Frontend Essencial

HTML, CSS e Javascript

O que vamos aprender?

- Arquitetura de uma aplicação WEB
- Os pilares do desenvolvimento WEB

  Frontend
- Criação de templates
- Construção de documentos HTML
- Estilização de documentos com CSS
- Utilização de Bootstrap no desenvolvimento
- Lógica de programação com Javascript
- Apresentação Javascript ES6
- O que é o DOM (Document Object Model)
- Manipulação do DOM com Javascript
- Requisições AJAX
- Hospedagem de sites (Heroku)

Avaliação da Disciplina

- Avaliação conceito de 10 pontos
- Avaliação de 30 pontos
- Avaliação final de 60 pontos
  - Projeto em grupo
    - Construção de um site (60 pontos)

Documentação utilizada para disciplina

- W3Schools
- MDN Web Docs
- Bootstrap

Apostila Caelum - (Material Auxiliar)

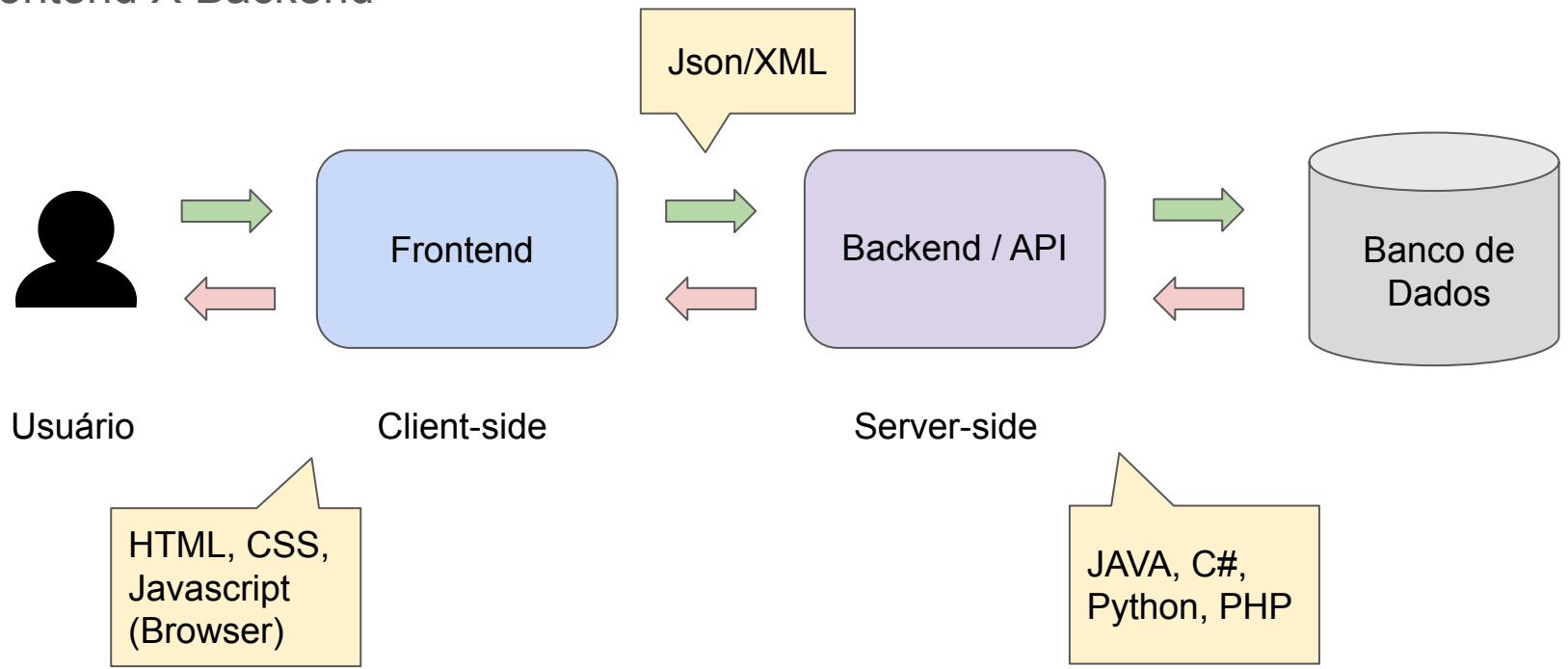
- https://www.caelum.com.br/apostila-html-css-javascript/

Ferramentas para o desenvolvimento

- Visual Studio Code (Editor de texto)
- Browser (Chrome)

Arquitetura de uma aplicação WEB

Frontend X Backend



Estrutura de uma aplicação WEB

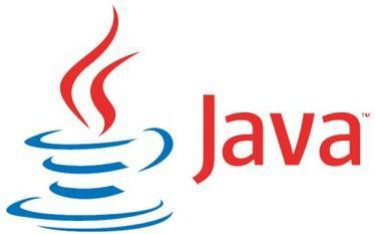
Frontend X Backend

- Frontend é responsável basicamente pela parte visual e interação do usuário com a aplicação onde utilizamos tecnologias como HTML, CSS e Javascript
- Backend é responsável por fazer a ponte entre os dados recebidos do frontend até o banco de dados ou vice-versa, responsável por garantir as regras de negócio, segurança, validações, realizar inserções e recuperar dados no banco de dados e realizar a manipulação dos dados.

Estrutura de uma aplicação WEB

Linguagens de programação X Linguagens de marcação

Linguagens de programação é usada na manipulação dos dados, enviando instruções para CPU que reescrevem os dados de entrada e saída.



Estrutura de uma aplicação WEB

- **Linguagens de marcação** é usada para controlar a apresentação dos dados, como representar esses nomes de usuário como uma lista de marcadores ou como uma tabela.

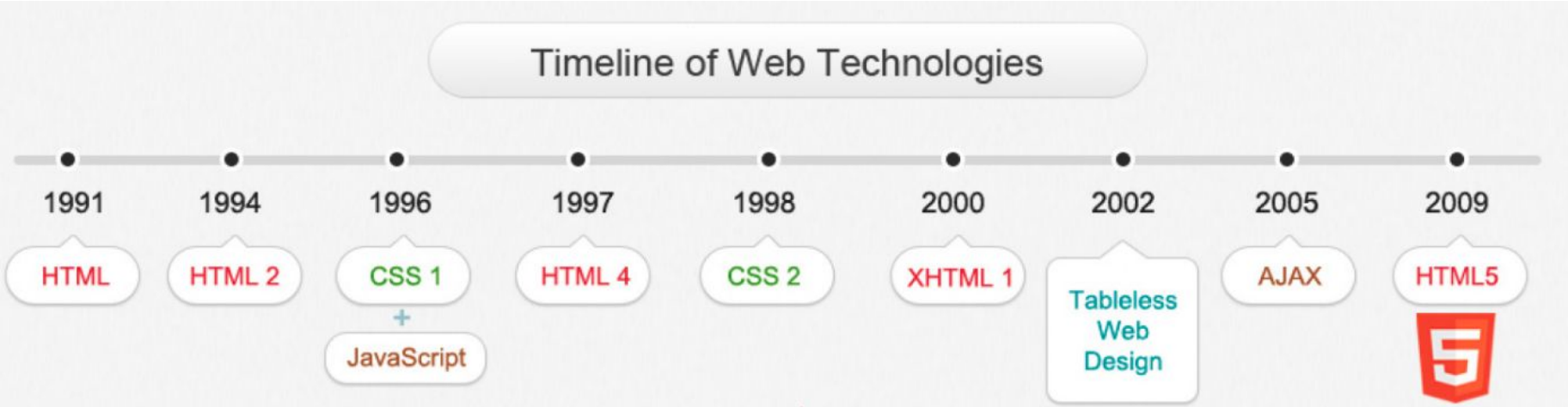
HTML

HTML

- HTML significa Hyper Text Markup Language
- HTML é a linguagem de marcação padrão para a criação de páginas da web
- HTML descreve a estrutura de uma página da web
- HTML consiste em uma série de elementos
- Os elementos HTML informam ao navegador como exibir o conteúdo
- Os elementos HTML identificam partes de conteúdo como "isto é um título", "isto é um parágrafo", "isto é um link", etc.

HTML

O HTML foi desenvolvido em 1991 por Tim Berners-Lee originalmente para o compartilhamento e disseminação de documentos científicos entre ele e seu grupo de colegas.



HTML - Estrutura básica

<!DOCTYPE html> ➔ Define que é um documento HTML 5

<html> ➔ Elemento raiz de uma página HTML

<head> ➔ Elemento contém meta informações sobre a página HTML

<title>Título</title> ➔ Especifica o título do documento

</head>

<body> ➔ Define o corpo do documento e é um container para todo o conteúdo visível.

<h1>Cabeçalho</h1> ➔ Elemento define um grande título.

Parágrafo ➔ Elemento define um parágrafo.

</body>

</html>

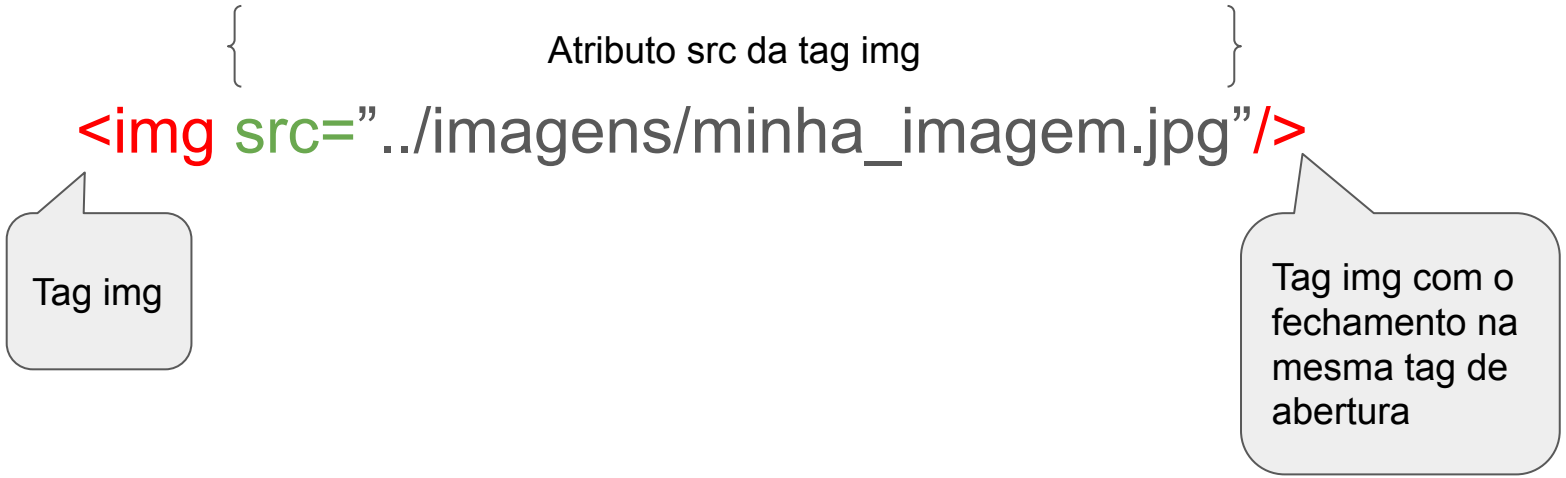
HTML - Tag</>

O que é uma tag?

- As tags são marcações que são identificadas pelo browser e tem como objetivo formatar o conteúdo contido em seu escopo.
- No documento HTML na maioria dos casos teremos uma tag de abertura e uma tag de fechamento.

HTML - Tag</>

- No entanto podemos ter tags que não englobam um conteúdo e nesse caso não necessitamos de uma tag de fechamento.
- Uma tag ainda pode receber atributos como apresentado no exemplo abaixo… Vamos explorar mais esse conceito um pouco mais a frente.



HTML - Elemento

- Um elemento HTML é definido por uma tag inicial, algum conteúdo e uma tag final.

<nomeDaTag> algum conteúdo… </nomeDaTag>

- É considerado um elemento HTML toda a estrutura desde da tag de abertura até a tag de fechamento

<h1>Titulo</h1>

Parágrafo

HTML - Declaração <!DOCTYPE>

- A <!DOCTYPE>declaração representa o tipo de documento e ajuda os navegadores a exibir as páginas da web corretamente.
- Deve aparecer apenas uma vez, no topo da página (antes de quaisquer tags HTML).
- A <!DOCTYPE>declaração não diferencia maiúsculas de minúsculas.
- A <!DOCTYPE>declaração para HTML5 é:

```
<!DOCTYPE html>
```

HTML - Tag <html>

- Na estrutura do documento HTML a primeira tag a ser inserida é a tag **<html>** indicando o início do documento.
- Outras duas tags obrigatórias na estrutura são **<head>** e **<body>**

```
<html>

    <head></head>

    <body></body>

</html>
```

HTML - Tag <html>

- Algo muito comum e “importante” de informarmos nessa tag é o idioma no qual o documento HTML está escrito.
- Fazemos isso incluindo a “palavra chave” **lang** na tag de abertura <html>

```
<html lang="pt-BR">
```

HTML - Tag <head>

- O conteúdo da tag <head> não será exibida no navegador, esta tag serve para inserção de metadados que serão consumidos pelo navegador. Ele contém informações como title , links para <css> (se você deseja modelar seu conteúdo HTML com CSS), links para favicons personalizados e outros metadados (dados sobre o HTML, como quem o escreveu, e palavras-chave importantes que descrevem o documento.)
- Um metadado obrigatório que se insere na tag **<head>** é a tag **<title>** que define o titulo de todo o documento HTML

HTML - Tag <head>

- Outro metadado bastante comuns vistos em documentos HTML escritos em portugues (por nosso idioma possuir acentos e cedilha) é a tag de codificação de caracteres **<meta>**

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

HTML - Tag <head>

Mais metadados que podem ser adicionados na tag <head>

- Adicionando autor e descrição

```
<meta name="author" content="Chris Mills">

<meta name="description" content="A Área de Aprendizagem do MDN tem como objetivo

proporcionar iniciantes em Web com tudo o que eles precisam saber para começar a desenvolver sites e aplicativos.">
```

HTML - Tag <head>

Mais metadados que podem ser adicionados na tag <head>

- Inserindo um favicon

```
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```

HTML - Tag <head>

Mais metadados que podem ser adicionados na tag <head>

- Importando folha de estilo externa CSS

```
<link rel="stylesheet" href="meu-arquivo-css.css">
```

- Importando arquivos Javascript

```
<script src="meu-arquivo-js.js"></script>
```

Vamos explorar mais a fundo todos esses tipos de imports mais a frente!!!!

HTML - Tag <body>

- Essa tag é responsável por todo o conteúdo exibido pelo navegador.
- É necessário que o <body> tenha pelo menos um elemento filho, ou seja, uma ou mais tags HTML em seu escopo.

  Exemplo:

```
<body>    => Elemento pai

    <h1>Título Qualquer</h1>    => Elemento filho de body

    <p>Primeiro parágrafo.</p>    => Elemento filho de body

</body>
```

Explorando algumas tags para marcação de textos...

- Temos tags para formatação de **títulos**, **parágrafos** e podemos criar **ênfase** para alguma palavra ou trecho do texto.
- Para formatação de títulos utilizamos as tags:

```
<h1>Heading 1</h1>

<h2>Heading 2</h2>

<h3>Heading 3</h3>

<h4>Heading 4</h4>

<h5>Heading 5</h5>

<h6>Heading 6</h6>
```

Explorando algumas tags para marcação de textos...

- Para formatação de textos podemos utilizar a tag <p> para definirmos um parágrafo.

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

<p>This is another paragraph.</p>
```

- Para criar uma linha horizontal, usada para separar conteúdos, podemos utilizar a tag <hr> e para uma quebra de linha podemos utilizar a tag
, ambos elementos não possuem tag de fechamento.

Explorando algumas tags para marcação de textos...

- Caso precisamos que algum texto seja exibido em um formatação pré-definida, ou seja, não queremos que nosso texto sofra uma formatação do navegador podemos utilizar o elemento <pre> esse elemento permite apresentar o texto na formatação em que foi escrito no documento.

Explorando algumas tags para marcação de textos...

Exemplo:

```
<p>
  My Bonnie lies over the ocean.
  My Bonnie lies over the sea.
  My Bonnie lies over the ocean.
  Oh, bring back my Bonnie to me.
</p>
```

Explorando algumas tags para marcação de textos...

Exemplo:

```
<pre>
  My Bonnie lies over the ocean.
  My Bonnie lies over the sea.
  My Bonnie lies over the ocean.
  Oh, bring back my Bonnie to me.
</pre>
```

Tags HTML - Marcações de ênfase

- Elementos para exibir tipos especiais de texto

- <b> - texto em negrito
- - texto importante
- <i> - texto em itálico
- - texto enfatizado
- <mark> - texto marcado

- <small> - texto menor
- <del> - texto excluído
- <ins> - texto inserido
- <sub> - texto subscrito
- <sup> - texto sobrescrito

Aplicando estilos

- Podemos estilizar os elementos HTML através do atributo **style.** Alguns dos muitos estilos que podemos aplicar são cor, fonte, tamanho e muitos outros.

Eu sou vermelho

Eu sou azul

Eu sou grande

Aplicando estilos

- Como fazemos isso?

<*tagname* style="*property*:*value*;">

- *property* => Propriedade **CSS**
- *value* => Valor **CSS**

Aplicando estilos

- Definir a cor de fundo de um elemento

```
<body style="background-color:powderblue;">

<h1 style="background-color:powderblue;">This is a heading</h1>

<p style="background-color:tomato;">This is a paragraph.</p>
```

Aplicando estilos

- Definir a cor do texto

```
<body style="color:powderblue;">

<h1 style="color:blue;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>
```

Aplicando estilos

- Definir a font do texto

```
<h1 style="font-family:verdana;">This is a heading</h1>

<p style="font-family:courier;">This is a paragraph.</p>
```

Aplicando estilos

- Definir o tamanho do texto

```
<h1 style="font-size:300%;">This is a heading</h1>

<p style="font-size:160%;">This is a paragraph.</p>
```

Aplicando estilos

- Definir o alinhamento do texto

```
<h1 style="text-align:center;">Centered Heading</h1>

<p style="text-align:center;">Centered paragraph.</p>
```

Aplicando estilos

- Atributo style e algumas propriedades CSS

- Use o style atributo para estilizar os elementos HTML
- Use background-color para a cor de fundo
- Use color para cores de texto
- Use font-family para fontes de texto
- Use font-size para tamanhos de texto
- Use text-align para alinhamento de texto

Vamos aprofundar nos estilos CSS mais a frente!!!

Inserindo comentários

- Podemos incluir comentários em nosso documento HTML, que não será interpretado pelo navegador ao exibir o conteúdo.

```
<!-- This is a comment -->
<p>This is a paragraph.</p>
<!-- Remember to add more information here -->
```

Vamos começar a COLORIR as coisas

- As cores HTML são especificadas com nomes de cores predefinidos ou com valores RGB, HEX, HSL, RGBA ou HSLA.
- Podemos definir a cor de fundo para elementos HTML

  Exemplo utilizando o nome das cores:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Vamos começar a COLORIR as coisas

- As cores HTML são especificadas com nomes de cores predefinidos ou com valores RGB, HEX, HSL, RGBA ou HSLA.
- Podemos definir a cor do texto.

Exemplo utilizando o nome das cores:

```
<h1 style="color:Tomato;">Hello World</h1>

<p style="color:DodgerBlue;">Lorem ipsum...</p>

<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Vamos começar a COLORIR as coisas

- As cores HTML são especificadas com nomes de cores predefinidos ou com valores RGB, HEX, HSL, RGBA ou HSLA.
- Podemos definir a cor da borda.
- Para a borda precisamos definir alguns atributos a mais. Além da cor precisamos informar também o tamanho e modelo da borda.

Exemplo utilizando o nome das cores:

```
<h1 style="border:2px solid Tomato;">Hello World</h1>

<h1 style="border:2px solid DodgerBlue;">Hello World</h1>

<h1 style="border:2px solid Violet;">Hello World</h1>
```

Vamos começar a COLORIR as coisas

- Podemos representar as cores por nomes ou valores de cores RGB, HEX, HSL
- Nomes de cores (O HTML suporta 140 nomes de cores padrão)
    - Tomato
    - Orange
    - Gray
    - Blue
    - Red

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">Lorem ipsum...</p>

<h1 style="border:2px solid Violet;">Hello World</h1>
```

Vamos começar a COLORIR as coisas

- RGB e RGBA
- Um valor de cor RGB representa as fontes de luz VERMELHA, VERDE e AZUL.
- Um valor de cor RGBA é uma extensão de RGB com um canal Alfa (opacidade).

rgb (*vermelho, verde* , *azul*)

- Cada parâmetro (vermelho, verde e azul) define a intensidade da cor com um valor entre 0 e 255.
- Isso significa que existem 256 x 256 x 256 = 16777216 cores possíveis!

https://www.w3schools.com/html/html_colors_rgb.asp

Vamos começar a COLORIR as coisas

- RGBA
- Os valores de cores RGBA são uma extensão dos valores de cores RGB com um canal Alfa - que especifica a opacidade de uma cor.
- O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (nem um pouco transparente)
- Um valor de cor RGBA é especificado com:

rgba (*vermelho, verde* , *azul, alfa*)

https://www.w3schools.com/html/html_colors_rgb.asp

Vamos começar a COLORIR as coisas

- HEX (hexadecimal)
- Meio mais comum de representação de cores no HTML e CSS
- Uma cor hexadecimal é especificada com: #RRGGBB, onde os inteiros hexadecimais RR (vermelho), GG (verde) e BB (azul) especificam os componentes da cor.

#rrggbb

https://www.w3schools.com/html/html_colors_hex.asp

Vamos começar a COLORIR as coisas

- HSL e HSLA
- HSL significa matiz, saturação e luminosidade.
- Os valores de cor HSLA são uma extensão do HSL com um canal Alfa (opacidade).

hsl (*matriz* , *saturação* , *luminosidade*)

- Matiz é um grau na roda de cores de 0 a 360. 0 é vermelho, 120 é verde e 240 é azul.
- A saturação é um valor percentual, 0% significa um tom de cinza e 100% é a cor total.
- A luminosidade também é um valor percentual, 0% é preto e 100% é branco.

https://www.w3schools.com/html/html_colors_hex.asp

Vamos começar a COLORIR as coisas

- HSLA
- Os valores de cor HSLA são uma extensão dos valores de cor HSL com um canal Alfa - que especifica a opacidade de uma cor.
- O parâmetro alfa é um número entre 0,0 (totalmente transparente) e 1,0 (nem um pouco transparente)

hsla (*matriz* , *saturação* , *luminosidade, alfa*)

https://www.w3schools.com/html/html_colors_hex.asp

Vamos começar a COLORIR as coisas

- Obviamente não temos como memorizar esses códigos de todos esses códigos de cores, mas a boa notícia é que não precisamos nos preocupar com qual código vai gerar tal cor?
- Temos softwares onlines que nos auxiliam nas paletas de cores.
- Precisamos apenas escolher a cor e copiar o código no formato que quisermos.

Exemplo:

https://htmlcolorcodes.com/

Inserindo imagem

- Através da tag <img> conseguimos referenciar imagens para nosso documento.
- Alguns atributos são esperados nessa tag:
  - **src** => Caminho do arquivo de origem
  - **alt** => Texto alternativo **(importante para questões de acessibilidade e má conexão)**
  - **width** => Largura
  - **height** => Altura

```
<img src="foto.jpg" alt="FotoDePerfil" width="104" height="142">
```

Exercicios sugeridos

https://www.w3schools.com/html/exercise.asp

- HTML Attributes
- HTML Headings
- HTML Paragraphs
- HTML Styles
- HTML Formatting

CSS - Cascading Style Sheets

- CSS é utilizado para formatar layout de uma página web.
- Com CSS, você pode controlar a cor, a fonte, o tamanho do texto, o espaçamento entre os elementos, como os elementos são posicionados e dispostos, quais imagens ou cores de fundo devem ser usadas, diferentes exibições para diferentes dispositivos e tamanhos de tela e muito mais!

CSS - Cascading Style Sheets

- Podemos adicionar CSS ao nosso documento HTML de 3 maneiras.
- **Inline** - usando o atributo style dentro dos elementos HTML
- **Interno -** usando um elemento <style> inserido dentro do elemento HTML <head>
- **Externo -** usando um <link> elemento para vincular a uma arquivo CSS externo

Vamos ver como utilizamos essas 3 maneiras...

CSS - Cascading Style Sheets

- A maneira mais comum e indicada para adicionar CSS ao seu documento HTML é com arquivos CSS externos.
- Cada maneira possui suas particularidades…
- **Inline:**
  - Um CSS embutido é usado para aplicar um estilo único a um único elemento HTML.
  - Um CSS embutido usa o atributo style de um elemento HTML.

```
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

CSS - Cascading Style Sheets

- **Interno:**
  - Um CSS interno é usado para definir um estilo para uma única página HTML..
  - Um CSS interno é definido na <head>seção de uma página HTML, dentro de um <style>elemento.

```
<style>

    body {background-color: powderblue;}

    h1   {color: blue;}

    p    {color: red;}

</style>
```

CSS - Cascading Style Sheets

- **Externo:**
  - Uma folha de estilo externa é usada para definir o estilo de muitas páginas HTML.
  - Para usar uma folha de estilo externa, adicione um link a ela na <head>seção de cada página HTML.

```
<head>
    <link rel="stylesheet" href="styles.css">
</head>
```

CSS - Cascading Style Sheets

```
<!DOCTYPE html>
<html>
        <head>
          <link rel="stylesheet" href="styles.css">
        </head>
        <body>

                <h1>This is a heading</h1>
                <p>This is a paragraph.</p>

        </body>
</html>
```

CSS - Cascading Style Sheets

- A folha de estilo externa pode ser escrita em qualquer editor de texto. O arquivo não deve conter nenhum código HTML e deve ser salvo com uma extensão .css.
- Styles.css

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

CSS - Manipulando Fontes

- Da mesma forma que alteramos cores, também podemos alterar o texto. Definimos fontes com o uso da propriedade **font-family.**
- Por padrão, o navegadores mais conhecidos exibem texto em um tipo que conhecemos como “**serif”**. Elas são chamadas de **fontes serifadas** pelos pequenos ornamentos em suas terminações.

```
h1 {
    font-family: serif;
}
```

```
h2 {
    font-family: sans-serif;
}
```

Alinhamento e decoração de textos

- Vamos conhecer algumas maneiras de alterarmos as disposições dos textos.
- A propriedade mais simples, é a responsável pelo alinhamento do text, a **text-align.**

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

- É possível configurar também uma série de espaçamentos de texto com o CSS :

```
p {
    line-height: 3px; /* tamanho da altura de cada linha */
    letter-spacing: 3px; /* tamanho do espaço entre cada letra */
    word-spacing: 5px; /* tamanho do espaço entre cada palavra */
    text-indent: 30px; /*tamanho da margem da primeira linha do texto*/
}
```

Inserindo imagem de fundo

- A propriedade **background-image** permite indicar um arquivo de imagem para ser exibido ao fundo do elemento. Por exemplo :

```
h1 {
    background-image: url(imagem-de-fundo.jpg);
}
```

- Com essa declaração, o navegador vai requisitar o arquivo com nome *imagem-de-fundo.jpg* , que deve estar na mesma pasta do arquivo CSS onde consta esta declaração

Inserindo bordas

- As propriedades da borda CSS permitem que você especifique o estilo, a largura e a cor da borda de um elemento.
- A propriedade border-style especifica o tipo de borda a ser exibida. Os seguintes valores são permitidos

dotted - borda pontilhada

dashed - borda tracejada

solid - borda sólida

double - borda dupla

groove - borda 3D

ridge - 3D estriado

inset - borda de inserção 3D

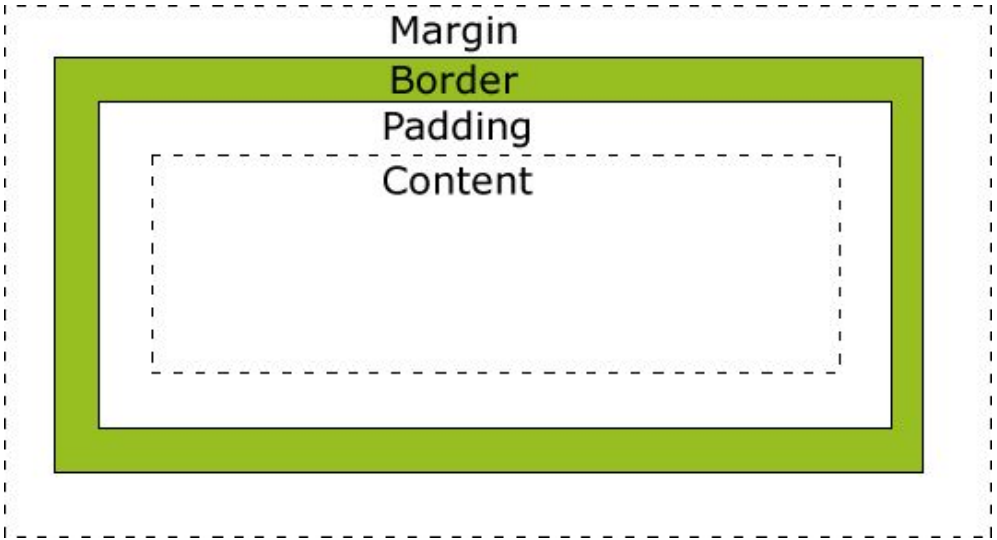
outset - borda de início 3D

hidden - borda oculta

CSS Margem - Margin

- A propriedade margin CSS define uma margem (espaço) fora da borda.

```
p {
  border: 2px solid powderblue;
  margin: 50px;
}
```



CSS - Margem - Margin

As propriedades de margem CSS são usadas para criar espaço ao redor dos elementos, fora de quaisquer bordas definidas. CSS tem propriedades para especificar a margem de cada lado de um elemento:

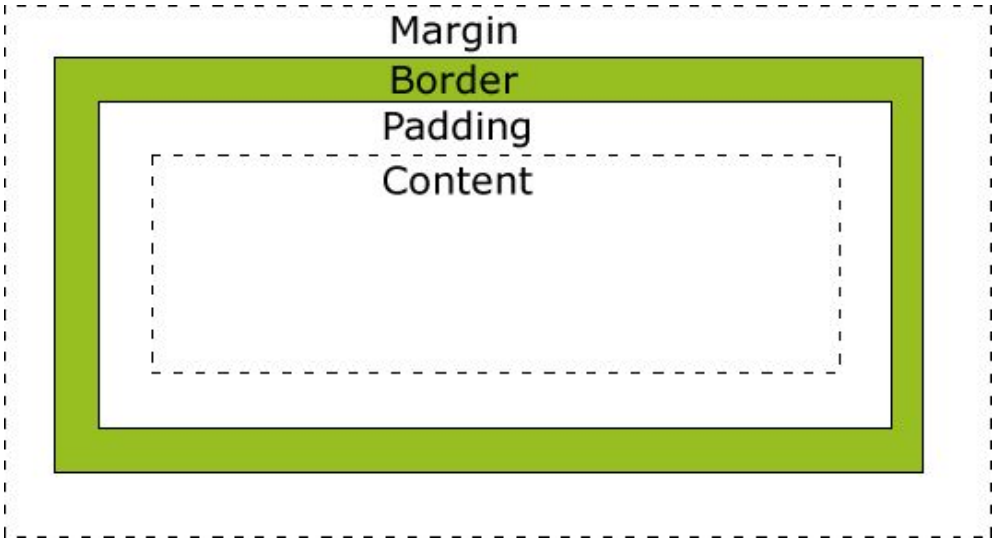
- margin-top
- margin-right
- margin-bottom
- margin-left

Todas as propriedades de margem podem ter os seguintes valores: auto, length, % e inherit

CSS Espaçamento - Padding

- A propriedade padding CSS define um preenchimento (espaço) entre o texto e a borda.

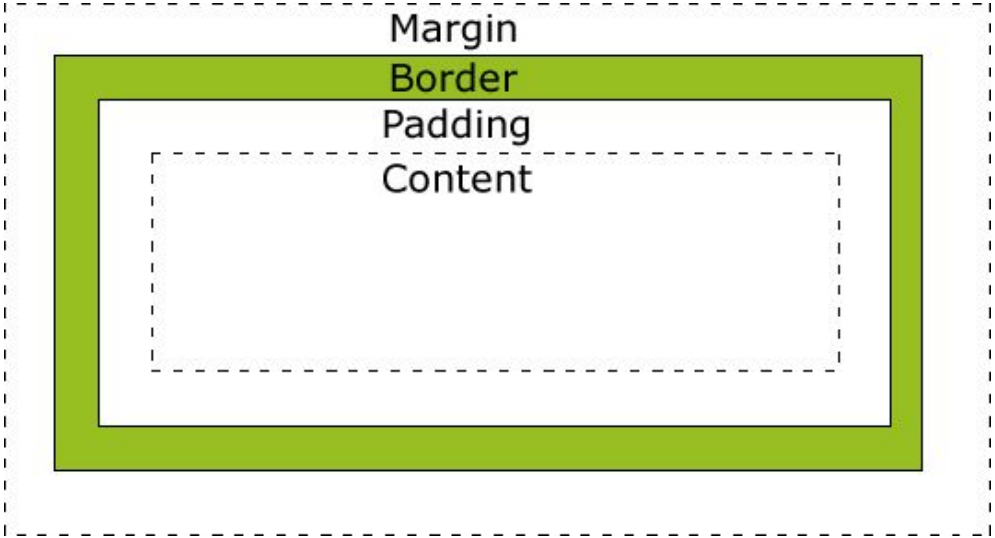
```
p {
  border: 2px solid powderblue;
  padding: 30px;
}
```



CSS - Espaçamento - Padding

As propriedades de preenchimento (padding) CSS são usadas para gerar espaço ao redor do conteúdo de um elemento, dentro de quaisquer bordas definidas. A manipulação do **padding** funciona como a propriedade margin. Exemplo :

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```



Links para CSS externo

- As folhas de estilo externas podem ser referenciadas com um URL completo ou com um caminho relativo à página da web atual.

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">

<link rel="stylesheet" href="/html/styles.css">

<link rel="stylesheet" href="styles.css">
```

Links HTML - HiperLinks

- Os links são encontrados em quase todas as páginas da web. Os links permitem que os usuários cliquem para ir de uma página para outra.
- Você pode clicar em um link e ser direcionado para outro documento
- Podemos incluir link em qualquer elemento HTML. Textos, imagens, div entre outros.
- A tag <a> define um hiperlink.
- O atributo href indica o destino do link. O conteúdo no escopo da tag será a parte visível e clicável exibida em tela.

Sintaxe:

```
<a href="https://www.w3schools.com/">Visit W3Schools.com!</a>
```

Links HTML - HiperLinks

- Por padrão os links aparecem da seguinte forma em todos os navegadores
  - Um link não visitado está sublinhado e em azul
  - Um link visitado está sublinhado e roxo
  - Um link ativo está sublinhado e vermelho
- Os links podem ser estilizados com CSS para possuírem outra aparência
- Por padrão ao acessar um hiperlink o novo documento abrirá na janela atual sobrepondo a página atual. Com o atributo target podemos modificar isso e especificar onde abrir o documento vinculado.

Links HTML - HiperLinks

- Valores que podemos atribuir ao atributo target

  - _self- Predefinição. Abre o documento na mesma janela / guia em que foi clicado
  - _blank - Abre o documento em uma nova janela ou guia
  - _parent - Abre o documento no quadro pai
  - _top - Abre o documento em todo o corpo da janela

```
<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

Links HTML - URLs absolutos vs. URLs relativos

- Os exemplos utilizados até o momento utilizamos **URL absolutos** (endereço da web completo) no href atributo.

```
<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

- utilizamos **URL relativo** quando precisamos referenciar um documento, arquivo ou imagem local (Um link para uma página dentro do mesmo site) não havendo a necessidade de informar “https://www”

```
<p><a href="html_images.asp">HTML Images</a></p>

<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

Links HTML - Imagens

- Podemos utilizar qualquer elemento HTML para ser um link clicável, vamos ver como ficaria utilizando uma imagem como link.
- Basta englobar no escopo da tag <a> o elemento que se deseja usar como link.

```
<a href="default.asp">
    <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">
</a>
```

Links HTML - Endereço de e-mail

- Podemos criar links para endereços de e-mail inserindo o parâmetro mailto: dentro do atributo href.
- Abre o programa de email do usuário para permitir o envio de um novo e-mail para o endereço informado no parâmetro mailto: do atributo href.

```
<a href="mailto:someone@example.com">Send email</a>
```

Links HTML - Botão como link

- Para usar um botão HTML como link, precisaremos da ajuda de algum código Javascript.
- Javascript permite programar ações em determinados eventos que ocorrem no seu site, nesse caso vamos disparar uma ação quando ocorrer o evento do usuário clicar no botão.

```
<button onclick="document.location='default.asp'">HTML Tutorial</button>
```

Vamos nos aprofundar em Javascript mais a frente!!!

Links HTML - Títulos de link

- O atributo title especifica informações extras sobre um elemento. As informações geralmente são mostradas como um texto de dica de ferramenta quando o mouse se move sobre o elemento.

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">Visit our HTML Tutorial</a>
```

Links HTML

- Use o elemento <a> para definir um link
- Use o atributo href para definir o endereço do link
- Use o atributo target para definir onde abrir o documento vinculado
- Use o elemento <img> (dentro de <a>) para usar uma imagem como um link
- Use o esquema mailto: dentro do atributo href para criar um link que abre o programa de e-mail do usuário

HTML - Inserindo Imagem

- Voltando a manipulação de imagens para algumas observações, vimos mais no início como importar imagens e utiliza-lás nos documentos HTML.
- Mas agora já aprendemos mais coisas e podemos deixar a nossa importação de imagem mais interessante.
- A tag <img> possui os atributos para definir a largura (width) e altura (height) porém é mais indicado realizar as dimensões pelo CSS utilizando o atributo style.
- Podemos também trabalhar com urls absolutas ou relativas no atributo src

```
<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">
```

HTML - Inserindo Imagens

- Utilizando a propriedade float do CSS podemos fazer uma imagem flutuar à esquerda(left) ou direita(right) permitindo alinharmos com textos e muito mais.

```
<p><img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">

The image will float to the right of the text.</p>


<p><img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px;">

The image will float to the left of the text.</p>
```

Exercicios Sugeridos

https://www.w3schools.com/html/exercise.asp

- HTML CSS
- HTML Links
- HTML Images

HTML - Tabelas

- HTML possui tags específicas para a criação de tabelas, isso permite organizar os dados em linhas e colunas.
- Em uma tabela no HTML podemos definir bordas, tamanho, cabeçalhos, espaçamento entre as células, o estilo da tabela, mesclar linhas ou colunas.
- Podemos incluir nas células não apenas texto ou dados, mas também botão, link, imagem ou qualquer elemento html.

Vamos ver como definir uma tabela em HTML

HTML - Tabelas

```
<table>
    <tr>
        <th>Company</th>
        <th>Company</th>
        <th>Company</th>
    </tr>
    <tr>
        <td>Company</td>
        <td>Company</td>
        <td>Company</td>
    </tr>
</table>
```

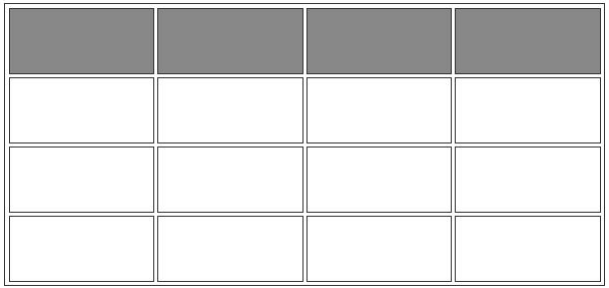
Legenda:

<tr></tr> - Representa a linha da tabela
<th></th> - Representa os cabeçalhos da tabela
<td></td> - Representa os dados da tabela

HTML - Tabelas - Bordas

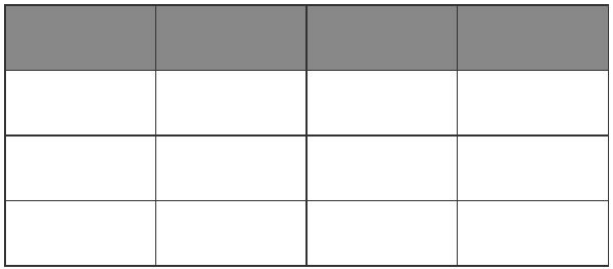
- Ao adicionar uma borda a uma tabela, você também adiciona bordas ao redor de cada célula da tabela:

```
table, th, td {
  border: 1px solid black;
}
```



- Para evitar bordas duplas podemos definir a propriedade border-collapse do CSS

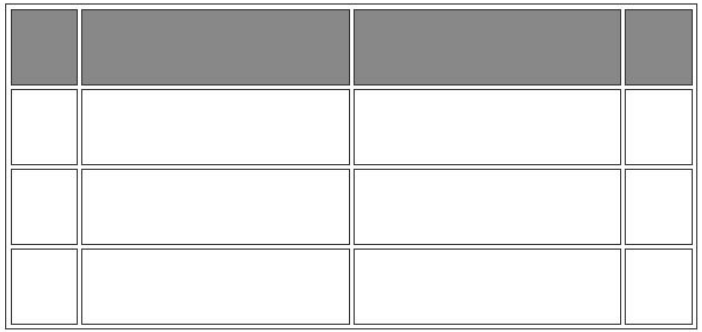
```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```



HTML - Tabelas - Tamanhos

- Podemos definir o tamanho da tabela inteira, mas podemos definir tamanhos diferentes para cada coluna ou linha.
- Utilizamos a propriedade width e height

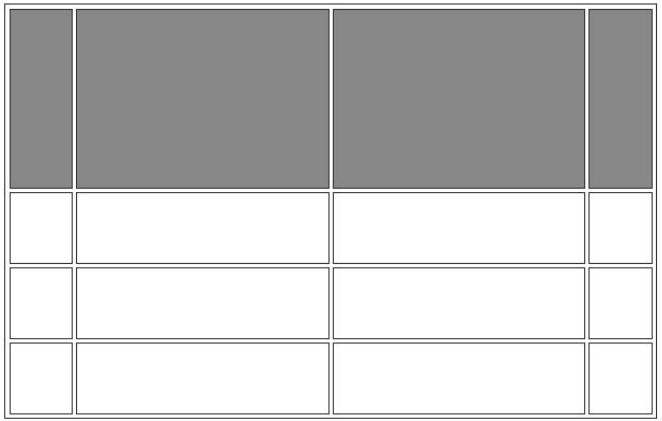
```
<tr>
    <th></th>
    <th style="width: 40%"></th>
    <th style="width: 40%"></th>
    <th></th>
</tr>
```



HTML - Tabelas - Tamanhos

- Podemos definir o tamanho da tabela inteira, mas podemos definir tamanhos diferentes para cada coluna ou linha.
- Utilizamos a propriedade width e height

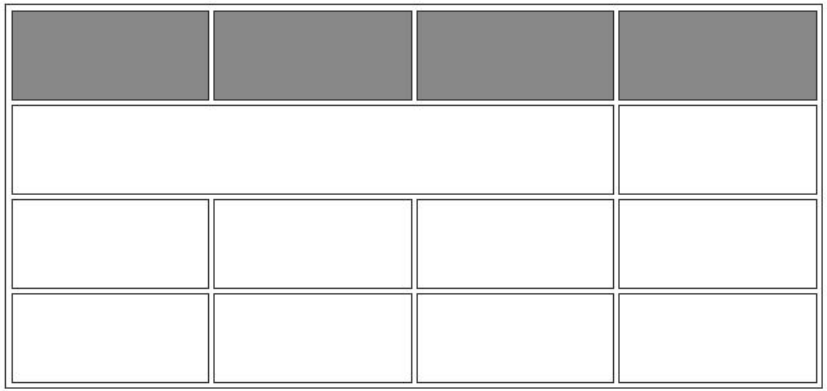
```
<tr style="height: 110px">
    <th></th>
    <th style="width: 40%"></th>
    <th style="width: 40%"></th>
    <th></th>
</tr>
```



HTML - Tabelas - Colspan & Rowspan

- Podemos mesclar células horizontalmente ou verticalmente.
- Utilizamos a propriedade colspan para mesclar colunas e a rowspan para mesclar linhas.

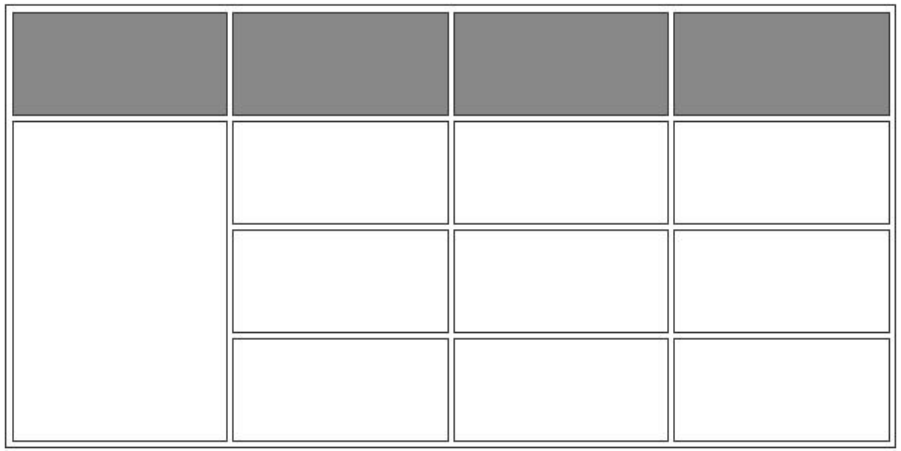
```
<tr>
    <td colspan="3"></td>
    <td></td>
</tr>
```



HTML - Tabelas - Colspan & Rowspan

- Podemos mesclar células horizontalmente ou verticalmente.
- Utilizamos a propriedade colspan para mesclar colunas e a rowspan para mesclar linhas.

```
<tr>
    <td rowspan="3"></td>
    <td></td>
    <td></td>
    <td></td>
 </tr>
```



HTML - Tabelas

https://www.w3schools.com/html/html_tables.asp

Exercício Sugerido

https://www.w3schools.com/html/exercise.asp

- HTML Tables

Desafio sugerido

Desenvolver uma tabela de pedidos da loja SerraTec. (Modelo no próximo slide)

Link das imagens utilizadas

Ícone computador

Logo SerraTec

serratec
Parque Tecnológico Região Serrana

PEDIDO DE COMPRA

		Data:	13/10/2021
Cliente:	Abner Jóia	**Contato:**	(24) 2222-7454
Email:	abnerjoia@gmail.com	**Telefone:**	(24) 2255-2314

Descrição do Produto	Valor	Qtde.	Valor Total
IPAD PRO	R$ 4.999,00	1	R$ 4.999,00
Notebook Dell Vostro	R$ 7.850,00	1	R$ 7.850,00
Cartão de memoria 1TB	R$ 59,90	3	R$ 179,70
Kit Teclado e Mouse	R$ 129,90	2	R$ 259,80
Bala 7Belo	R$ 0,10	10	R$ 1,00
		Valor total da compra:	R$ 13.289,50

HTML - Listas

- No HTML podemos criar listas para agrupar itens relacionados
- Podemos criar listas ordenadas, listas não ordenadas e listas de descrição

Uma Lista não ordenada:

- Item
- Item
- Item
- Item

Uma lista ordenada:

1. Primeiro item
2. Segundo item
3. Terceiro item
4. Quarto item

Uma lista descritiva:

Pao
- Pao francês
- Pão integral

Suco
- Suco de laranja

HTML - Lista não ordenada

- Uma lista não ordenada começa com a tag `<ul>`. Cada item da lista começa com a tag `<li>`.

```
<ul>

  <li>Café</li>

  <li>Chá</li>

  <li>Leite</li>

</ul>
```

- Café
- Chá
- Leite

HTML - Lista não ordenada

- Em uma lista não ordenada podemos definir o estilo do marcador dos itens da lista incluindo no atributo style a propriedade CSS list-style-type informando algum dos valores abaixo.
- disc (default)
- circle
- square
- none

```
<ul style="list-style-type:disc;">
  <li>Café</li>
  <li>Chá</li>
  <li>Leite</li>
</ul>
```

HTML - Lista Ordenada

- Uma lista ordenada começa com a tag `<ol>`. Cada item da lista começa com a tag `<li>`.

```
<ol>

  <li>Café</li>

  <li>Chá</li>

  <li>Leite</li>

</ol>
```

1. Café
2. Chá
3. Leite

HTML - Lista Ordenada

- Em uma lista ordenada podemos definir com no atributo type o tipo de marcador da lista.

```
<ol type="A">
  <li>Café</li>
  <li>Chá</li>
  <li>Leite</li>
</ol>
```

A. Café
B. Chá
C. Leite

HTML - Lista Ordenada

Tipo	Descrição
type=”1”	Ordenação com números (default)
type=”A”	Letras maiúsculas
type=”a”	Letras minúsculas
type=”I”	Algarismos romanos maiúsculos
type=”i”	Algarismos romanos minúsculos

HTML - Listas de Descrição

- HTML também oferece suporte a listas de descrição. Uma lista de descrição é uma lista de termos, com uma descrição de cada termo.
- A tag `<dl>` define a lista de descrição, a tag `<dt>` define o termo (nome) e a tag `<dd>` descreve cada termo:

```
<dl>
  <dt>Pao</dt>
  <dd>- Pão Francês</dd>
  <dt>Suco</dt>
  <dd>- Suco de Laranja</dd>
</dl>
```

- Pão
  - Pão Francês
- Suco
  - Suco de Laranja

Exercícios Sugeridos

https://www.w3schools.com/html/exercise.asp

- HTML Lists

Elementos estruturais: Tags e

- A tag define uma divisão ou seção em um documento HTML. A tag é usada como um contêiner para elementos HTML - que são estilizados com CSS ou manipulados com JavaScript. Qualquer tipo de conteúdo pode ser colocado dentro da tag !
- O elemento e não tem atributos obrigatórios, mas style, class e id são comuns.
- A tag é um contêiner de linha usado para marcar uma parte de um texto ou uma parte de um documento. A tag é muito parecida com o elemento , mas é um elemento de nível de bloco e é um elemento de linha.

Vamos testar alguns exemplos!

Propriedade CSS - Flexbox

- A propriedade flexbox é muito utilizada para alinharmos containers div no documento HTML.
- Podemos definir a disposição dos container em padrão de linhas (row) ou colunas (column).
- Precisamos primeiro criar um container flexível definindo a propriedade CSS display: flex isso permitirá que os elementos filho deste container sejam flexíveis.

Propriedade CSS - Flexbox

- Definir propriedade flex-direction
  - row
  - row-reverse
  - column
  - column-reverse

Vamos ver alguns exemplos!

Atributo class e Seletores CSS

- O atributo class é freqüentemente usado para apontar para um nome de classe em uma folha de estilo. Ele também pode ser usado por um JavaScript para acessar e manipular elementos com o nome de classe específico.

```
<div class="city">
  <h2>London</h2>
  <p>London is the capital of England.</p>
</div>
```

```
.city {
  background-color: tomato;
  color: white;
  border: 2px solid black;
  margin: 20px;
  padding: 20px;
}
```

Atributo class e Seletores CSS

- Podemos definir classes para qualquer elemento HTML
- A mesma classe pode ser utilizada por diferentes elementos

Sintaxe do atributo class no documento HTML

```
<tag class=”nomeDaClasse”></tag>
```

Sintaxe seletor CSS

```
.nomeDaClasse{
    propriedades CSS aqui
}
```

Atributo id e seletores CSS

- O atributo id especifica um id único para um elemento HTML. O valor do atributo id deve ser exclusivo no documento HTML.
- O atributo id é usado para apontar para uma declaração de estilo específica em uma folha de estilo. Ele também é usado pelo JavaScript para acessar e manipular o elemento com o id específico.

```html
<h1 id="titulo">Titulo</h1>
```

```css
#titulo {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
```

Seletor Hierárquico

Podemos ainda utilizar um seletor hierárquico que permite aplicar estilos aos elementos filhos de um elemento pai:

```
#rodape img {
    margin-right: 30px;
    vertical-align: middle;
    width: 94px;
}
```

Neste exemplo, o elemento pai **rodape** é selecionado pelo seu id. O estilo será aplicado apenas nos elementos **img** filhos do elemento com id=rodape.

Seletores CSS

- Podemos aplicar o mesmo estilo para diversos elementos HTML

```
<h1 class=”titulo”>Titulo</h1>

<p class=”paragrafo”>Meu parágrafo</p>
```

```
.titulo, .paragrafo{
        text-align: center;
}
```

Seletores CSS

- Podemos referenciar um elemento HTML para um ou mais estilos CSS

```
<h1 class=”formataTitulo tituloGrande”>Titulo</h1>
```

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

.tituloGrande{
        font-size: 100px;
}
```

Exercício Sugeridos

https://www.w3schools.com/html/exercise.asp

- HTML Classes
- HTML Id

HTML Semântico : O que são Elementos Semânticos?

Um elemento semântico descreve claramente seu significado para o navegador e o desenvolvedor.

Exemplos de elementos não semânticos: `<div>` e `<span>` - Não diz nada sobre seu conteúdo.

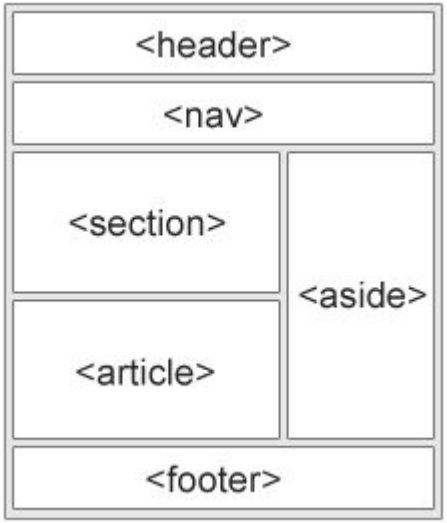
Exemplos de elementos semânticos: `<form>`, `<table>` e `<article>` - define claramente seu conteúdo.

Elementos Semânticos no HTML

Muitos sites contêm código HTML como: para indicar navegação, cabeçalho e rodapé.

Em HTML, existem alguns elementos semânticos que podem ser usados para definir diferentes partes de uma página da web:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



Cabeçalho (header)

O elemento <header> representa um contêiner para conteúdo introdutório ou um conjunto de links de navegação.

Um elemento <header> normalmente contém:

- um ou mais elementos de título (<h1> - <h6>)
- logotipo ou ícone
- informação de autoria

Observação : Você pode ter vários elementos <header> em um documento HTML. No entanto,<header> não pode ser colocado em um <footer>, <address> ou outro elemento <header>.

Elemento HTML(nav)

O elemento `<nav>` define um conjunto de links de navegação

- Nem todos os links de um documento devem estar dentro de um elemento <nav>. Este destina-se apenas ao bloco principal de links de navegação.
- Navegadores e leitores de tela para usuários com deficiência, podem usar esse elemento para determinar se a renderização inicial desse conteúdo deve ser omitida.

Elemento HTML(section)

- O elemento <section> define uma seção em um documento.
- De acordo com a documentação HTML do W3C: "Uma seção é um agrupamento temático de conteúdo, normalmente com um título."

Exemplos de onde um elemento <section> pode ser usado:

- Capítulos
- Introdução
- Novos itens
- Informações de Contato

Elemento HTML(article)

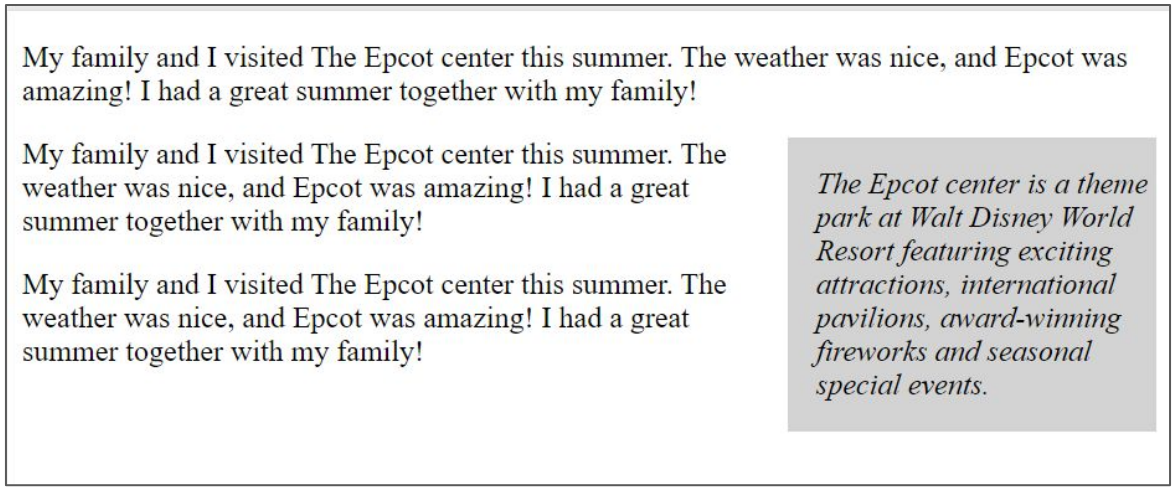
- O <article>elemento especifica conteúdo independente e autocontido.
- Um artigo deve fazer sentido por si só e deve ser possível distribuí-lo independentemente do resto do site.

Exemplos de onde o <article>elemento pode ser usado:

- Postagens do fórum
- Postagens no blog
- Comentários do usuário
- Cartões de produto
- Artigos de jornal

Elemento HTML(aside)

- O elemento <aside> define algum conteúdo além do conteúdo em que é colocado (como uma barra lateral).
- O conteúdo <aside> deve estar indiretamente relacionado ao conteúdo circundante.



Rodapé (footer)

O elemento <footer> define um rodapé para um documento ou seção.

O elemento <footer> define um rodapé para um documento ou seção.

- Informação de autoria
- Informação de copyright
- Informação de contato
- Mapa do site
- Retorno para os links superiores
- Documentos relacionados

Podemos ter vários elementos <footer> em um documento

Formulários HTML

- Um formulário serve para a coleta de informações que são inseridas pelo usuário, os dados geralmente são enviados a um servidor para o processamento.
- Definimos um formulário com o elemento <form>. O elemento form é um container para diferentes tipos de elementos de entrada, como: campos de texto, caixa de seleção, botões de opção, botões de envio, etc...

```
<form>

    elementos do formulário

</form>
```

Formulários HTML - Atributos de Formulário

- Atributo action define a ação a ser executada quando o formulário é enviado. normalmente, os dados do formulário são enviados para um arquivo no servidor quando o usuário clica no botão enviar.

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

Formulários HTML - Atributos de Formulário

- Atributo target atributo especifica onde exibir a resposta recebida após o envio do formulário. Os mais utilizados são _blank e _self (default)

```
<form action="/action_page.php" target=”_blank” >
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

Formulários HTML - Método (Method)

- Atributo method especifica o método HTTP a ser usado ao enviar os dados do formulário.
- Os dados do formulário podem ser enviados como variáveis de URL (com method="get") ou como transação HTTP post (com method="post").

```
<form action="/action_page.php" method=”get” >

<form action="/action_page.php" method=”post” >
```

Formulários HTML - Método (Method)

Notas sobre GET:

- Acrescenta os dados do formulário ao URL, em pares nome / valor
- NUNCA use GET para enviar dados confidenciais! (os dados do formulário enviado são visíveis no URL!)
- O comprimento de um URL é limitado (2.048 caracteres)
- Útil para envios de formulários onde um usuário deseja marcar o resultado
- GET é bom para dados não seguros, como strings de consulta no Google

Formulários HTML - Método (Method)

Notas no POST:

- Acrescenta os dados do formulário dentro do corpo da solicitação HTTP (os dados do formulário enviado não são mostrados no URL)
- O POST não tem limitações de tamanho e pode ser usado para enviar grandes quantidades de dados.
- Os envios de formulários com POST não podem ser marcados

Formulários HTML - Tipos de entrada input

- O <input> é o elemento mais utilizado em formulários HTML. Este elemento permite a entrada de dados pelo usuário e possui diferentes tipos que informamos no atributo type.

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

Formulários HTML - Tipos de entrada input

- <input> pode receber diversos tipos de atributos como name placeholder required entre muitos outros.

Vamos explorar um pouco essas possibilidades na W3school

https://www.w3schools.com/html/html_form_attributes.asp

Validação HTML5 : Required e Pattern

O atributo **required** é um atributo booleano. Quando presente, específica que o elemento deve ser preenchido antes de enviar o formulário.

O atributo **required** pode ser usado nos elementos <*input*> , <*select*> e <*textarea*>.

Exemplo :

```
<form action="/action_page.php">
    Username: <input type="text" name="username" required>
    <input type="submit">
</form>
```

Validação HTML5 : Required e Pattern

Já o atributo pattern especifica uma expressão regular com a qual o valor do elemento <input> é verificado no envio do formulário.

O atributo padrão funciona com os seguintes tipos de entrada: text, date, search, url, tel, email, and password.

Exemplo (pattern que determina que o password conterá 8 ou mais caracteres:

```
<form action="/action_page.php">
  <label for="pwd">Password:</label>
  <input type="password" id="pwd" name="pwd"
  pattern=".{8,}" title="Eight or more characters">
  <input type="submit">
</form>
```

Formulários : Elemento <label>

A tag <label> define um rótulo para muitos elementos do formulário.

O elemento <label> é útil para usuários de leitores de tela, porque o leitor de tela lerá o rótulo em voz alta quando o usuário focar no elemento de entrada.

O elemento <label> também ajuda os usuários que têm dificuldade em clicar em regiões muito pequenas (como botões de rádio ou caixas de seleção) - porque quando o usuário clica no texto dentro do elemento <label>, ele alterna o botão / caixa de seleção.

O atributo for da tag <label> deve ser igual ao atributo id do elemento <input> para vinculá-los.

Formulários : O botão submit

O <input type="submit"> define um botão para enviar os dados do formulário a um manipulador de formulários.

O manipulador de formulários é normalmente um arquivo no servidor com um script para processar dados de entrada.

O manipulador de formulários é especificado no atributo de ação (action) do formulário.

Exercícios sugeridos

https://www.w3schools.com/html/exercise.asp?filename=exercise_html_form_elements3

- HTML Forms
- HTML Forms Attributes
- HTML Form Elements
- HTML Input Types
- HTML Input Attributes

Exercícios sugeridos

Desafio

Desenvolver um formulário de cadastro utilizando HTML e CSS, aplicando as validações(required) e padronizações(pattern) do HTML entre outros atributos. Os tipos dos inputs devem ser condizentes com os dados esperados de entrada. Modelo sugestivo no próximo slide.

Formulário de Cadastro

Nome

Sobrenome

Data Nascimento

dd/mm/aaaa

E-mail

Telefone

24 98856-2565

CPF

Tecnologia de preferência

HTML CSS Javascript

Área de atuação

Front-end

Nome de Usuário

Senha

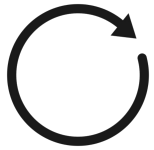
Enviar Cancelar

Exercícios sugeridos

https://www.w3schools.com/css/exercise.asp

- CSS Selectors
- CSS How To…
- CSS Background
- CSS Border
- CSS Margin
- CSS Padding
- CSS Height/Width
- CSS Box Model
- CSS Outline
- CSS Text
- CSS Font
- CSS Tables
- CSS Align

CSS

CSS Resets 

Quando não especificamos nenhum estilo para nossos elementos HTML, o navegador utiliza uma série de estilos padrão, diferente para cada navegador.

Para evitar diferenças de layout entre navegadores, alguns desenvolvedores e empresas criaram alguns estilos que chamamos de CSS Reset.

A intenção é setar um valor básico para todas as características do CSS, sobrescrevendo totalmente os estilos padrão do navegador.

Reset CSS: O que é, Exemplos, Como criar e Utilizar - Alura

CSS

- Já vimos alguns tipos de seletores que permitem selecionar o elemento HTML e estilizá lo.
- Seletores simples (selecione os elementos com base no nome, id, classe)
- Seletores combinadores (selecione os elementos com base em uma relação específica entre eles)
- Seletores de pseudoclasse (selecione os elementos com base em um determinado estado)
- Seletores de pseudoelementos (seleciona e estiliza uma parte de um elemento)
- Seletores de atributos (seleciona os elementos com base em um atributo ou valor de atributo)

CSS

- Antes de vermos seletores mais avançados um seletor que não vimos até o momento é o seletor universal.
- O Seletor universal seleciona todos os elementos HTML na página.
- No Exemplo abaixo todos os elementos HTML da página serão afetados.

```
* {
  text-align: center;
  color: blue;
}
```

Seletores Combinadores

- Um seletor CSS pode conter mais de um seletor simples. Entre os seletores simples, podemos incluir um combinador.

Existem quatro combinadores diferentes em CSS:

- seletor descendente (espaço)
- seletor filho (>)
- seletor irmão adjacente (+)
- seletor irmão geral (~)

Seletores Combinadores - Descendente (espaço)

- O seletor descendente corresponde a todos os elementos descendentes de um elemento especificado.

```
div p {
  background-color: yellow;
}
```

Seletores Combinadores - Filho (>)

- O seletor filho seleciona todos os elementos que são filhos de um elemento especificado.

```
div > p {
  background-color: yellow;
}
```

Seletores Combinadores - Irmão adjacente(+)

- O seletor irmão adjacente é usado para selecionar um elemento que está diretamente após outro elemento específico.
- Os elementos irmãos devem ter o mesmo elemento pai e "adjacente" significa "imediatamente após".

```
div + p {
  background-color: yellow;
}
```

Seletores Combinadores - Seletor geral de irmãos(~)

- O seletor irmão geral seleciona todos os elementos que são os próximos irmãos de um elemento especificado.

```
div ~ p {
  background-color: yellow;
}
```

Seletores de Pseudo classes

O que são Pseudo-classes?

Uma pseudoclasse é usada para definir um estado especial de um elemento.

Por exemplo, pode ser usado para:

- Definir o estilo de um elemento quando o usuário passa o mouse sobre ele
- Estilizar links visitados e não visitados de maneira diferente
- Defina o estilo de um elemento quando ele obtiver o foco

Seletores de Pseudo classes

Sintaxe

```
selector:pseudo-class {
  property: value;
}
```

Seletores de Pseudo classes

- Vamos apresentar exemplos dos seletores mais utilizados, porém existe uma grande quantidade de pseudo classes além das apresentadas aqui.
- Para estilização de links

  a:link {}

  a:visited {}

  a:hover {}

  a:active {}

Vamos aplicar os exemplos pelo W3School

Seletores de Pseudo classes

- As pseudo classes podem ser combinadas com classes HTML:

```
a.highlight:hover {
  color: #ff0000;
}
```

Vamos aplicar os exemplos pelo W3School

Seletores de Pseudo classes

- Um exemplo de uso da :hover pseudo-classe em um elemento :

```
div:hover {
  background-color: blue;
}
```

Vamos aplicar os exemplos pelo W3School

Seletores de Pseudo classes

- Pseudo classe do primeiro filho
- A pseudoclasse :first-child corresponde a um elemento especificado que é o primeiro filho de outro elemento.

```
p:first-child {
  color: blue;
}
```

Vamos aplicar os exemplos pelo W3School

Seletores de Pseudo classes

Índice de pseudo-classes padrão (Documentação MDN)

- :active
- :checked
- :default (en-US)
- :dir() (en-US)
- :disabled
- :empty
- :enabled
- :first (en-US)
- :first-child
- :first-of-type
- :fullscreen
- :valid
- :visited
- :focus
- :hover
- :indeterminate (en-US)
- :in-range (en-US)
- :invalid
- :lang() (en-US)
- :last-child
- :last-of-type
- :left (en-US)
- :link
- :not()
- :right (en-US)
- :root
- :nth-child()
- :nth-last-child()
- :nth-last-of-type() (en-US)
- :nth-of-type()
- :only-child
- :only-of-type
- :optional
- :out-of-range
- :read-only (en-US)
- :read-write
- :required
- :scope (en-US)
- :target

Seletores de Pseudo-elementos

O que são pseudo-elementos?

Um pseudoelemento CSS é usado para estilizar partes específicas de um elemento.

Por exemplo, pode ser usado para:

- Defina o estilo da primeira letra ou linha de um elemento
- Insira conteúdo antes ou depois do conteúdo de um elemento

Seletores de Pseudo-elementos

Sintaxe

```
selector::pseudo-element {
  property: value;
}
```

Seletores de Pseudo-elementos

Pseudo-elemento de primeira linha

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

- Observação: O pseudoelemento ::first-line só pode ser aplicado a elementos de nível de bloco.

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

O pseudo-elemento ::first-letter é usado para adicionar um estilo especial à primeira letra de um texto.

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

- Observação: o pseudoelemento ::first-letter só pode ser aplicado a elementos de nível de bloco.

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

O pseudoelemento ::before pode ser usado para inserir algum conteúdo antes do conteúdo de um elemento.

```
h1::before {
  content: url(smiley.gif);
}
```

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

O pseudoelemento ::after pode ser usado para inserir algum conteúdo após o conteúdo de um elemento.

```
h1::after {
  content: url(smiley.gif);
}
```

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

O pseudoelemento ::marker seleciona os marcadores dos itens da lista.

```
::marker {
  color: red;
  font-size: 23px;
}
```

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

O pseudoelemento ::selection corresponde à parte de um elemento que é selecionado por um usuário.

As seguintes propriedades CSS pode ser aplicada a ::selection: color, background, cursor, e outline.

```
::selection {
  color: red;
  background: yellow;
}
```

Vamos conferir pelo W3School!

Seletores de Pseudo-elementos

Índice de pseudo-elementos comuns (Documentação MDN)

- `::after`
- `::before`
- `::cue` (en-US)
- `::first-letter`
- `::first-line`
- `::selection`
- `::slotted` (en-US)
- `::backdrop`
- `::placeholder` (en-US)
- `::marker` (en-US)
- `::spelling-error` (en-US)
- `::grammar-error` (en-US)

Seletores de atributo CSS

O seletor [attribute] é usado para selecionar elementos com um atributo especificado.

```
a[target] {
  background-color: yellow;
}
```

Vamos conferir pelo W3School!

Seletores de atributo CSS

O seletor [attribute="value"] é usado para selecionar elementos com um atributo e valor especificados.

```
a[target="_blank"] {
  background-color: yellow;
}
```

Vamos conferir pelo W3School!

Seletores de atributo CSS

O seletor [attribute~="value"] é usado para selecionar elementos com um valor de atributo contendo uma palavra especificada.

```
[title~="flower"] {
  border: 5px solid yellow;
}
```

Vamos conferir pelo W3School!

Seletores de atributo CSS

O seletor [attribute|="value"] é usado para selecionar elementos com o atributo especificado começando com o valor especificado.

```
[class|="top"] {
  background: yellow;
}
```

Observação: o valor deve ser uma palavra inteira, sozinha, como class = "top", ou seguida por um hífen (-), como class = "top-text"!

Vamos conferir pelo W3School!

Seletores de atributo CSS

O seletor [attribute$="value"] é usado para selecionar elementos cujo valor de atributo termina com um valor especificado.

```
[class$="test"] {
  background: yellow;
}
```

Observação: o valor não precisa ser uma palavra inteira!

Vamos conferir pelo W3School!

Seletores de atributo CSS

O seletor [attribute*="value"] é usado para selecionar elementos cujo valor de atributo contém um valor especificado.

```
[class*="te"] {
  background: yellow;
}
```

Observação: o valor não precisa ser uma palavra inteira!

Vamos conferir pelo W3School!

Seletores de atributo CSS

Os seletores de atributo podem ser úteis para estilizar formulários sem classe ou ID:

```
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}
```

```
input[type="button"] {
  width: 120px;
  margin-left: 35px;
  display: block;
}
```

Exercícios Sugeridos

https://www.w3schools.com/css/exercise.asp

- CSS Combinators
- CSS Pseudo-classes
- CSS Pseudo-elements
- CSS Attribute Selector

Layout CSS

- A propriedade display é a propriedade CSS mais importante para controlar o layout.
- A propriedade display especifica se / como um elemento é exibido.
- Cada elemento HTML tem um valor de exibição padrão dependendo de que tipo de elemento ele é. O valor de exibição padrão para a maioria dos elementos é block ou inline.

Block vs Inline

Elementos HTML podem se comportar basicamente de duas maneiras com relação à sua interferência no documento como um todo: em bloco (**block**) ou em linha (**inline**)

Um elemento de **nível de bloco** sempre começa em uma nova linha e ocupa toda a largura disponível (se estende para a esquerda e para a direita o máximo que pode). Abaixo, a lista com os elementos em nível de bloco no HTML :

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>
	<dl>	<dt>	<fieldset>	<figcaption>	<figure>
<footer>	<form>	<h1>-<h6>	<header>	<hr>	
<main>	<nav>	<noscript>		<p>	<pre>
<section>	<table>	<tfoot>		<video>	

Elementos Inline

Um elemento em linha não começa em uma nova linha e só ocupa a largura necessária.

Abaixo os elementos inline no HTML :

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>
 	<button>	<cite>	<code>	<dfn>	
<i>	<img>	<input>	<kbd>	<label>	<map>
<object>	<output>	<q>	<samp>	<script>	<select>
<small>			<sub>	<sup>	<textarea>
<time>	<tt>	<var>			

Display Inline-Block

Comparado com `display: inline`, a principal diferença é que `display: inline-block` permite definir uma largura e altura no elemento.

Além disso, com `display: inline-block`, as margens / preenchimentos superior e inferior são respeitados, mas com `display: inline` não são.

Comparado com `display: block`, a principal diferença é que `display: inline-block` não adiciona uma quebra de linha após o elemento, então o elemento pode ficar próximo a outros elementos.

Vamos brincar com alguns exemplos no w3schools.

Layout CSS

Elementos de nível de bloco

- Um elemento de nível de block sempre começa em uma nova linha e ocupa toda a largura disponível (se estende para a esquerda e para a direita, tanto quanto pode).

  -
  - <h1> - <h6>
  - <p>
  - <form>
  - <cabeçalho>
  - <footer>
  - <seção>

Layout CSS

Elementos Inline

- Um elemento inline não começa em uma nova linha e só ocupa a largura necessária.

-
- <a>
- <img>

Layout CSS

Display none

- display: none; é comumente usado com JavaScript para ocultar e mostrar elementos sem excluí-los e recriá-los.
- Ocultar um elemento pode ser feito definindo a propriedade display como none. O elemento ficará oculto e a página será exibida como se o elemento não existisse:

```
h1.hidden {
  display: none;
}
```

Layout CSS

Display none

- visibility:hidden; também esconde um elemento. No entanto, o elemento ainda ocupará o mesmo espaço de antes. O elemento ficará oculto, mas ainda afetará o layout:

```
h1.hidden {
  visibility: hidden;
}
```

Exercícios Sugeridos

https://www.w3schools.com/css/exercise.asp

- CSS Display/Visibility

Usando largura, largura máxima e margem: auto;

max-width e margin: auto;

- Conforme mencionado no capítulo anterior; um elemento de nível de bloco sempre ocupa toda a largura disponível (se estende para a esquerda e para a direita, tanto quanto pode).
- Definir o width de um elemento de nível de bloco impedirá que ele se estenda até as bordas de seu contêiner. Em seguida, você pode definir as margens como automáticas, para centralizar horizontalmente o elemento em seu contêiner. O elemento ocupará a largura especificada e o espaço restante será dividido igualmente entre as duas margens:

Layout CSS - Usando largura, largura máxima e margem: auto;

- Nota: O problema acima ocorre quando a janela do navegador é menor que a largura do elemento. O navegador então adiciona uma barra de rolagem horizontal à página.
- Em vez disso, o uso de max-width, nessa situação, melhorará o manuseio do navegador de pequenas janelas. Isso é importante ao tornar um site utilizável em dispositivos pequenos:

https://www.w3schools.com/css/css_max-width.asp

Posicionamento estático, relativo e absoluto

Existe um conjunto de propriedades que podemos utilizar para posicionar um elemento na página, como o *top*, *left*, *bottom* e *right*. Porém essas propriedades, por padrão, não são obedecidas por nenhum elemento, pois elas dependem de uma outra propriedade, a position

A propriedade position especifica o tipo de método de posicionamento usado para um elemento (static, relative, fixed, absolute ou sticky). Exemplo :

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

Layout CSS - A propriedade position

- A propriedade position especifica o tipo de método de posicionamento usado para um elemento (static, relative, fixed, absolute or sticky)
- A propriedade position especifica o tipo de método de posicionamento usado para um elemento.

Existem cinco valores de posição diferentes:

- static
- relative
- fixed
- absolute
- sticky

Layout CSS - A propriedade position

- Os elementos são posicionados usando as propriedades top, bottom, left e right. No entanto, essas propriedades não funcionarão a menos que a position propriedade seja definida primeiro. Eles também funcionam de forma diferente dependendo do valor da posição.

Layout CSS - A propriedade position

static

- Os elementos HTML são posicionados estáticos por padrão.
- Os elementos posicionados estáticos não são afetados pelas propriedades top, bottom, left e right.
- Um elemento com position: static;não é posicionado de nenhuma maneira especial; está sempre posicionado de acordo com o fluxo normal da página:

Layout CSS - A propriedade position

relative

- Um elemento com position: relative;está posicionado em relação à sua posição normal.
- Definir as propriedades top, right, bottom e left de um elemento relativamente posicionado fará com que ele seja ajustado para longe de sua posição normal. Outro conteúdo não será ajustado para caber em qualquer lacuna deixada pelo elemento.

Layout CSS - A propriedade position

fixed

- Um elemento com position: fixed;é posicionado em relação à janela de visualização, o que significa que ele sempre permanece no mesmo lugar, mesmo se a página for rolada. As propriedades top, right, bottom e left são usadas para posicionar o elemento.
- Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado.

Layout CSS - A propriedade position

absolute

- Um elemento com position: absolute;é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à janela de visualização, como fixo).
- Contudo; se um elemento posicionado de forma absoluta não tiver ancestrais posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página.

Nota: Elementos posicionados absolutos são removidos do fluxo normal e podem sobrepor elementos.

Exercícios Sugeridos

https://www.w3schools.com/css/exercise.asp

- CSS Positioning

Layout CSS - A propriedade z-index

- A propriedade z-index especifica a ordem da pilha de um elemento.
- A propriedade z-index especifica a ordem da pilha de um elemento (qual elemento deve ser colocado na frente ou atrás dos outros).
- Um elemento pode ter uma ordem de empilhamento positiva ou negativa:

Nota: z-index só funciona em elementos posicionados (position: absolute, position: relative, position: fixed ou position: sticky) e itens flexíveis (elementos que são filhos diretos de display: flex).

Exercício

https://www.w3schools.com/css/exercise.asp

- CSS Z-index

CSS - Unidades

- CSS tem várias unidades diferentes para expressar um tamanho/comprimento.
- Muitas propriedades CSS podem possuir valores especificando um "tamanho", tais como width, margin, padding, font-size, etc.
- Informamos um número seguido por uma unidade de tamanho, tais como 10px, 2em, etc.
- Existem dois tipos de unidades: absoluto e relativo .

CSS - Unidades - Absolutas

- As unidades de comprimento absoluto são fixas e um comprimento expresso em qualquer uma delas aparecerá exatamente com esse tamanho.
- As unidades de comprimento absoluto não são recomendadas para uso na tela, porque os tamanhos da tela variam muito. No entanto, eles podem ser usados se o meio de saída for conhecido, como para layout de impressão.

- cm
- mm
- in
- px*
- pt
- pc

* Pixels (px) são relativos ao dispositivo de visualização. Para dispositivos de baixo dpi, 1px é um pixel de dispositivo (ponto) da tela. Para impressoras e telas de alta resolução, 1px implica vários pixels de dispositivo.

CSS - Unidades - Relativas

- As unidades de comprimento relativo especificam um comprimento relativo a outra propriedade de comprimento. As unidades de comprimento relativo escalam melhor entre diferentes meios de renderização.

- em
- ex
- ch
- rem
- vw
- vh
- vmin
- vmax
- %

Dica: as unidades em e rem são práticas na criação de um layout perfeitamente escalável!
* Viewport = o tamanho da janela do navegador. Se a janela de visualização tiver 50 cm de largura, 1vw = 0,5 cm.

A propriedade border-radius

A propriedade border-radius define o raio dos cantos do elemento, permitindo adicionar bordas arredondadas aos elementos.

Esta propriedade pode ter de um a quatro valores, seguindo as regras :

- Quatro valores, ex: 15px 50px 30px 5px. Os valores são aplicados em sentido horário, começando pelo canto superior esquerdo, gerando uma imagem assim :

A propriedade border-radius

- Três valores (ex: 15px 50px 30px) : Nesse caso o segundo valor é aplicado aos cantos superior direito e inferior esquerdo

- Dois valores (ex: 15px 50px) : Primeiro valor aplicado aos cantos superior esquerdo e inferior direito, e segundo valor os lados opostos.

- Um valor (ex: 15px) : Valor aplicado em todos os cantos.

As propriedades text-shadow e box-shadow

A propriedades `text-shadow` e `box-shadow` adiciona sombra ao texto e a um elemento, respectivamente.

Esta propriedade aceita uma lista separada por vírgulas de sombras a serem aplicadas ao texto.

Exemplo :

```
h1 {
  text-shadow: 2px 2px #ff0000;
}
```

```
#exemplo {
  box-shadow: 2px 2px #ff0000;
}
```

Opacidade

A propriedade opacity define o nível de opacidade de um elemento.

O nível de opacidade descreve o nível de transparência, onde 1 não é transparente, 0,5 é 50% transparente e 0 é completamente transparente.

opacidade 0.2

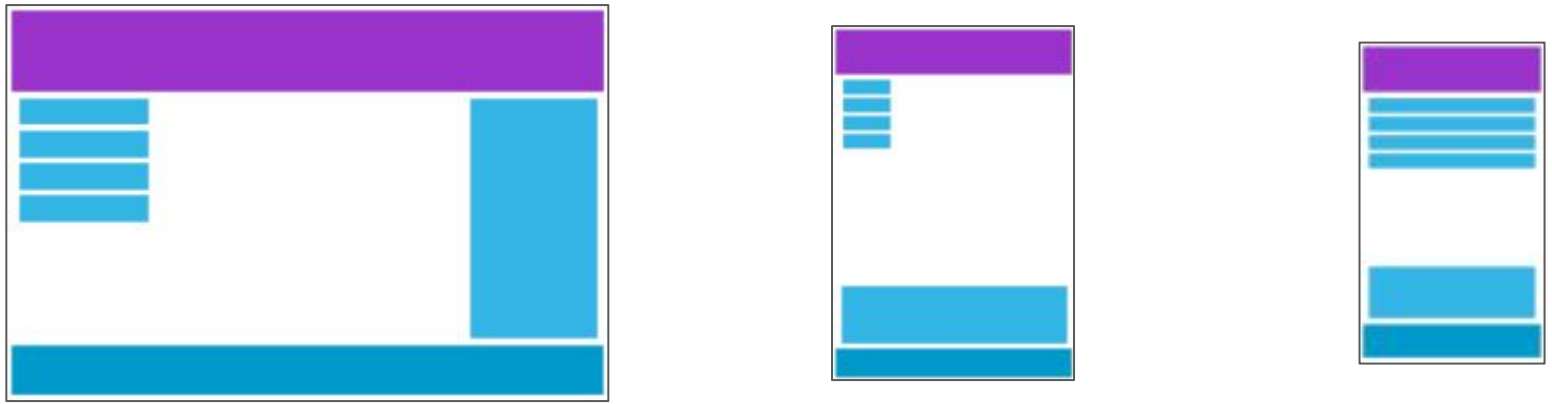
opacidade 0.5

opacidade 1 (padrao)

CSS Responsivo

O que seria Design responsivo?

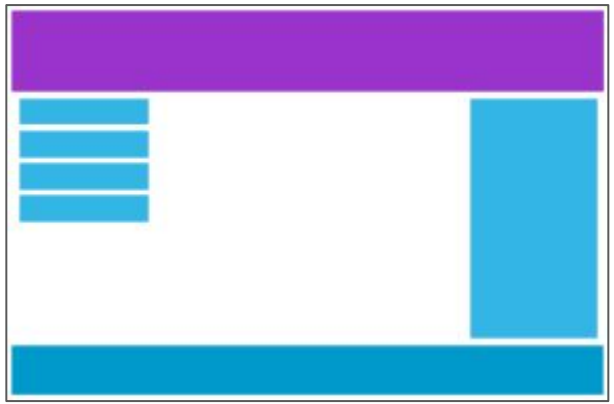
- Um design modular para diversos tamanhos de telas, fazendo com que o seu website tenha uma boa aparência em diferentes tipos de dispositivos.
- Utilizamos apenas HTML e CSS para modularizar nossa aplicação para que o design se adapte a diferentes tamanhos de tela.

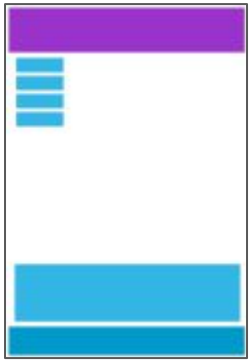


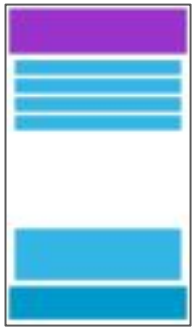
CSS Responsivo

O que seria Design responsivo?

- É chamado de web design responsivo quando você usa CSS e HTML para redimensionar, ocultar, encolher, ampliar ou mover o conteúdo para que fique bem em qualquer tela.







Viewport

O viewport é a área visível do usuário em uma página da web.

Antes smartphones, as páginas da web eram projetadas apenas para telas de computador e era comum que elas tivessem o design estático e o tamanho fixo.

Então, quando com o advento dos dispositivos móveis, páginas da web de tamanho fixo eram muito grandes para caber na janela de visualização. Para corrigir isso, os navegadores nesses dispositivos reduziram a página da web inteira para caber na tela. Esta é uma medida de resolução temporária.

Configurando Viewport

- O HTML5 introduziu um método para permitir que os web designers controlem a janela de visualização, por meio da <meta>tag.
- Você deve incluir elemento <meta> para viewport em todas as suas páginas da web:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Isso fornece ao navegador instruções sobre como controlar as dimensões e a escala da página.
- A width=device-width define a largura da página para seguir a largura da tela do dispositivo (que irá variar dependendo do dispositivo).
- A initial-scale=1.0 define o nível de zoom inicial quando a página é carregada na primeira vez pelo navegador.

CSS Responsivo

- Os usuários estão acostumados a rolar os sites verticalmente em dispositivos desktop e móveis - mas não horizontalmente!
- Portanto, se o usuário for forçado a rolar horizontalmente ou diminuir o zoom para ver toda a página da web, isso resultará em uma experiência ruim para o usuário.

CSS Responsivo

- **1. NÃO use elementos grandes de largura fixa** - Por exemplo, se uma imagem for exibida com uma largura maior do que a janela de visualização, isso pode fazer com que a janela de visualização role horizontalmente. Lembre-se de ajustar este conteúdo para caber na largura da janela de visualização.
- **2. NÃO deixe o conteúdo depender de uma largura de janela de visualização específica para renderizar bem** - Como as dimensões e a largura da tela em pixels CSS variam amplamente entre os dispositivos, o conteúdo não deve depender de uma largura de janela de visualização específica para renderizar bem.
- **3. Use consultas de mídia CSS para aplicar estilos diferentes para telas pequenas e grandes** - A definição de larguras CSS absolutas grandes para os elementos da página fará com que o elemento seja muito largo para a janela de visualização em um dispositivo menor. Em vez disso, considere o uso de valores de largura relativos, como largura: 100%. Além disso, tome cuidado ao usar grandes valores de posicionamento absolutos. Isso pode fazer com que o elemento fique fora da janela de visualização em dispositivos pequenos.

CSS Responsivo Media Types

A regra @media, introduzida no CSS2, possibilitou definir diferentes regras de estilo para diferentes tipos de mídia.

Exemplos: você pode ter um conjunto de regras de estilo para telas de computador, um para impressoras, um para dispositivos portáteis, um para dispositivos do tipo televisão e assim por diante.

Infelizmente, esses tipos de mídia nunca tiveram muito suporte por dispositivos, além do tipo de mídia de impressão. Logo, o conceito de media types fica apenas a título de conhecimento, e veremos a seguir sua extensão no CSS3 as media queries, bem mais utilizadas.

CSS Responsivo Media Queries

As media queries (consultas de mídia) no CSS3 estenderam a ideia de tipos de mídia CSS2: em vez de procurar um tipo de dispositivo, eles olham para a capacidade do dispositivo.

As consultas de mídia podem ser usadas para verificar muitas coisas, como:

- largura e altura do viewport
- Largura e altura do dispositivo
- orientação (o tablet / smartphone está no modo paisagem ou retrato?)
- resolução

O uso de consultas de mídia é uma técnica popular para fornecer uma folha de estilo personalizada para desktops, laptops, tablets e smartphones

CSS Responsivo Media Queries

O que é uma consulta de mídia?

- A consulta de mídia é uma técnica CSS introduzida no CSS3.
- Ele usa a @media regra para incluir um bloco de propriedades CSS apenas se uma determinada condição for verdadeira.

```
@media only screen and (max-width: 600px) {
  body {
      background-color: lightblue;
  }
}
```

CSS Responsivo Media Queries

Vamos explorar mais no W3School

Bootstrap um framework para facilitar nossa vida?

Links Úteis

Tutorial W3Shcool
https://www.w3schools.com/bootstrap5/index.php

Bootstrap
https://getbootstrap.com/docs/4.0/getting-started/introduction/

Bootstrap e Frameworks de CSS

Uma tendência no mundo front-end é o uso de frameworks CSS com estilos base para nossa página.

Ao invés de começar todo o projeto do zero, criando todo estilo na mão, existem frameworks que já trazem toda uma base construída de onde partiremos com nossa aplicação.

Dentre as diversas opções no mercado, uma das mais famosas é o Bootstrap.

Bootstrap e Frameworks de CSS

O Bootstrap traz uma série de recursos, tais como :

- Reset CSS
- Estilo visual base para maioria das tags
- Ícones
- Grids prontos para uso
- Componentes CSS
- Plugins JavaScript
- Tudo responsivo e mobile-first

Assim, podemos começar logo o projeto sem perder tempo com design no início!

Estilo e Componentes Base

Para usar o Bootstrap, apenas incluímos seu CSS na página :

```
<link rel="stylesheet" href="css/bootstrap.css">
```

Fazendo isso já temos :

- Reset aplicado
- Tags com estilo e tipografia base
- Classes com componentes adicionais que podemos aplicar na página.

Estilo e Componentes Base

Por exemplo, para criar um título com uma frase de abertura em destaque, usamos o Jumbotron:

```
<div class="jumbotron jumbotron-fluid">
    <div class="container">
        <h1 class="display-4">Ótima escolha!</h1>
        <p class="lead">Obrigado por comprar na Mirror
    Fashion.</p>
    </div>
</div>
```

Vamos olhar mais exemplos na documentação do bootstrap

Grid Responsivo do Bootstrap

Uma das dificuldades de um projeto front-end é o posicionamento de elementos.

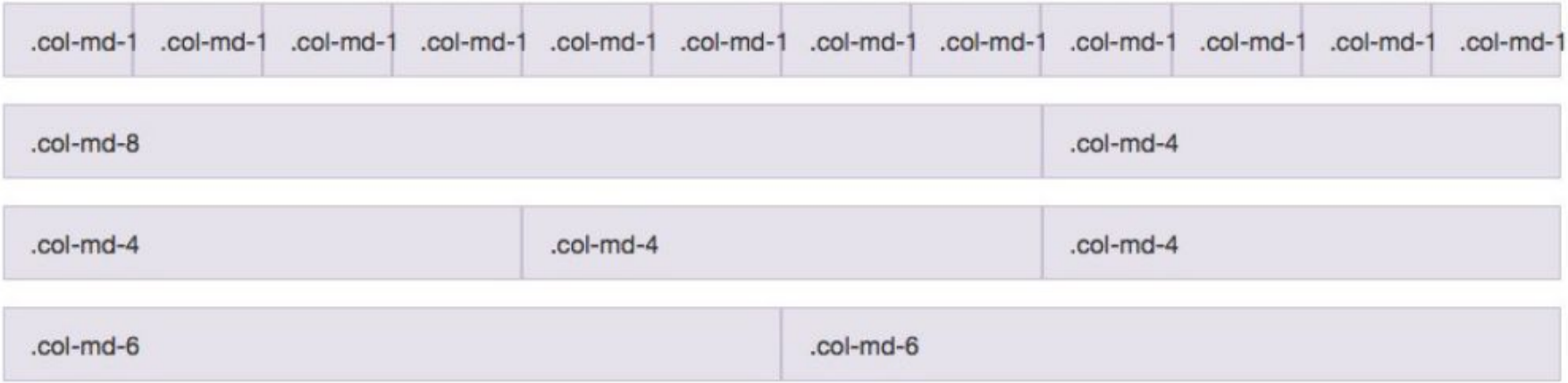
A solução mais comum é o uso de grids, onde divide-se a tela em colunas e os elementos vão sendo encaixados dentro dessas colunas.

Todo framework CSS moderno traz um grid pronto para utilização.

O grid do Bootstrap trabalha com a ideia de 12 colunas, onde podemos escolher quantas colunas iremos ocupar através do nosso código

Grid Responsivo do Bootstrap

Alguns exemplos da divisão de grids no bootstrap :




JS

Javascript

Documentação

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

https://www.w3schools.com/js/default.asp

Javascript

O Javascript consolida o terceiro pilar do desenvolvimento WEB, trazendo para nossa aplicação uma maior interatividade com o usuário.

JavaScript é uma linguagem de script orientada a objetos, multiplataforma. É uma linguagem pequena e leve. Dentro de um ambiente de host (por exemplo, um navegador web) o JavaScript pode ser ligado aos objetos desse ambiente para prover um controle programático sobre eles.

Pode atuar tanto no lado do cliente quanto no lado do servidor.

É suportado por todos Browser (Navegadores) modernos.

Javascript

O JavaScript, como o próprio nome sugere, é uma linguagem de scripting.

Uma linguagem de scripting é comumente definida como uma linguagem de programação que permite ao programador controlar uma ou mais aplicações de terceiros.

No caso do JavaScript, podemos controlar alguns comportamentos dos navegadores através de trechos de código que são enviados na página HTML.

Javascript

Outra característica comum nas linguagens de scripting é que normalmente elas são linguagens interpretadas, ou seja, não dependem de compilação para serem executadas.

Essa característica é presente no JavaScript: o código é interpretado e executado conforme é lido pelo navegador, linha a linha, assim como o HTML.

Javascript

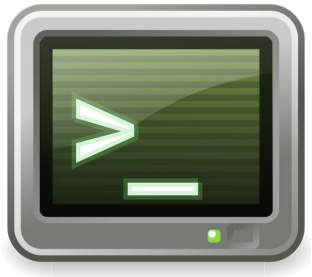
Javascript é uma linguagem interpretada
Fracamente tipada e dinâmica

```
var texto = "Um texto qualquer";
var numero = 10;
texto = numero;
console.log(texto + 10)
```

O Console do Navegador

Existem várias formas de executar códigos JavaScript em um página. Uma delas é executar códigos no que chamamos de Console.

A maioria dos navegadores desktop já vem com essa ferramenta instalada. No Chrome, é possível chegar ao Console apertando F12 e em seguida acessar a aba "Console" ou por meio do atalho de teclado Control + Shift + C; no Firefox, pelo atalho Control + Shift + K.



Sintaxe Básica

Vamos olhar pelos exemplos da W3Schools a sintaxe básica do Javascript, que compreende os seus valores (literais e variáveis), operadores e tipos de dados.

A tag script

A tag <script> é usada para incorporar no código HTML um script do lado do cliente (JavaScript).

O elemento <script> contém instruções de script ou aponta para um arquivo de script externo por meio do atributo src.

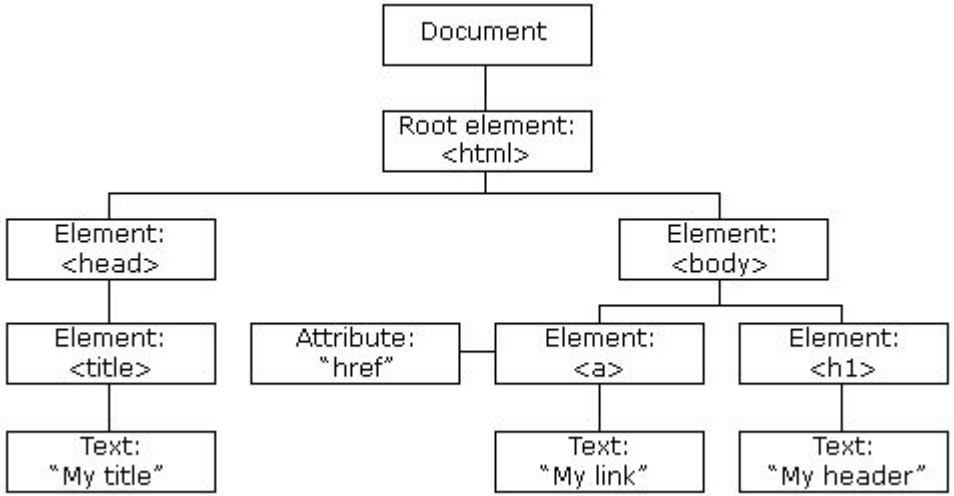
```
<script src="script.js"></script>
```

```
<script>
    document.getElementById('exemplo')
</script>
```

DOM: a página no mundo Javascript

Quando uma página da web é carregada, o navegador cria um modelo de objeto de documento (do inglês **D**ocument **O**bject **M**odel) da página.

O modelo HTML DOM é construído como uma árvore de objetos:



DOM: a página no mundo Javascript

Com o modelo de objeto, o JavaScript obtém todo o poder de que precisa para criar HTML dinâmico, sendo capaz de:

- Alterar todos os elementos HTML na página
- Alterar todos os atributos HTML na página
- Alterar todos os estilos CSS na página
- Remover elementos e atributos HTML existentes
- Adicionar novos elementos e atributos HTML
- Reagir a todos os eventos HTML existentes na página
- Criar novos eventos HTML na página

JavaScript HTML DOM Document

O objeto HTML DOM **document** é o proprietário de todos os outros objetos em sua página da web.

Se você deseja acessar qualquer elemento em uma página HTML, você sempre começa acessando o objeto **document**.

Veremos alguns exemplos de como podemos usar o objeto **document** para acessar e manipular HTML

Funções e os eventos do DOM

Os métodos/funções HTML DOM são ações que você pode executar (em elementos HTML).

Propriedades HTML DOM são valores (de elementos HTML) que você pode definir ou alterar.

Veremos alguns exemplos à seguir.

Funções e os eventos do DOM

Alguns exemplos de métodos :

Método	Descrição
document.getElementById(*id*)	Encontra um elemento por id de elemento
element.setAttribute*(attribute, value)*	Altera o valor do atributo de um elemento HTML
document.createElement(*element*)	Cria um elemento HTML
document.getElementById(*id*).onclick = function(){*code*}	Adicionando código de manipulador de eventos a um evento onclick

Funções e os eventos do DOM

Alguns exemplos de propriedades :

Propriedade	Descrição
element.innerHTML = *new html content*	Alterar o HTML interno de um elemento
element.attribute = new value	Altere o valor do atributo de um elemento HTML
element.style.*property = new style*	Altere o estilo de um elemento HTML

Tipos de dados

Vamos conhecer alguns tipos de dados existentes em JavaScript

- String
- Number
- Boolean
- Undefined e Null
- Object
- Array

String

String é uma cadeia de caracteres. Podem ser representados através das aspas simples (‘ ’), aspas duplas (“ ”) ou template strings (` `).

As Template Strings permitem que trabalhemos com strings multi-linhas e utilizemos expressões de linguagem utilizando a formatação ${}.

```
var nome = "Carlos";
var frase = `${nome} é um cara legal.`;
//Carlos é um cara legal.
```

Number

São tipos de dados os quais conseguimos manipular de forma numérica.

```
//Numbers
33 //Int (Inteiro)
15.6 //Float (Real)
NaN // Not a Number
Infinity //Infinito
```

Boolean

True ou False. Muito utilizados em funções condicionais, entre outros.

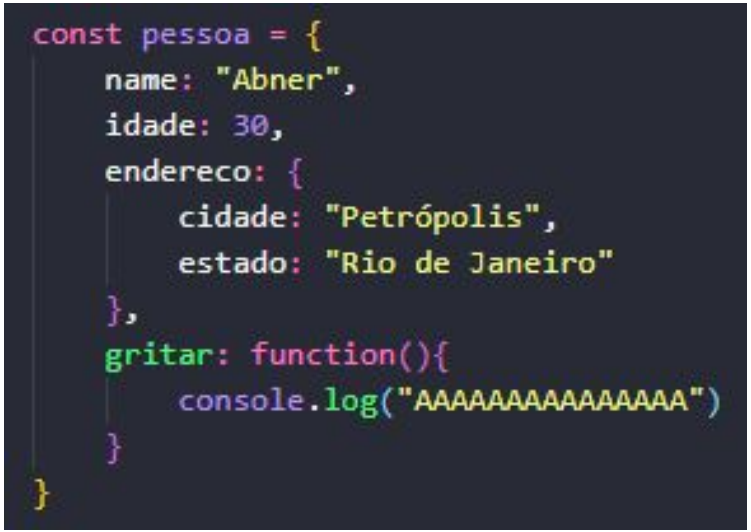
Undefined e Null

Undefined e null são dois tipos de caracteres muito confundidos no JavaScript, porém, entender a diferença entre eles é crucial durante o desenvolvimento das nossas aplicações.

- Undefined: Valor indefinido - Algo que não existe
- Null: Valor nulo - Objecto que não tem nada dentro dele

Object

Um objeto é composto de Propriedades/Atributos e Funcionalidades/Métodos. É necessária uma atenção especial neste tipo de dado pois ele estará presente frequentemente em nosso cotidiano.



Object

Acessando Propriedades/Atributos e Funcionalidades/Métodos.

```
const pessoa = {
    nome: "Abner",
    idade: 30,
    endereco:{
        cidade: "Petrópolis",
        estado: "Rio de Janeiro"
    },
    gritar: function() {
        console.log("AAAAAAAAAAAAAAAA");
    }
}

console.log(pessoa.nome)
console.log(pessoa.endereco.cidade)
pessoa.gritar()
```

Array (Vetores)

É uma lista, ou seja um agrupamento de dados. Podem receber qualquer tipo de dados dentro de si;

```
const meuArray = ["Banana", "Maçã", "Mamão"]

const meuArray = [10, 256, "Abner", true, {nome: "Carlos", idade: 26}, 15.8]
```

Funções Anônimas

Em casos como esse, onde não há um outra parte do código onde queremos referenciar uma função e ela será apenas referenciada ao invés de chamar a função, podemos usar o conceito de função anônima, já criando a função no lugar onde antes apenas indicamos seu nome. Por exemplo :

```
var inputTamanho = document.getElementById("inputTamanho");
var outputTamanho = document.getElementById("outputTamanho");

inputTamanho.oninput = function(){
    outputTamanho.value = inputTamanho.value;
};
```

Manipulando Strings

Uma variável que armazena um string faz muito mais que isso! Ela permite, por exemplo, consultar o seu tamanho e realizar transformações em seu valor. Por exemplo :

```
var palavraTosca = 'Catapora'
palavraTosca.length; // Tamanho da string
palavraTosca.replace("pora", "pimba");
```

Manipulando Strings

Assim como em Java, podemos converter uma String para inteiro ou ponto flutuante usando o método parseInt e parseFloat :

```
var textoInteiro = "10";
var inteiro = parseInt(textoInteiro, 10);
var textoFloat = "10.22";
var float = parseFloat(textoFloat)
```

Manipulando Números

Números, assim como strings, também são imutáveis. O exemplo abaixo altera o número de casas decimais com a função toFixed. Esta função retorna uma string, mas, para ela funcionar corretamente, seu retorno precisa ser capturado:

```
var milNumber = 1000;
var milString = milNumber.toFixed(2); //Recebe o retorno da função
console.log(milString); //Imprime a string "1000.00"
```

Arrays em Javascript

A utilização de arrays em javascript não é muito diferente do que foi visto em Portugol ou Java. Os pontos interessantes são que, por javascript não ser tipado, podemos armazenar valores de tipos diferentes em vetores no Javascript.

```
var variosTipos = ["Mamona", 10, [1,2]];
console.log(variosTipos[0]);
```

Arrays em Javascript

Para adicionar elementos ao vetor, podemos utilizar a função **push** que adiciona um elemento na última posição do array ou adicionar direto em um índice selecionado :

```
var palavras = ["RSW", "Ensino"];
palavras.push("Inovação");
palavras[9] = "Criatividade";
console.log(palavras);
```

Arrays em Javascript são semelhantes às ArrayLists em Java

Arrays em Javascript

Manipulando Arrays

- .length => Permite ver o comprimento do array
- .join(“,”) => Junta todos os elementos do array em uma string
- .pop() => Remove o último elemento do array
- .shift() => Remove o primeiro elemento do array
- .push() => Adiciona um elemento ao final do array
- .indexOf() => Localiza um elemento no array
- .find() => Localiza um elemento no array

Arrays em Javascript

Concatenando Arrays

```
var meuArray = [1,2,3]
var meuArray2 = [6,7,8]
var meuArrayConcatenado = [...meuArray, 4, 5, ...meuArray2, 9, 10, 11]
console.log(meuArrayConcatenado)
```

ES6

Laços de Repetição e Condicionais

Todas as estruturas de laços de repetição e condicionais que vimos em disciplinas passadas funciona em javascript. Apenas alguns exemplos de sintaxe:

```
while(contador <= 10){
    //código a ser repetido
}
```

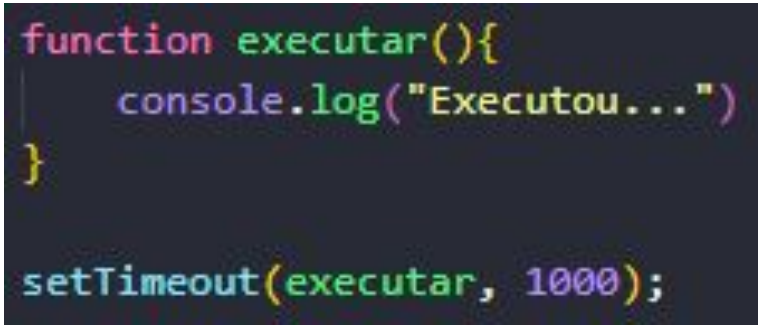
```
if(condicao){
    //código a ser executado
}
```

```
for(/*variável de controle*/;/* condição */; /* pós execução*/){
    // código a ser repetido
}
```

Funções Temporais

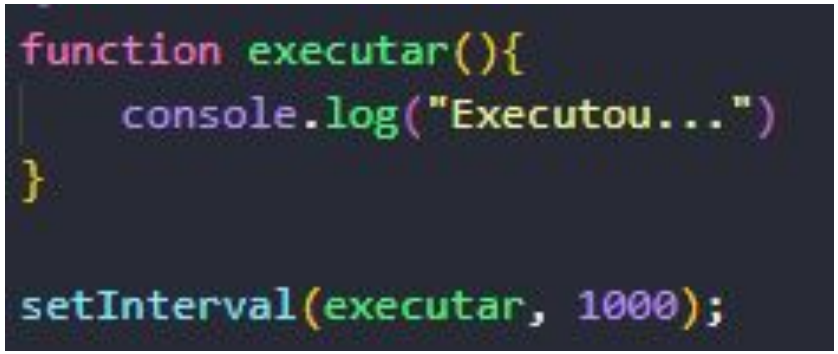
Em JavaScript, podemos criar um timer para executar um trecho de código após um certo tempo, ou ainda executar algo de tempos em tempos.

A função **setTimeout** permite que agendemos alguma função para execução no futuro e recebe o nome da função a ser executada e o número de milissegundos a esperar:



Funções Temporais

Se for um código recorrente, podemos usar o setInterval que recebe os mesmos argumentos mas executa a função indefinidamente de tempos em tempos:



É uma função útil para, por exemplo, implementar um banner rotativo, apresentado no exercício à seguir.

Desestruturação no JavaScript

Desestruturação consiste em você extrair do objeto apenas as propriedades que precisa. Muito útil para objetos que possuem muitas propriedades.

```
const {nome, idade, endereco, gritar} = pessoa;
const {cidade} = endereco;

console.log(nome)
console.log(idade)
console.log(cidade)
gritar()
```

ES6

Desestruturação no JavaScript

Funciona também com arrays, porém como o array utiliza [ ] a desestruturação fica dessa maneira:

```
var letras = ["A","B","C","D"]
const[primeiro, segundo] = letras
```

```
var letras = ["A","B","C","D"]
const[primeiro,, terceiro] = letras
```

ES6

Spread & Rest Operator...

Ambos são bem parecidos e podem gerar muita confusão porém, existem suas diferenças. Spread tem a ideia de “espalhar” os dados, enquanto o Rest tem a ideia de “pegar o resto dos dados”. Este operador é muito útil para manipulação de Arrays, Objetos e também na implementação de funções mais dinâmicas.

...Spread

```
var meuArray = [1,2,3]
var meuArray2 = [6,7,8]
var meuArrayConcatenado = [...meuArray, 4, 5, ...meuArray2, 9, 10, 11]
console.log(meuArrayConcatenado)
```

ES6

Spread & Rest Operator...

...Rest

```
var nomes = ["João", "Juliana", "Alexandre", "Roberto"];
const [primeiro, ...resto] = nomes;

console.log(primeiro)
console.log(resto)
```

ES6

Spread & Rest Operator...

Aplicação em objetos ...Spread

```
const pessoa = {
    nome: "Abner",
    idade: 30,
    endereco:{
        cidade: "Petrópolis",
        estado: "Rio de Janeiro"
    },
    gritar: () => {
        console.log("AAAAAAAAAAAAAAAA");
    }
}

const pessoa2 = {...pessoa, idade: 18}
console.log(pessoa2)
```

ES6

Spread & Rest Operator...

Aplicação em objetos ...Rest

```
const pessoa = {
    nome: "Abner",
    idade: 30,
    endereco:{
        cidade: "Petrópolis",
        estado: "Rio de Janeiro"
    },
    gritar: () => {
        console.log("AAAAAAAAAAAAAAAA");
    }
}

const { nome, ...resto} = pessoa;
console.log(nome)
console.log(resto)
```

ES6

Spread & Rest Operator...

Aplicação em funções ...Spread

```
function somarNumeros(n1, n2){
    console.log(n1 + n2)
}

var numeros = [2, 2]
somarNumeros(...numeros)
```

ES6

Spread & Rest Operator...

Aplicação em funções ...Rest

```
function nomesComRest(...nomes){
    console.log(nomes)
}

nomesComRest("João", "Juliana", "Alexandre", "Roberto")
nomesComRest("João", "Juliana")
```

ES6

Arrow Functions =>

Uma expressão arrow function possui uma sintaxe mais curta quando comparada a uma expressão de função.

```
ola = _ => 'Olá'
```

ES6

Arrow Functions =>

```
let ola = function () {
    return 'Olá!'
}
```

```
ola = _ => 'Olá'
```

ES6