Frontend Essencial

Professor: Weberson Rodrigues de Araujo de Oliveira

serratec
Parque Tecnológico Região Serrana

O que esperar desta disciplina?

Fazer um brainstorm com os alunos do que veremos aqui

serratec
Parque Tecnológico Região Serrana

Referências para disciplina:

- Documentação da W3Schools
- Documentação oficial do Bootstrap
- Documentação da MDN

serratec
Parque Tecnológico Região Serrana

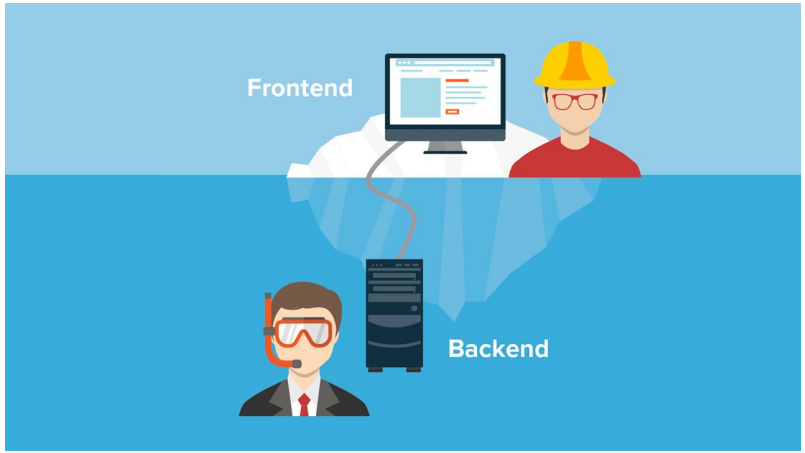
Frontend x Backend

De maneira simples, Frontend pode ser entendido como a parte visual de uma aplicação, ou seja, o que o usuário final consegue interagir. Geralmente as tecnologias mais usadas em Frontend são as tecnologias base da Web, como HTML, CSS e JavaScript.

serratec
Parque Tecnológico Região Serrana

Frontend x Backend

Já o Backend, vem da ideia do que há por trás de uma aplicação, fazendo a ponte entre os dados armazenados no banco de dados com a informação apresentada e recuperada na interface gráfica da aplicação, realizando as devidas regras de negócio, validações e segurança sobre a manipulação dos dados.

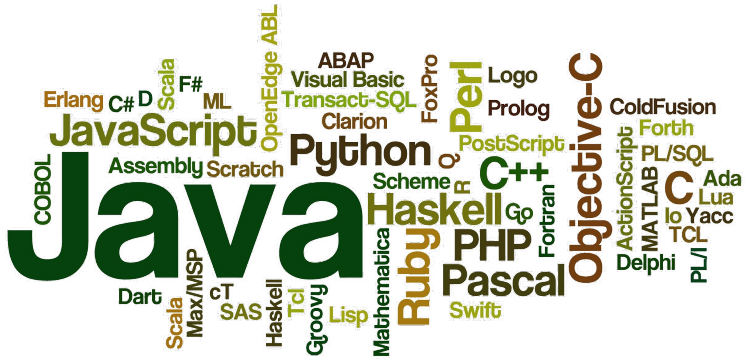


serratec
Parque Tecnológico Região Serrana

Linguagens de Programação X Linguagens de Marcação

Até o momento vimos algumas linguagens de programação (Portugol, Java), uma linguagem de consulta (SQL) e agora veremos algumas linguagens de marcação (HTML, CSS).

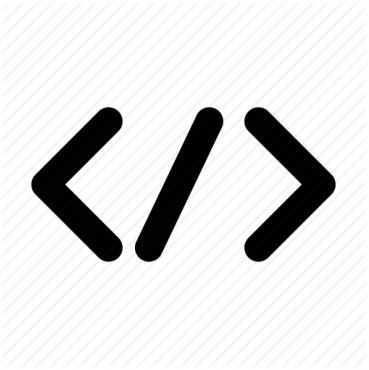
Uma **linguagem de programação** é usada para *transformar* dados. Isso é feito criando instruções da CPU que reescrevem os dados de entrada na saída.

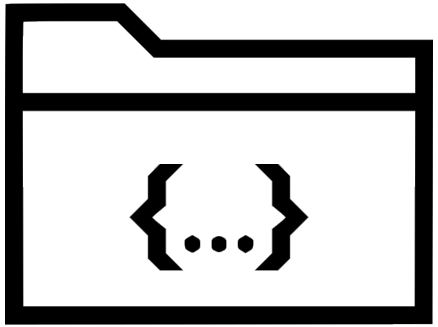


serratec

Linguagens de Programação X Linguagens de Marcação

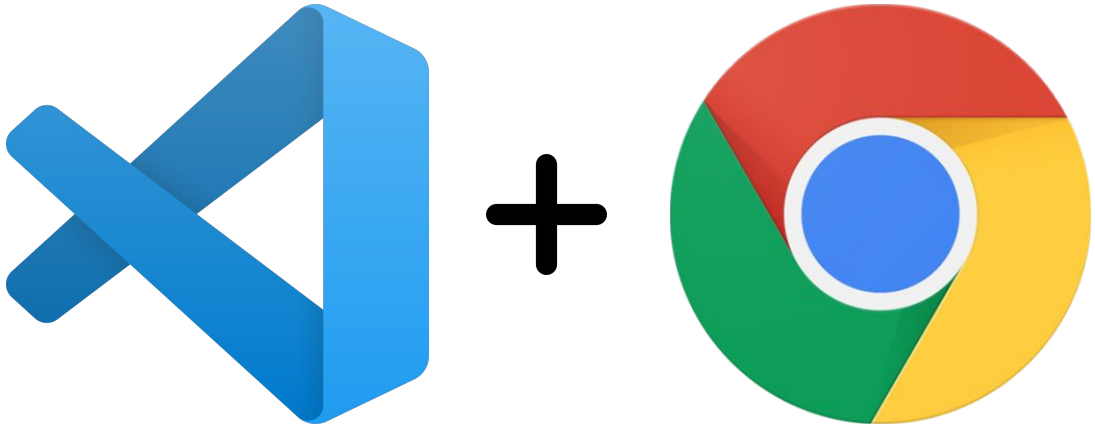
Uma **linguagem de marcação** é usada para controlar a apresentação dos dados, ou seja, como “representar esses nomes de usuário como uma lista de marcadores ou uma tabela”.



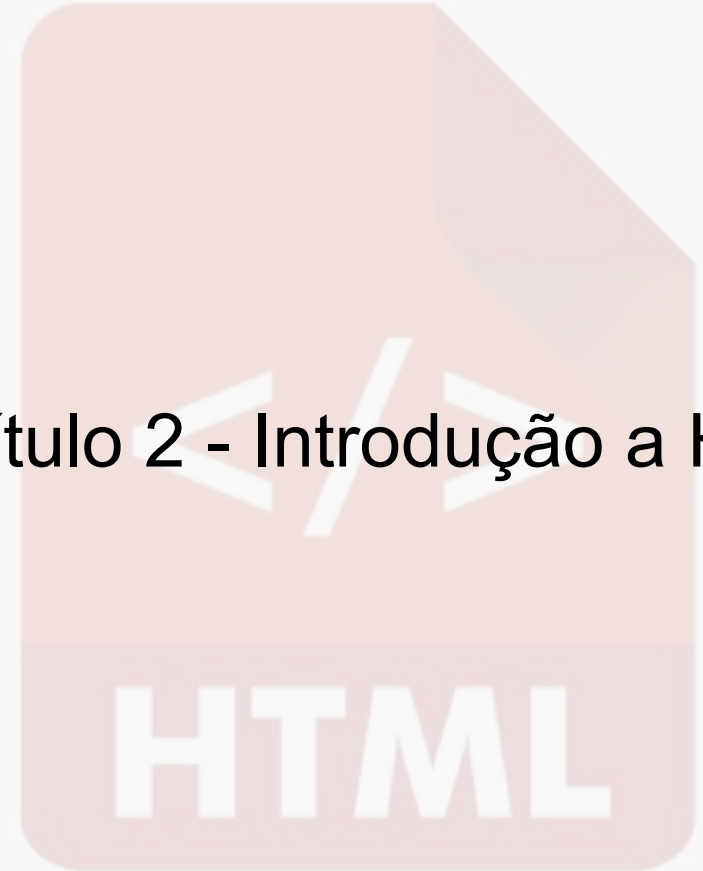


Qual Ferramenta usar para esse curso?

Inicialmente, testaremos nossas próprias aplicações direto em nosso web browser. Assim, um simples editor de texto e seu web browser favorito poderão fazer a mágica.



Capítulo 2 - Introdução a HTML e CSS



Introdução ao HTML

A única linguagem que o navegador consegue interpretar para exibição de conteúdo é o HTML.

Vamos testar como o clássico “Olá Mundo” pode ficar usando o HTML :

- Em qualquer editor, crie um aquivo com nome index.html
- Escreva “Olá Mundo” no aquivo
- Abra no navegador da sua escolha

O que aconteceu?

Melhorando o exemplo - Nosso e-commerce

Vamos adicionar um texto mais chamativo para nossa página principal. Algo do tipo :

Lojas Tabajara
Especialista em eletrônicos, carros, salgados e doces.
Entregamos em domicílio e pela internet
Não emitimos nota fiscal
Garantia de 2 horas

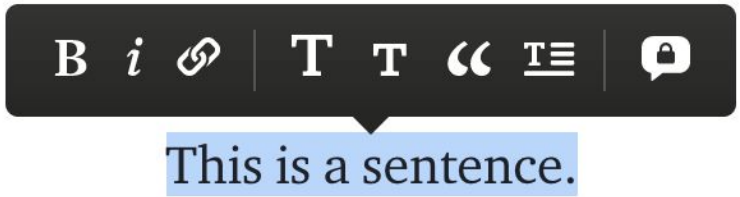
Como ficou o resultado final?

Formatando nossa página principal

Vimos que o exemplo não ficou muito como esperado.

Para podermos exibir as informações desejadas com a **formatação**, é necessário que cada trecho de texto tenha uma **marcação** indicando qual é o significado dele.

Vamos ver um exemplo em nosso arquivo de texto com a marcação adequada.



Formatando nossa página principal

```
<!DOCTYPE html>
<html>
<head>
   <title>Lojas Tabajara</title>
   <meta charset="utf-8">
</head>
<body>
   <h1>Lojas Tabajara</h1>
   <h2>Especialista em eletrônicos, carros, salgados e doces.</h2>
   <ul>
       <li>Entregamos em domicílio e pela internet</li>
       <li>Não emitimos nota fiscal</li>
       <li>Garantia de 2 horas</li>
   </ul>
</body>
</html>
```

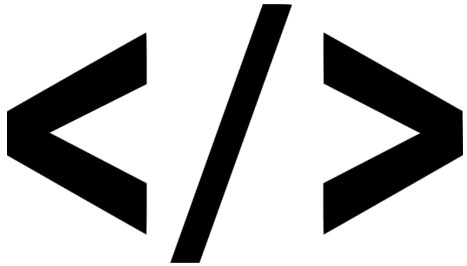
Vamos tentar juntos deduzir o que cada parte deste código está fazendo!

Percepções do exemplo anterior

Vimos que o resultado exibido após a inserção das **tags** HTML é um texto muito mais agradável e legível.

O que são essas **tags** então?

- As **tags** HTML são basicamente uma maneira de dar significado a um texto contido em seu escopo



O HTML foi originalmente desenvolvido para...

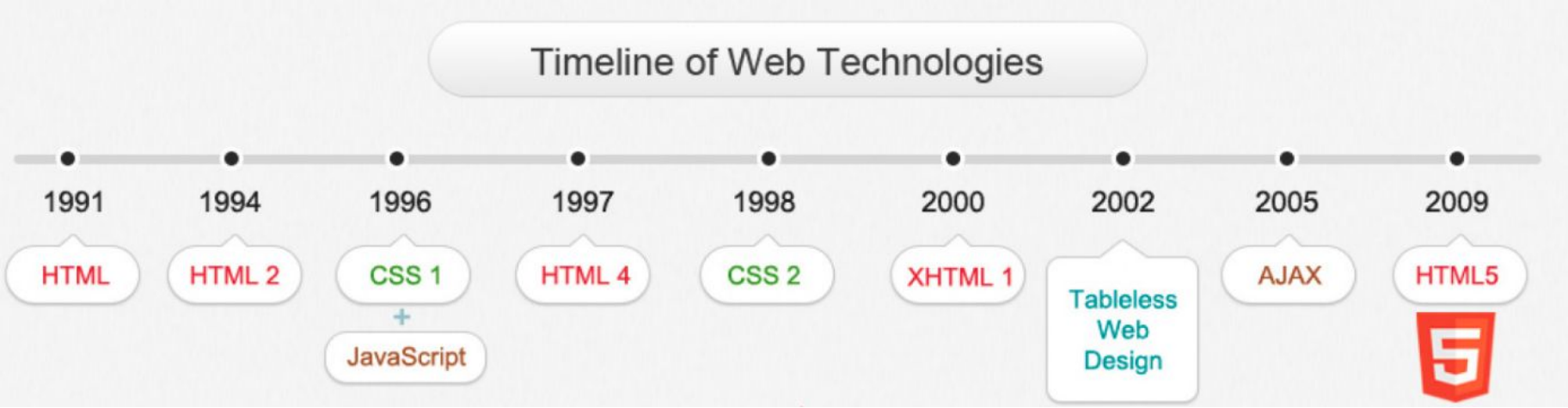
Originalmente, o HTML foi desenvolvido para exibição de documentos científicos.

É como se a Web fosse desenvolvida para exibir monografias formatados pela ABNT. Porém, com o tempo, a evolução da Web e de seu potencial comercial tornou-se necessário a exibição de informações com riquezas de elementos gráficos e interações.

Atualmente, a linguagem de marcação/sistema de preparação de documentos mais utilizada para escrita de trabalhos científicos é o Latex.

Histórico de Evolução do HTML

Desde os primeiros dias da World Wide Web, houve muitas versões de HTML:



Sintaxe do HTML

O HTML é um conjunto de **tags** responsáveis pela marcação do conteúdo de uma página no navegador.

Uma tag é definida com caracteres “<” e “>”. Quando quisermos englobar algum tipo de conteúdo em uma tag, precisamos fechar a respectiva tag com uma tag respectiva de fechamento.

Por exemplo: `<h1>Lojas Tabajara</h1>`

O que a Tag a seguir faz com o texto do exemplo?

Sintaxe do HTML

Entretanto, algumas tags podem receber atributos dentro de sua própria definição. Por exemplo, se quisermos adicionar uma imagem ao nosso site, podemos fazer:

```
<img src="../imagens/casa_de_praia.jpg>
```

Note que nesse exemplo a tag **img** não possui algum conteúdo externo para englobar. Nesses casos não é necessário usar uma tag de fechamento

Estrutura de um documento HTML

Um documento HTML válido precisa seguir obrigatoriamente a estrutura composta pelas tags `<html>`, `<head>` e `<body>`, além da instrução `<!DOCTYPE>`. A seguir, veremos a importância de cada uma:

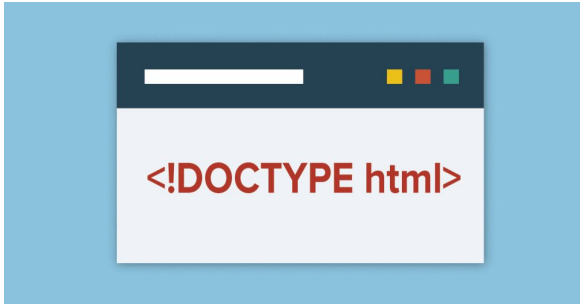
```
<!DOCTYPE html>

<html>
    <head>
       <title>Titulo da página</title>
    </head>
    <body>
       <h2>Estou no corpo da página.</h2>
    </body>
</html>
```

A instrução DOCTYPE

Todos os documentos HTML devem começar com a declaração de tipo de documento: <! DOCTYPE html>.

A declaração <! DOCTYPE> 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 tag <html>

Na estrutura do nosso documento, a primeira tag a ser inserida é a tag **<html>.**

Dentro dessa tag, precisamos declarar outras duas tags: <head> e <body>. Temos assim, uma estrutura inicial padrão da seguinte forma:

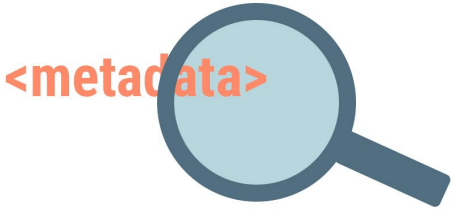
```
<html>

    <head></head>

    <body></body>

</html>
```

A tag <head>

A tag **<head>** contém informações sobre o documento que são de interesse somente do navegador.

Assim, estas informações não serão exibidas na tela do navegador.

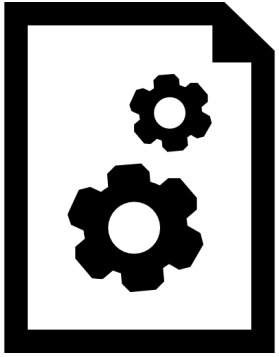
A especificação obriga a presença da tag <title> dentro do <head> permitindo especificar o título do nosso documento, que é geralmente exibido na barra de título do navegador.



Mais sobre tag <head>

Uma outra configuração, muito utilizada em documentos escritos em idiomas como o português (possui caracteres como acento e cedilha) é a configuração de codificação de caracteres (**encoding**, ou **charset**).

Esta configuração é passada dentro da tag **<meta>** responsável por configurar as metasconfigurações, as configurações do nosso documento.



A tag <body>

A tag `<body>` contém o corpo do documento que é exibido pelo navegador.

É necessário que o `<body>` tenha ao menos um elemento “filho”, ou seja, uma ou mais tags HTML dentro dele. Exemplo:

```
<body>

    <h1>Título Qualquer</h1>

    <p>Primeiro parágrafo. blablablabla</p>

</body>
```

Mais tags - Títulos, parágrafos e ênfases

O HTML é composto de diversas tags, cada uma com sua função e significado. Veremos diversas tags ao longo do curso.

Neste momento, vamos focar em tags que representam **títulos**, **parágrafos** e **ênfase**.

Meu título!

Parágrafo 1, blablabla.

Parágrafo 2 com **ênfase**

Tags HTML - Titulos (Cabeçalhos)

Os cabeçalhos HTML são definidos com as tags de `<h1>` a `<h6>`.

`<h1>` define o cabeçalho mais importante. `<h6>` define o título menos importante

Exemplo:

```
<h1>Título principal</h1>

<h2>Subtitulo</h2>

<h3>Sub - subtítulo?</h3>
```

Vamos Testar!

Tags HTML - Parágrafos

Os parágrafos HTML são definidos com a tag <p>:

```
<p>Isto é um parágrafo.</p>

<p>Isto é um outro parágrafo.</p>
```

Vamos Testar!

Tags HTML - Marcações de ênfase

O HTML contém vários elementos para definir o texto com um significado especial.

Os elementos de formatação foram projetados para exibir tipos especiais de texto:

- <b> - Texto em negrito
  - - Texto importante
- <i> - Itálico
  - - Texto enfatizado
- <mark> - Texto marcado
- <small> - Texto menor
- <del> -Texto deletado
- <ins> - Texto inserido
- <sub> - Texto subscrito
- <sup> - Texto sobrescrito

Manipulando Imagens

As imagens HTML são definidas com a tag <img>.

O arquivo de origem (src), texto alternativo (alt), largura (width) e altura (height) são fornecidos como atributos:

Exemplo:

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

Exercício: Primeiros passos com HTML

Sugestão de exercícios:
Exercício:

- Exercícios extras: Lista W3schools

Estilizando com CSS

Vimos algumas tags que permitem manipularmos nosso documento até um certo nível.

Mas e se quisermos personalizar ainda mais nossa página, talvez mudando a cor do texto, a fonte, etc?

Antigamente, essas operações eram feitas no próprio HTML. Se quiséssemos colorir um título por exemplo, poderíamos fazer:

```
<h1><font color="red">Lojas Tabajara anos 90</font></h1>
```

Estilizando com CSS

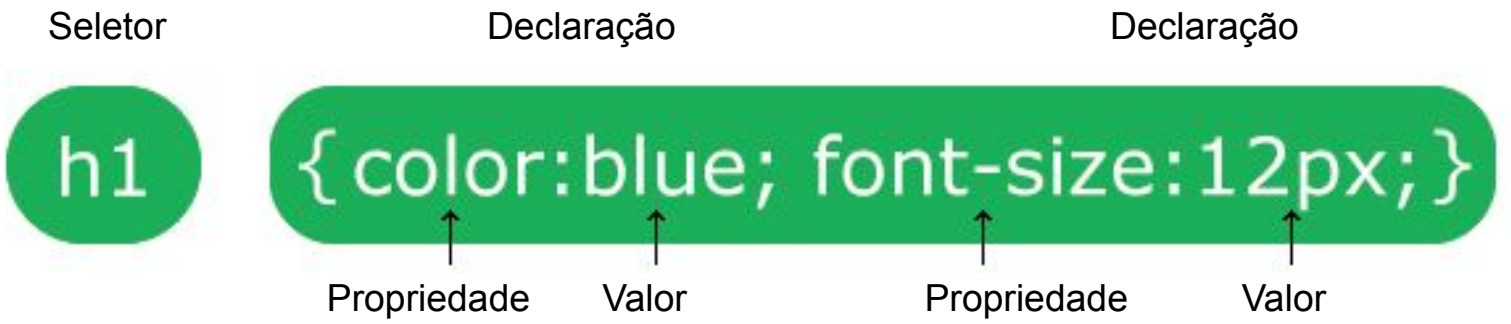
Além da tag <font>, várias outras tags de estilo existiam. Hoje em dia, entretanto utilizar tags HTML para estilo é considerado uma **má prática** e não deve ser usado.

Em seu lugar, surgiu o CSS, que é outra linguagem, separada do HTML, com objetivo especial de cuidar da estilização da página.

Assim, o conteúdo fica por responsabilidade do HTML e a estilização e formatação visual fica por responsabilidade do CSS.

Sintaxe e Inclusão de CSS

A sintaxe do CSS é bem simples: é uma declaração de propriedades e valores separados por um sinal de dois pontos “:”, e cada propriedade é separada por um sinal de ponto e vírgula “;”. Por exemplo:



Utilizando o CSS dentro do arquivo HTML

Uma forma de utilizar o CSS dentro do próprio arquivo HTML, sem quebrar a função de cada uma das linguagens é através da tag **<style>.**

Assim, definimos qual estrutura no HTML iremos aplicar o estilo referenciando o seletor específico. Veja o exemplo a seguir

```
<head>
<style>
        p {color : blue; background-color: yellow; }
</style>
</head>
<body>
        <h1> Lojas Tabajara </h1>
        <p> Fundadas pelo casseta e planeta, as lojas tabajara revolucionaram o mundo </p>
</body>
```

Utilizando o CSS como um arquivo externo

Outra forma de declararmos os estilos de nossos documentos é com um arquivo externo, geralmente com a extensão .css.

A principal vantagem dessa abordagem é que além de deixar o projeto mais organizado, a mesma folha de estilo pode ser utilizada em diversos documentos.

Para referenciarmos o CSS em um arquivo separado utilizamos a tag <link> dentro da tag <head> do documento HTML.

```
<head>
    <meta charset="utf-8">
    <title>Sobre as lojas tabajara</title>
    <link rel="stylesheet" href="estilos.css">.
</head>
```

Manipulando Fontes

Da mesma forma que alteramos cores, também podemos alterar o texto. Definimos fontes com o uso da propriedade **font-family.**

Por padrão, os 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 Texto

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*/
}
```

Imagem de Fundo

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

```
body {
    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

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

Exercício : Primeiros Passos com CSS

Sugestão de exercícios:

- Exercícios extra: Lista de CSS da W3Schools

Cores

Propriedades como **background-color, color, border-color** entre outras aceitam uma cor como valor. Existem várias maneiras de definir cores quando utilizamos o CSS

A mais simples é usando o nome da cor:

```
h1 {
    color: red;
}
```

Existem 140 nomes padrão de cores suportadas pelo CSS/HTML!

Cores

A dificuldade da abordagem anterior é acertar a exata variação de cor que queremos no design. Outra abordagem é utilizar o sistema de cores RGB.

Com o RGB, podemos especificar até 16 milhões de cores com uma combinação das cores básicas Vermelho (Red), Verde(Green) e Azul(Blue). Cada cor tem uma intensidade que varia de 0 a 255

```
h3 {
color: rgb(255, 200, 0);
}
```

```
h3 {
color: #FFC800;
}
```

Listas HTML

As listas HTML permitem que os desenvolvedores da web agrupem um conjunto de itens relacionados em listas. Elas podem ser ordenadas, não ordenadas ou seguirem uma 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:

Pão
- Pão francês
- Pão integral

Suco
- Suco de laranja

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

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

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

- Pão
  - Pão Frances
- Suco
  - Suco de Laranja

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:

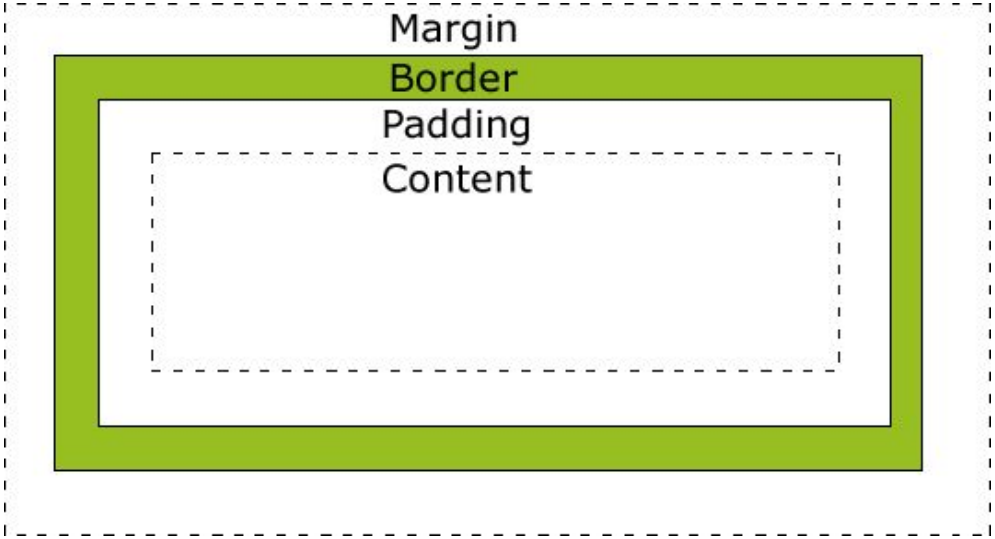
- margin-top
- margin-right
- margin-bottom
- margin-left

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

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;
}
```

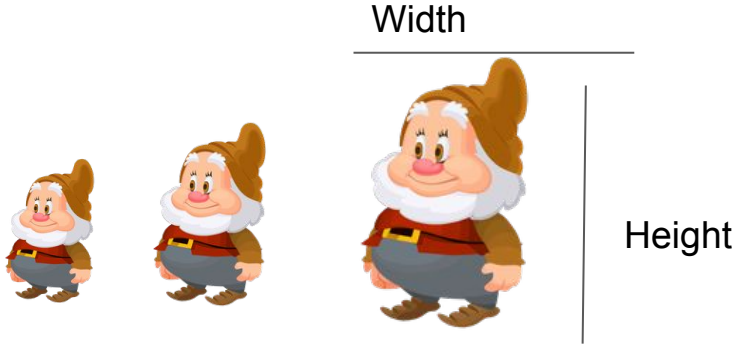


Definindo as dimensões de um elemento

É possível também definir diretamente as dimensões de um elemento, por exemplo:

```
p {
   background-color: red;
   height: 300px;
   width: 500px;
}
```

Assim, todos os parágrafos ocuparão 300 pixels de altura e 500 de largura, com cor de fundo vermelha



Adicionando Links

Quando precisamos indicar que um trecho de texto se refere a outro conteúdo, seja ele no mesmo documento ou em outro endereço, utilizamos a tag de âncora <a>.

Existem dois usos para as âncoras. Um deles é a definição de links:

```
<p>
Visite o site
<a
href="https://www.w3schools.com/html/html_links.asp">W3scools
</a>.
</p>
```

Adicionando Links

Outro uso para a tag de âncora `<a>` é a demarcação de destinos para links dentro do próprio documento.

```
<p>
    Mais informações <a href=”#info”>aqui</a>.
</p>

<p> Conteúdo da página </p>

<h2 id =”info”> Mais informações sobre o assunto: </h2>
<p> Informações... </p>
```

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 !

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 de divs e spans no W3Schools!

Seletores CSS

Já vimos como selecionar elementos no CSS usando simplesmente o nome da tag:

```
p {
    color: red;
}
```

Apesar de simples, é uma maneira muito limitada de selecionar. Às vezes não queremos pegar todos os parágrafos por exemplo, mas apenas algum determinado.

Seletor de ID

É possível aplicar propriedades visuais a um elemento selecionado pelo valor de seu atributo id. Para isso, o seletor deve iniciar com o caractere “#” seguido do valor correspondente.

```
#cabecalho {
    color: white;
    text-align: center;
}
```

O seletor acima fará que o elemento com atributo id de valor “cabecalho” tenha seu texto renderizado na cor branca e centralizado.

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.

Propriedade Float e Clear

A propriedade float do CSS especifica como um elemento deve flutuar.

A propriedade clear do CSS especifica quais elementos podem flutuar ao lado do elemento “cleared” e em que lado.

Estas são propriedades de posicionamento e formatação de conteúdo, usados para orientar containers e imagens.

Como em web, tudo é muito visual, vamos olhar mais alguns exemplos dessas propriedades aqui

Exercícios de Seletores CSS e Flutuação de Elementos

A prática leva a perfeição!

Sugiro praticar com os exemplos que vimos até o momento.

Capítulo 3 - HTML semântico e posicionamento

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.

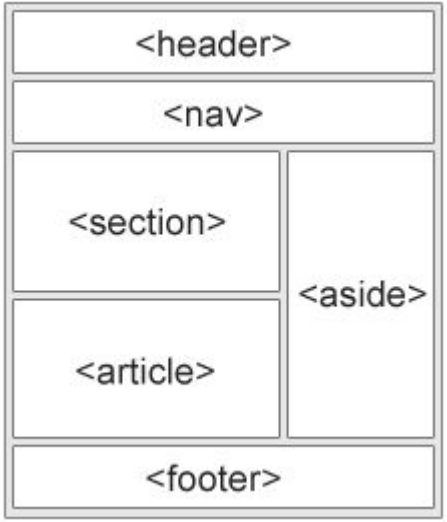
Exemplos de elementos semânticos: `<form>`, `<table>` e `<article>` - define claramente seu conteúdo.

Elementos Semanticos 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.

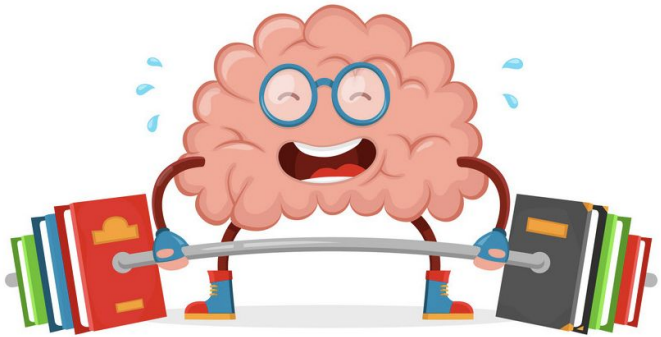
Um elemento <header> normalmente contém:

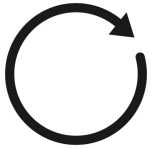
- um ou mais elementos de título (<h1> - <h6>)
- logotipo ou ícone
- informação de autoria

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>.

Exercícios : Header semântico

Faça exercícios propostos pelo professor.



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.

Alguns exemplos: https://html5boilerplate.com/

https://meyerweb.com/eric/tools/css/reset/

Block vs Inline

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

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 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;
}
```

Capítulo 4 - Mais HTML e CSS

Formulários

Um formulário HTML é usado para coletar a entrada do usuário. A entrada do usuário geralmente é enviada a um servidor para processamento.

O elemento HTML <form> é usado para criar um formulário HTML para entrada do usuário:

```
<form>
.
form elements
.
</form>
```

O elemento <form> é um contêiner para diferentes tipos de elementos de entrada, como: campos de texto, caixas de seleção, botões de opção, botões de envio, etc.

serratec

Cascata e Herança no CSS

Algumas propriedades de elementos pais, quando alteradas, são aplicadas automaticamente para seus elementos filhos em cascata. Por exemplo:

```
<div id="pai">
    <h1>Sou um título</h1>
    <h2>Sou um subtítulo</h2>
</div>
#pai {
    color: blue;
}
```

As propriedades que não são aplicadas em elementos filhos são as que afetam diretamente a caixa(box) do elemento, como width, height, margin e padding

No exemplo acima, todos os elementos filhos herdaram a propriedade color do elemento pai a qual pertencem.

Exercícios: Menu e destaque

Faça exercícios propostos pelo professor.

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.

Exercícios: Painéis Flutuantes

Faça exercícios propostos pelo professor.

Seletores de Atributo do CSS3

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

O exemplo a seguir seleciona todos os elementos <a> com um atributo de destino:

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

Vamos olhar o exemplo no link para termos uma melhor ideia do que podemos fazer com os seletores de atributos

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

Exercícios: Rodapé

Faça exercícios propostos pelo professor.

CSS Avançado - Capítulo 5

Seletores Avançados

Os seletores CSS mais comuns são os que vimos anteriormente: por ID, classes e por descendência.

No entanto, há muitos outros seletores novos que são bastante úteis. Os exemplos que veremos são:

- Seletor de irmãos
- Seletor de irmão adjacente
- Seletor de filho direto
- Negação

Pseudo Classes

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

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

Pseudo Classes - Sintaxe

A sintaxe das pseudo-classes é a seguinte:

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

Vamos olhar os exemplos da w3schools (link no título) dos tipos de pseudo-classes possíveis.

Pseudo Elementos

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

Pode ser usado em, por exemplo:

- Estilizar a primeira letra ou linha de um elemento
- Inserir conteúdo antes ou depois do conteúdo de um elemento

Pseudo Elementos - Sintaxe

A sintaxe das pseudo-classes é a seguinte :

```
seletor::pseudo-elemento {

  propriedade: valor;

}
```

Vamos olhar os exemplos da w3schools (link no título) dos tipos de pseudo-elementos possíveis.

Exercícios: Seletores, Pseudo Classes e Elementos

Faça exercícios propostos pelo professor.

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

As 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;
}
```

Google Fonts

Um dos elementos visuais que mais se destaca em um Site ou WebApp é a fonte do sistema.
Pensando nisso, muitos sites escolhem bem suas fonts para que elas combinem com o layout da aplicação.

Diante disso, ai que entra o Google fonts, lá podemos encontrar vários tipos de fontes de forma gratuita, para podermos utilizar via HTML ou CSS.

Aqui você pode escolher uma infinidade de fontes.

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)

Gradientes

Os gradientes CSS permitem que você exiba transições suaves entre duas ou mais cores especificadas.

O CSS define dois tipos de gradientes:

- Linear Gradients (orientado de baixo paracima, esquerda para a direita, diagonalmente)
- Radial Gradients (definido pelo centro)

```
#grad {
  background-image: linear-gradient(red, yellow);
}
```

Exercícios: Visual CSS3

Faça exercícios propostos pelo professor.

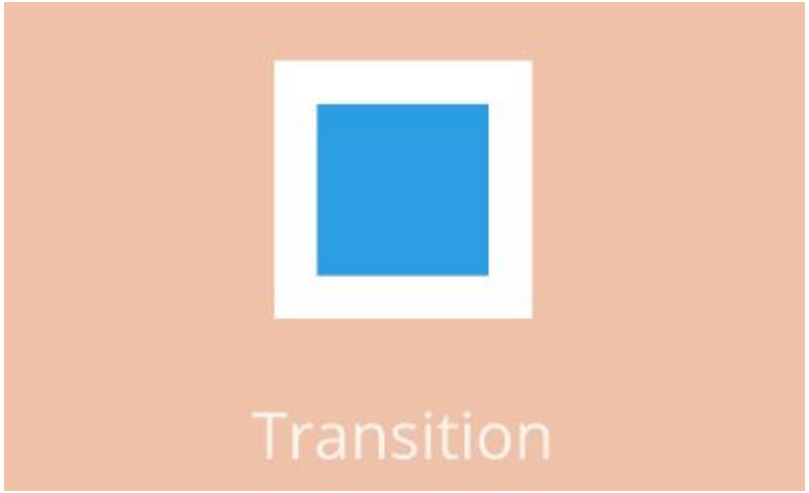
CSS: Transições

As transições CSS permitem que você altere os valores das propriedades suavemente, durante um determinado período.

Principais propriedades de transições

(que veremos nos exemplos práticos):

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

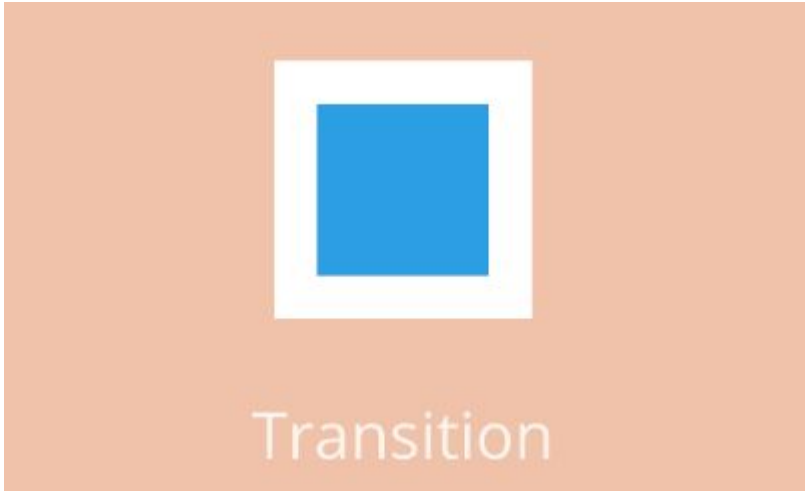


CSS3: Transformações

A propriedade transform aplica uma transformação 2D ou 3D a um elemento. Esta propriedade permite girar, dimensionar, mover, inclinar, etc., elementos.

Exemplos :

```
div.a {
  transform: rotate(20deg);
}
div.b {
  transform: skewY(20deg);
}
```



Exercícios: Transform e Transition

Faça exercícios propostos pelo professor.

Web para dispositivos móveis - Capítulo 6

serratec

CSS 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 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

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.

Viewport

A Apple criou então, uma solução que depois foi copiada para outros smartphones, que é configurar o valor que julgamos mais adequado para o viewport:

```
<meta name="viewport" content="width=320">
```

ou

```
<meta name="viewport" content="width=device-width">
```

Exercícios: Adaptações para Mobile

Faça exercícios propostos pelo professor.

Bootstrap e Fomulários HTML5

serratec

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

Podemos também utilizar via CDN:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
```

Vantagem em utilizar o CDN:

Muitos usuários já baixaram o Bootstrap 4 do jsDelivr ao visitar outro site. Como resultado, ele será carregado do cache quando eles visitarem seu site, o que leva a um tempo de carregamento mais rápido. Além disso, a maioria dos CDNs garante que, uma vez que um usuário solicite um arquivo, ele será servido a partir do servidor mais próximo a eles, o que também leva a um tempo de carregamento mais rápido.

Exercícios: Página de Checkout

Faça exercícios propostos pelo professor.

Mais Sobre Formulários

Vimos superficialmente a estrutura de um formulário em HTML. Vamos dar mais atenção aos seus atributos agora que já avançamos mais em nosso projeto.

Formulários: Elemento <input>

O elemento HTML <input> é o elemento de formulário mais usado.

Um elemento <input> pode ser exibido de várias maneiras, dependendo do atributo type. Alguns exemplos:

- <input type = “text”> : Exibe um campo de entrada de texto de linha única
- <input type="radio"> : Exibe um botão de opção (para selecionar uma das muitas opções)
- <input type="checkbox"> : Exibe uma caixa de seleção (zero ou mais de muitas opções)
- <input type="submit"> : Exibe um botão de envio (para enviar o formulário)
- <input type="button"> : Exibe um botão clicável

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.

Formulários: O atributo action

O 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. Por exemplo:

```
<form action="/pagina_destino.php">
```

Se o atributo de ação for omitido, a ação será definida para a página atual.

Observação: Usaremos o Javascript ao invés de usar o atributo action.

Exercícios : Formulários

Faça exercícios propostos pelo professor.

Validação HTML5 : Required e Pattern

O atributo **required** é um atributo booleano. Quando presente, especifica 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="usrname" 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>
```

Exercícios: Validação com HTML5

Faça exercícios propostos pelo professor.

Grid Responsivo do Bootstrap

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

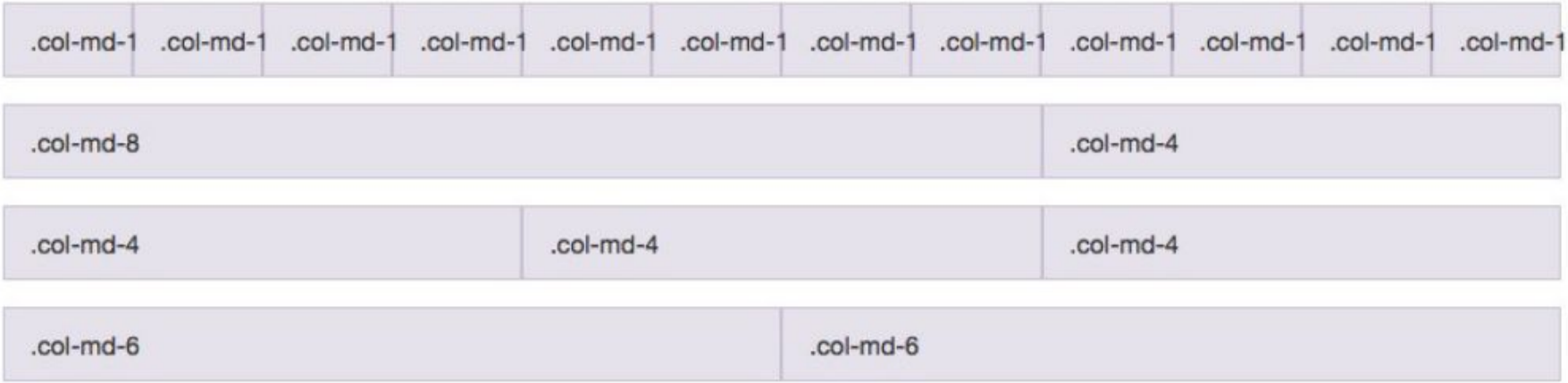
A solução mais comum é o uso de grids, onde divide-se a tela em colunas e os elementos vão sendo encaixados dentro dessas colunas.

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:



Grid System

O gridsystem do bootstrap é uma mão na roda, ajuda a organizar layouts de maneira muito simples e intuitiva.
Paginas que demorariam horas para posicionar elementos, com ele podemos fazer em minutos.

Vamos ver alguns exemplos aqui.

Exercícios: Grid

Faça exercícios propostos pelo professor.

Buttons

Atualmente muitos sites utilizam os layouts dos botões bootstrap, com isso, eles são uma aprender a utilizá-los poderá ser uma ótima ideia.

Vamos ver alguns exemplos aqui.

Modal

O componente Modal é uma caixa de diálogo/janela pop-up exibida na parte superior da página atual.
Todo modal pode ser chamado direto pelo HTML ou por uma função javascript.

Vamos ver alguns exemplos aqui.

Tabelas

Um dos componentes mais bonitos do Bootstrap são as tabelas.
Temos várias classes que nos permitem criar tabelas com um layout bem agradável para o usuário.

Vamos ver alguns exemplos aqui.

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Bootstrap Input Groups

O Bootstrap tem uma série de grupos de input já estilizados que podemos utilizar em nossos projetos.

Vamos dar uma olhada?
Clique aqui.

Bootstrap Form Inputs

O Bootstrap ajuda muito na criação de formulários, pois tem vários tipos de inputs, selects, textarea e outros que podemos usar de maneira bem simples.

Vamos dar uma olhada?
Clique aqui.

Exercícios: Elementos Bootstrap

Faça exercícios propostos pelo professor.

Capítulo 9 - Javascript e Interatividade na Web

serratec

Por que usamos Javascript?

JavaScript é uma das 3 linguagens que todos os desenvolvedores da web devem aprender:

1. Enquanto o HTML é usado para definir o conteúdo das páginas da web
2. E o CSS para especificar o layout das páginas da web
3. O JavaScript é a linguagem responsável pela programação do comportamento das páginas da web

Os usos comuns do JavaScript são manipulação de imagens, validação de formulários e mudanças dinâmicas de conteúdo.

Características da linguagem

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.

Características da linguagem

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.

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 Contrl + 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>
document.getElementById("demo").innerHTML = "Olá mundo!";
</script>
```

A tag script

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

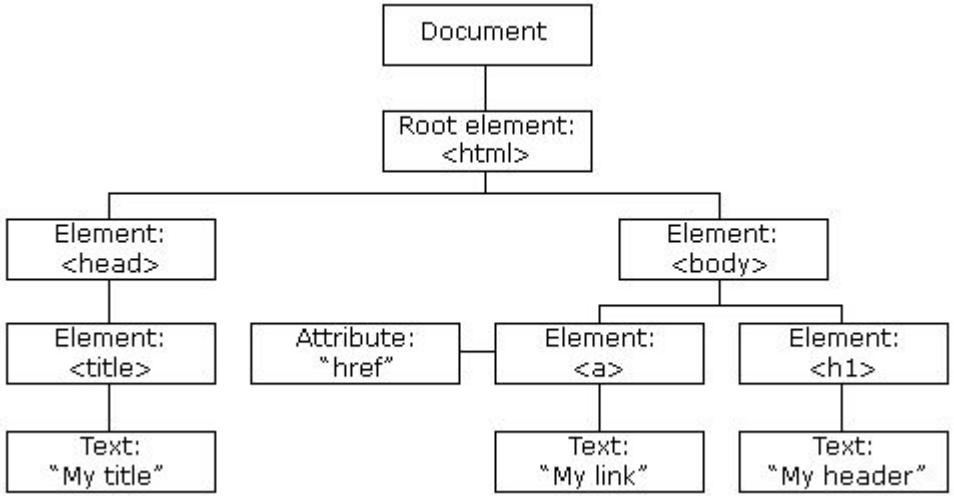
O elemento `<script>` contém instruções de script ou aponta para um arquivo de script externo por meio do atributo src.

```
<script>
document.getElementById("demo").innerHTML = "Olá mundo!";
</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 a 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

Exercícios: Mostrando exemplos Javascript

Funções

Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.

Uma função JavaScript é executada quando "algo" a invoca (chama).

Exemplo:

```
function multiplicar(p1, p2) {
  return p1 * p2;   // A função retornará o valor de P1 * P2
}
```

Funções Anônimas

Em muitos casos, onde não há um outra parte do código onde queremos referenciar uma função e ela será apenas referenciando-a 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:

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

Este e outros exemplos pode ser encontrados aqui.

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 empresa = "Caelum";
empresa.length; // tamanho da string
empresa.replace("lum","tano"); // retorna Caetano
```

Note que o valor de empresa não foi modificado. Teste chamar empresa após o replace.

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);
var textoFloat = "10.22";
var float = parseFloat(textoFloat);
```

Existem várias funções para manipular Strings, clique aqui para conhecê-las.

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"
```

Existem várias funções para manipular Números, clique aqui para conhecê-las.

Exercícios: Manipulando Strings e Funções

Faça exercícios propostos pelo professor.

Laços de Repetição e Condicionais

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

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

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

```
if (condicao) {
  //  código a ser executado se condiçao for verdadeira
}
```

Outros laços muito comuns em javascript são: For in e For of

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:

```
// executa a minhaFuncao daqui um segundo
setTimeout(minhaFuncao, 1000);
```

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:

```
// executa a minhaFuncao de um em um segundo
setInterval(minhaFuncao, 1000);
```

É uma função útil para, por exemplo, implementar um cronometro, apresentado no exercício a seguir.

Exercícios: Cronômetro

Faça um sistema que tenha um cronômetro ao meio da tela.
Esse sistema deve ter um botão para iniciar, pausar e parar.
O botão pausar só para o cronômetro, já o botão parar, ele pausa e zera o cronometro.

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 = ["W3Scools", 10, [1,2]];
```

Para recuperar um valor, basta referenciar o vetor e sua respectiva posição

```
console.log(variosTipos[1]) // imprime o número 10
```

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 = ["W3scools", "Ensino"];
palavras.push("Inovação"); // adiciona a string "Inovação" por último no array
palavras[9] = "Criatividade";
```

Arrays em Javascript são semelhantes às ArrayLists em Java

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 = ["W3scools", "Ensino"];
palavras.push("Inovação"); // adiciona a string "Inovação" por último no array
palavras[9] = "Criatividade";
```

Arrays em Javascript são semelhantes às ArrayLists em Java

Metodos de Arrays

Os métodos em arrays são muito comuns, pois o Array é um tipo de objeto que é muito utilizado no javascript, por isso é muito importante conhecer os principais métodos de array como: **map**, **filter**, **find**, **reduce** e outros.

Clique aqui para conhecer os métodos-base.
Clique aqui para conhecer os métodos de interação.

Observação: Quando mais você souber sobre os métodos de array, mas ficará fácil evoluir em frontend de forma geral. Principalmente os métodos: **map**, **filter**, **find, findIndex, splice** e **reduce**.

Objetos em JavaScript

Assim como aprendemos em **Java** a utilizar objetos, podemos fazer o mesmo em JavaScript.
Objetos no javascript são muito uteis para representar informações reais.
Exemplo de um objeto pessoa:

```
var pessoa = {
        nome: ‘Fulano’,
        Idade: 18,
        casado: false,
        Preferencias: [‘Game’, ‘Série’, ‘Musica’, ‘Esportes’]
}
```

Objetos em JavaScript

Em JavaScript os objetos são reis, se você entende objetos, você entende JavaScript.

Em JavaScript, quase "tudo" é um objeto.

- Booleanos podem ser objetos (se definidos com a palavra - new chave)
- Os números podem ser objetos (se definidos com a palavra - new chave)
- Strings podem ser objetos (se definidos com a palavra - new chave)
- Datas são sempre objetos
- Matemática são sempre objetos
- Expressões regulares são sempre objetos
- Arrays são sempre objetos
- Funções são sempre objetos
- Objetos são sempre objetos

Todos os valores JavaScript, exceto primitivos, são objetos.

Arrow Function

As Arrow Functions (funções de seta ou flexa) foram introduzidas no ES6.

As funções de seta nos permitem escrever uma sintaxe de função mais curta:

```
var multiplicar = (a, b) => a * b;
```

Vamos aprender um pouco mais, clique aqui para olhar alguns exemplos.

Observação: Esse tipo de função é muito comum no dia a dia, você verá muitos casos utilizando esse tipo de função, então pratique bem como utilizar essas funções.

Classes

O ECMAScript 2015, também conhecido como ES6, introduziu as classes JavaScript.

Classes JavaScript são modelos para objetos JavaScript.
Exemplo:

```
let myCar1 = new Car("Ford", 2014);

let myCar2 = new Car("Audi", 2019);
```

Clique aqui para conhecer as classes.

JSON

JSON é um formato para armazenar e transportar dados. Ele é frequentemente usado quando os dados são enviados de um servidor para uma página da web.

- JSON significa Java **S** cript **O** bject **N** otation
- JSON é um formato de intercâmbio de dados leve
- JSON é independente de idioma *
- JSON é "autodescritivo" e fácil de entender

* A sintaxe JSON é derivada da sintaxe de notação de objeto JavaScript, mas o formato JSON é somente texto. O código para leitura e geração de dados JSON pode ser escrito em qualquer linguagem de programação.

JavaScript Promises

"Eu prometo um resultado!"

"Produzir código" é um código que pode levar algum tempo

"Consumindo código" é o código que deve aguardar o resultado

Uma promessa é um objeto JavaScript que vincula a produção de código e o código de consumo.

As Promises são códigos que prometem fazer algo, mas não garantem que vão conseguir.
Geralmente utilizamos elas para consumir alguma API.

Fetch API

A função fetch() foi introduzida do JavaScript com intuito de acessar e manipular requisições HTTP.

Usaremos ela para integrar com APIs através do famoso CRUD. (GET, POST, PUT e DELETE

Veremos alguns exemplos aqui.

Async / Await

As palavras-chave async e await, implementadas a partir do ES2017, são uma sintaxe que simplifica a programação assíncrona, facilitando o fluxo de escrita e leitura do código; assim é possível escrever código que funciona de forma assíncrona, porém é lido e estruturado de forma síncrona. O async/await também trabalha com o código assíncrono baseado em Promises, porém esconde as promessas para que a leitura e a escrita seja mais fluída.

Segue um artigo muito legal sobre. Artigo

Exercícios: Consumir APIs através de Promises

Faça exercícios propostos pelo professor.

The end!

Finalizamos mais uma disciplina.
Continue estudando e ganhando conhecimento para crescer profissionalmente.

Um abraço e até a próxima.

serratec
Parque Tecnológico Região Serrana