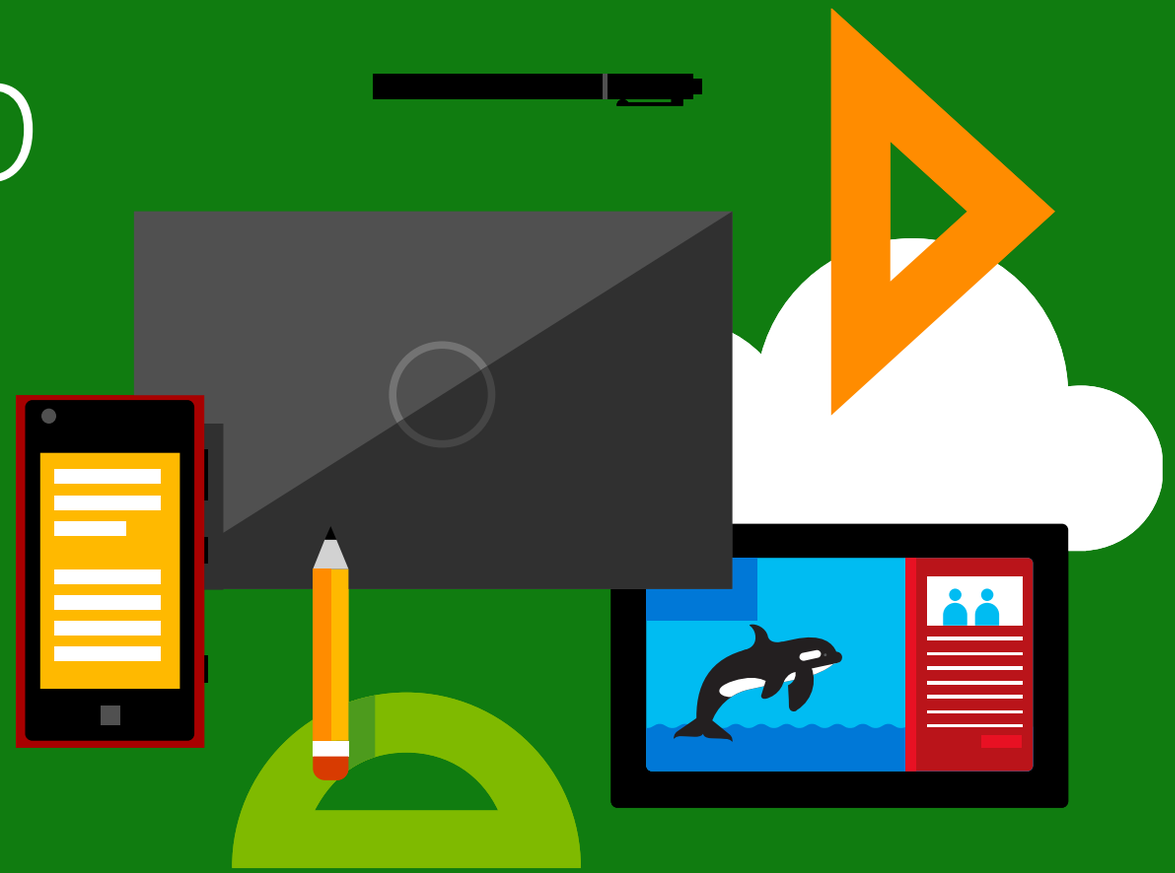
Dell IT Academy

HTML e CSS

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

Interface com o Usuário na Web



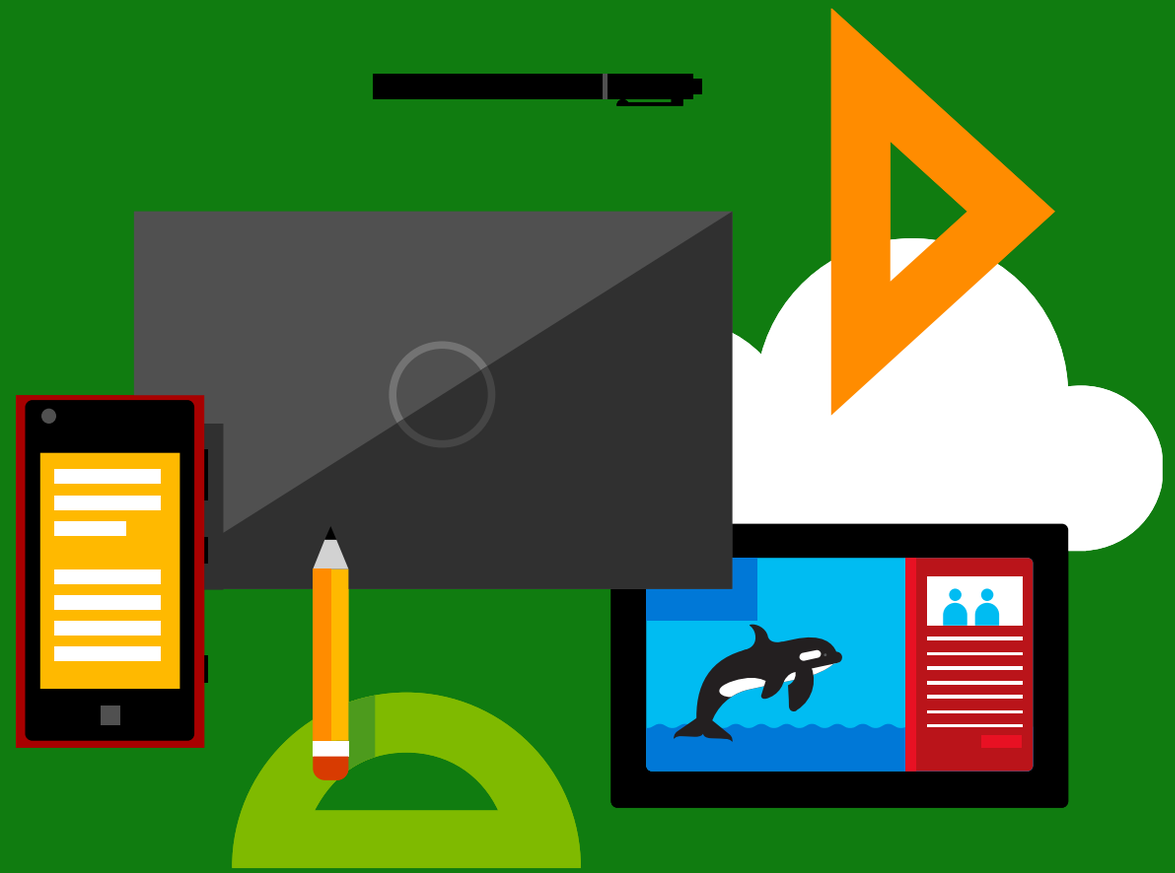
Tecnologias

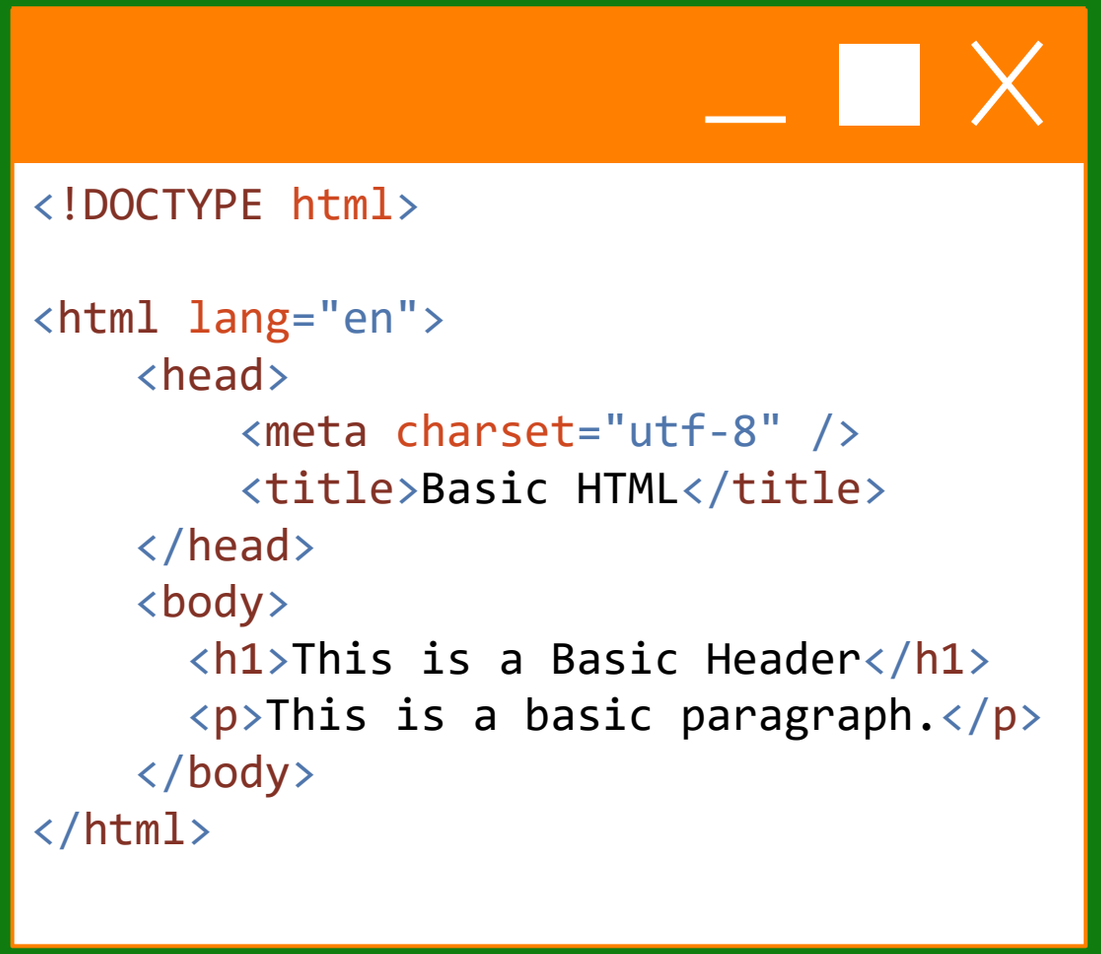
- Múltiplos navegadores com diferentes *engines*
- HTML
- CSS
- JavaScript

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





```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>
```

Hypertext Markup Language (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 iteratividade 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 Techonology 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

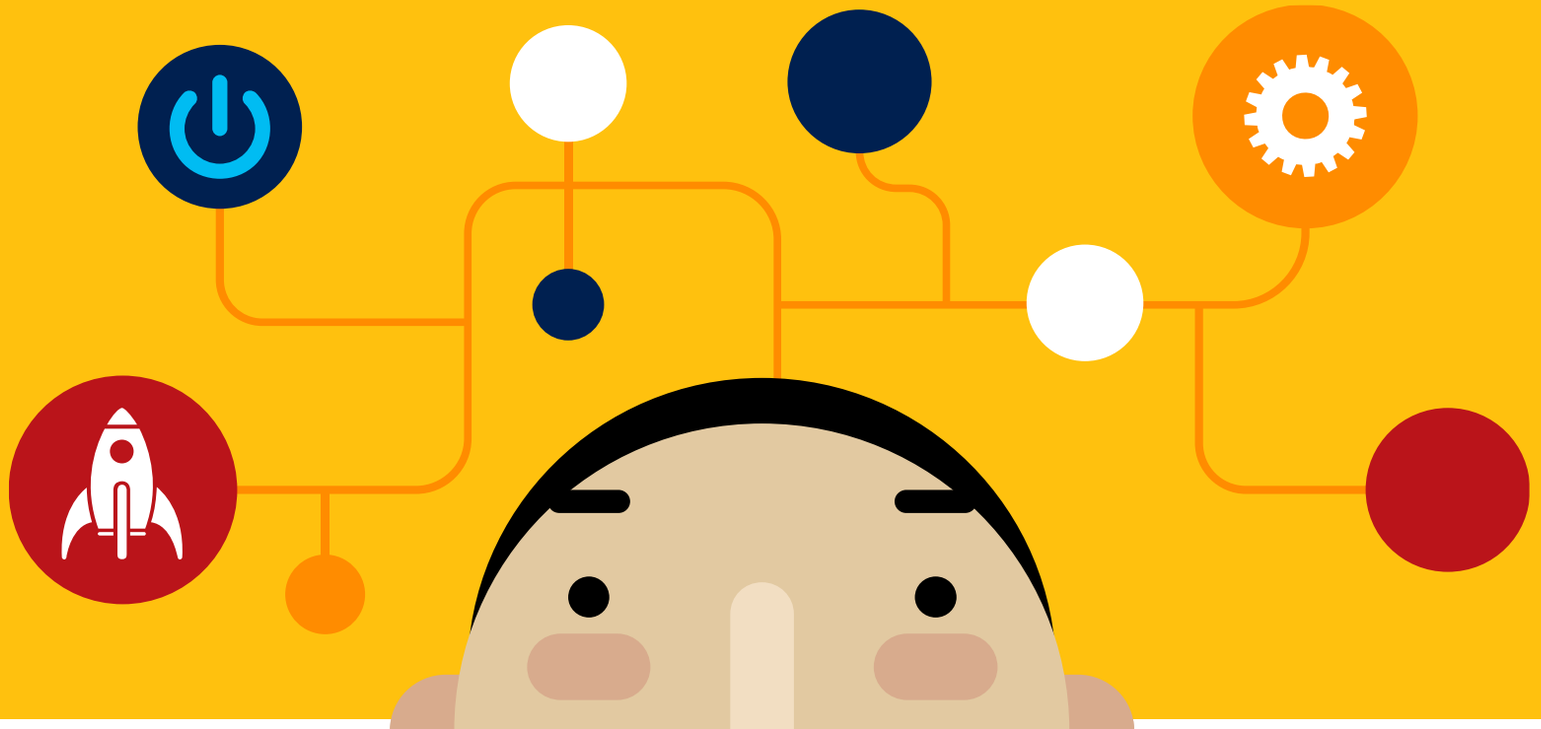
- Codesandbox
  - https://codesandbox.io/
- Stackblitz
  - https://stackblitz.com/
- Replit
  - https://replit.com/

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

```
<!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 abertura

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 lemento 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**

I *love* Centro de Inovação.

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	`&`	`&#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">
```

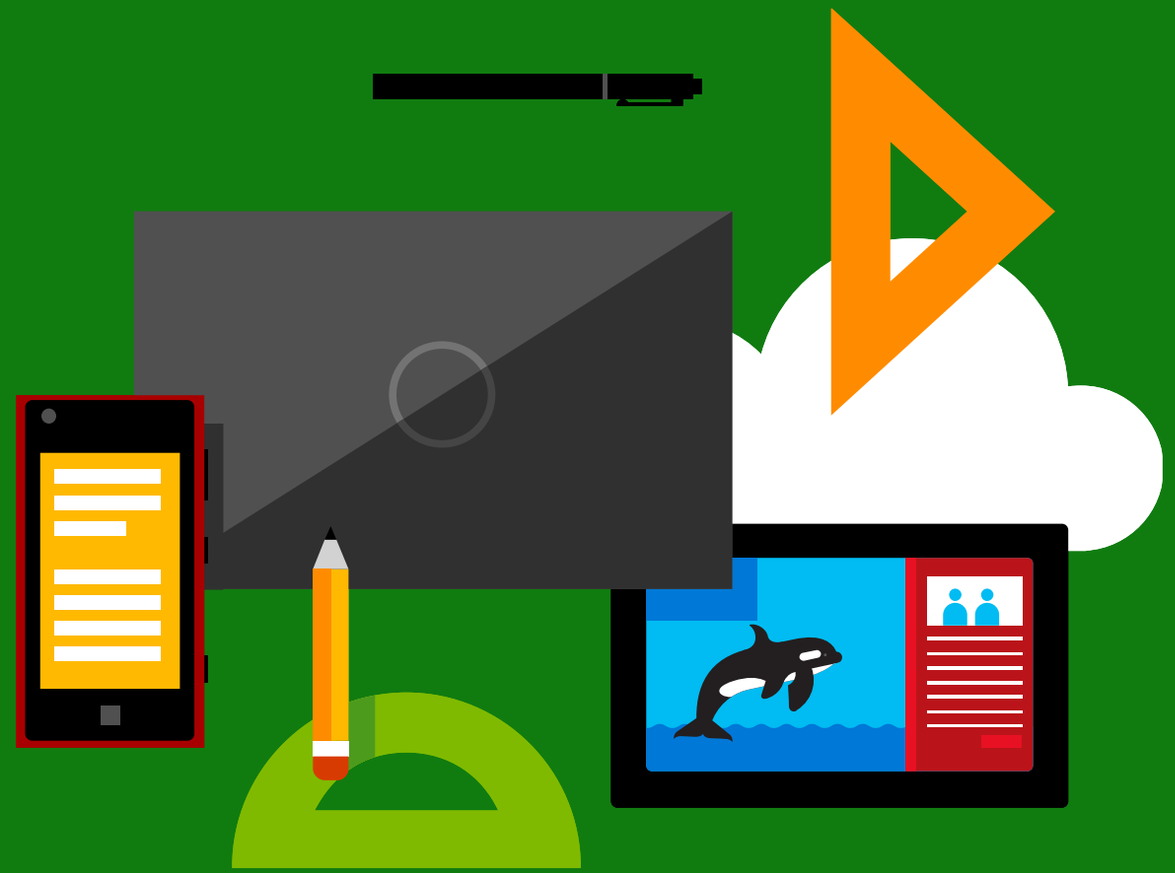
HTML5

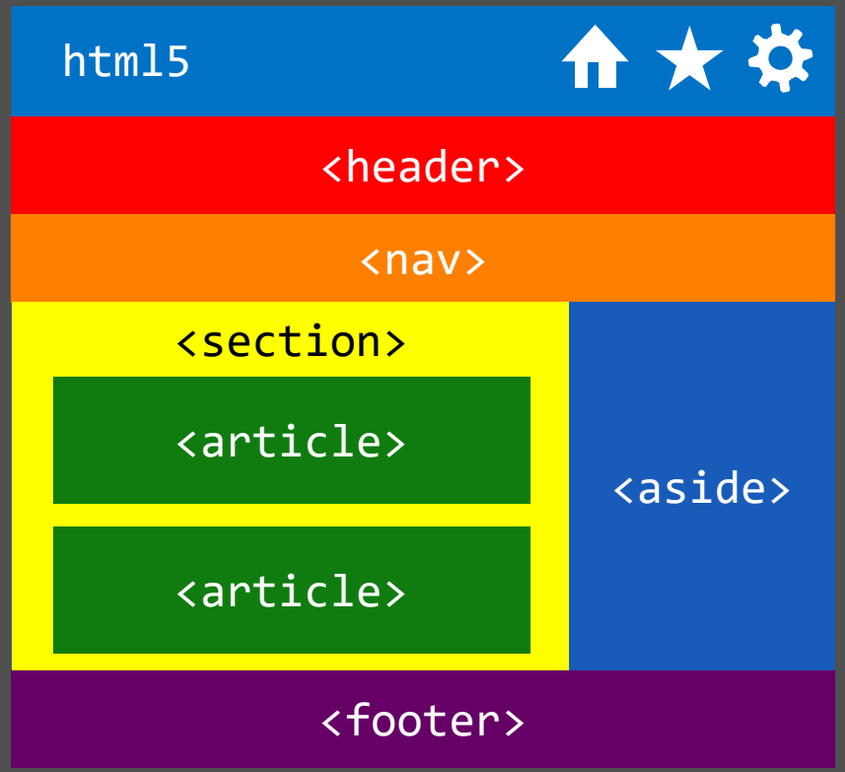
```
<!DOCTYPE html>
```

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

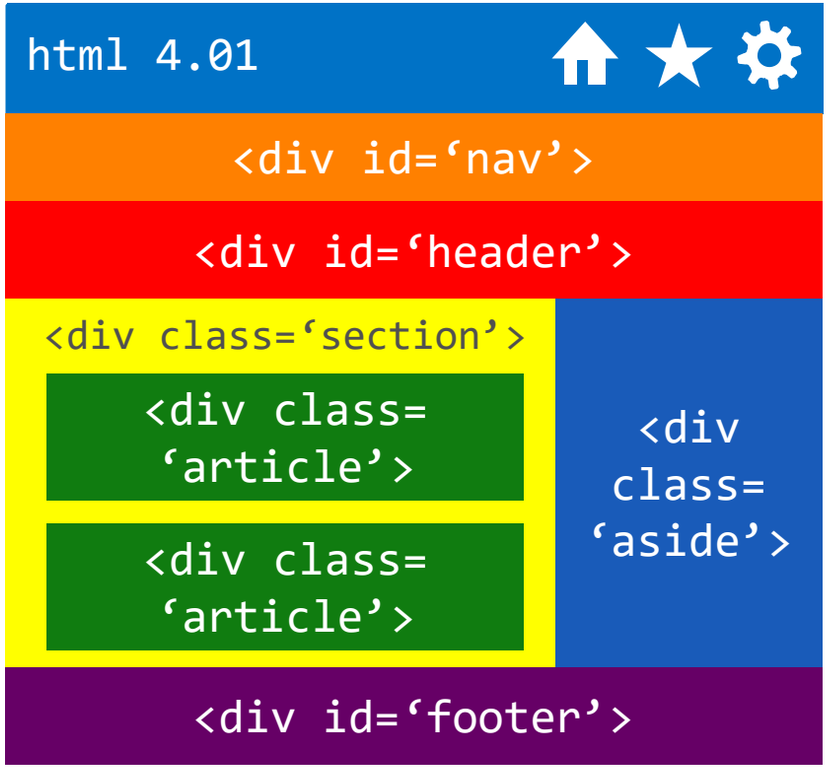
```
<div id="footer">Rodapé</div>
```

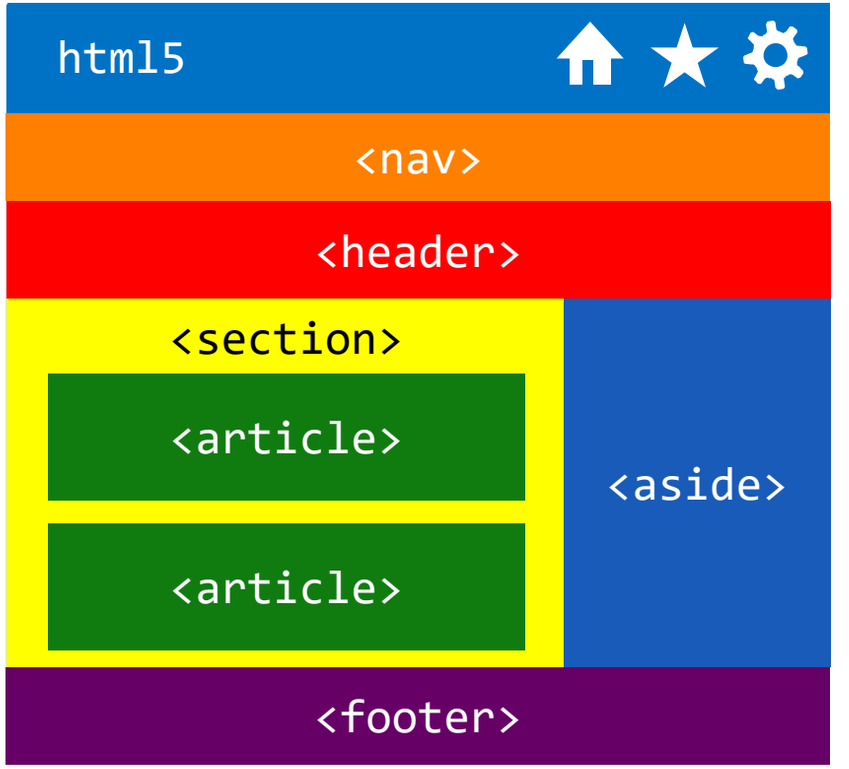
Tag `<div>`

- 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">
     <img src="dog.jpg" alt="My awesome dog, 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 contêm 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 infomações 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 independete de outro conteúdo na página	`article`
Planos para sindicalizar um bloco de conteúdo	`article`
Criar uma barra lateral	`aside`
Empacotar e posicionar múltiplas seções que não estão relacionados entre si	`div`

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>
```

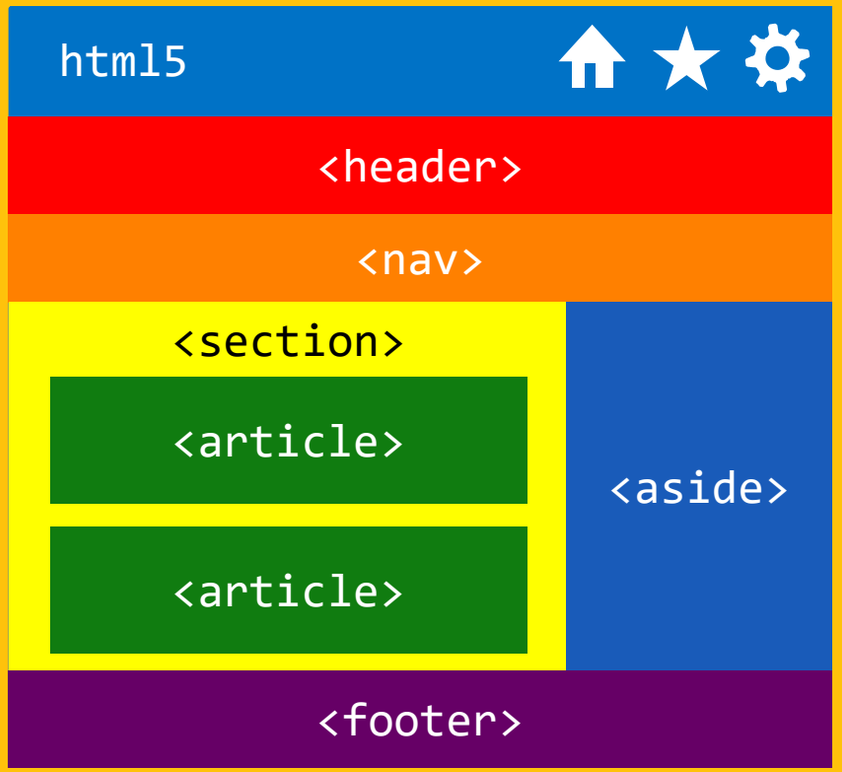
Elemento **nav**

- O elememtno **<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

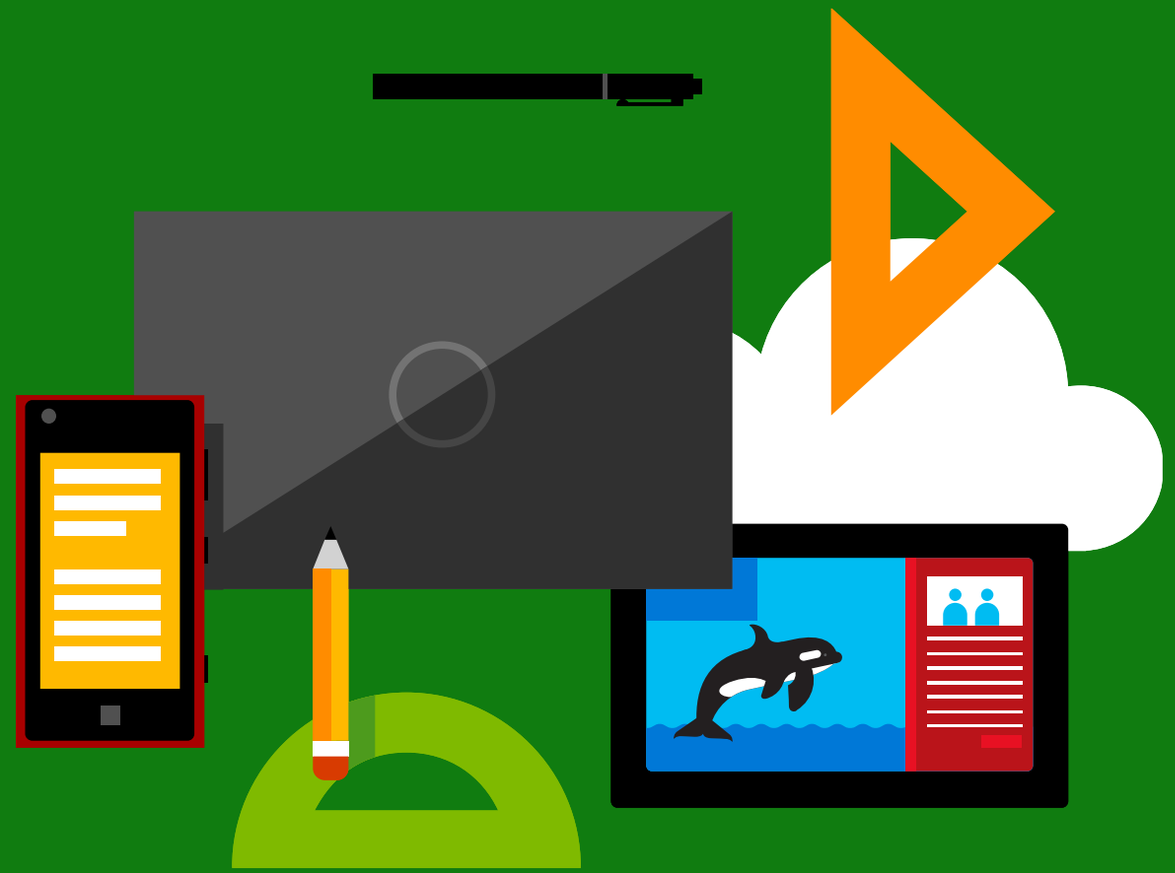
```
<nav>
      <a href="/html/">HTML</a>
      <a href="/css/">CSS</a>
      <a href=”/javascript/">JavaScript</a>
</nav>
```

Elementos `article` e `aside`

- O elemento `<article>` define um conteúdo indpendente e auto-contido
  - Um exemplo pode ser um artigo de notícia ou um post de blog
- O elemento `<aside>` é utilziado 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 altenativa 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

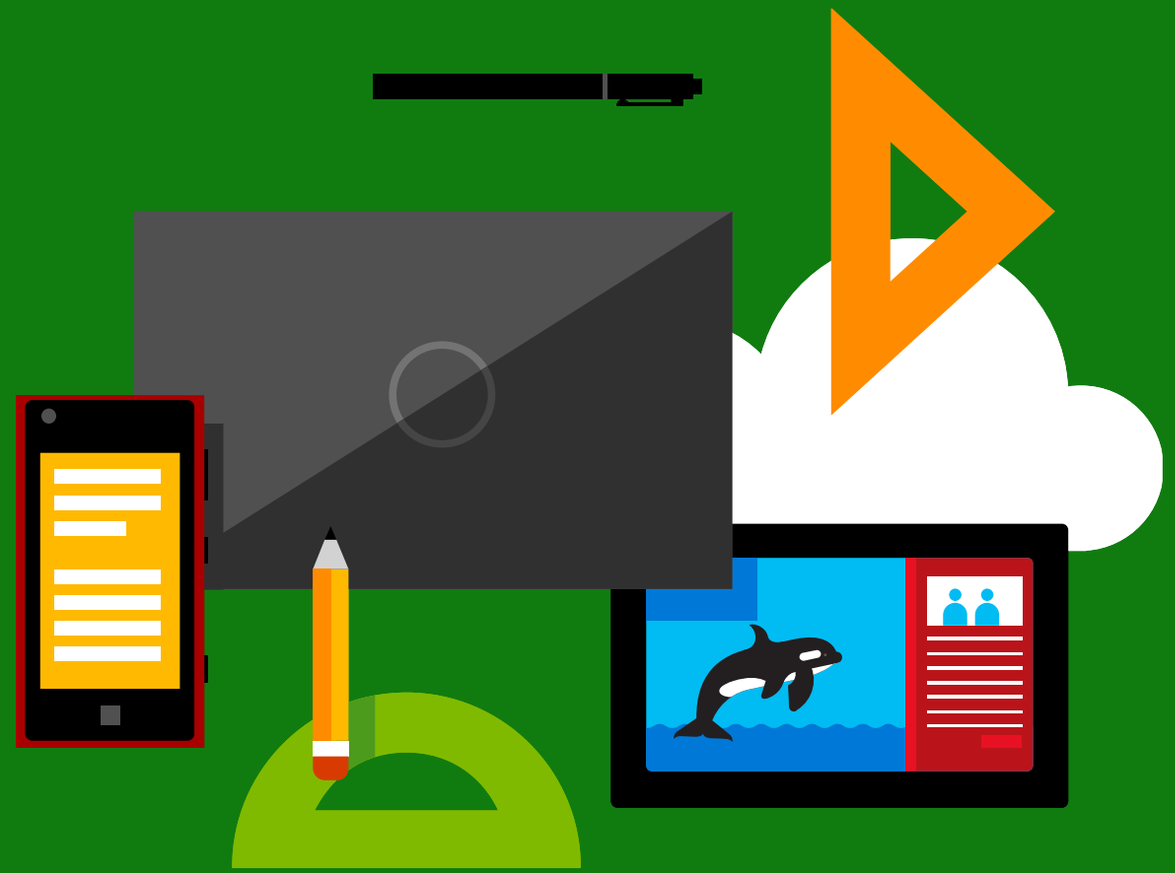
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>HTML Example</title>
 </head>
 <body>
    <h1>This is a sample page.</h1>
    <p>This page includes <em>some</em> <strong>nested
    elements</strong>.</p>
 </body>
</html>
```

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
`<table>`	Cria uma tabela
`<tr>`	Cria uma linha
`<th>`	Cria um cabeçalho
`<td>`	Cria uma célula dentro de uma linha -> uma coluna
`<colgroup>`	Aplicar estilos a uma seleção de colunas
`<thead>`	Marca um grupo de cabeçalhos de linhas
`<tfoot>`	Marca um grupo de rodapés de linhas
`<tbody>`	Utilizado para formatar grupod de linhas
`<caption>`	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

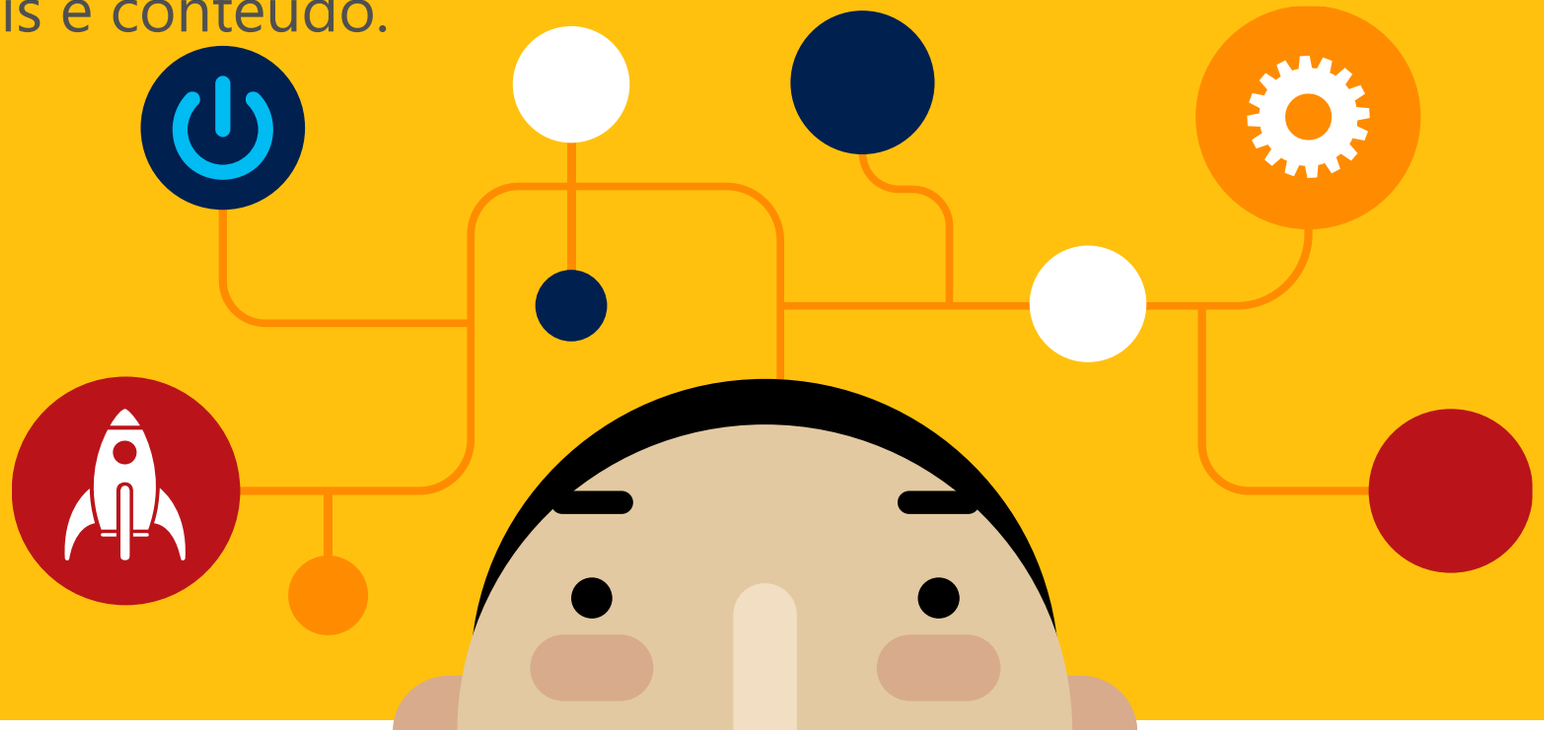
```
<h3>Seattle To-Do List</h3>
<ul>
    <li>Visit Space Needle</li>
    <li>Buy rain jacket</li>
</ul>
```

Exemplo

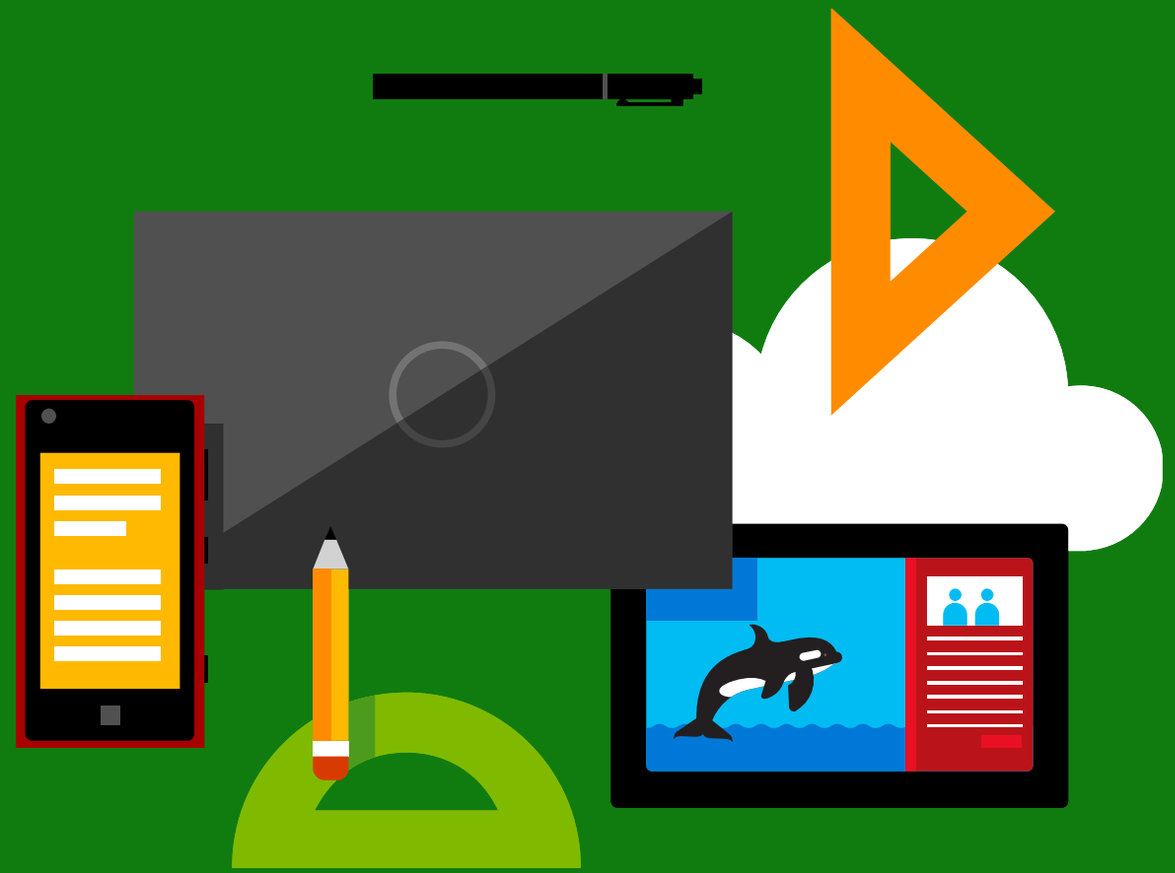
```
<body>
    <h3>Favorite Foods</h3>
    <ol>
        <li>Pizza</li>
        <li>Cake</li>
    </ol>

    <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

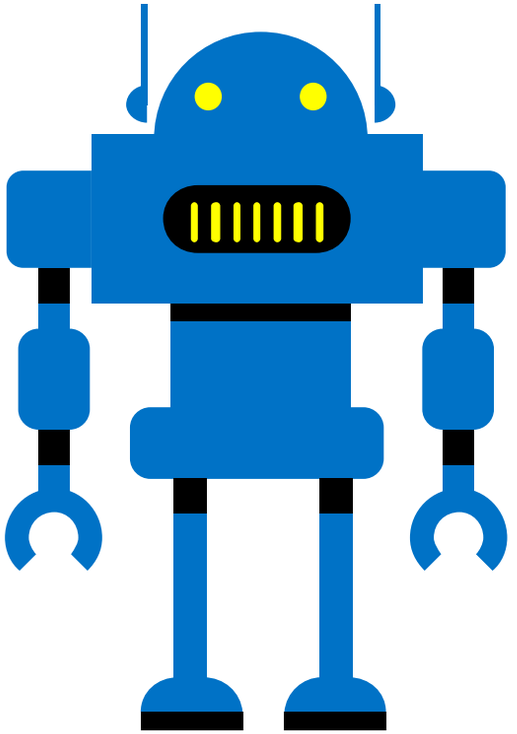


Imagens e Gráficos no HTML

- 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 manêm 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`

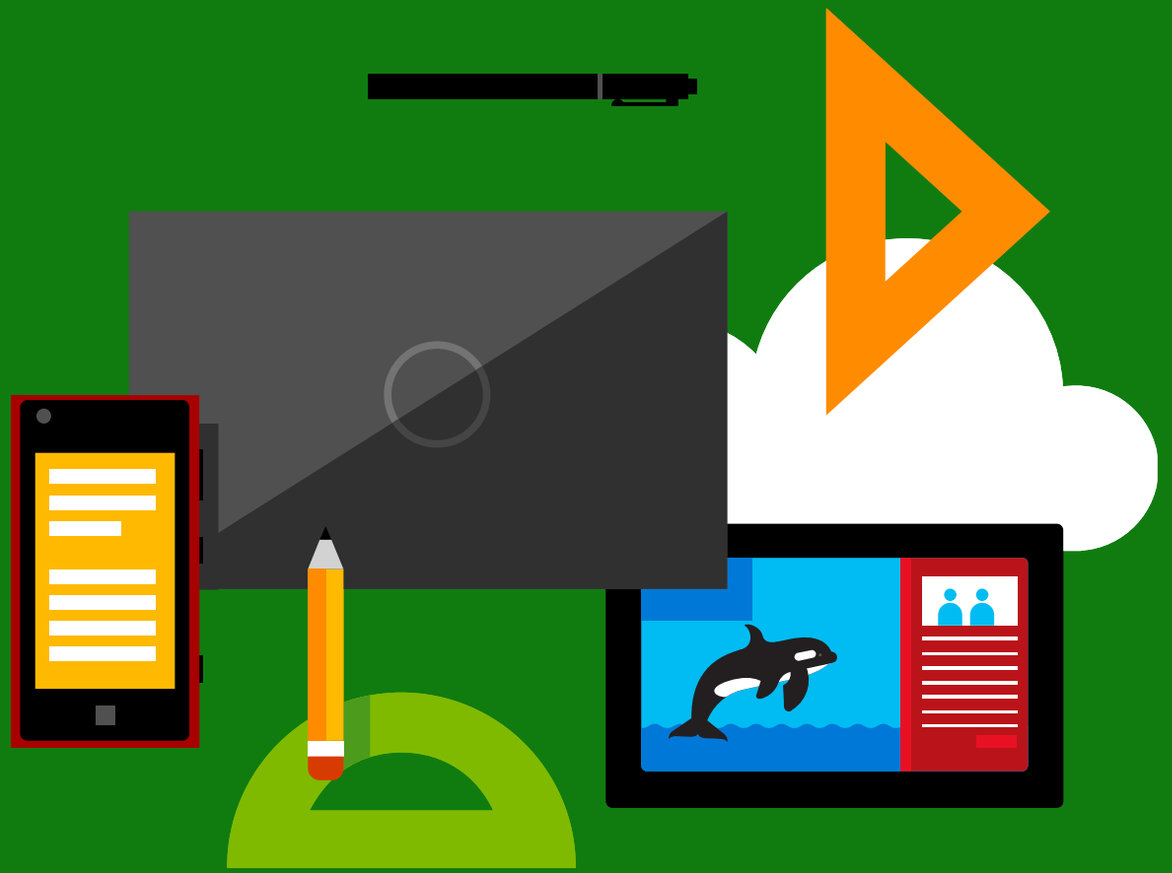
ATRIBUTO	VALOR	DESCRIÇÃO
`src`	URL	Especifica a localização da imagem
`alt`	Texto	Especifica texto acessível para a imagem
`height`	pixels	Especifica a altura da imagem
`width`	pixels	Especifica a largura da imagem
`ismap`	ismap	Especifica o uso de um mapa de imagem gerenciado pelo lado-servidor
`usemap`	#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>
    <img src="dog.jpg" alt="This is a picture of my dog, Stella." title="Stella"/>
</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

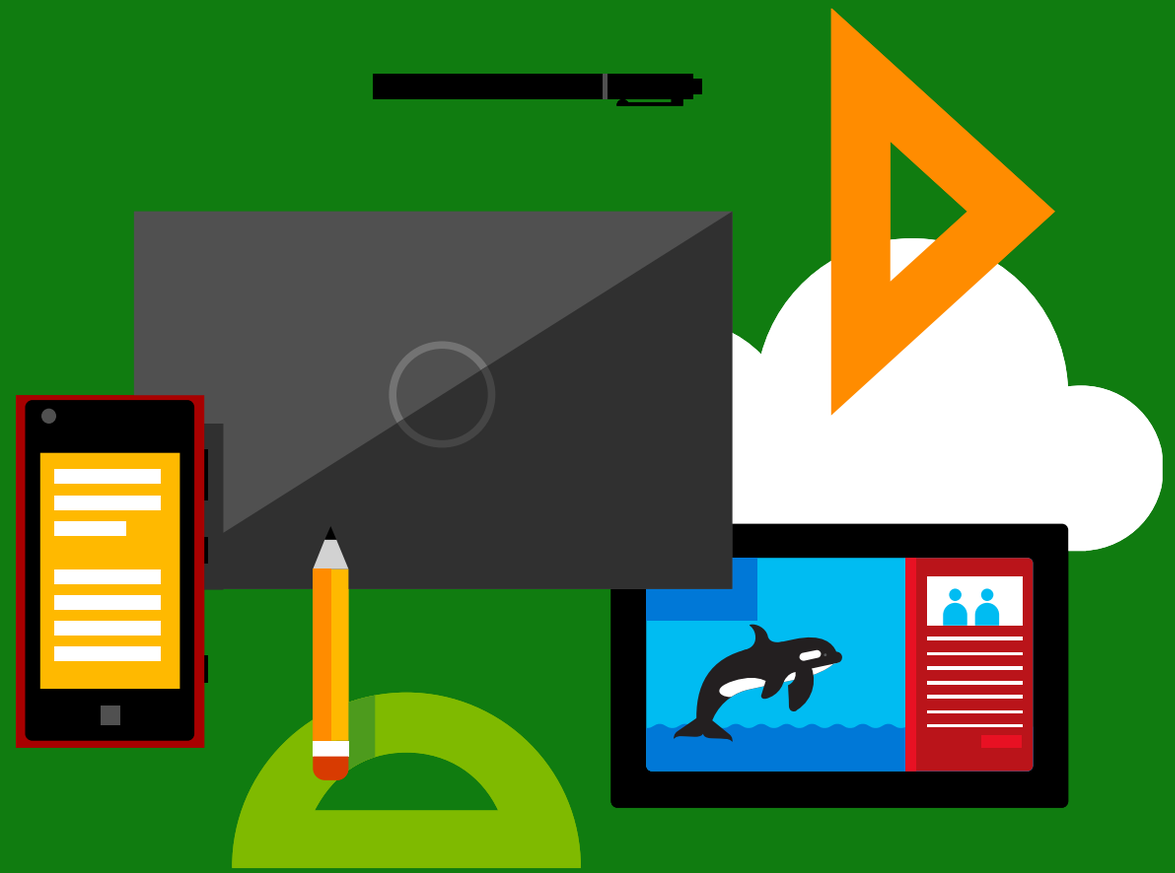
```
<canvas id="smlRectangle2" height="100" width="200">
        <img
         src="http://www.example.com/images/smlRectangle.jpg"
             alt="A blue rectangle" />
</canvas>
```

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
poster	Apresenta uma imagem estática enquanto o vídeo carrega
autoplay	Inicia o vídeo automaticamente ao caregar a página
controls	Apresenta controles na página para manipulação do vídeo
loop	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 comumento 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`

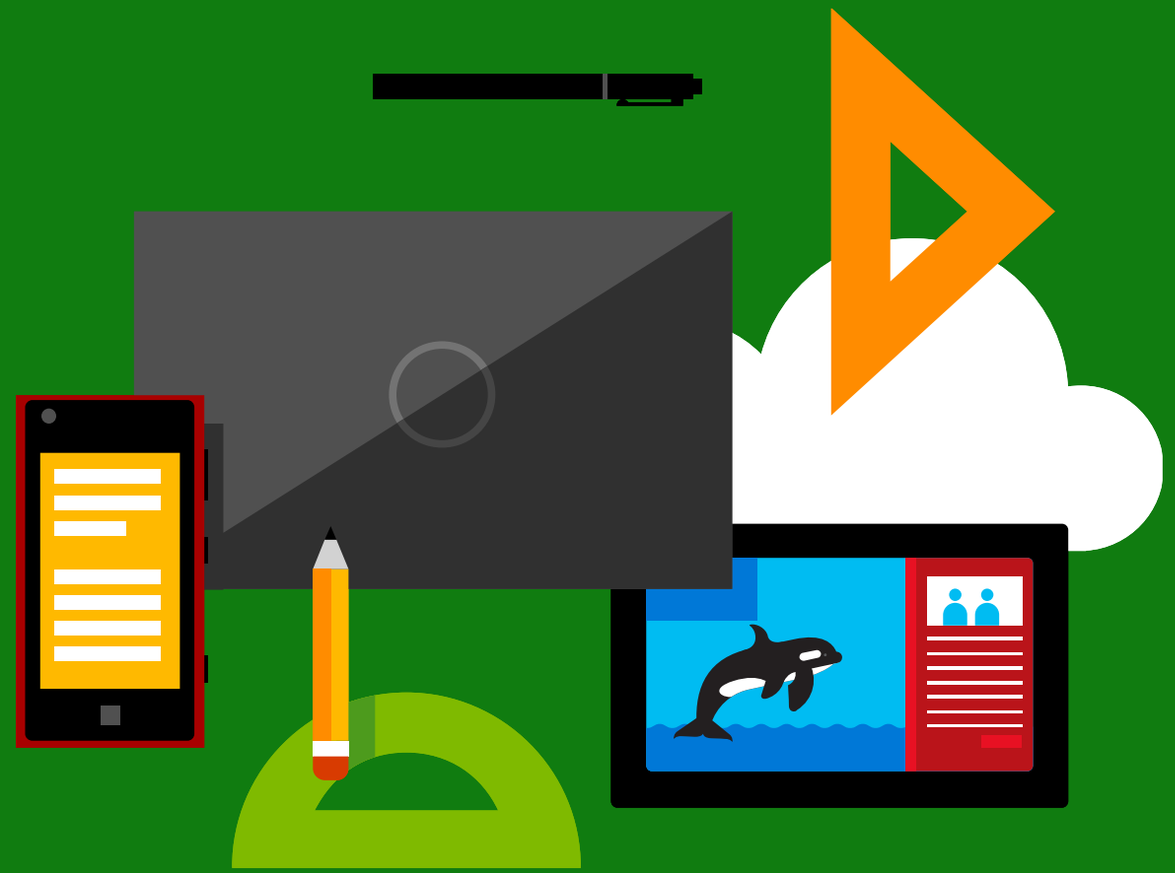
```
<audio src="myaudio.mp3" controls="controls"></audio>
```

Formatos de Áudio

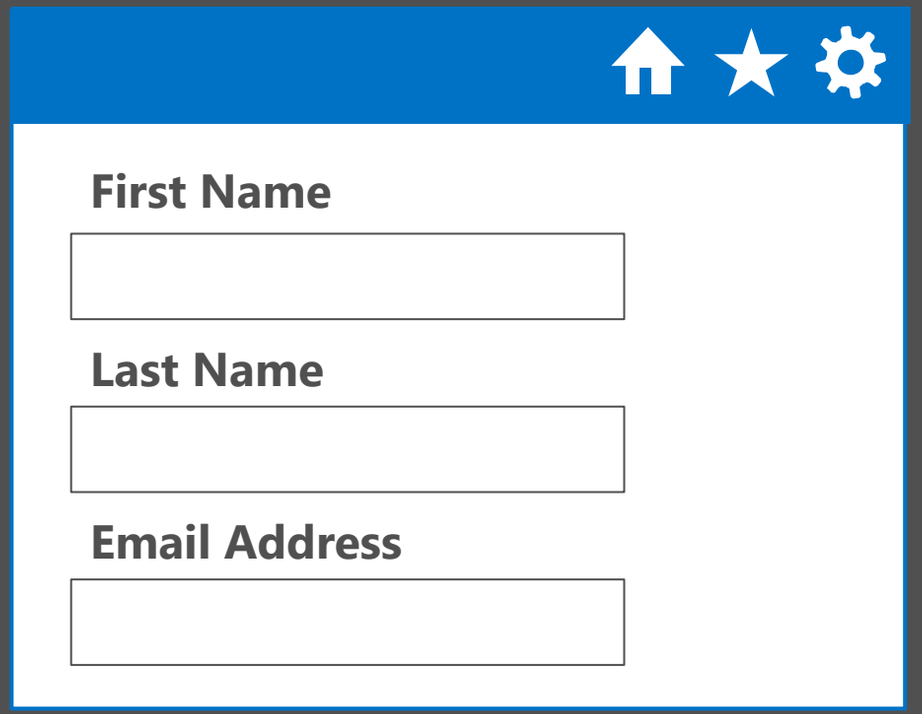
- Existem três formatos principais de áudio suportados pelos navegadores:
  - OGG
  - MP3
  - WAV
- Nem todo navegador suporta todo tipo de áudio
- Utiliza-se o atributo **`source`** para incluir múltiplos formatos

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

Entradas e Formulários



Formulário Web



- 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 formlário 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 dágua que ajuda o usuário com o conteúdo de um campo

autofocus

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

required

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

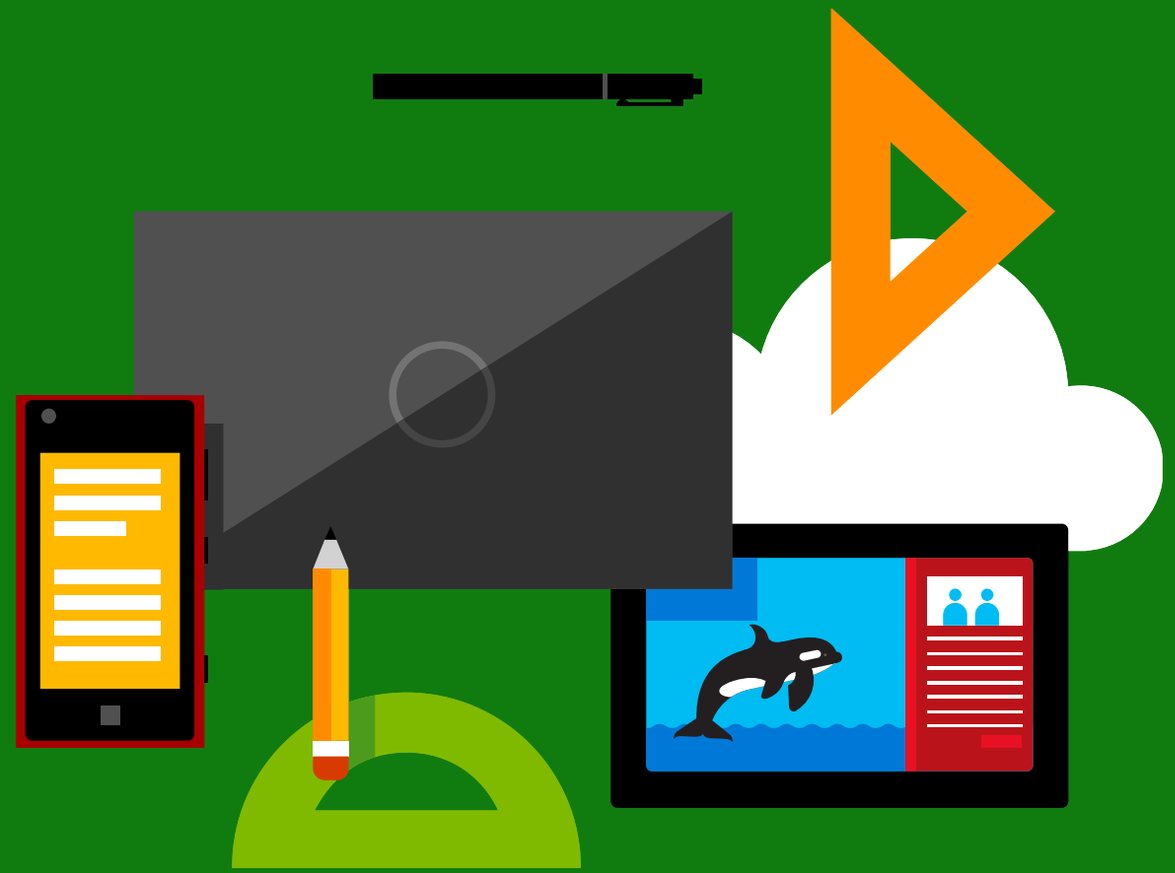
placeholder

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

Exemplo

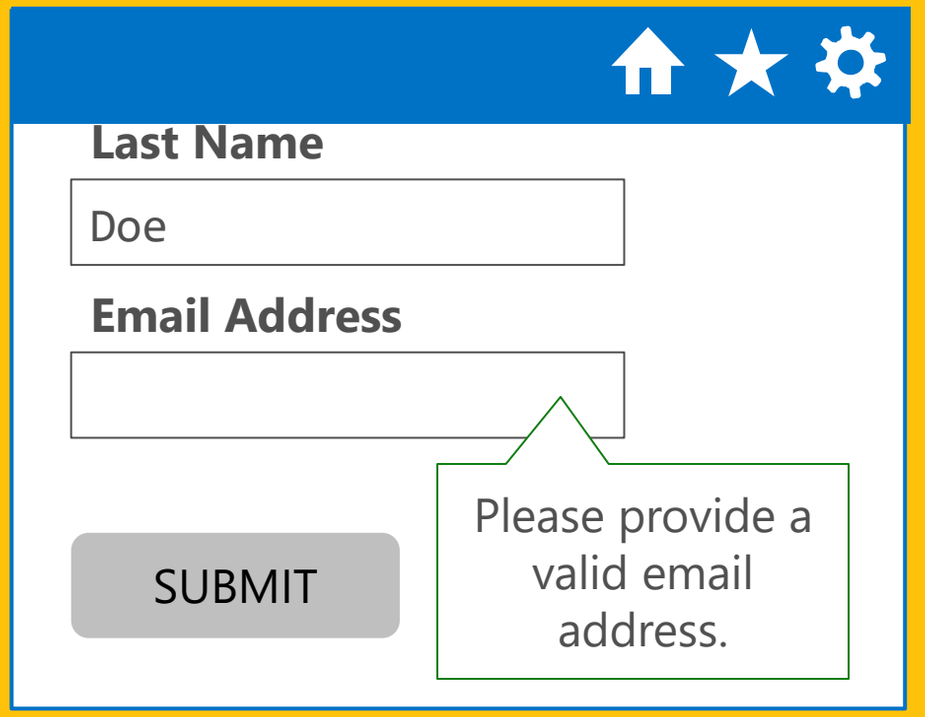
```
<body>
    <h1>Form Demo!</h1>
    <form id="contact" method="post" action="">
          <label for=“firstName">First Name</label>
          <input type="text" name=“firstName" placeholder="First Name"/><br/>
          <label for=“lastName">Last Name</label>
          <input type="text" name=“lastName" 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>
```

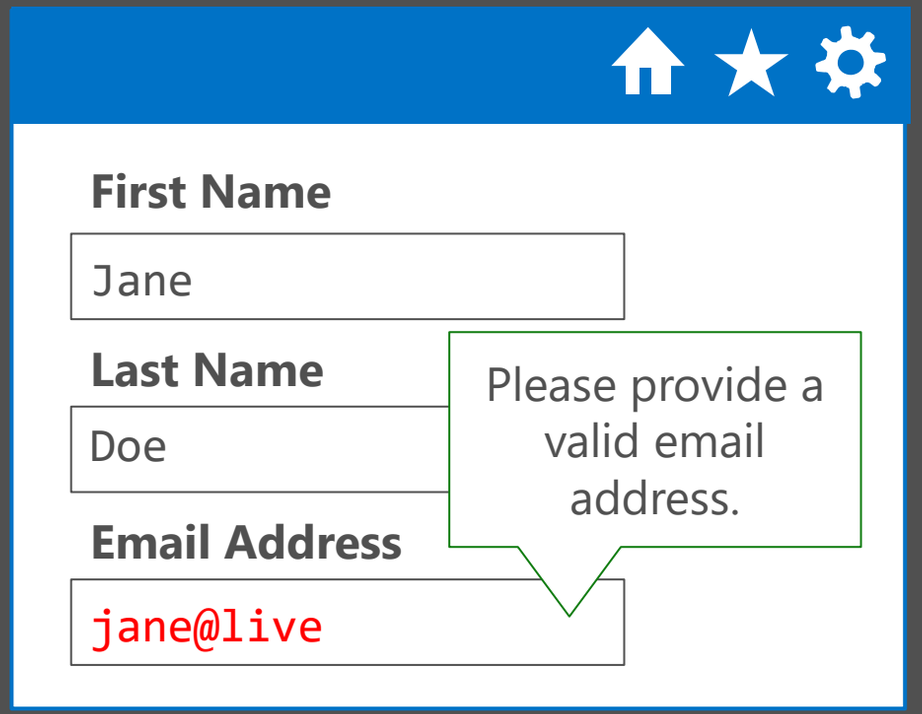
Validação de Formulários



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**



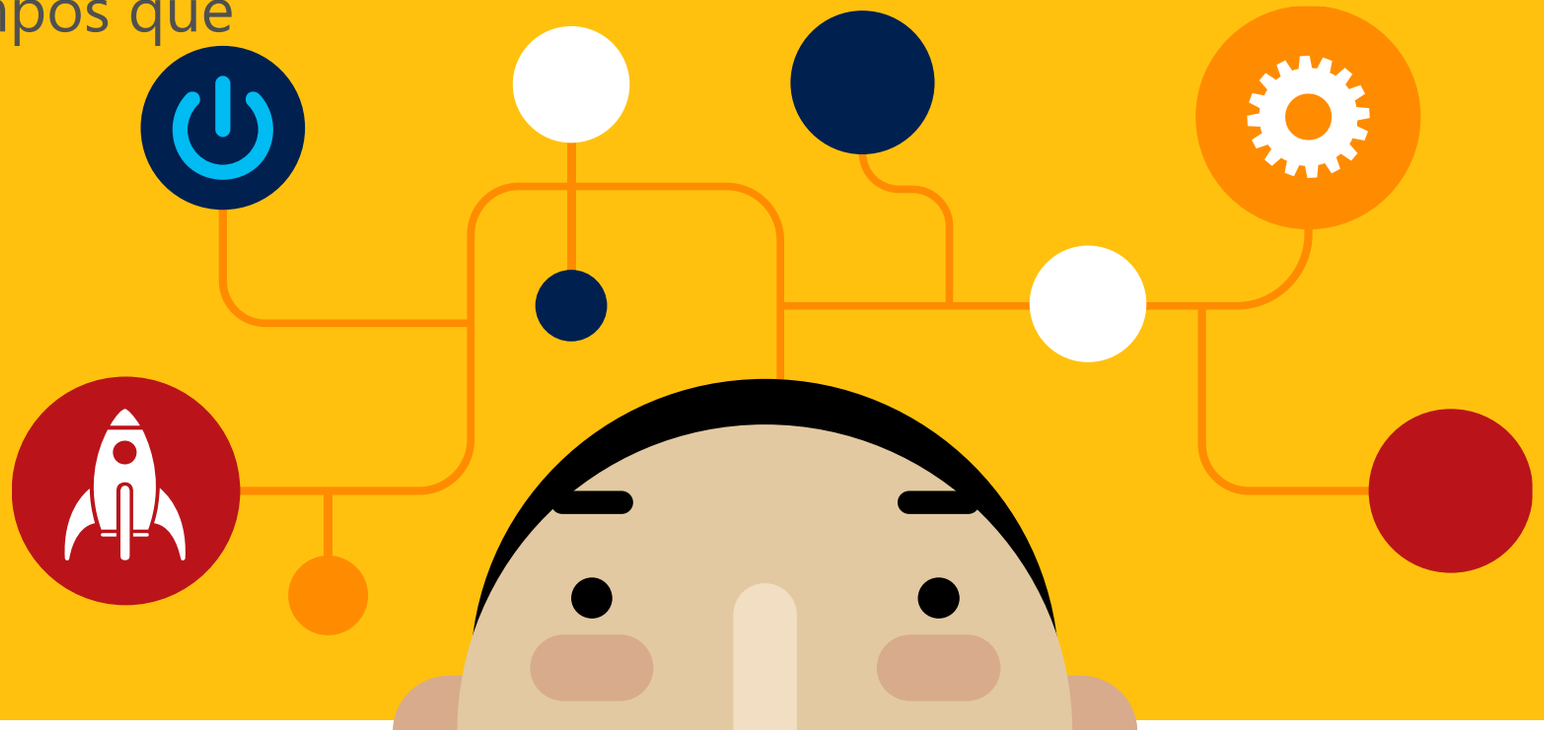


Validação no Lado-Cliente

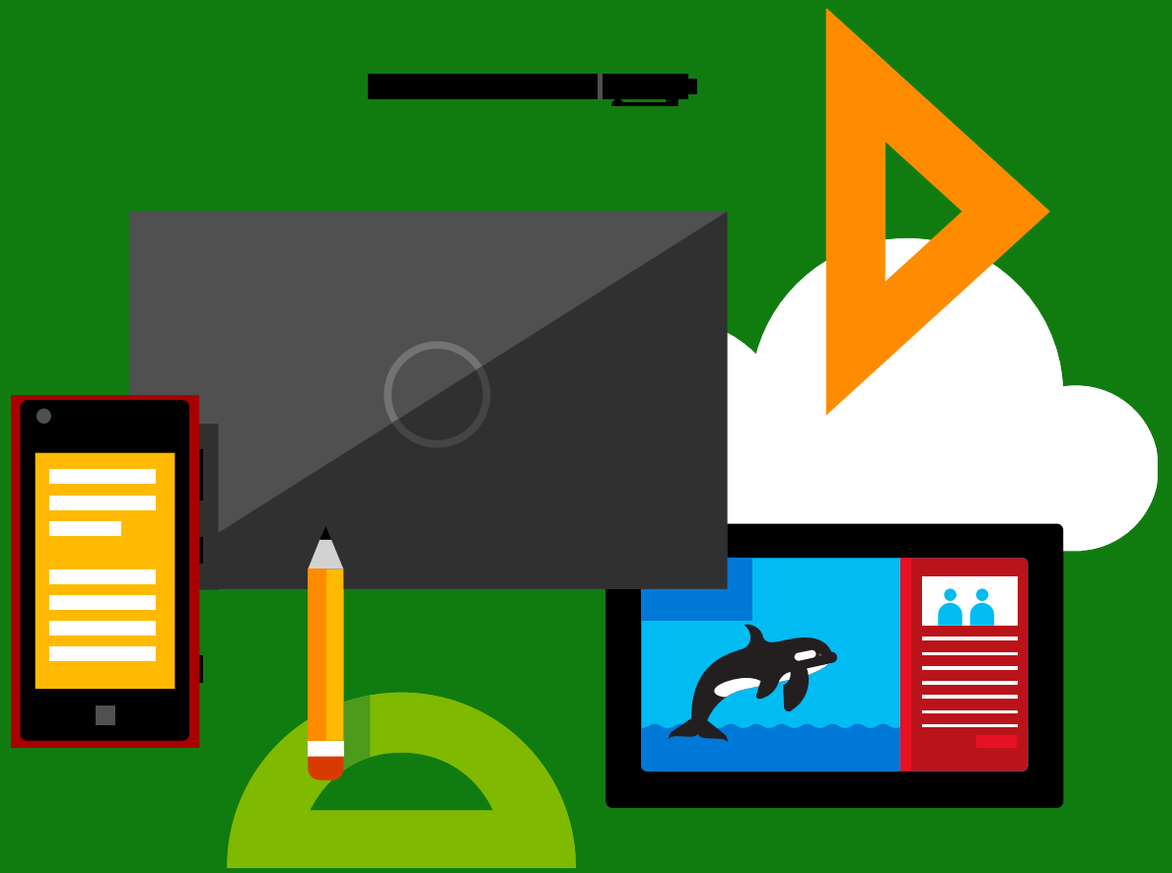
- Quando um navegador valida os dados providos por um usuário, chamamos de **validação no lado-cliente**
  - Se a validação é executado por um servidor, então chamamos de validação no lado-servidor
- Se um usuário informa um valor em formato incorreto em um campo, então o navegador instrui o usuário 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.

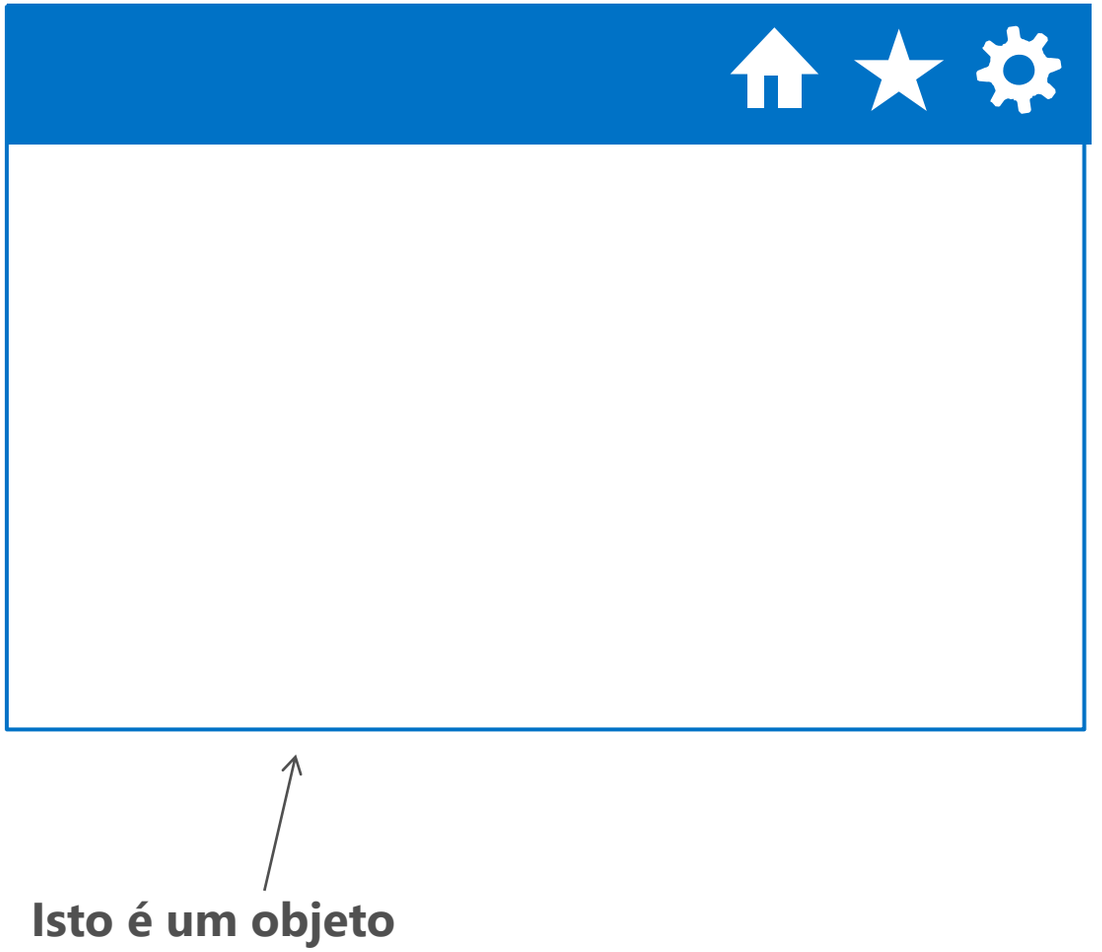


Manipulação do DOM



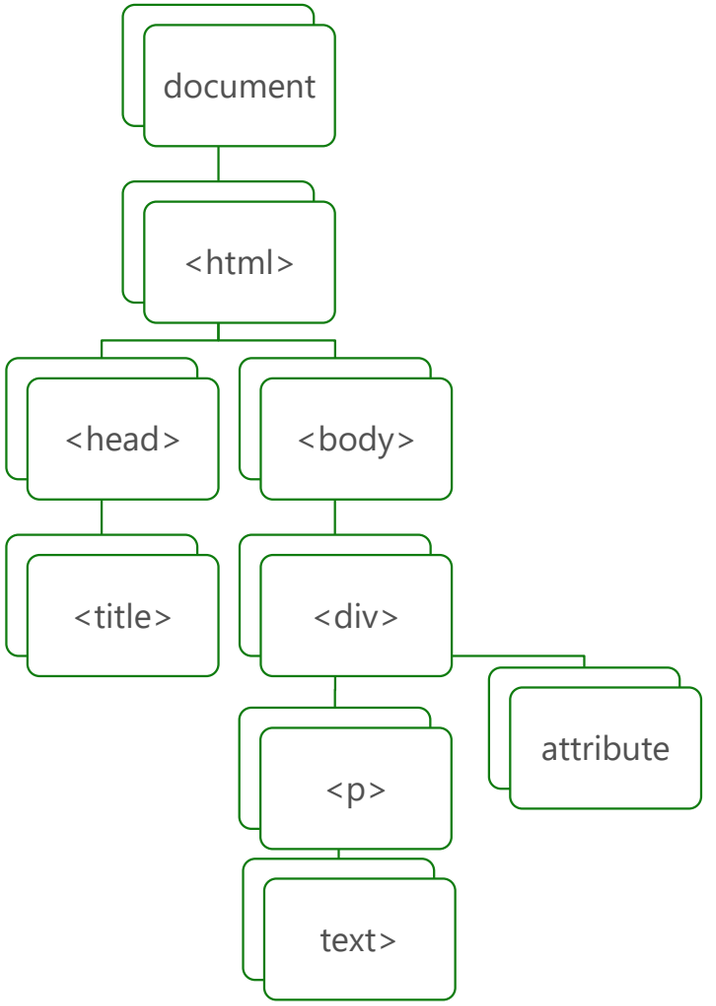
DOM

- 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



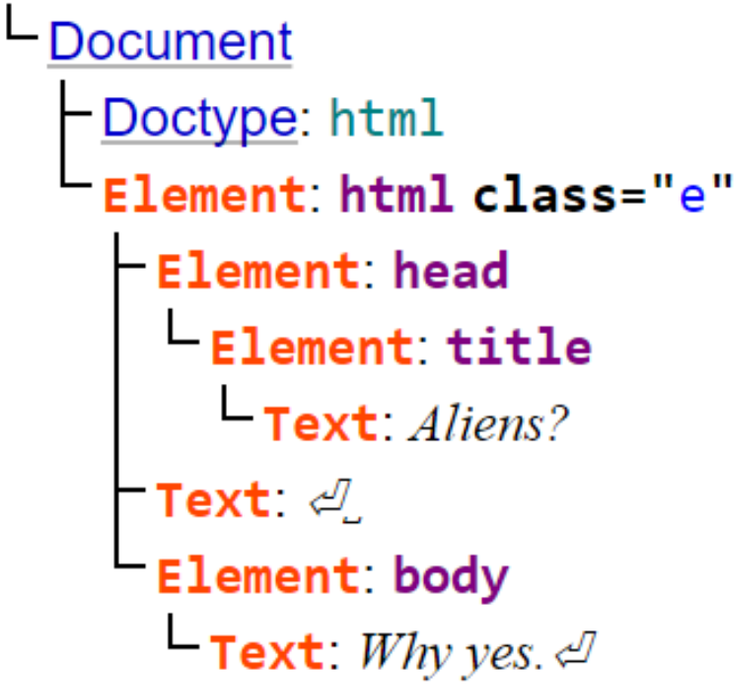
Document Object Model (DOM)

- 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>
```

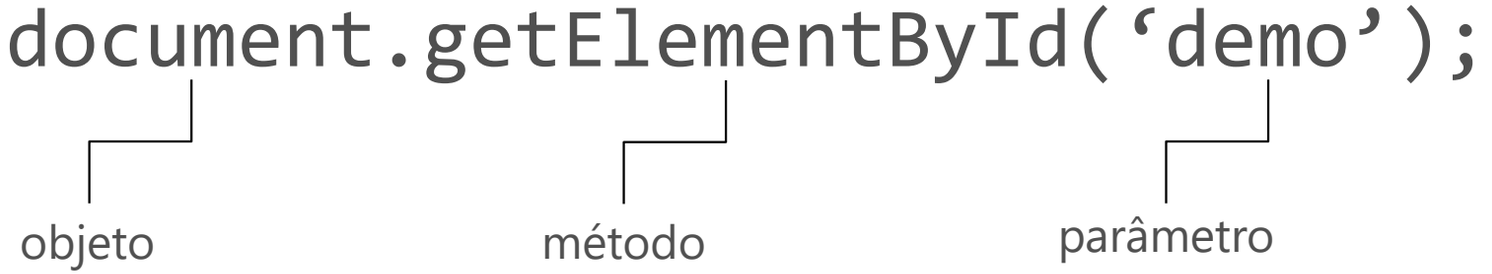


Document Object Model (DOM)

- 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



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 contéudo
  - 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

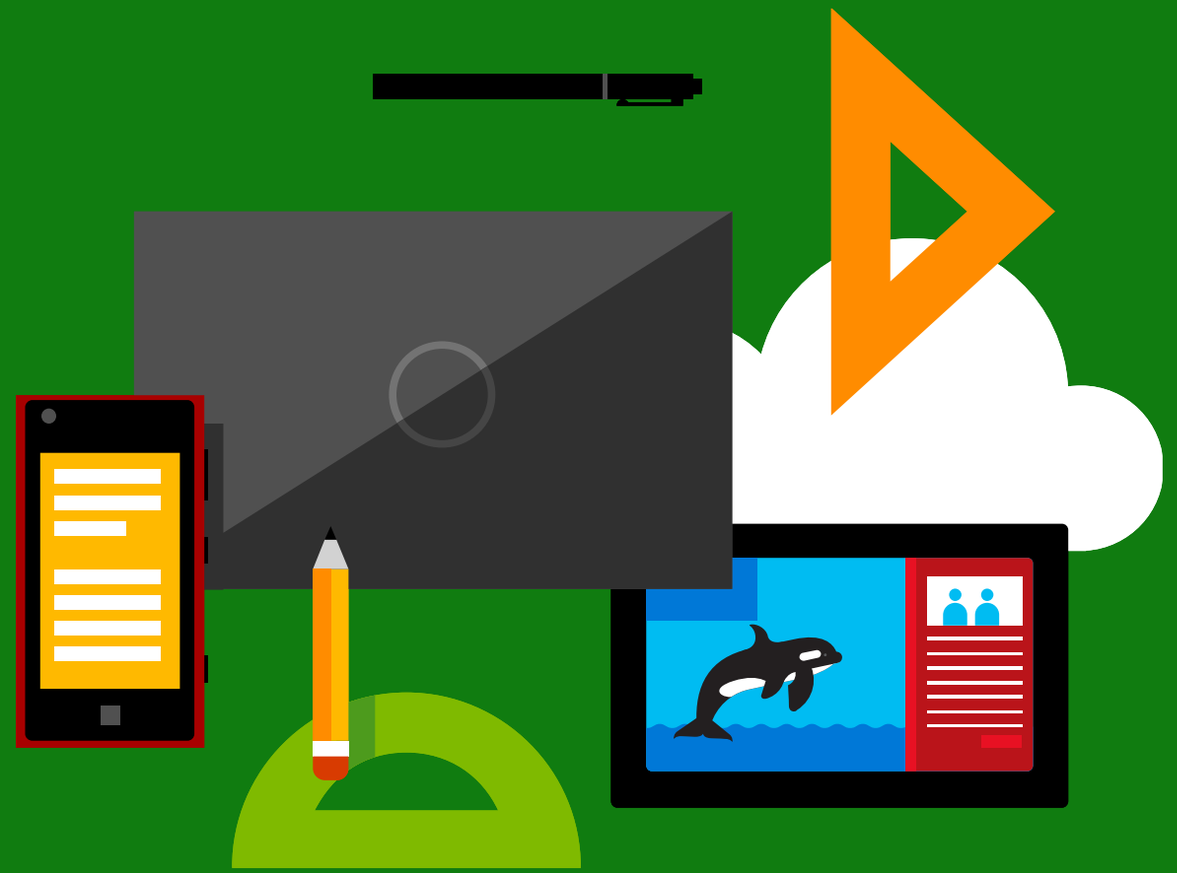
```
<button
    onclick="show_image (’dog.jpg’,
        276,110, ’Stella');">
        Display an image!
</button>
```

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
`onsubmit`	Submissão de formulário

`onkeydown`
`onkeypress`
`onkeyup`
	Uso de teclas

`onclick`
`onmousedown`
`onmouseup`
	Cliques de mouse ou touchpad

`onload` `onunload`	Carregamento ou descarregamento da página
`onselect`	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()`**