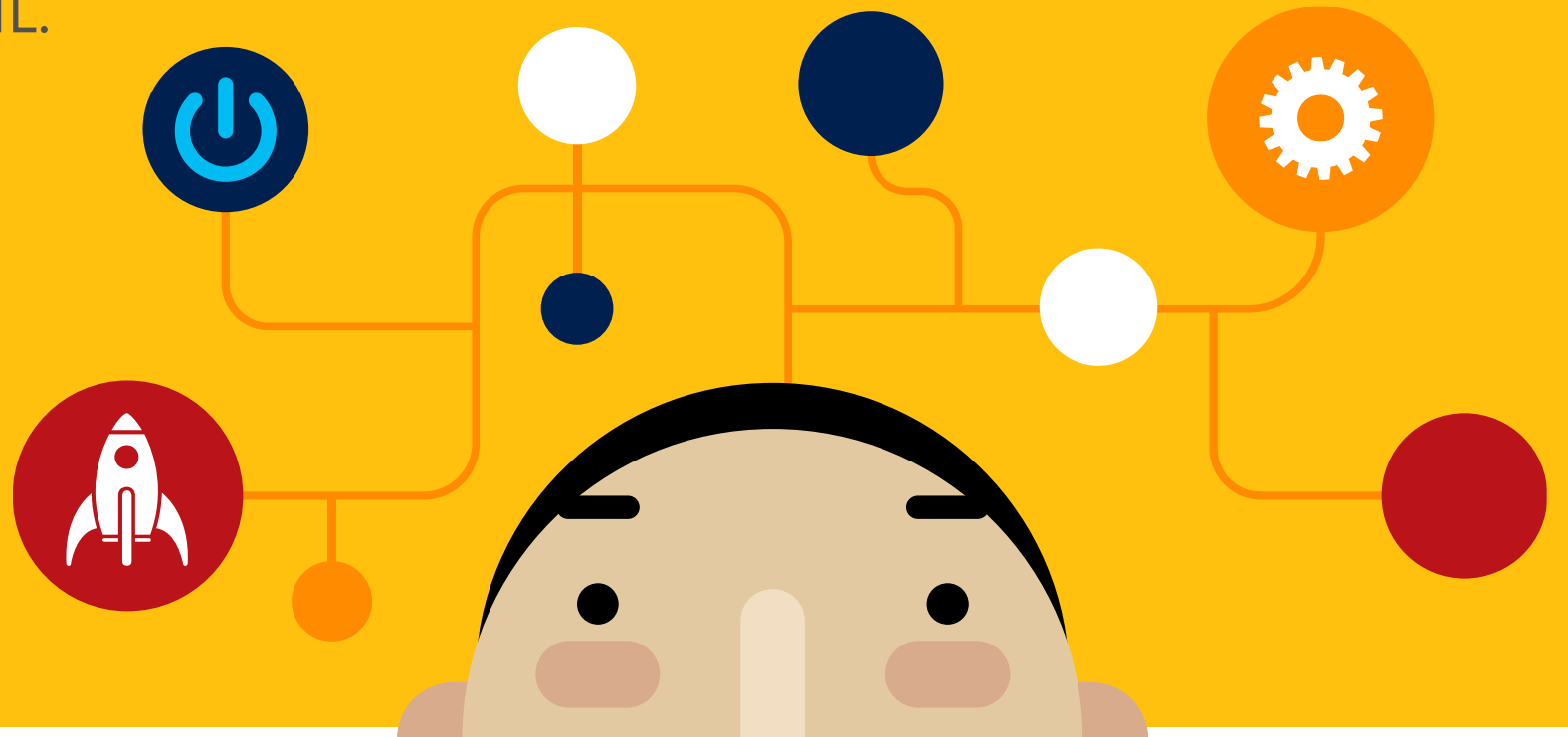
# Laboratório

Pesquise na Web por repositórios que fornecem páginas CSS com licença gratuita. Escolha uma e analise sua estrutura. Procure entender exatamente o quê cada propriedade faz.

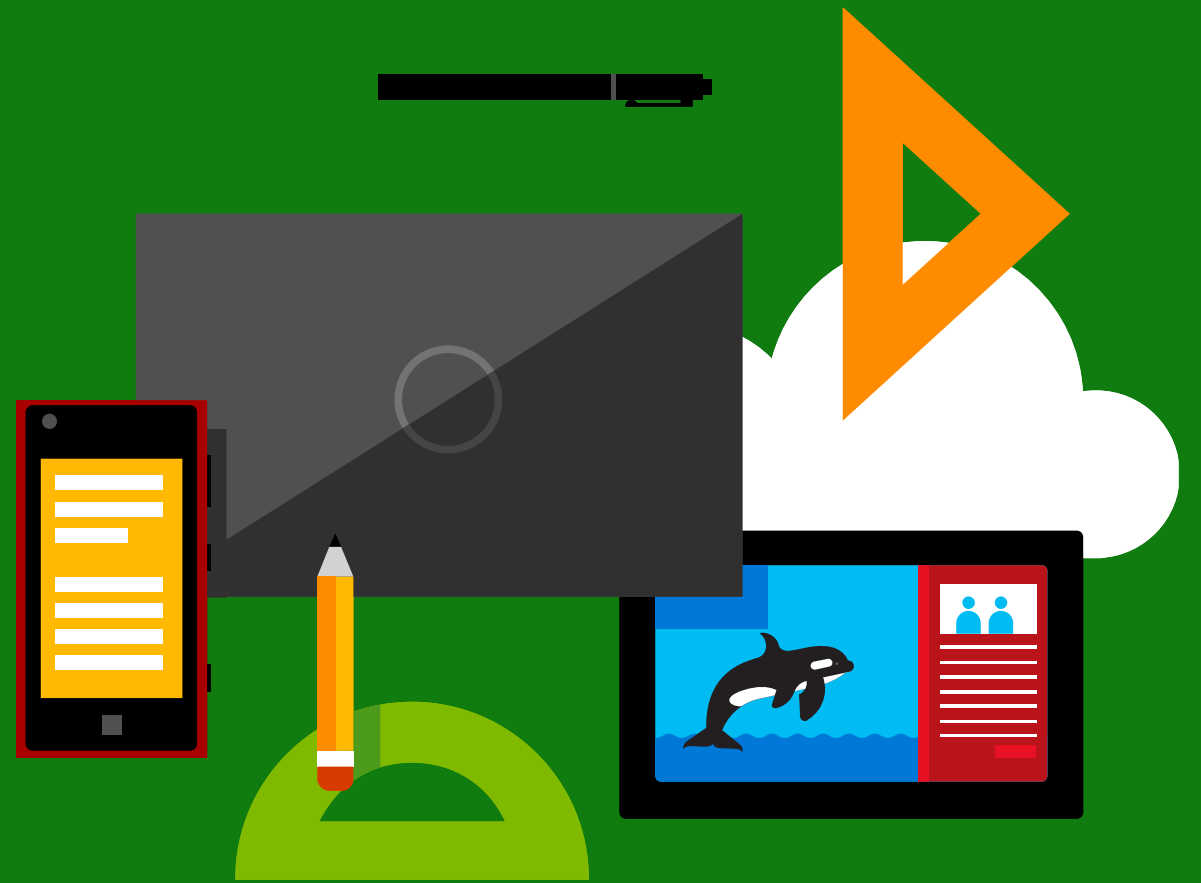


# Laboratório

Pense na estrutura e informações de um currículo. Crie um documento HTML com as informações essenciais do seu currículo. Depois, crie uma página de estilos CSS e aplique ao documento HTML.



# Processadores de CSS



# Extensões ao CSS

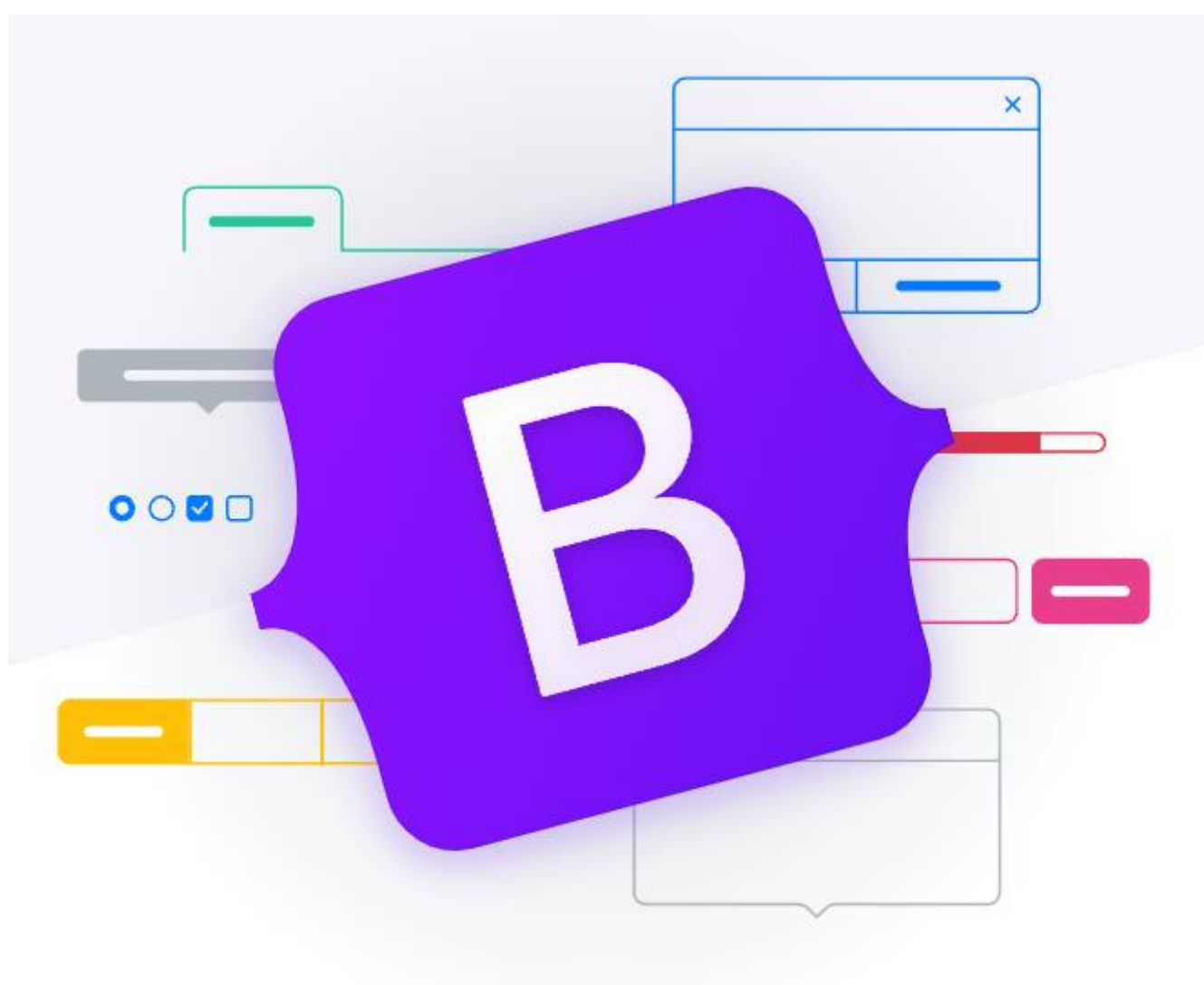
- Alguns frameworks fazem uso de processadores de CSS que permitem adicionar elementos novos à linguagem
  - LESS <https://lesscss.org/>
  - SASS <https://sass-lang.com/>
  - Tailwind <https://tailwindcss.com/>

# Bootstrap



# Bootstrap

- Uso da linguagem CSS pode ser bastante complexo
- Suporte a diferentes navegadores é um desafio
- Bootstrap é um framework de temas e layout criado pelo Twitter
  - Resolve rapidamente problemas básicos de layout
  - Torna o uso dos recursos de CSS muito mais simples
- Utiliza CSS3 para criação de páginas responsivas, com isto a página se adapta dinamicamente para diferentes navegadores e tamanhos de tela



<http://getbootstrap.com/>

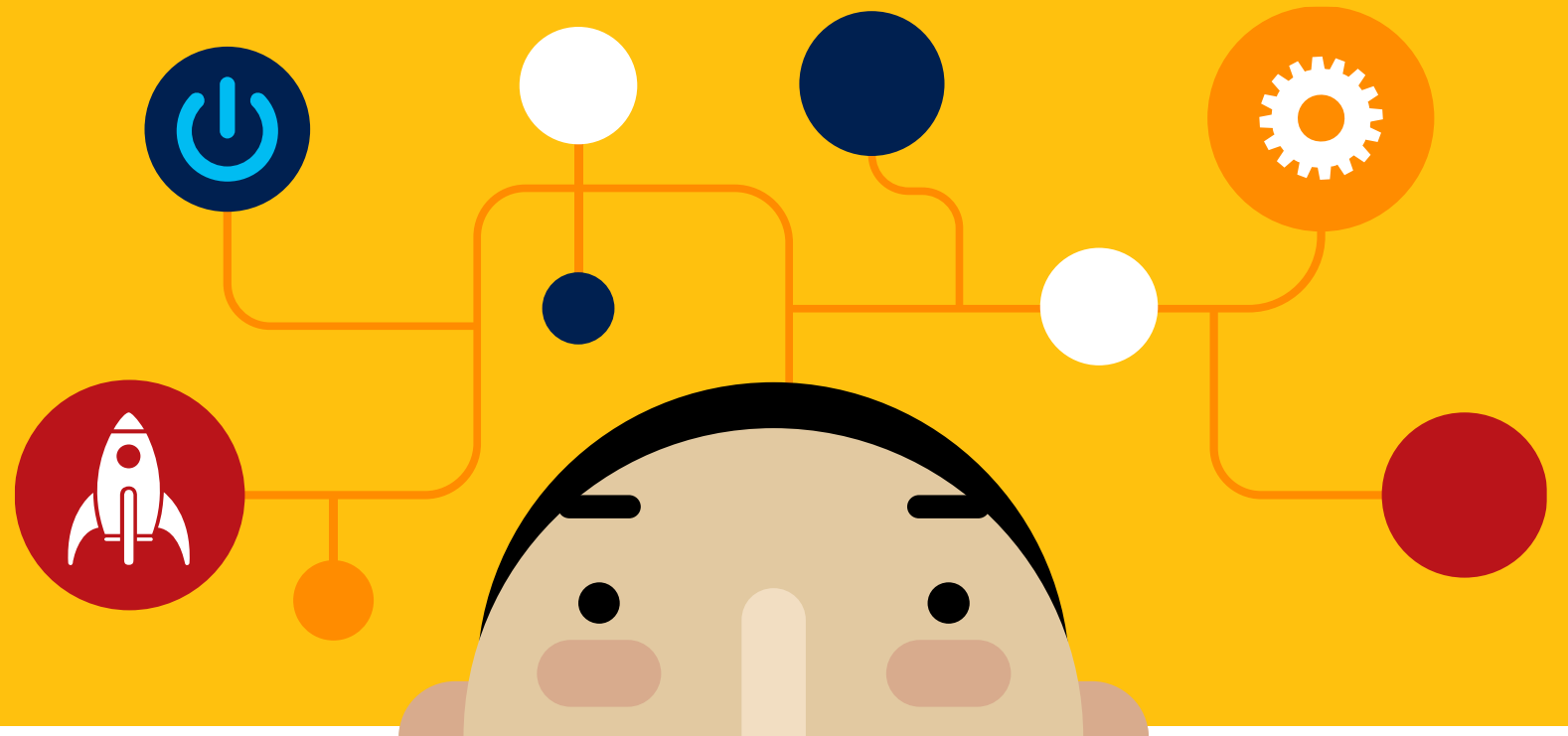


# Bootstrap

- Cuidado!
- O Bootstrap possui uma coletânea de componentes de UI opcionais que dependem de bibliotecas de JavaScript adicionais (tais como jQuery)
- Evite, sempre que possível, utilizar essas bibliotecas adicionais de forma conjunta com outros frameworks a fim de não causar conflitos

# Laboratório

Retome o documento do exercício sobre uma página de currículo. Aplique o Bootstrap sobre o documento. Procure também aplicar um tema do Bootstrap sobre a página.

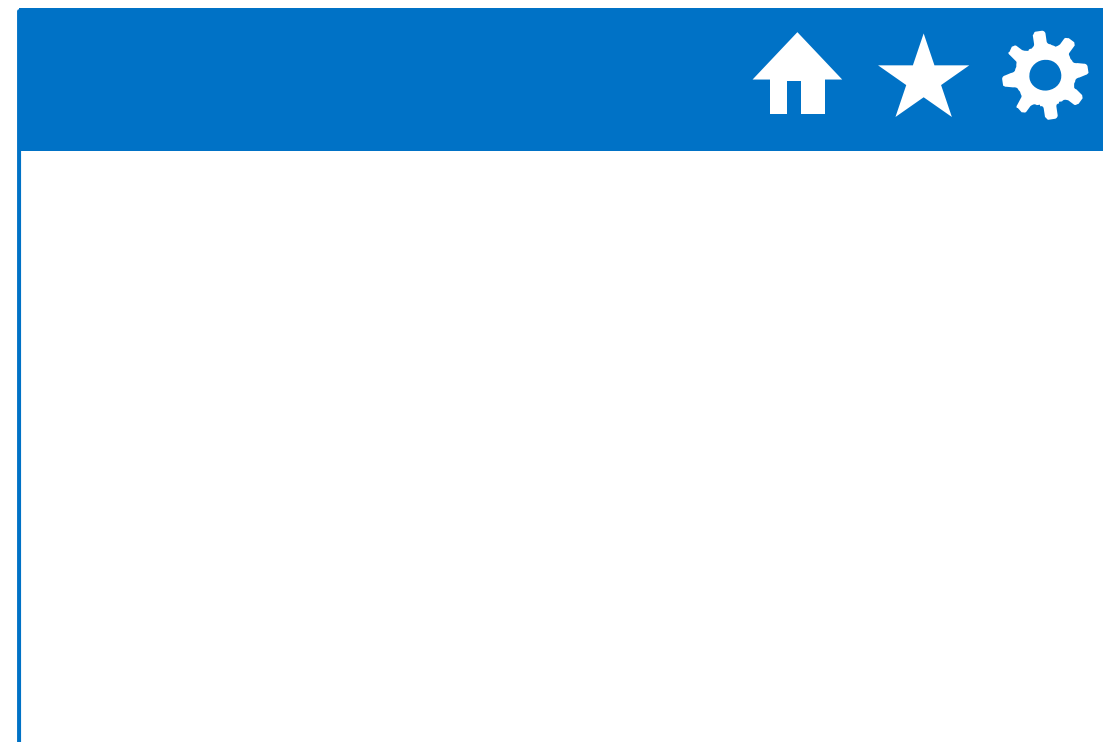


# Manipulação do DOM



# DOM

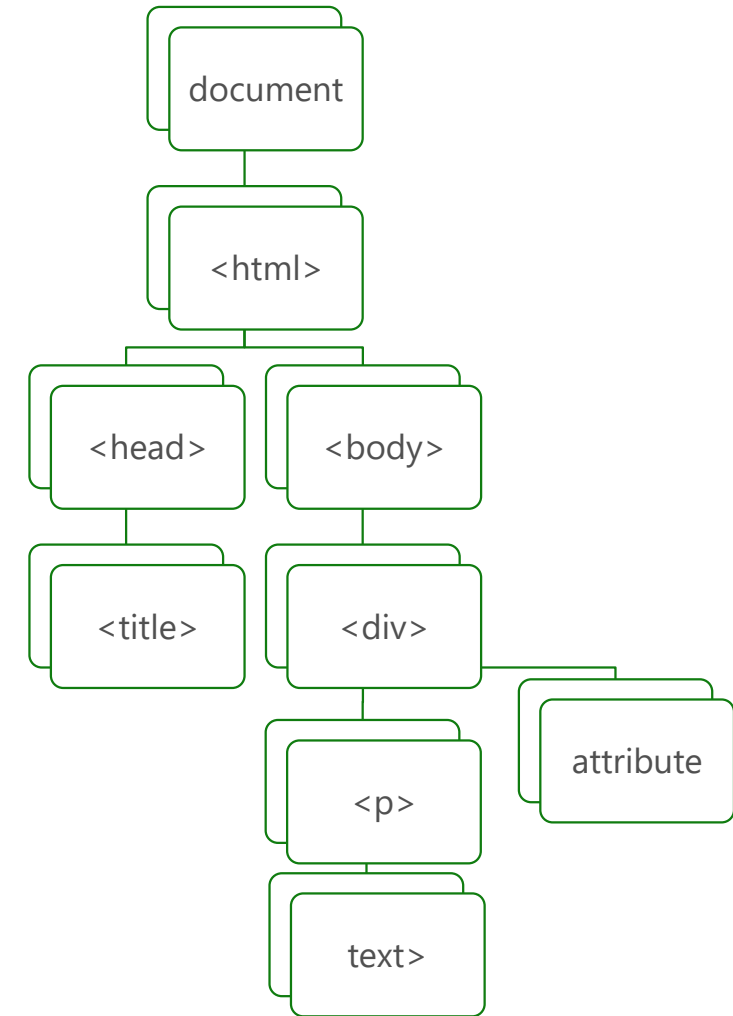
- Um elemento HTML é um objeto
- É possível acessar e modificar os objetos HTML que estão dentro de um navegador
- A criação de páginas Web interativas está baseada na capacidade de manipulação desses objetos
- Objetos são agrupados em **modelos de objetos**, os quais são usados para representar os navegadores e páginas Web



Isto é um objeto

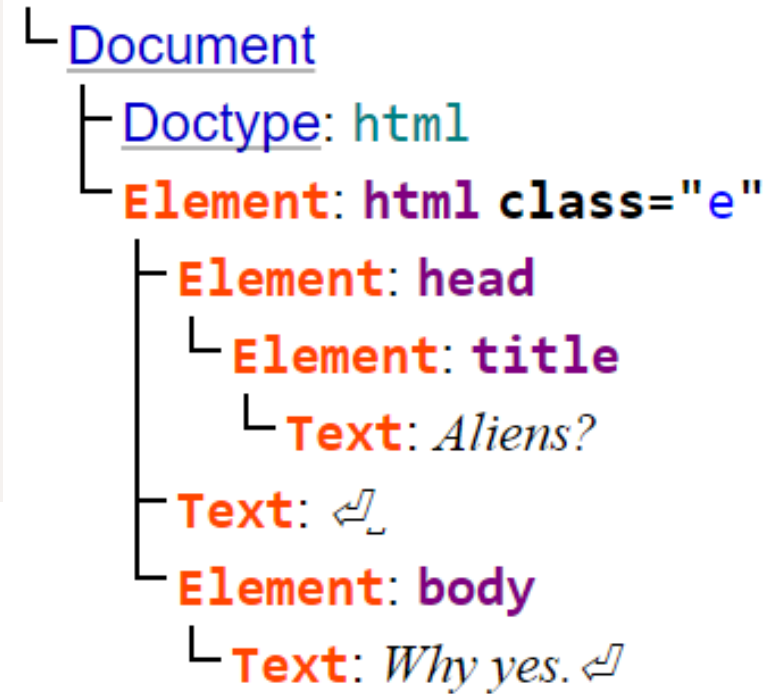
# Document Object Model (DOM)

- O *Document Object Model* (DOM) cria um modelo em árvore de uma página Web
- O DOM é utilizado para atualizar conteúdo, estrutura e estilos, de forma programática
  - DOM é uma das diversas API disponibilizadas pelos navegadores que se pode consumir em JavaScript
- O objeto mais ao topo é o **document object**, que representa a página como um todo
  - Seus objetos filhos representam os demais elementos individuais da página



# Document Object Model (DOM)

```
<!DOCTYPE html>
<html class=e>
  <head><title>Aliens?</title></head>
  <body>Why yes.</body>
</html>
```



# Document Object Model (DOM)

- Documentos de padronização:
  - <https://dom.spec.whatwg.org/>

# Localizando e Acessando Elementos

- É possível acessar objetos na árvore DOM utilizando o ID do elemento
- Método **getElementById()**
  - O element deve possuir o atributo ID setado no HTML
- Retorna um objeto específico representando o elemento selecionado

```
document.getElementById( 'demo' );
```

objeto

método

parâmetro



# getElementById() Exemplo

```
<body>
```

```
  <h1>Get today's date and add it to an element on the page.</h1>
```

```
  <p id="demo"></p>
```

```
  <script type="text/javascript">  
    document.getElementById("demo").innerHTML=Date();  
  </script>
```

```
</body>
```

# Atualizando Conteúdo dos Elementos

- Utiliza-se a propriedade **innerHTML** para alterar ou inserir conteúdo
  - Pode ser utilizado sobre qualquer elemento
- Para alterar o conteúdo, definir o valor de **innerHTML** para a string desejada
  - Utilize uma atribuição (=) em JavaScript
- Para remover o conteúdo, defina o valor da propriedade como uma string vazia

# innerHTML Exemplo

```
<body>
```

```
  <h1>Updating Content</h1>
```

```
  <p id="demo"></p>
```

```
  <script type="text/javascript">
```

```
    document.getElementById("demo").innerHTML='Using  
      JavaScript is super fun!';
```

```
  </script>
```

```
</body>
```

# Criando Elementos

- Para criar novos elementos, o objeto **document** fornece o método **createElement()**
- Adicione o elemento recém criado como filho de outro elemento na árvore DOM via os métodos
  - `appendChild()`
  - `insertBefore()`
  - `removeChild()`
  - `replaceChild()`

## JavaScript

```
function show_image(src, width, height, alt) {  
    var img = document.createElement("img");  
    img.src = src;  
    img.width = width;  
    img.height = height;  
    img.alt = alt;  
    // Adds it to the <body> tag  
    document.body.appendChild(img);  
}
```

## HTML

```
<button  
    onclick="show_image ('dog.jpg',  
        276,110, 'Stella');">  
    Display an image!  
</button>
```

# createElement Exemplo

```
<body>

  <h1>Creating Elements</h1>
  <p id="demo"></p>

  <button onclick="show_image
('dog.jpg',300,400,'Stella');">Display
an image!
  </button>

</body>
```

```
<script>
function show_image(src, width, height, alt) {
  var img = document.createElement("img");
  img.src = src;
  img.width = width;
  img.height = height;
  img.alt = alt;
  // Adds it to the <body> tag
  document.body.appendChild(img);
}
</script>
```

# Manipulação de Eventos



# Eventos

- Eventos são ações originadas pelo usuário ou qualquer outra entidade (inclusive objetos)
- DOM provê **tratadores de eventos**, os quais respondem a determinados eventos, efetuando algo em resposta
- Por exemplo, o tratador de evento `onClick` responde a uma ação de clicar algo

Tratador	Evento Associado
<code>onsubmit</code>	Submissão de formulário
<code>onkeydown</code> <code>onkeypress</code> <code>onkeyup</code>	Uso de teclas
<code>onclick</code> <code>onmousedown</code> <code>onmouseup</code>	Cliques de mouse ou touchpad
<code>onload</code> <code>onunload</code>	Carregamento ou descarregamento da página
<code>onselect</code>	Seleção de um item

# Eventos

- Para registrar um tratador de evento:
  - Modelo tradicional
    - Inline no atributo do elemento dentro do documento
    - Via script, associando uma função à propriedade correspondente ao evento
  - Modelo DOM
    - Permite manipular objetos que representam os eventos



# Eventos Exemplo

```
<body>
  <p>Select some of the text:
    <input type="text" value="Hello, World!"
      onselect="myFunction()"></p>
  <p id="demo"></p>

  <script>
    function myFunction() {
      document.getElementById('demo').innerHTML =
        "Selecting text is awesome!";
    }
  </script>
</body>
```

# Eventos

- Objeto **Event**
- Fornece várias propriedades com informações associado ao evento
- Geralmente é passado como primeiro argumento na chamada do tratador do evento
- Propriedades
  - *target* – o alvo do evento
  - *type* – o tipo (nome) do evento

# Eventos

- Modelo de eventos do DOM envolve três conceitos básicos:
  - Captura
  - "Bubbling"
  - Cancelamento

# Eventos

- Exemplo de fluxo de evento: o usuário passa o mouse sobre um link de um documento
  - Mouse sobre o documento
  - Mouse sobre qualquer elemento que contenha o elemento A
  - Mouse sobre o elemento A
  - Mouse sobre qualquer elemento que contenha o elemento A
  - Mouse sobre o documento

# Eventos

- Captura:
- Refere-se ao processo onde um tratador registrado em um ancestral da fonte do evento pode interceptar o evento antes que ele seja recebido pelo alvo
- Opera do topo da árvore em direção às folhas
- Para habilitar o tratador, utilizar **useCapture** como *true* no processo de registro
- Método **stopPropagation()** é utilizado para cancelar o processo

# Eventos

- “Bubbling”:
- Processo “inverso” ao processo de captura
- Refere-se ao processo onde um tratador registrado em um ancestral da fonte do evento tratar o evento depois que ele tenha sido recebido pelo alvo
- Opera do alvo do evento em direção ao topo da árvore
- Para habilitar o tratador, utilizar **useCapture** como *true* no processo de registro
- Método **stopPropagation()** é utilizado para cancelar o processo

# Eventos

- Cancelamento:
- Eventos que permitem o cancelamento (por exemplo, ativação de um link) podem ter sua ação impedida através de um tratador que chame o método **preventDefault()**