

# Aula – 1

## Apresentação

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Conteúdo Programático
- ❑ Cronograma
- ❑ Avaliação
- ❑ Ambiente de Programação

# **Conteúdo Programático**



# O que está na ementa?

Arquitetura Web. HTML e CSS. JavaScript. Introdução aos frameworks Javascript para front-end. Introdução à programação server-side em Javascript. Persistência de dados na Web.

**COM222 e XDES03 possuem equivalência**

**HTML**

**HTML**



**HTML diz ao navegador onde e  
quais elementos estão na tela:**

- Botões
- Formulários
- Tabelas
- Títulos
- Cabeçalhos
- Rodapé



**HTML recomenda pensarmos no significado do elemento ou na área da página.**

- Barra de Navegação <nav>
- Cabeçalho <header>
- Seções <section>
- Rodapé <footer>
- Parte principal <main>





## Por que preocupar com semântica?

- Manutenção
- Legibilidade
- Organização
- Aprimora acessibilidade

CSS

CSS



# CSS diz ao navegador como os elementos serão exibidos. Estilização

- Cor
- Margens
- Animações
- Cor de fundo
- Fonte
- Ordem



# Como aplicamos a estilização?

css

- Criando regras
- Seletores
- Condição de corrida
- Modelo de Caixa (Box Model)
- Flexbox



# JavaScript

# JS

**JS permite adicionar comportamento nos elementos exibidos no navegador.**

- Tratamento de Eventos
- Clique em botões
- Envio/Recebimento de requisições
- Submissão de formulários
- Tratamento de erros
- Manipular o DOM

JS

# **É essencial compreender como funciona a programação assíncrona com JS**

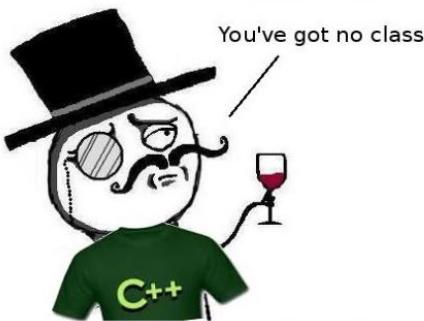
- Promise
- Async/Await
- Callbacks



JS

# JS e a Programação Orientada a Objetos

- ❑ Prototypes
- ❑ Classes em JS



# JS

# Navegador



**Navegador é capaz  
de exibir páginas  
web interpretando  
instruções HTML,  
CSS e JavaScript**



# NodeJS



**NodeJS é um ambiente de execução (*runtime environment*) que permite a execução de código JS fora do navegador.**



- Criar código do lado do servidor
- Popular com boa empregabilidade
- Possui Gerenciador de versões – NVM  
*(node version manager)*

# NPM



**NPM é um gerenciador de pacotes para projetos baseado em NodeJS. Permite a automatizar a *build* (construção do programa)**

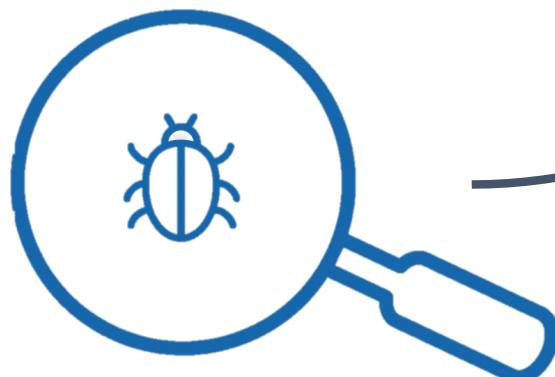
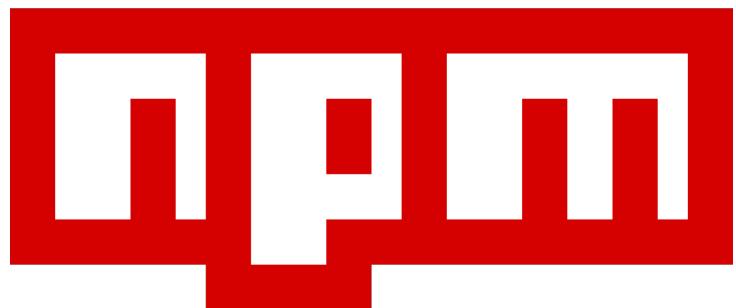
- Gerencia as dependências
- Executa os testes automatizados
- Gera o pacote executável
- Projetos NodeJS podem conter diversas dependências.

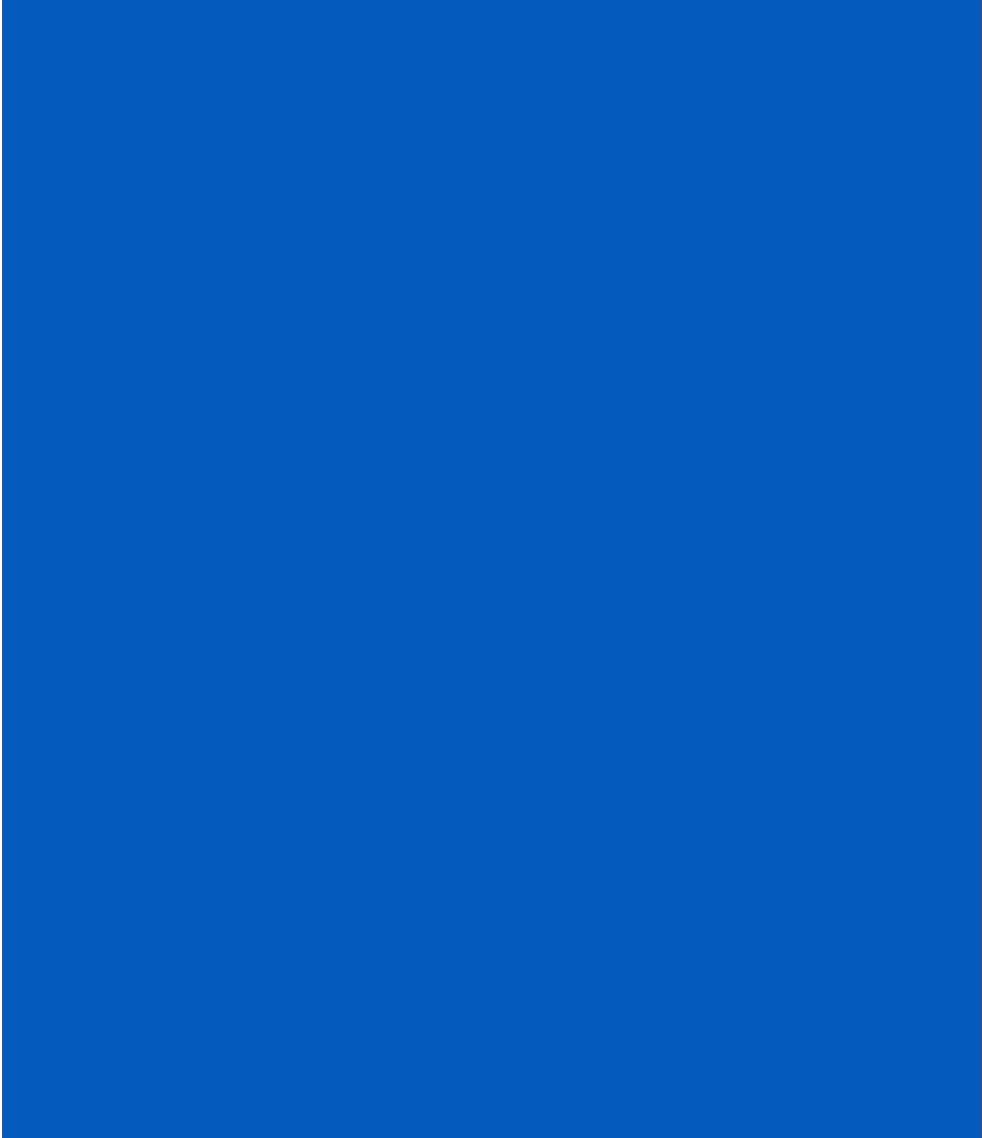
**Para termos um código como um produto não basta apenas a compilação. Precisamos construir (build). Geralmente envolve:**

- Compilar
- Executar testes automatizados
- Empacotar com todas as dependências
- Gerar o Executável



BUILDER





Express 

# Framework para facilitar a criação de aplicações web do lado do servidor. Backend.

- ❑ Criação de rotas (*endpoints*)
- ❑ Gerenciar requisições HTTP  
(HyperText Transfer Protocol)

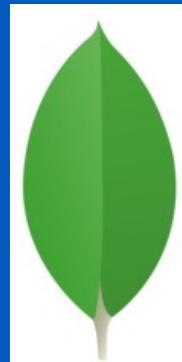




**Protocolo de comunicação para a troca de hipertexto. Padrão na comunicação de páginas e conteúdo Web. Baseado nos seguintes verbos:**

- GET -> buscar no servidor
- POST -> colocar no servidor
- DELETE -> apagar no servidor
- PUT -> atualizar no servidor

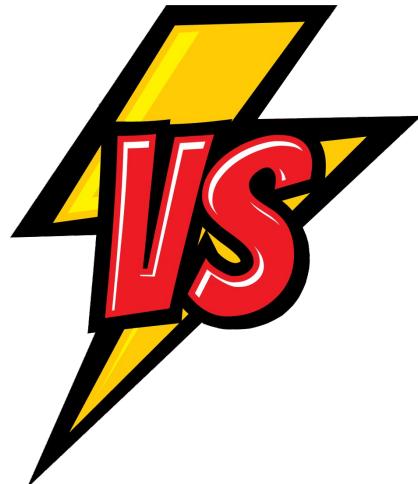




mongoDB

# Banco de dados NoSQL baseado em documentos JSON (JavaScript Object Notation).

```
[  
  {  
    "id": 1,  
    "nome": "Maria",  
    "idade": 18  
  },  
  {  
    "id": 2,  
    "nome": "Jaum",  
    "idade": 19  
  }  
]
```



Id	Nome	idade
1	Maria	18
2	Jaum	19



**mongoose**

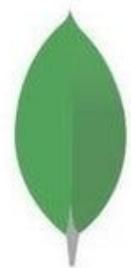


**É um ODM (Object Data Mapper).  
Facilita a conexão e mapeamento de  
dados entre código JS e o MongoDB**

```
[  
  {  
    "id": 1,  
    "nome": "Maria",  
    "idade": 18  
  },  
  {  
    "id": 2,  
    "nome": "Jaum",  
    "idade": 19  
  }  
]
```



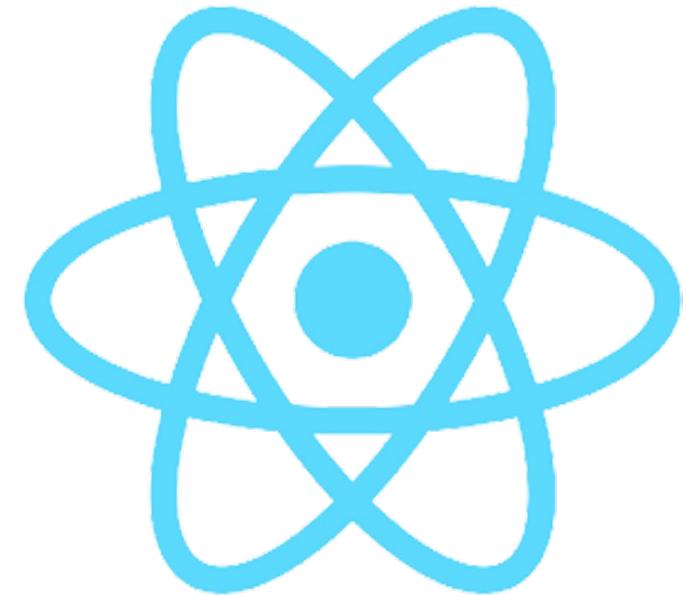
Id	Nome	idade
1	Maria	18
2	Jaum	19



**mongoDB**<sup>®</sup>  
Atlas

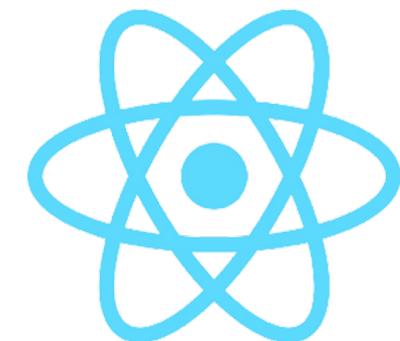
**Para adiantar, criem um conta no serviço MongoDB Atlas. É de graça e sem custo adicional.**

□ <https://www.mongodb.com/atlas>



# **Biblioteca JS para facilitar a construção de aplicações web do lado do cliente. Frontend**

- Utiliza NPM para gerenciar as dependências
- Leve
- Pode requerer a instalação de diversas outras dependências



**React**

# Avaliação

## Divisão de notas

---

- $N1 = \text{Exercícios} * 0,4 + \text{Prova} * 0,6$
- $N2 = \text{Exercícios} * 0,2 + \text{Projeto} * 0,8$

- Nota final =  $(N1 + N2)/2$





Nota Final  $\geq$  6 😊

Nota Final  $<$  6 😞  
**Substitutiva**

## Prova Substitutiva

---

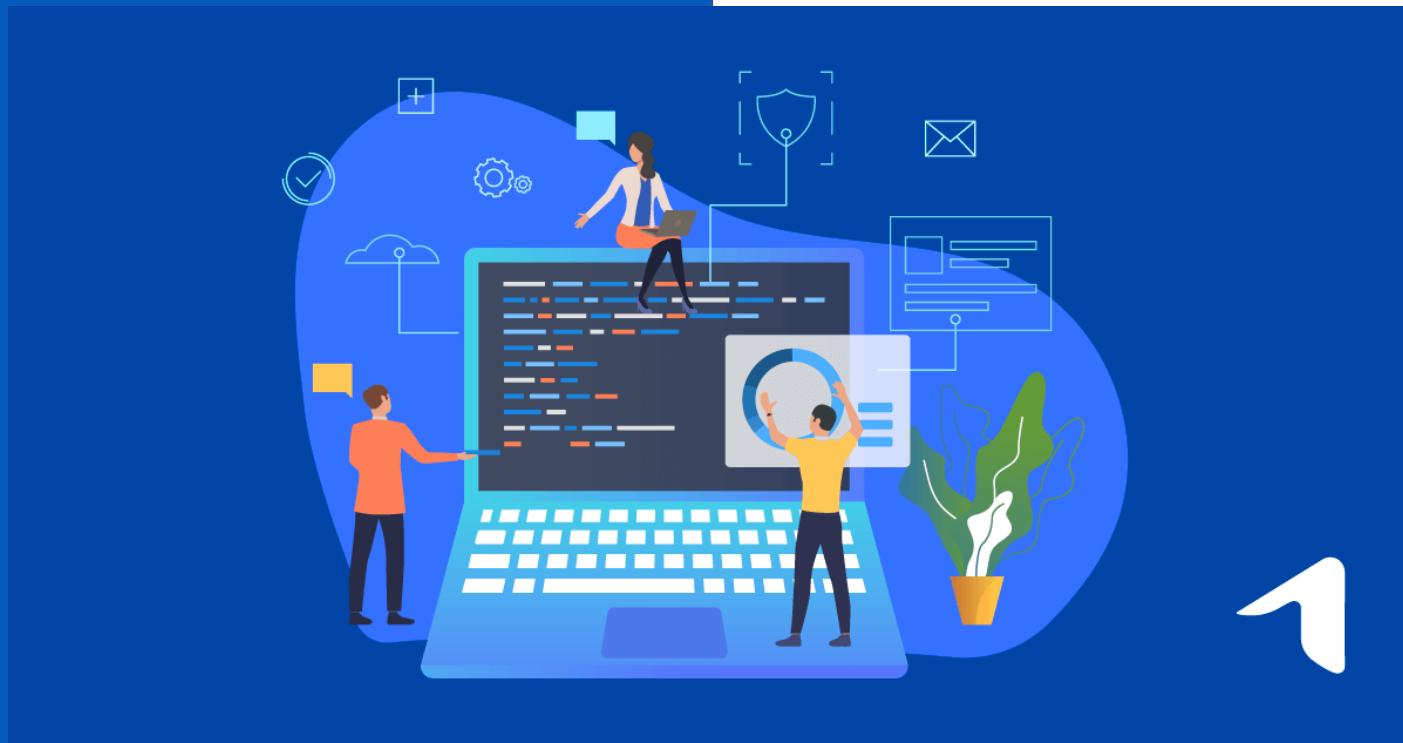
- A substitutiva irá substituir a nota N1 ou N2, a que for menor
- Prova prática
- Sério!!!!!!

## Exercícios Práticos

---

- Todos os exercícios deverão ser entregues durante o horário de aula e somente serão validados se o discente estiver presente.
- As exceções serão informadas explicitamente.
- Não pode entregar da praia





# Critérios do Projeto Final – Parte 1

---

- ❑ Obrigatoriamente uma aplicação web.
- ❑ Utilizar framework frontend da escolha do grupo
  - ❑ React, Angular, Vue,...
- ❑ Utilizar framework backend da escolha do grupo
  - ❑ ExpressJS, SpringBoot, ASP.NET, Flask,...
- ❑ Poderá utilizar framework de estilização
  - ❑ Bootstrap
- ❑ Comunicação front-back através de verbos HTTP

## Critérios do Projeto Final – Parte 2

---

- ❑ CRUD completo com um banco de dados
  - ❑ MongoDB, MySQL, PostgreSQL, Voldemort
  - ❑ Não é necessário utilizar banco na nuvem
- ❑ Versionado em algum repositório git público
  - ❑ GitHub, GitLab, BitBucket
  - ❑ Documentação com README “no capricho”

## Critérios do Projeto Final – Parte 3

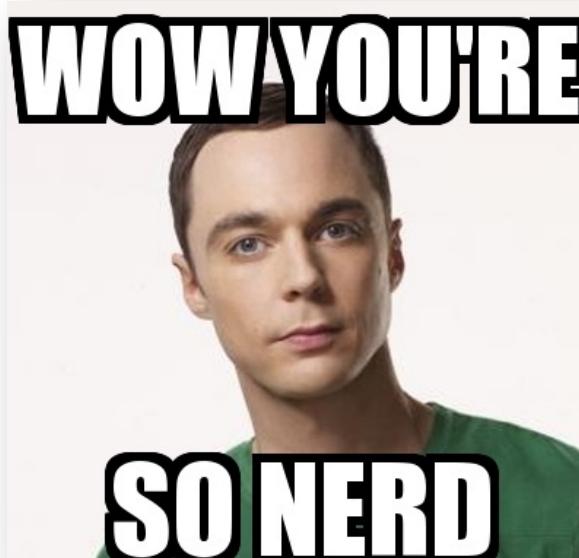
---

- ❑ Documentação do negócio:
  - ❑ Quais são as regras do negócio?
  - ❑ Qual problema que resolve?
  - ❑ Por que é um problema importante?

# exception



Posso fazer um jogo?



# Critérios para Jogos – Parte 1

---

- ❑ Obrigatoriamente utilizar um framework para desenvolvimento de jogos que **seja web**.
- ❑ Em outras palavras, é necessário ser baseado em HTML5/JS
- ❑ **Não será permitido** utilizar **game engines** que não sejam web, mas que possibilitam gerar um executável WebGL.
  - ❑ Por exemplo o Unity

## Critérios para Jogos – Parte 2

---

- ❑ CRUD e conexão com banco de dados permanecem os mesmos critérios para aplicação web.
- ❑ Na prática todo os critérios de ***backend*** permanecem inalterados.
- ❑ O ***frontend*** que poderá se adaptar para um jogo.

The image features a vibrant, space-themed advertisement for the Chaser game framework. On the left, two cartoonish aliens are depicted: one orange with a purple hood and a sword-like weapon, and another green with multiple antennae and a large yellow eye. They are positioned behind the word "CHASER", which is written in large, blue, blocky letters that appear to be made of ice or crystal. The background is a dark purple gradient with a bright, glowing light source on the left. In the bottom left corner, there is an orange shield-shaped icon with the text "HTML 5" and a large white number "5".

**Desktop and  
Mobile HTML5  
game framework**

A fast, free and fun open source  
framework supporting both  
JavaScript and TypeScript.

**DOWNLOAD & GET STARTED**  
Download or Fork via Github

VER  
2.0

# Apresentação do Projeto

---

- As apresentações ocorrerão nos dias 11/07 e 14/07
- 15 min de apresentação + 5 min de perguntas
- Grupos de 3 a 5 alunos
- Mínimo de 10 grupos
- Ajustes serão feitos para adequar a 10 grupos

# Avaliação do Projeto – Parte 1

---

## Nota individual

- Na entrega é essencial deixar claro as contribuições de cada membro.
- Os membros precisam ter o conhecimento mínimo do projeto todo, mesmo que não tenha sido o foco.
- Poderão ocorrer perguntas individuais
- Os membros precisão contribuir com ***commits***

## Avaliação do Projeto – Parte 2

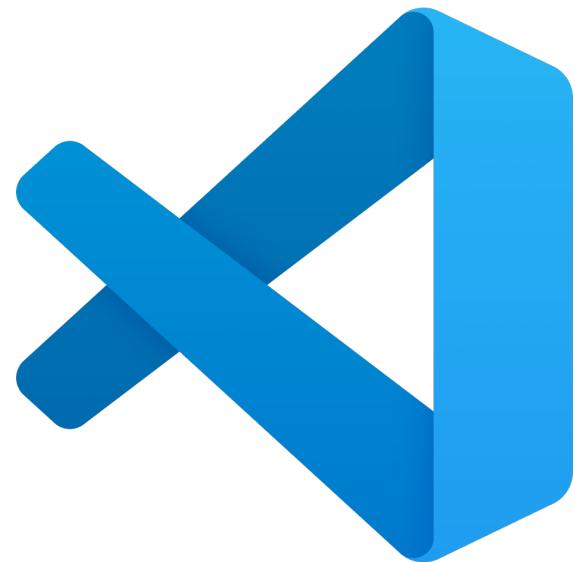
---

- Os critérios técnicos irão compor 75% da nota
- A postura na apresentação, seriedade, comprometimento, capricho e organização irão compor 25% da nota.

# Ambiente de Programação

# Visual Studio Code e Terminal UNIX

---

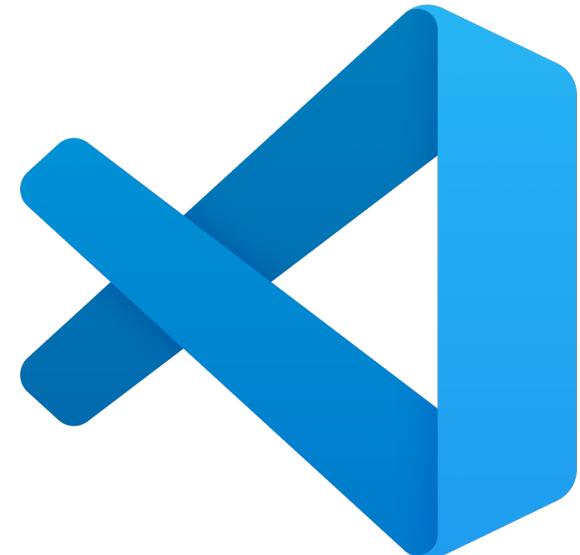
A screenshot of a terminal window titled 'phillima@DESKTOP-CM6Q9KD'. The window shows the following text:

```
phillima@DESKTOP-CM6Q9KD:/mnt/c/WINDOWS/system32$ 0 que eu to fazendo aqui?  
0: command not found  
phillima@DESKTOP-CM6Q9KD:/mnt/c/WINDOWS/system32$
```

# Visual Studio Code

---

- ❑ Editor de texto da Microsoft
- ❑ Popular para o desenvolvimento web
- ❑ Não confundir com Visual Studio, a IDE também da Microsoft

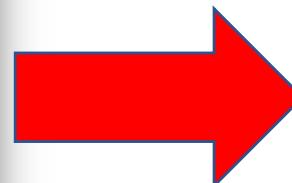


# UNIX

---

- ❑ Executar comandos UNIX básicos no terminal
- ❑ Navegar por pastas, compilação, execução

```
phillima@DESKTOP-CM6Q9KD:/mnt/c/WINDOWS/system32$ O que eu to fazendo aqui?  
O: command not found  
phillima@DESKTOP-CM6Q9KD:/mnt/c/WINDOWS/system32$
```



## Unix - História

---

- Sistema Operacional Desenvolvimento  
Originalmente no *Bell Labs* pela *at&t* da década de 1960.



at&t

## Unix - História

---

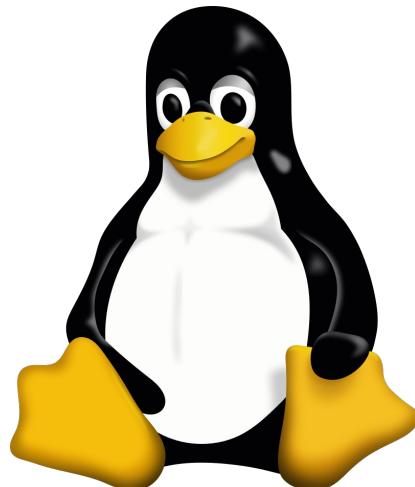
- A linguagem C foi utilizada para reescrever o sistema operacional Unix.
- Se tornou um dos sistemas mais populares.
- Popularizado por Dennis Ritchie e Ken Thompson.



# Unix-Like

---

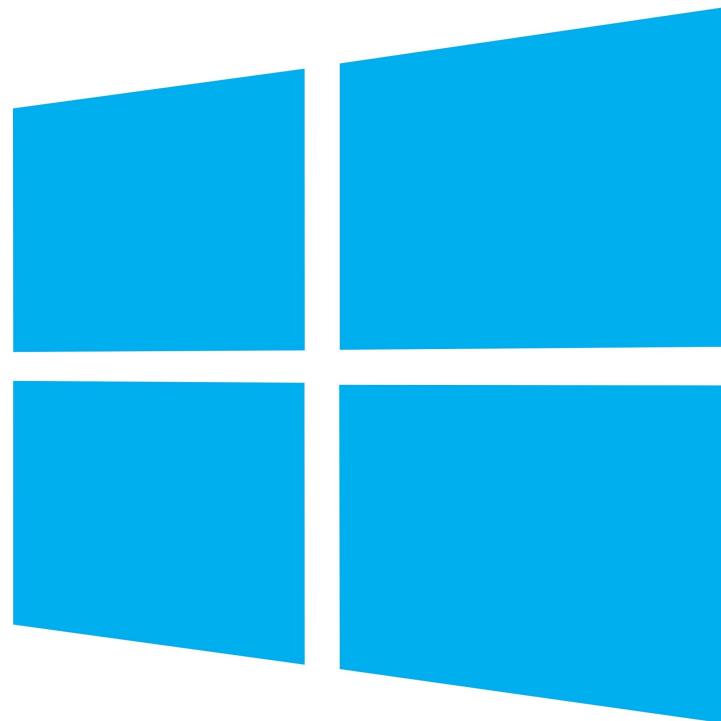
- Dada sua popularidade, passou a ser o sistema operacional favorito e foi adotado como parte de outros sistemas, dando origem ao *Unix-like*



# Unix-Like

---

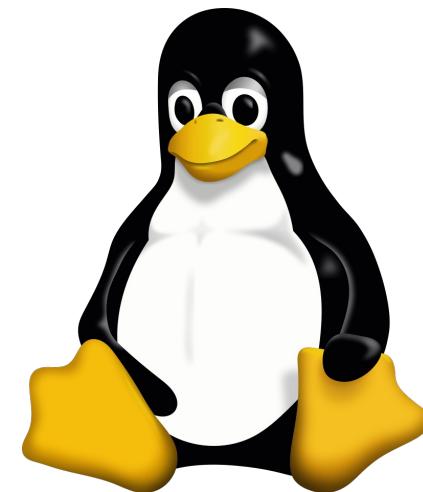
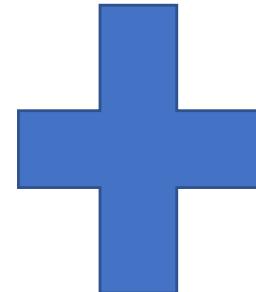
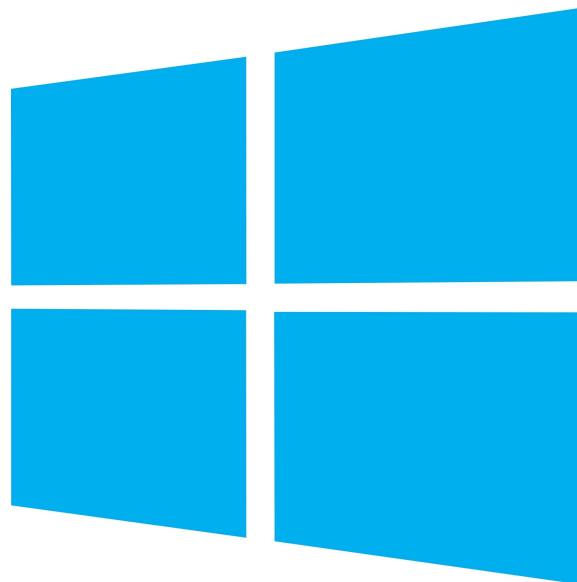
❑ E o Windows? ☹

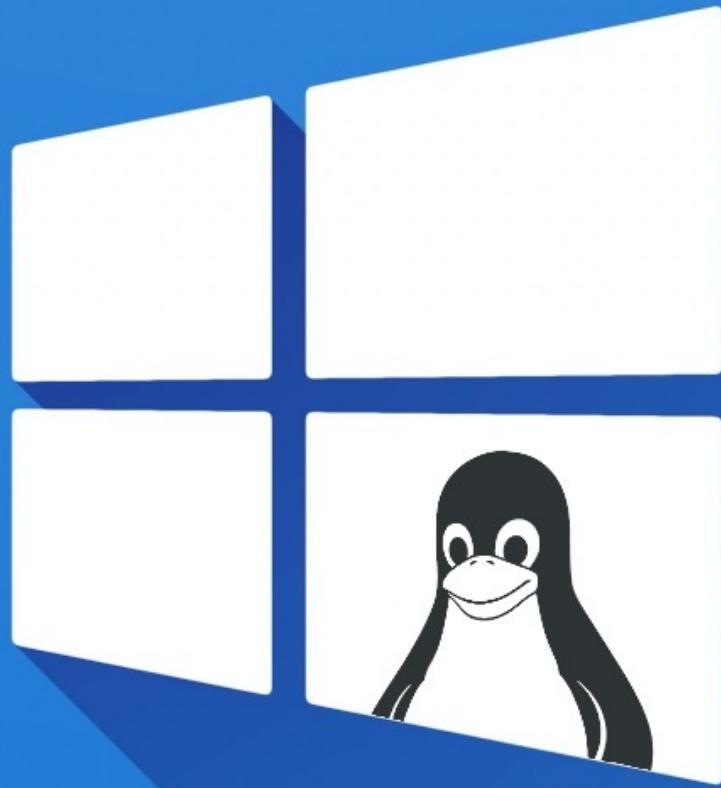


# Unix-Like

---

☐ WSL (Windows Subsystem Linux) para **nos salvar!**





phillima@DESKTOP-CM6Q9KD: /mnt/c/WINDOWS/system32



Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86\_64)

\* Documentation: <https://help.ubuntu.com>  
\* Management: <https://landscape.canonical.com>  
\* Support: <https://ubuntu.com/advantage>

System information as of Thu Aug 25 10:08:03 -03 2022

System load: 0.0 Processes: 8  
Usage of /: 1.3% of 250.98GB Users logged in: 0  
Memory usage: 0% IPv4 address for eth0: 192.168.28.234  
Swap usage: 0%

0 updates can be applied immediately.

This message is shown once a day. To disable it please create the  
/home/phillima/.hushlogin file.

phillima@DESKTOP-CM6Q9KD:/mnt/c/WINDOWS/system32\$ LINUX DENTRU DU UIINDOUS UHUUUUUUUUU

# Motivação

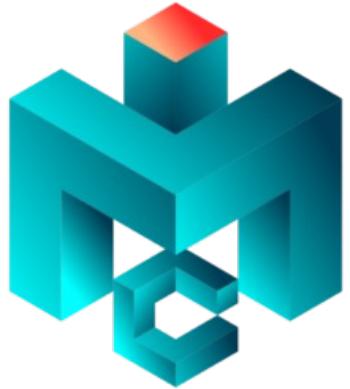
## Por que fazer COM222/XDES03?

---

- ❑ Alta empregabilidade.
- ❑ Tudo está na Web e poder ser acessado como serviço.
- ❑ Você precisa ser aprovado para formar e ser feliz.

Praise the sun!





# Aula – 2

## Introdução a Web

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

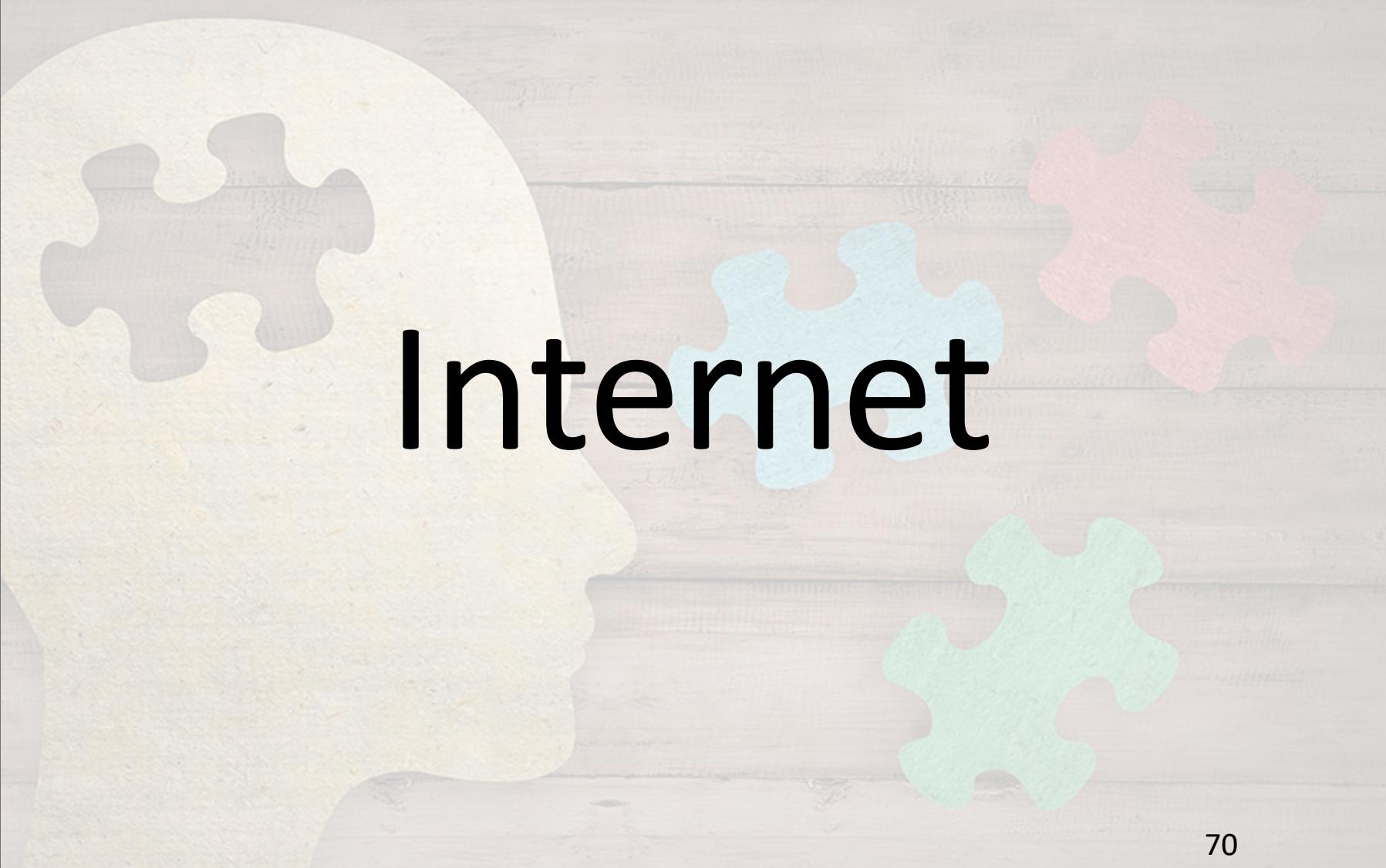
Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ O que é a Internet
- ❑ O que é a Web
- ❑ O ciclo Requisição e Resposta
- ❑ A Trinca HTML - CSS - JavaScript



# Internet

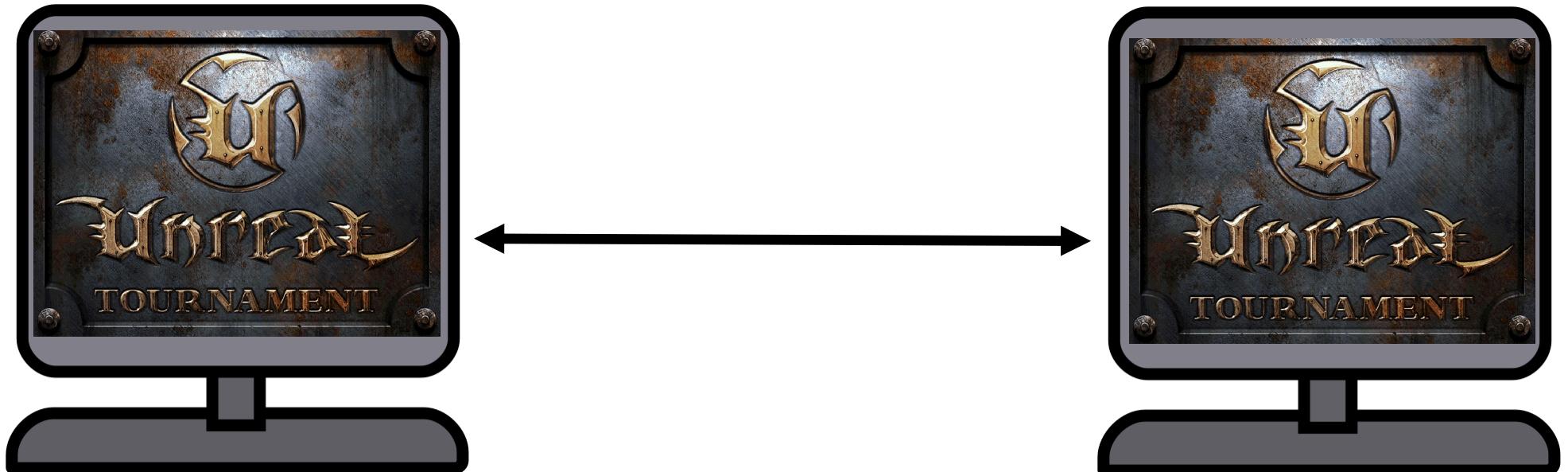


## O que é a Internet?

*“Uma rede de redes de computadores”*

(uma quantidade grande demais da conta de  
computadores conectados...sô)

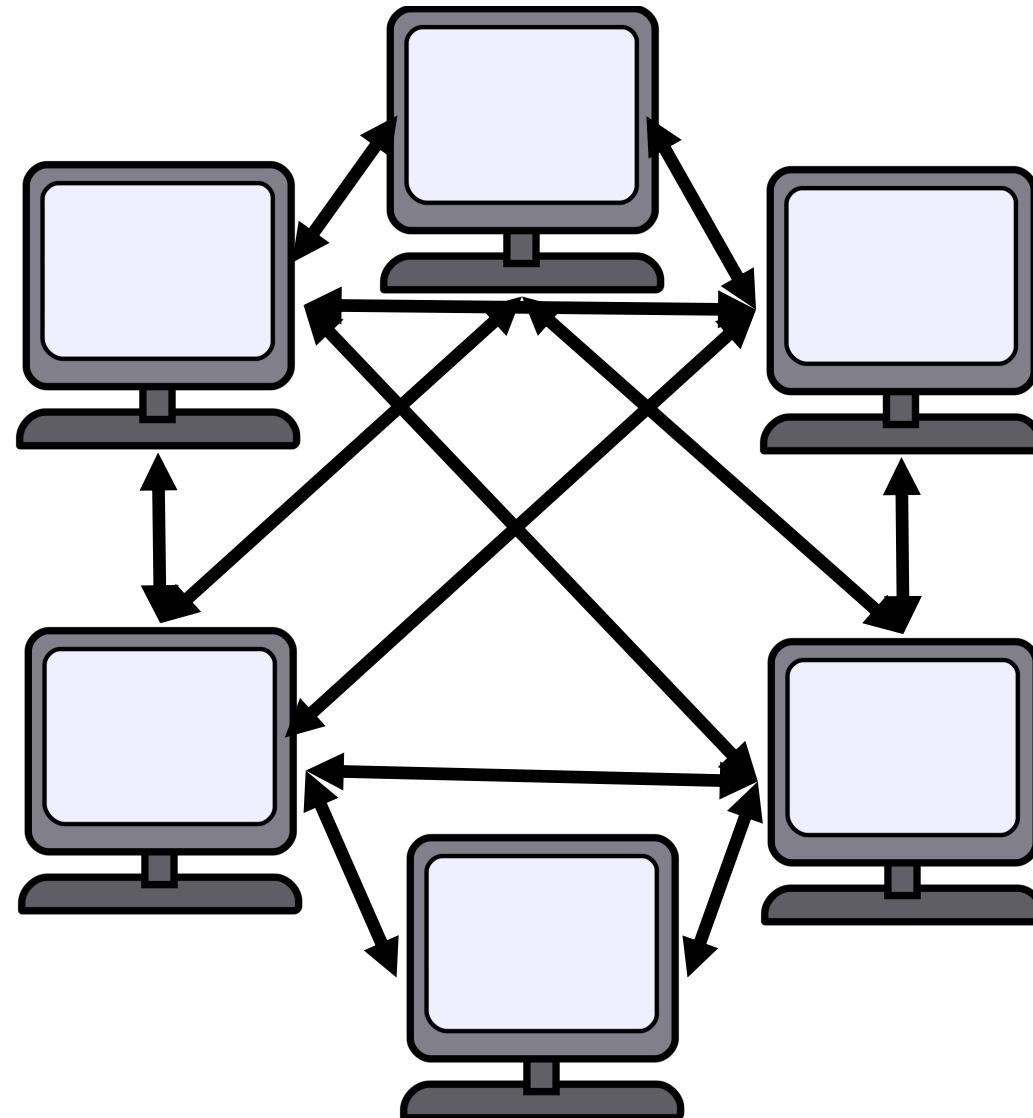
Para conectar dois computadores basta um cabo. Com isso eles conseguem trocar dados (e até jogar)





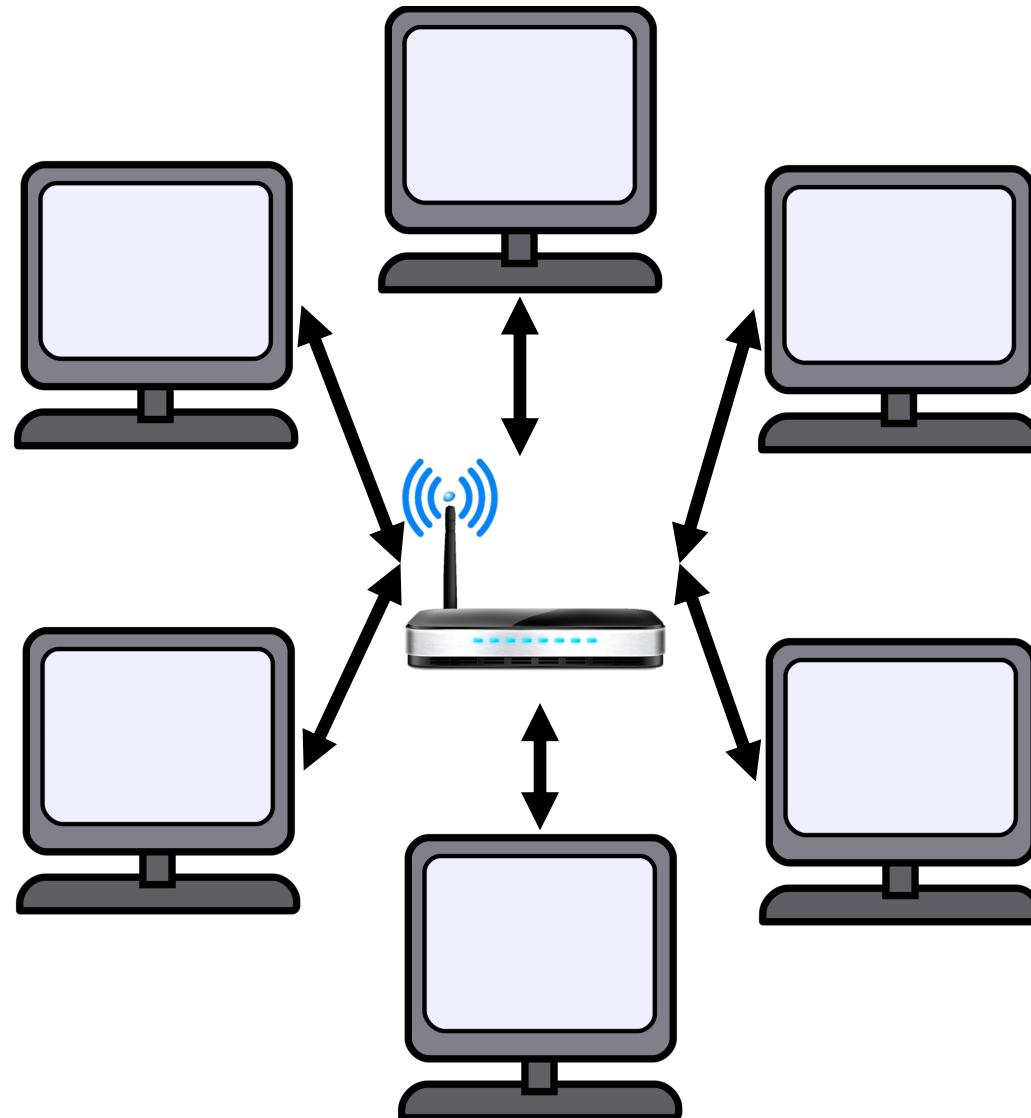
E se tivermos mais de  
dois computadores?

Será viável utilizar cabos conectando  
todos?



- Não parece adequado 😞
- E se precisarmos de mais computadores?



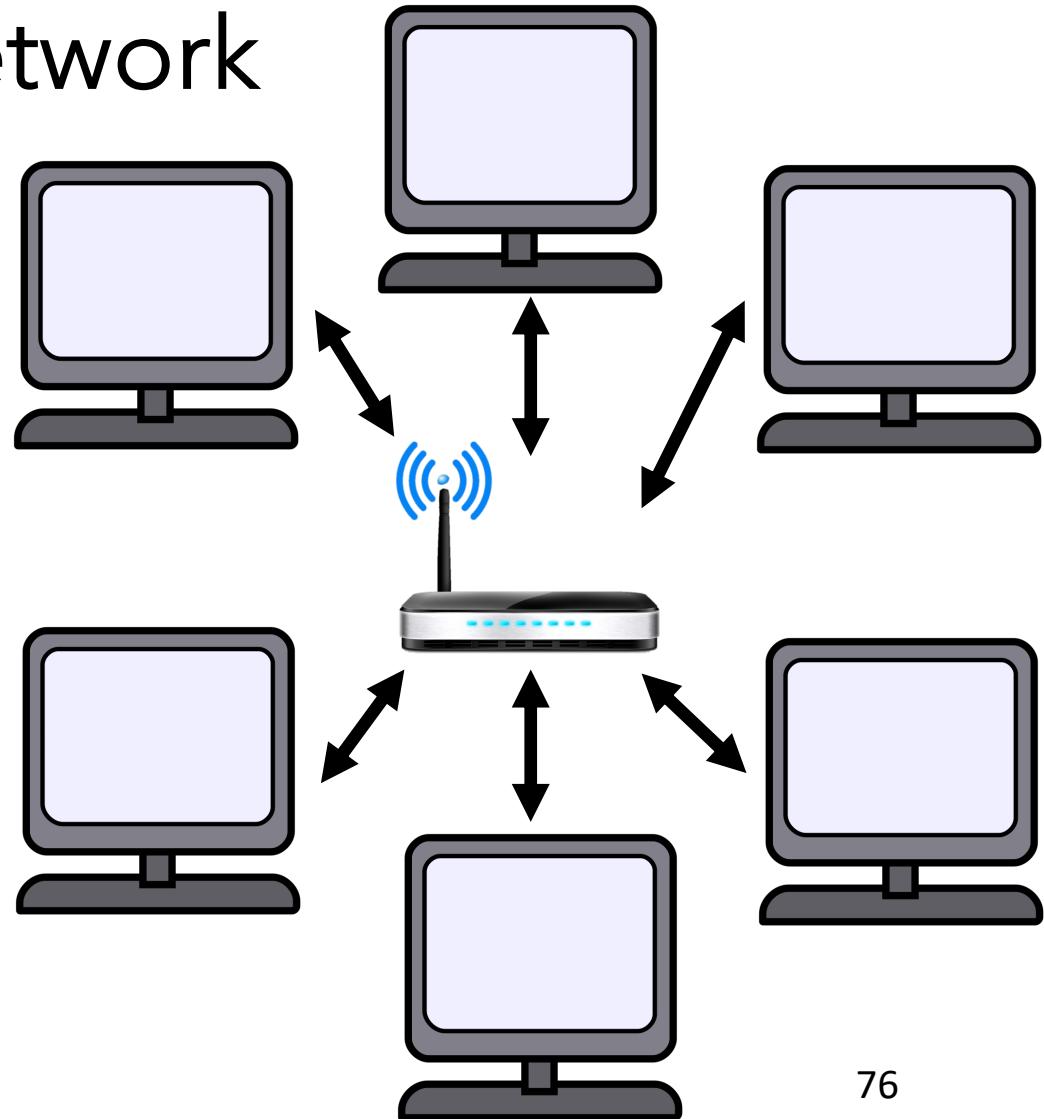


- Utilizando um **roteador** já melhora um pouco
- ☺



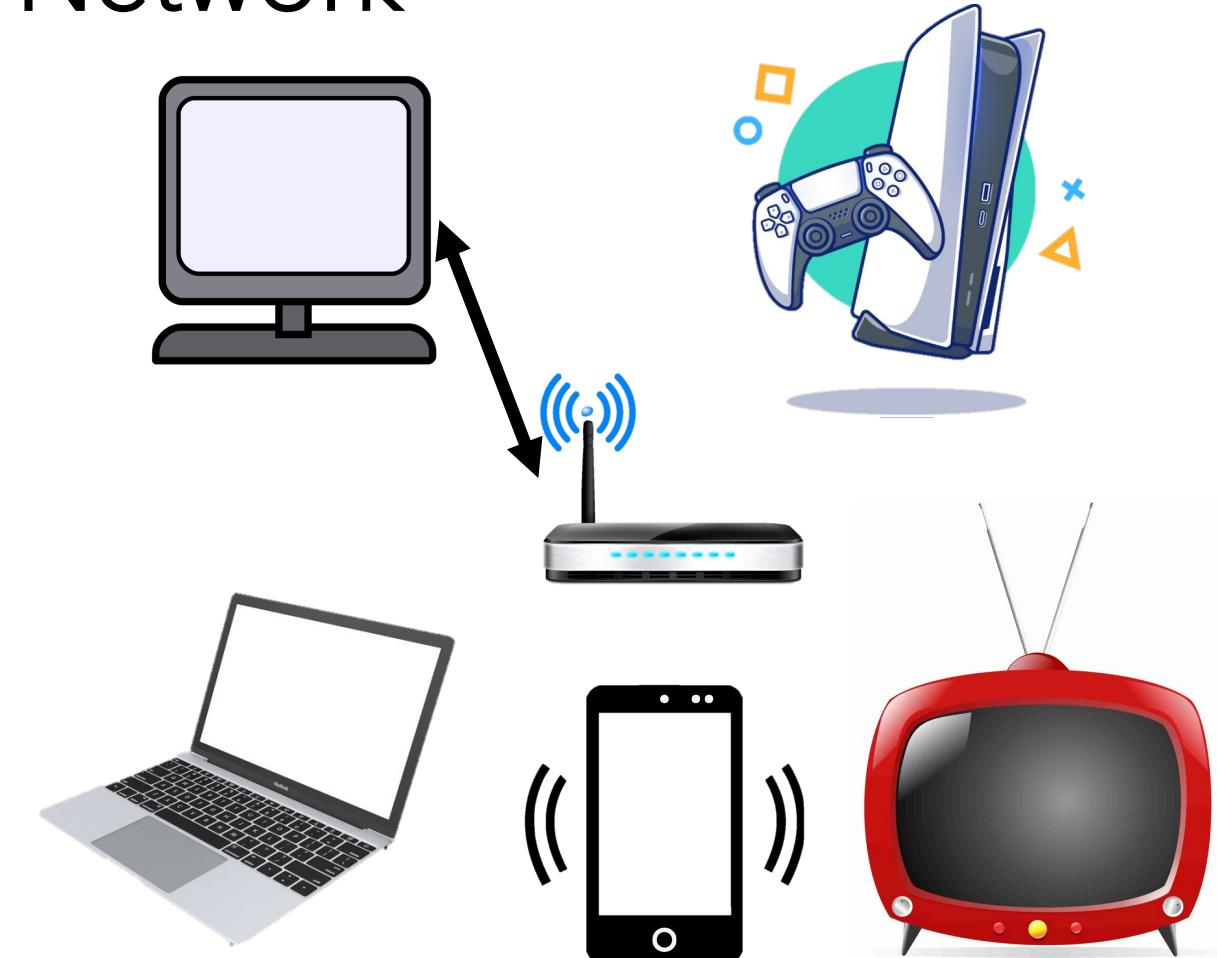
# LAN - Local Area Network

- ❑ Rede Local
- ❑ Dispositivos conseguem trocar dados apenas entre si
- ❑ Não consiste apenas em computadores pessoais



# LAN - Local Area Network

- LAN pode conter:
  - Televisores
  - Videogames
  - Dispositivos móveis

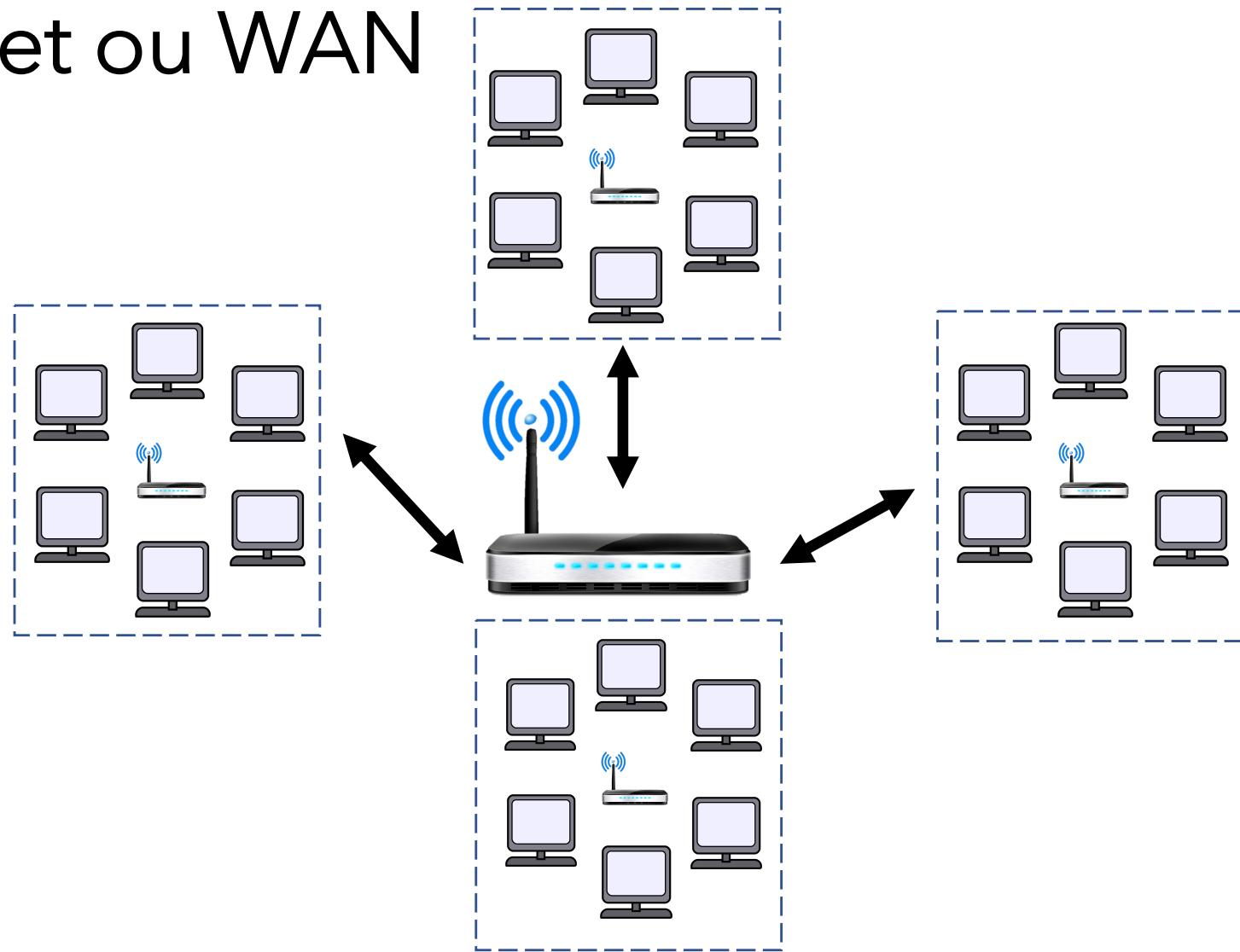




## E a internet? A rede de redes?

A LAN só permite que os equipamentos comuniquem localmente. A internet permite qualquer dispositivo conversar com qualquer outro conectado na rede

# Internet ou WAN





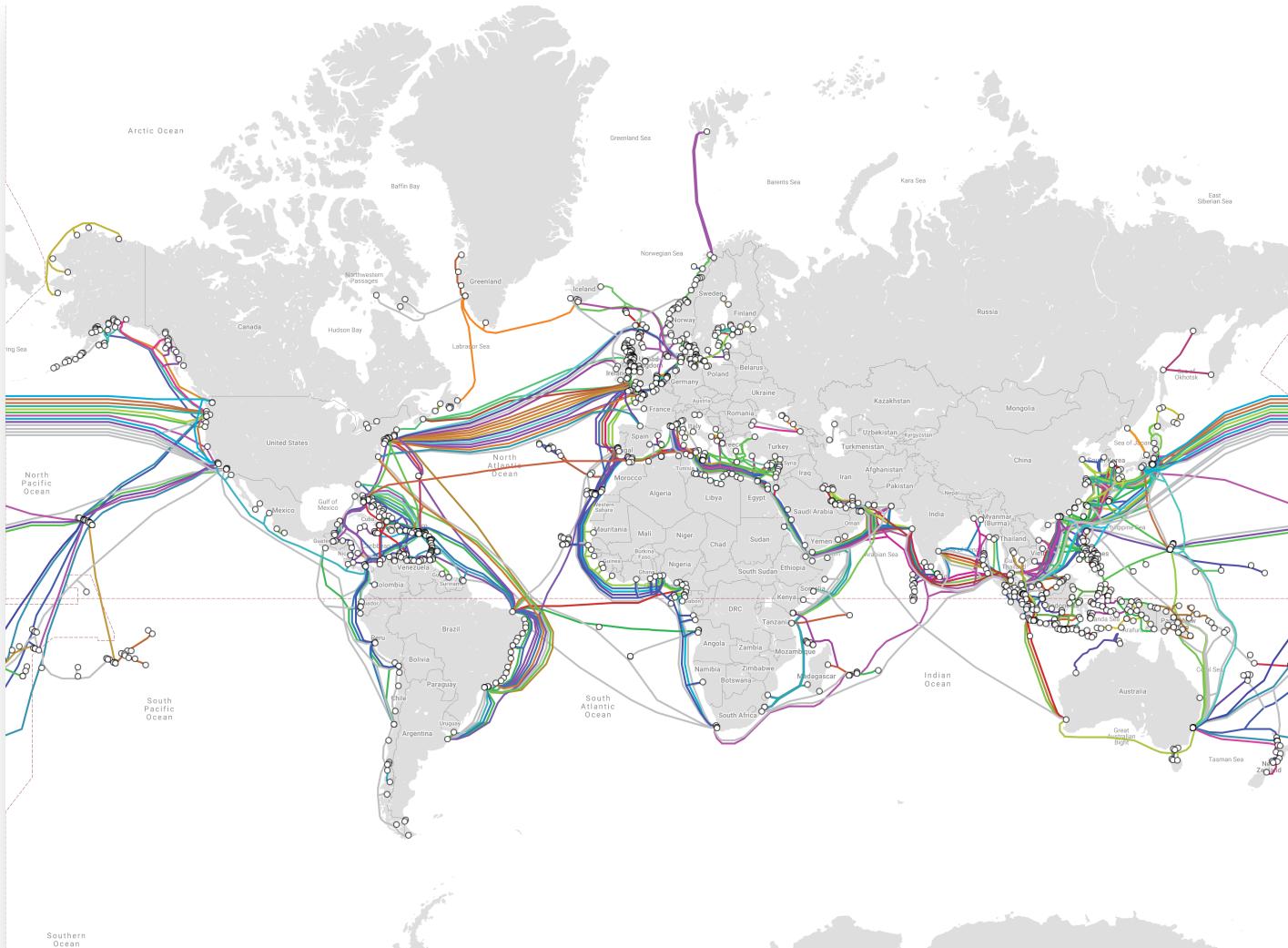
## Endereço IP, Endereço MAC, Roteador?

Para nosso curso não precisamos nos preocupar com esses detalhes! O equipamento e termo “roteador” é o suficiente.



Como as redes se  
conectam globalmente?

Temos grandes corporações responsáveis  
pelos equipamentos e cabos submarinos



Fonte: Reddit



O que fazer agora com  
todo mundo conectado?

As possibilidades são ilimitadas!



- ❑ Conectar no SIGAA e baixar o material da disciplina de Web.
- ❑ Entregar os exercícios dentro do prazo



- Conectar no Spotify e ouvir suas músicas favoritas.
- Descobrir músicas novas no “mix diário”



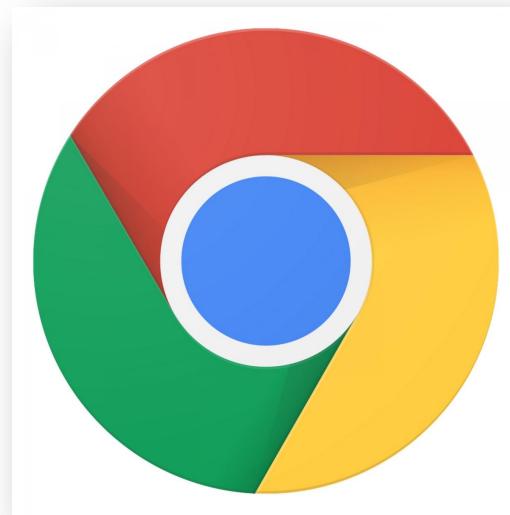
- Acessar serviços de e-mail
- Limpar a caixa de spam



- Acessar serviços de vendas de jogos digitais, como a steam.
- Jogar com colegas em outros países



- ❑ Acessar serviços de filmes e séries sob demanda tais como: Amazon Prime, HBO Max, Disney+, e etc.
- ❑ Ficar chateado quando sua série favorita é cancelada



Utilizar um navegador para requisitar páginas web.



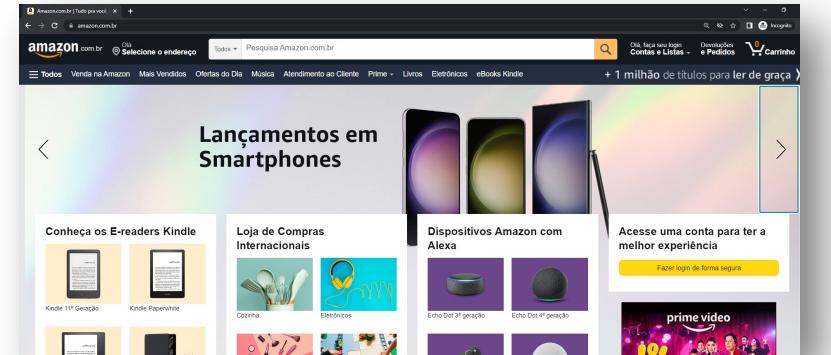
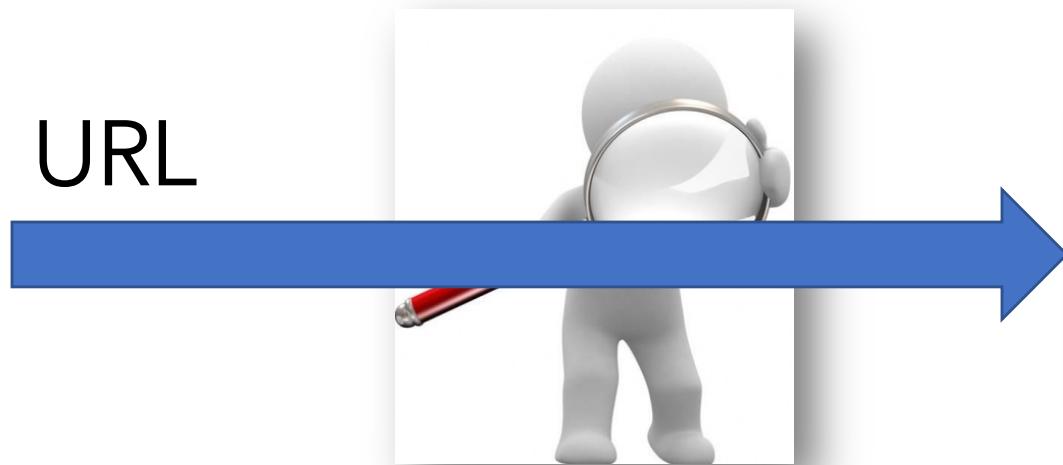
# Web



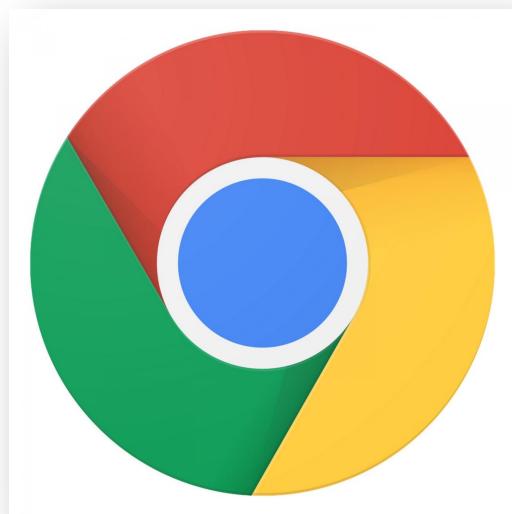
## O que é *web*?

Coleção de recursos (textos, imagens, vídeos) que são trafegados pela *internet* e renderizados em navegadores como uma página web.

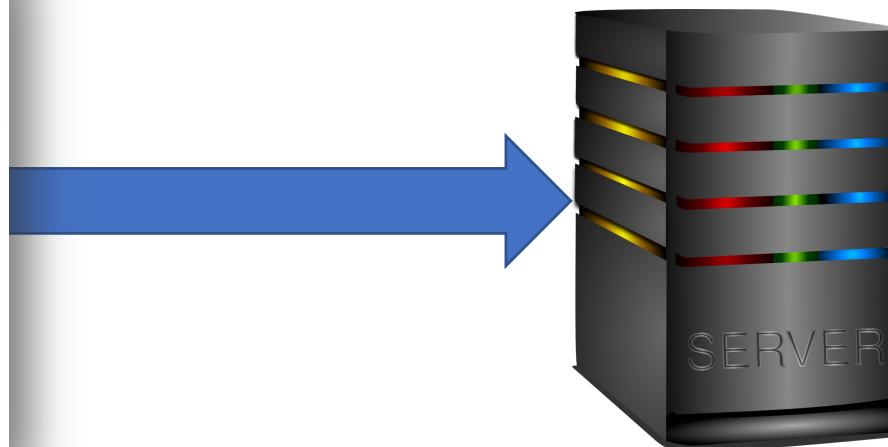
URL



- Os recursos são localizados através de URL (*Uniform Resource Locator*)
- Exemplo: [www.amazon.com.br](http://www.amazon.com.br) (URL)



Recursos ficam armazenados nos servidores



- Esse tipo de recurso é armazenado em servidores na internet.
- Eses recursos trafegam pela internet e são exibidos por navegadores.
- Navegadores são chamados também de “web browser”, ->“navegar na web”

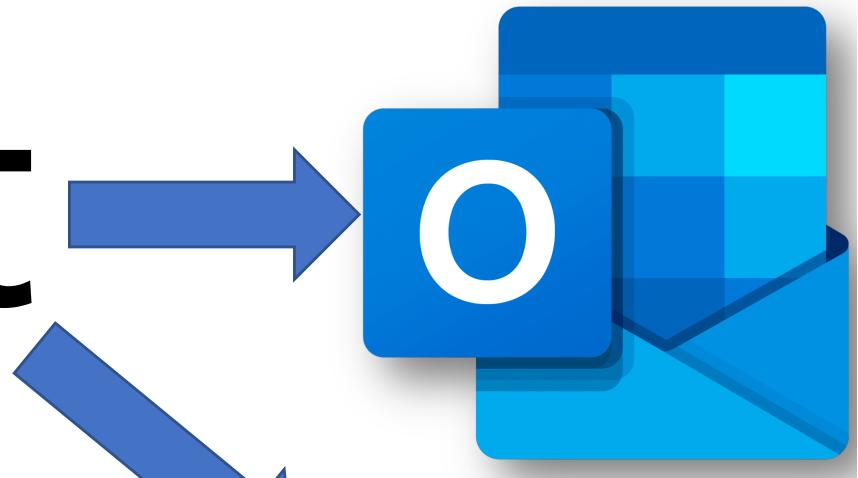
# Internet



# Web

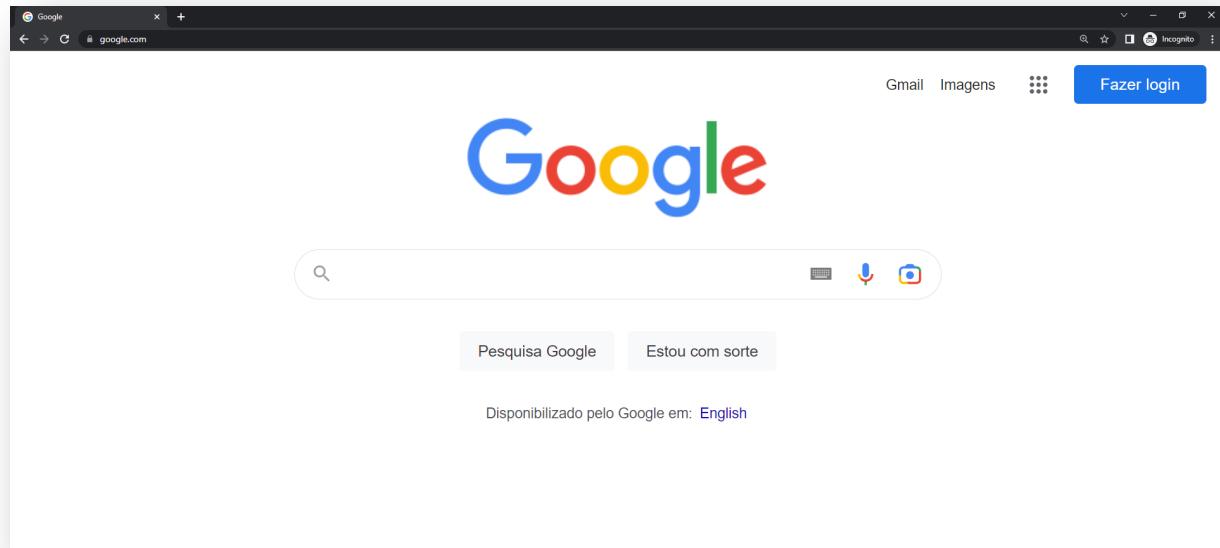
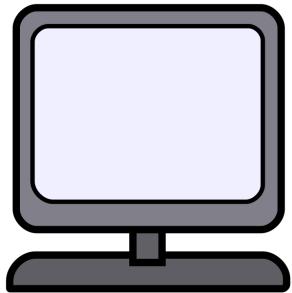
- A “web” é um dos tipos de serviços que trafegam pela Internet.
- A internet é a estrutura física de equipamentos e conexões

# Internet



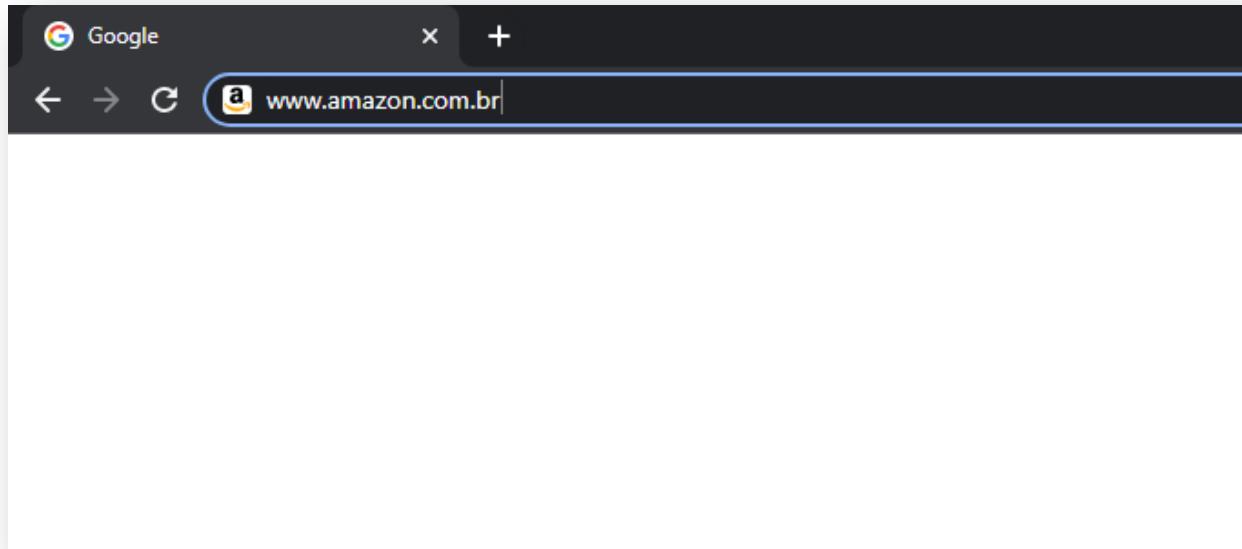
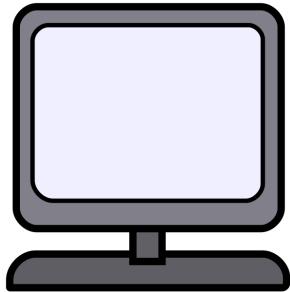
# Requisição e Resposta

# CLIENTE



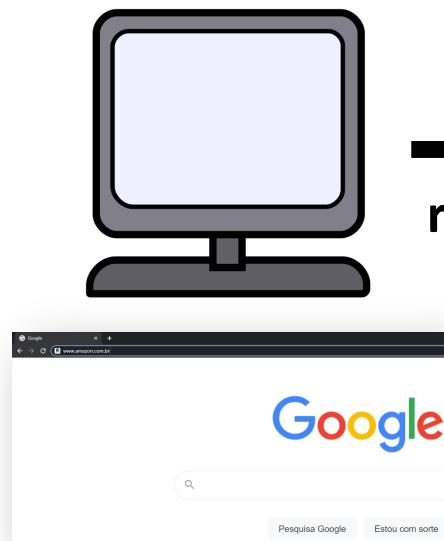
- Executa a ação de abrir o navegador
- Deseja requisitar uma página

# CLIENTE



- No campo de busca do navegador digita o endereço:  
[www.amazon.com.br](http://www.amazon.com.br)
- Ao pressionar enter uma **requisição** será enviada

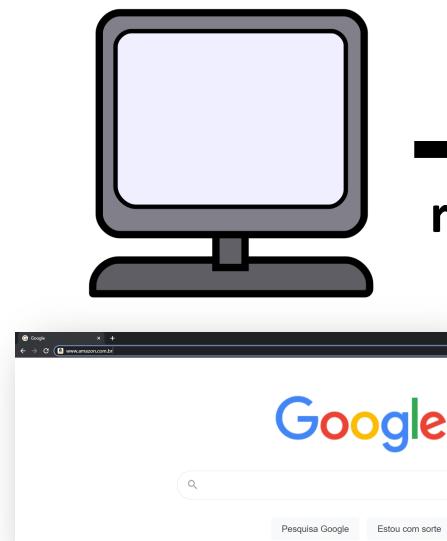
# CLIENTE



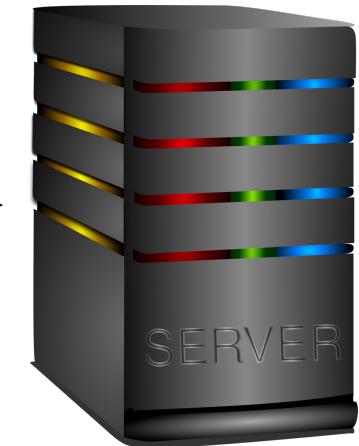
# SERVIDOR

- ❑ A **requisição**, através da Internet, chegará no computador de destino.
- ❑ Este computador tem o nome de “servidor” ou “servidor web”

# CLIENTE

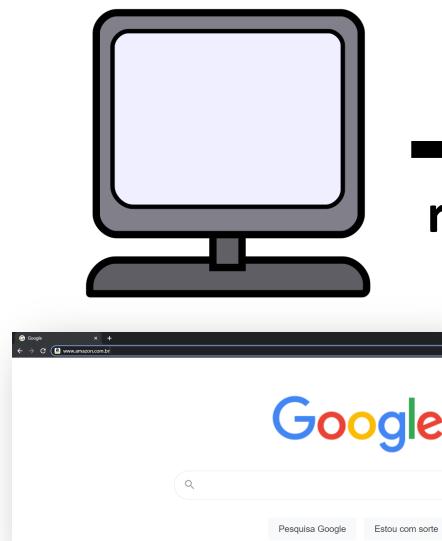


# SERVIDOR



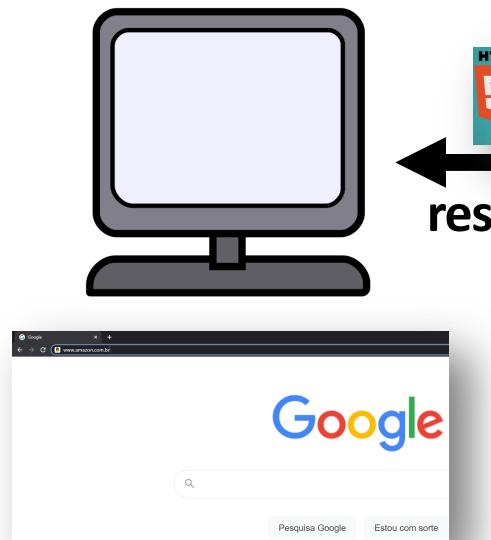
- A **requisição** contém um pedido com a página desejada.
- “Olá Sr. Servidor, tudo bem? Pode me devolver a página [www.amazon.com.br](http://www.amazon.com.br)?”

# CLIENTE



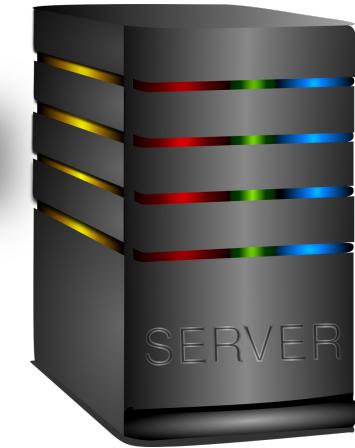
□ A partir desse momento o servidor irá processar a requisição e irá devolver uma resposta para o cliente.

# CLIENTE

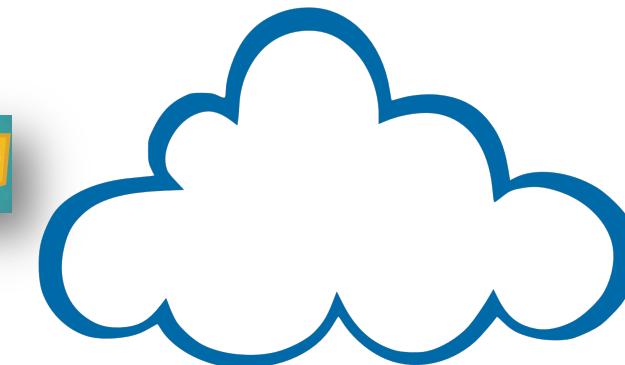


resposta

# SERVIDOR

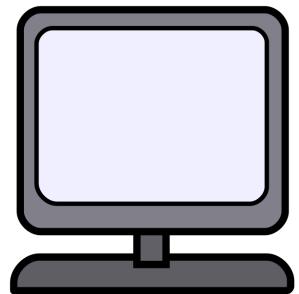


resposta

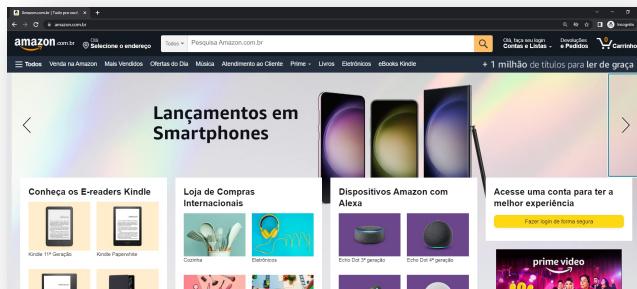
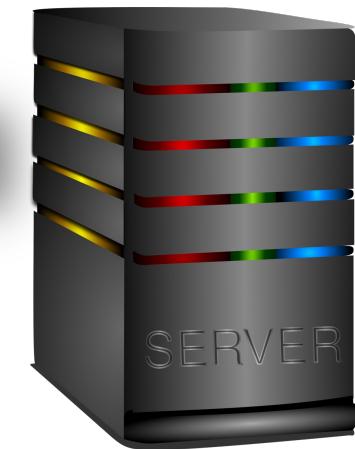


A resposta é uma página web e um código indicando o *status*

# CLIENTE

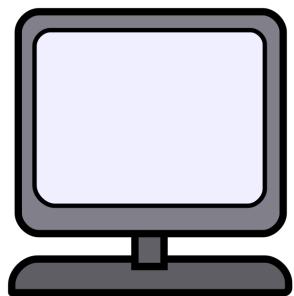


# SERVIDOR

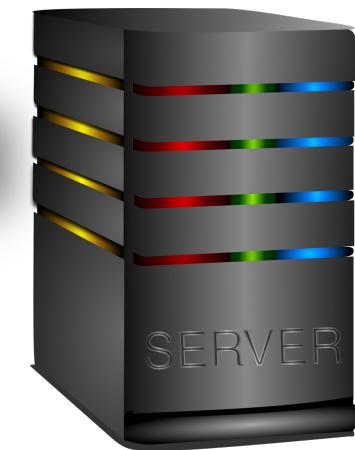


- Se a requisição for processada corretamente o servidor devolverá, como reposta, a página web da Amazon e um código 200.

# CLIENTE



# SERVIDOR



- ❑ Porém diversos erros podem ocorrer. Se a página solicitada não existir é retornada uma outra página informando o erro
- ❑ O famoso código 404 é retornado

Pra fechar!





- Cliente envia uma requisição, que será roteada pela internet



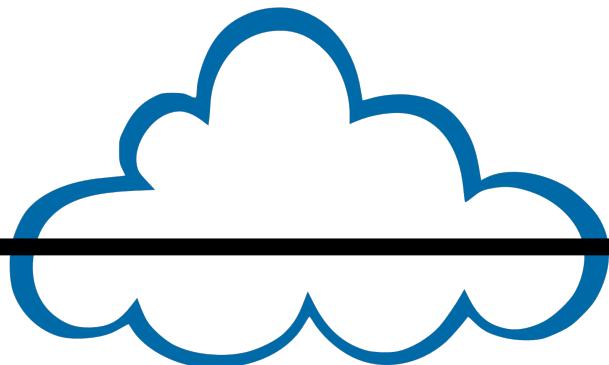
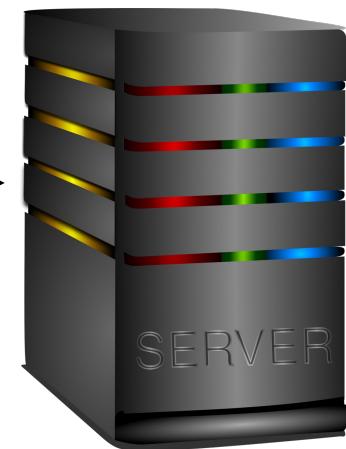
- Cliente envia uma requisição, que será roteada pela internet
- Servidor processa a requisição e devolve uma resposta em formato de página web.

# Frontend e Backend

# Frontend

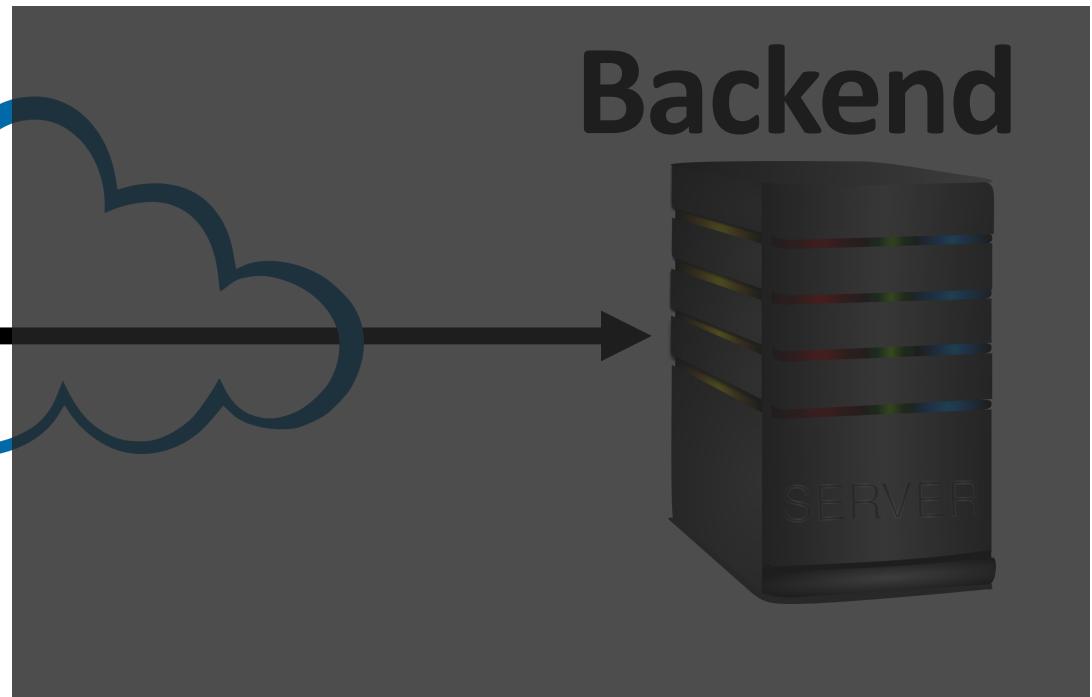


# Backend

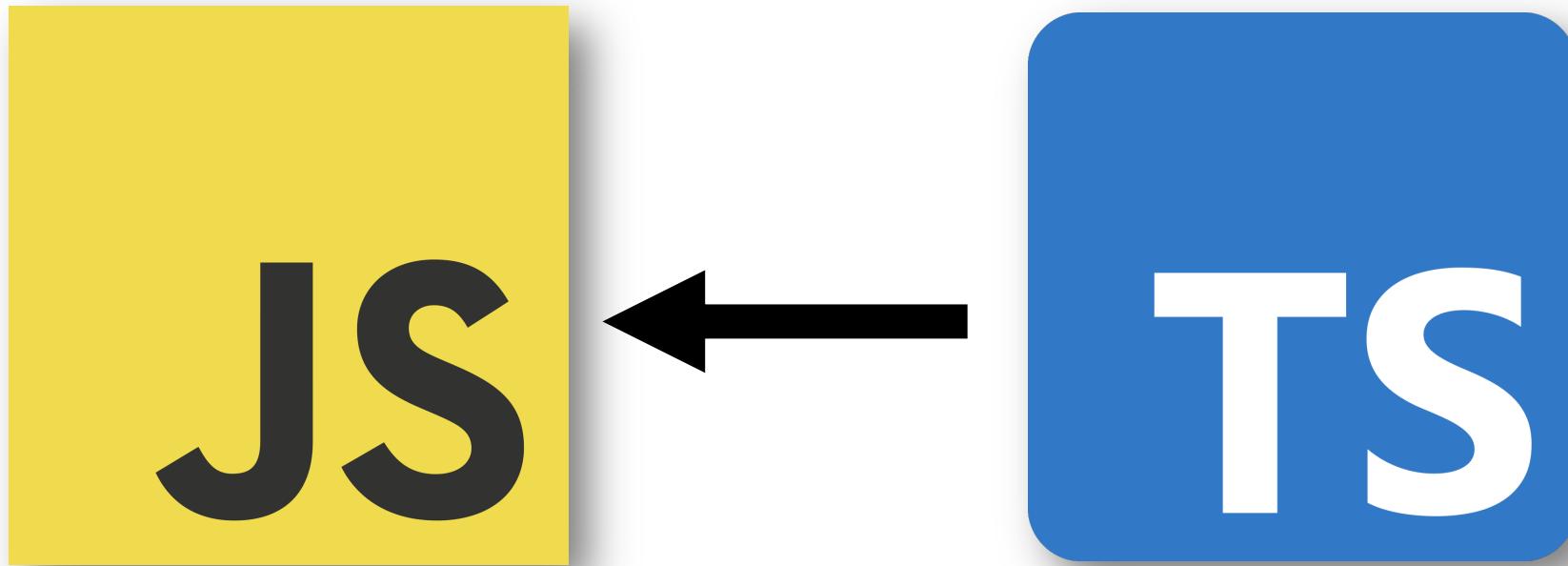


- Frontend* é a parte da web renderizada para o cliente
- Backend* é a parte da web que executa do lado do servidor

# Frontend



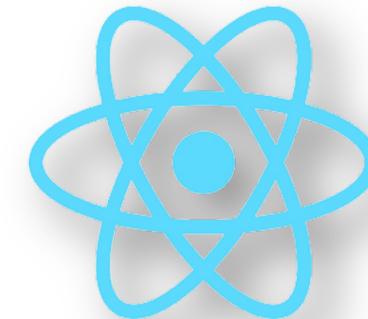
- A equipe responsável pelo frontend cuida de toda a interação/ações e lógica da página web visualizada na máquina cliente. Não necessariamente a estética.
- Eventos, submissão de formulários, ações



- A principal linguagem de programação utilizada no frontend é o JavaScript.
- Recentemente o TypeScript vem ganhando espaço. Mas o código final é convertido para JavaScript antes de ser interpretado pelo navegador.

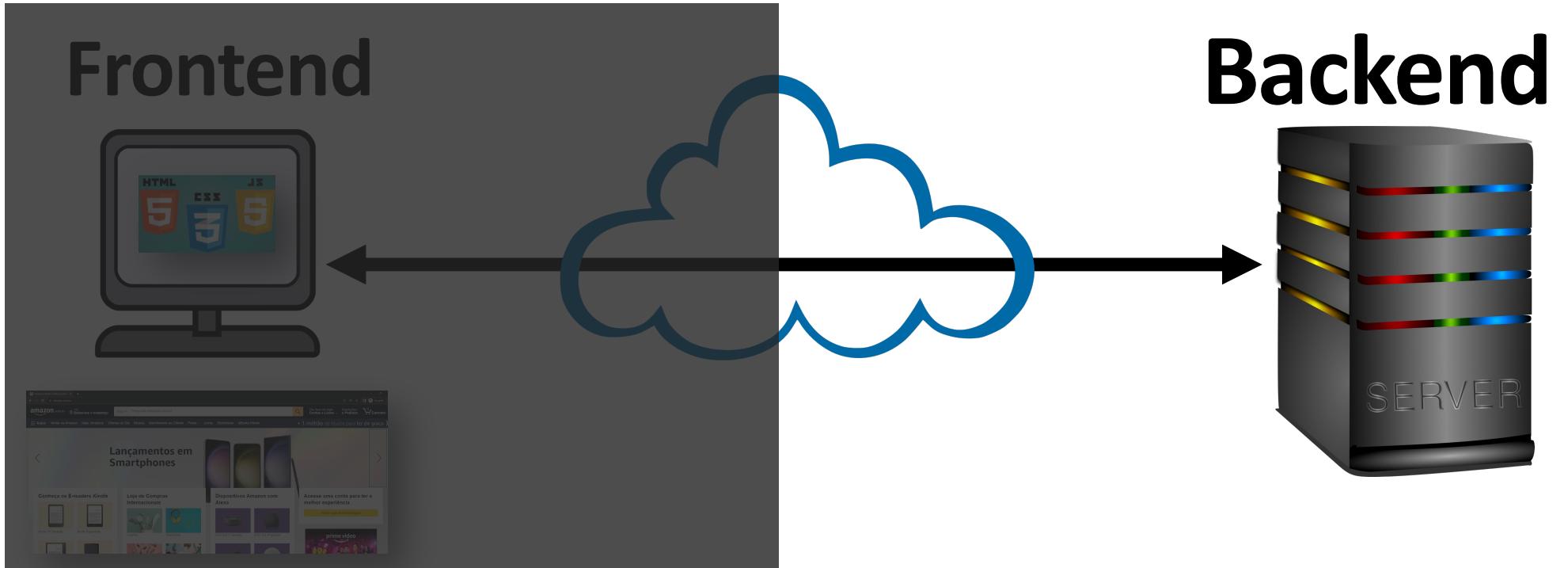


ANGULAR

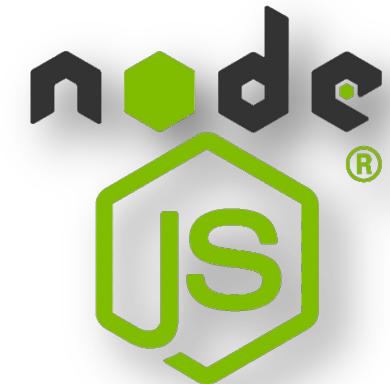


React

- ❑ Alguns frameworks/bibliotecas auxiliam esse desenvolvimento



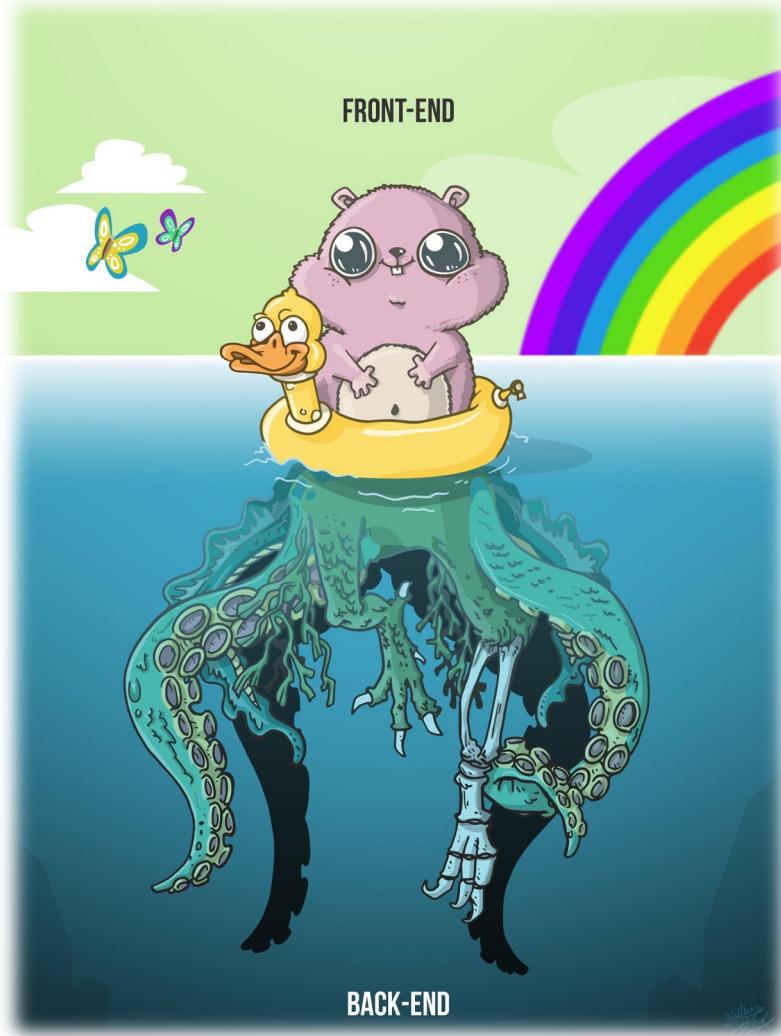
- A equipe responsável pelo backend cuida de todo o código que executa do lado do servidor.
- Acesso ao banco de dados, autenticação, autorização, processamento



- ❑ Diferente do frontend, existem diversas tecnologias e linguagens que podem ser utilizadas no backend, ou lado do servidor.

Pra fechar!





**HTML**



**CSS**



**JS**



**EU  
TOMEI  
CAFÉ  
EXPRESSO**

JS – Verbo

HTML - Substantivo

CSS - Adjetivo



**HTML**



**CSS**



**JS**



# HTML - HyperText Markup Language

**CAFÉ** HTML - Substantivo

# HTML - HyperText Markup Language

---

## □ HyperText

- Remete a ideia de um texto que “linkável”. Isto é, o texto pode ser um caminho para outro elemento. O objetivo inicial era navegar por páginas através desses “*hiper-links*”

# HTML - HyperText Markup Language

---

## □ *Markup Language*

- Significa marcação. Aqui entre a argumentação que HTML não é linguagem de programação e sim **marcação**. A marcação é informar **o que é aquele elemento**. Por exemplo, podemos **marcar um texto** dizendo que é um **botão**, ou um **cabeçalho**.
- A marcação é feita através de etiquetas (**tag**) HTML
- Temos **tag** específicas para **botões**, **imagens**, **cabeçalhos**, **formulários**, **tabelas**.
- Temos **tag** para **organização/agrupamento**
- Temos para **semântica**: Barra de navegação, rodapé, seção
  - Essência para acessibilidade

# HTML - HyperText Markup Language

---

## ❑ *Language*

- ❑ Concluindo que HTML é uma “Linguagem para a marcação de hipertexto”
- ❑ Sua extensão é “.html”

**HTML**



**CSS**



**JS**



# CSS - Cascading Style Sheet

---

## ❑ Cascading

- ❑ Em cascata. Significa que a estilização é aplicada “em cascata”.
- ❑ Veremos que é possível de definir um “identificador” para elementos HTML.
- ❑ A estilização pode ser aplicada para elementos de determinada tag e/ou para elementos com um identificador específico.
- ❑ Se o elemento possuir a tag e o identificador a estilização será aplicada “em cascata”.
- ❑ Mas devemos tomar cuidado com casos concorrentes, que veremos ao longo do curso.

# CSS - Cascading Style Sheet

---

## □ *Style Sheet*

- Folhas de estilização. As “folhas de estilização” podem ser vistas como os arquivos “.css” que contém as regras de estilização
- Essas regras podem definir diversos aspectos dos elementos HTML, tais como:
  - Cor, fonte, margem, posição, animação, ..., etc.

**HTML**



**CSS**



**JS**



# JavaScript (JS)

---

- ❑ Linguagem de programação multiparadigma que contém o comportamento (ações/verbos) de uma página com elementos HTML
- ❑ Possui sintaxe e comandos presentes em outras linguagens tais como:
  - ❑ if-else, for, switch, variáveis, constantes, expressões lambda, etc

## JavaScript (JS)

---

- ❑ Navegadores são capazes de interpretar os comandos descrito por código JS
- ❑ Utilizando o NodeJS é possível executar código JS fora do navegador.

## JavaScript (JS)

---

- ❑ Apesar de não ter nenhuma relação com a linguagem Java, seu nome foi pensando para fins comerciais.
- ❑ Na época, a linguagem Java estava popular

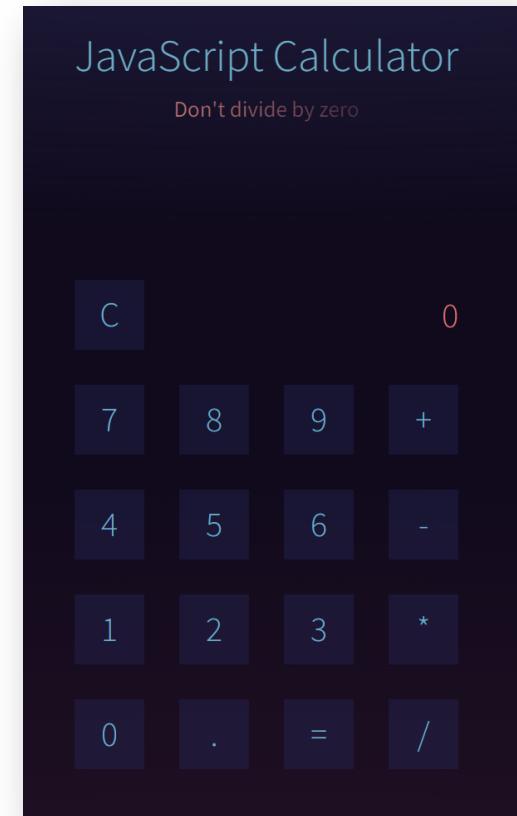
# Mão na Massa - HTML/CSS/JS

---

❑ Acesse o link:

<https://codepen.io/giana/pen/GJMBEv>

❑ É uma calculadora escrita em  
HTML/CSS/JS



JavaScript calculator

Giana PRO + Follow

HTML

```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12  <button data-ops="plus" class="ops">+</button>
13
14  <button class="num" data-num="4">4</button>
15  <button class="num" data-num="5">5</button>
16  <button class="num" data-num="6">6</button>
17  <button data-ops="minus" class="ops">-</button>
18
19  <button class="num" data-num="1">1</button>
```

CSS (SCSS)

```
45 &::after {
46   clear: both;
47 }
48
49
50 /* Calculator after dividing by zero */
51 .broken {
52   animation: broken 2s;
53   transform: translate3d(0,-200px,0);
54   opacity: 0;
55 }
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
```

JS

```
1 /*
2 TODO:
3   Limit number input
4   Disallow . from being entered multiple times
5   Clean up structure
6 */
7
8 (function() {
9   "use strict";
10
11 // Shortcut to get elements
12 var el = function(element) {
13   if (element.charAt(0) === "#") { // If passed an ID...
14     return document.querySelector(element); // ... return
15   }
16
17   return document.querySelectorAll(element); // Otherwise
18 }
19
20 // Variables
```

JavaScript Calculator

Don't divide by zero

The calculator interface features a dark background with light-colored buttons. It includes a numeric keypad (0-9), decimal point, and operators (+, -, \*, /). A 'C' button for clearing the display is located at the top left. The display area at the top shows the value '0'. Below the display, a warning message 'Don't divide by zero' is displayed in a small font.

The screenshot shows a code editor interface with three tabs: HTML, CSS (SCSS), and JS. The HTML tab contains the structure of the calculator, including an H1 title, a warning message, a clear button, a viewer, and a grid of number and operation buttons. The CSS (SCSS) tab contains SCSS code for styling, including a 'broken' class for handling division by zero and a 'viewer' class for styling the display area. The JS tab contains JavaScript code for the calculator's logic, including a function to get elements by ID or class, and a main function that includes TODO items like 'Limit number input' and 'Disallow . from being entered multiple times'. Red arrows point from the text to the corresponding code in each tab.

```
HTML
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12  <button data-ops="plus" class="ops">+</button>
13
14  <button class="num" data-num="4">4</button>
15  <button class="num" data-num="5">5</button>
16  <button class="num" data-num="6">6</button>
17  <button data-ops="minus" class="ops">-</button>
18
19  <button class="num" data-num="1">1</button>
20
```

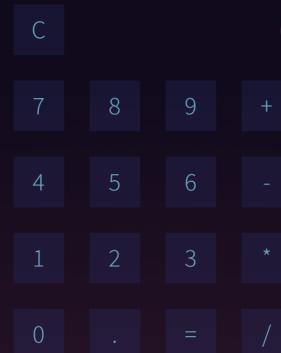
```
CSS (SCSS)
45 * &::after {
46   clear: both;
47 }
48
49 /* Calculator after dividing by zero */
50 .broken {
51   animation: broken 2s;
52   transform: translate3d(0,-2000px,0);
53   opacity: 0;
54 }
55
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
```

```
JS
1 /*
2 TODO:
3   Limit number input
4   Disallow . from being entered multiple times
5   Clean up structure
6 */
7
8 (function() {
9   "use strict";
10
11  // Shortcut to get elements
12  var el = function(element) {
13    if (element.charAt(0) === "#") { // If passed an ID...
14      return document.querySelector(element); // ... return
15    }
16
17    return document.querySelectorAll(element); // Otherwise
18  };
19
20 // Variables
```

## JavaScript Calculator

Don't divide by zero

- Nessa aba temos o código HTML
- Determina o que temos na tela
- A maioria são botões nesse exemplo



A screenshot of a code editor interface titled "JavaScript calculator". The editor has three tabs: "HTML", "CSS (SCSS)", and "JS". The "HTML" tab contains the structure of the calculator, including an H1 title, a warning message, a clear button, and a viewer div. The "CSS (SCSS)" tab contains SCSS code for styling, including rules for a broken state and a viewer element. The "JS" tab contains JavaScript code for the calculator's logic, including a function to get elements by ID or class, and a main function that includes TODO items like "Limit number input" and "Disallow . from being entered multiple times". Red arrows point from the text "A segunda aba é o CSS" and "Cores, posição, animação, opacidade, etc." to the "CSS (SCSS)" tab.

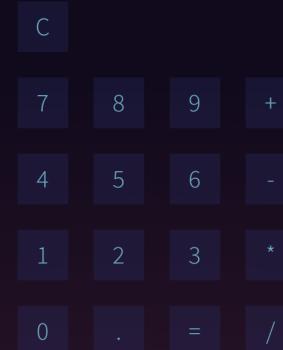
```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12
13  <button data-ops="plus" class="ops">+</button>
14
15  <button class="num" data-num="4">4</button>
16  <button class="num" data-num="5">5</button>
17  <button class="num" data-num="6">6</button>
18  <button data-ops="minus" class="ops">-</button>
19
20  <button class="num" data-num="1">1</button>
```

```
45 * &::after {
46   clear: both;
47 }
48
49 /* Calculator after dividing by zero */
50 .broken {
51   animation: broken 2s;
52   transform: translate3d(0,-2000px,0);
53   opacity: 0;
54 }
55
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
65 }
```

```
1 /*
2 TODO:
3   Limit number input
4   Disallow . from being entered multiple times
5   Clean up structure
6 */
7
8 (function() {
9   "use strict";
10
11   // Shortcut to get elements
12   var el = function(element) {
13     if (element.charAt(0) === "#") { // If passed an ID...
14       return document.querySelector(element); // ... return
15     }
16
17     return document.querySelectorAll(element); // Otherwise
18   };
19
20 // Variables
```

## JavaScript Calculator

Don't divide by zero



- A segunda aba é o CSS
- Elas determinam a estilização dos elementos.
- Cores, posição, animação, opacidade, etc.

The screenshot shows a code editor interface with three tabs: HTML, CSS (SCSS), and JS. A large red arrow points from the JS tab towards the calculator application below.

**HTML**

```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12  <button data-ops="plus" class="ops">+</button>
13
14  <button class="num" data-num="4">4</button>
15  <button class="num" data-num="5">5</button>
16  <button class="num" data-num="6">6</button>
17  <button data-ops="minus" class="ops">-</button>
18
19  <button class="num" data-num="1">1</button>
```

**CSS (SCSS)**

```
45 * &::after {
46   clear: both;
47 }
48
49
50 /* Calculator after dividing by zero */
51 .broken {
52   animation: broken 2s;
53   transform: translate3d(0,-2000px,0);
54   opacity: 0;
55 }
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
```

**JS**

```
1 /*
2 TODO:
3   Limit number input
4   Disallow . from being entered multiple times
5   Clean up structure
6 */
7
8 (function() {
9   "use strict";
10
11   // Shortcut to get elements
12   var el = function(element) {
13     if (element.charAt(0) === "#") { // If passed an ID,
14       return document.querySelector(element); // ... return
15     }
16
17     return document.querySelectorAll(element); // Otherwise
18   };
19
20});
```

**JavaScript Calculator**

Don't divide by zero

C 0

7	8	9	+
4	5	6	-
1	2	3	*
0	.	=	/

```
GitHub Gist: JavaScript calculator by Giana
```

**HTML**

```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12  <button data-ops="plus" class="ops">+</button>
13
14  <button class="num" data-num="4">4</button>
15  <button class="num" data-num="5">5</button>
16  <button class="num" data-num="6">6</button>
17  <button data-ops="minus" class="ops">-</button>
18
19  <button class="num" data-num="1">1</button>
```

**CSS (SCSS)**

```
45 * &::after {
46   clear: both;
47 }
48
49
50 * /* Calculator after dividing by zero */
51 .broken {
52   animation: broken 2s;
53   transform: translate3d(0,-2000px,0);
54   opacity: 0;
55 }
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
```

**JS**

```
1 /*
2 TODO:
3   Limit number input
4   Disallow . from being entered multiple times
5   Clean up structure
6 */
7
8 (function() {
9   "use strict";
10
11 // Shortcut to get elements
12 var el = function(element) {
13   if (element.charAt(0) === "#") { // If passed an ID...
14     return document.querySelector(element); // ... return
15   }
16
17   return document.querySelectorAll(element); // Otherwise
18 }
19
20 // Variables
```

JavaScript Calculator

Don't divide by zero

C	0		
7	8	9	+
4	5	6	-
1	2	3	*
0	.	=	/

## Removendo CSS e JS

---

- ❑ Vamos entender melhor o HTML e o que ele representa na página.
- ❑ Para isso vamos remover o código CSS e JS.
- ❑ Não se preocupe se achar o código confuso. Temos o semestre inteiro ainda.

The screenshot shows a code editor interface with three tabs: HTML, CSS (SCSS), and JS. The HTML tab contains the structure of a calculator. The CSS tab contains SCSS code with comments explaining various styles like clearing floats and handling broken states. The JS tab contains JavaScript code with comments explaining functions like getting elements by ID or class.

```
JavaScript calculator
Giana PRO + Follow
HTML
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5
6 <button id="clear" class="clear">C</button>
7
8 <div id="viewer" class="viewer">0</div>
9
10 <button class="num" data-num="7">7</button>
11 <button class="num" data-num="8">8</button>
12 <button class="num" data-num="9">9</button>
13 <button data-ops="plus" class="ops">+</button>
14
15 <button class="num" data-num="4">4</button>
16 <button class="num" data-num="5">5</button>
17 <button class="num" data-num="6">6</button>
18 <button data-ops="minus" class="ops">-</button>
19
20 <button class="num" data-num="1">1</button>

CSS (SCSS)
1 // &::after {
2   clear: both;
3 }
4
5 /* Calculator after dividing by zero */
6 .broken {
7   animation: broken 2s;
8   transform: translate3d(0,-2000px,0);
9   opacity: 0;
10 }
11
12 .viewer {
13   color: #c97874;
14   float: left;
15   line-height: 3em;
16   text-align: right;
17   text-overflow: ellipsis;
18   overflow: hidden;
19   width: 7.5em;
20 }

JS
1 // /*
2 // TODO:
3 //   Limit number input
4 //   Disallow . from being entered multiple times
5 //   Clean up structure
6 // */
7
8 // (function() {
9 //   "use strict";
10
11 // Shortcut to get elements
12 var el = function(element) {
13   if (element.charAt(0) === "#") { // If passed an ID...
14     return document.querySelector(element); // ... returns single element
15   }
16
17 //   return document.querySelectorAll(element); // Otherwise, returns a nodeList
18 };
19
20 // Variables
```

## JavaScript Calculator

Don't divide by zero  
C  
0  
7 8 9 + 4 5 6 - 1 2 3 × 0 . = /  
Reset Universe?

- 
- ❑ Observe os códigos CSS e JS comentados.
  - ❑ Cada linha possui // (duas barras) no início

```
JavaScript calculator
Giana PRO + Follow
```

HTML

```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5
6 <button id="clear" class="clear">C</button>
7
8 <div id="viewer" class="viewer">0</div>
9
10 <button class="num" data-num="7">7</button>
11 <button class="num" data-num="8">8</button>
12 <button class="num" data-num="9">9</button>
13 <button data-ops="plus" class="ops">+</button>
14
15 <button class="num" data-num="4">4</button>
16 <button class="num" data-num="5">5</button>
17 <button class="num" data-num="6">6</button>
18 <button data-ops="minus" class="ops">-</button>
19
20 <button class="num" data-num="1">1</button>
```

CSS (SCSS)

```
45 // &::after {
46 //   clear: both;
47 //}
48 //}
49
50 /* Calculator after dividing by zero */
51 .broken {
52   animation: broken 2s;
53   transform: translate3d(0,-2000px,0);
54   opacity: 0;
55 }
56
57 .viewer {
58   color: #c97874;
59   float: left;
60   line-height: 3em;
61   text-align: right;
62   text-overflow: ellipsis;
63   overflow: hidden;
64   width: 7.5em;
```

JS

```
1 /**
2 // TODO:
3 // Limit number input
4 // Disallow . from being entered multiple times
5 // Clean up structure
6 */
7
8 // (function() {
9 //   "use strict";
10
11 // // Shortcut to get elements
12 // var el = function(element) {
13 //   if (element.charAt(0) === "#") { // If passed an ID...
14 //     return document.querySelector(element); // ... returns single element
15 //   }
16
17 //   return document.querySelectorAll(element); // Otherwise, returns a nodeList
18 // };
19
20 // // Variables
```



- O resultado final é uma página HTML pura.
- Os elementos continuam existindo
- São botões e textos
- Não possuem nenhuma estilização (CSS)
- Não executam nenhuma ação (JS)

## Adicionando o CSS

---

- ❑ Vamos voltar com o código CSS
- ❑ Basta repetir os mesmos passos usados pra “comentar”
- ❑ Ao selecionar um código comentado e repetir os passos o código será descomentado.
- ❑ Faça isso apenas para o CSS (segunda aba)

The screenshot shows a GitHub repository titled "JavaScript calculator" by Giana. The repository has 1 star and 1 fork. The code is organized into three main sections: HTML, CSS (SCSS), and JS.

- HTML:**

```
1 <h1>JavaScript Calculator</h1>
2 <p class="warning">Don't divide by zero</p>
3
4 <div id="calculator" class="calculator">
5   <button id="clear" class="clear">C</button>
6
7   <div id="viewer" class="viewer">0</div>
8
9   <button class="num" data-num="7">7</button>
10  <button class="num" data-num="8">8</button>
11  <button class="num" data-num="9">9</button>
12  <button data-ops="plus" class="ops">+</button>
13
14  <button class="num" data-num="4">4</button>
15  <button class="num" data-num="5">5</button>
16  <button class="num" data-num="6">6</button>
17  <button data-ops="minus" class="ops">-</button>
18
19  <button class="num" data-num="1">1</button>
```
- CSS (SCSS):**

```
45 &:after {
46   clear: both;
47 }
48
49 /* Calculator after dividing by zero */
50 .broken {
51   animation: broken 2s;
52   transform: translate3d(0,-2000px,0);
53   opacity: 0;
54 }
55
56 .viewer {
57   color: #c97874;
58   float: left;
59   line-height: 3em;
60   text-align: right;
61   text-overflow: ellipsis;
62   overflow: hidden;
63   width: 1.5em;
64 }
```
- JS:**

```
1 // /*
2 // TODO:
3 //   Limit number input
4 //   Disallow . from being entered multiple times
5 //   Clean up structure
6 // */
7
8 // (function() {
9 //   "use strict";
10
11 //   // Shortcut to get elements
12 //   var el = function(element) {
13 //     if (element.charAt(0) === "#") { // If passed an ID
14 //       return document.querySelector(element); // ...
15 //     }
16 //
17 //     return document.querySelectorAll(element); // Otherwise
18 //   };
19 //
20 //   // variables
```

Below the code snippets is a screenshot of the calculator application interface. The title bar says "JavaScript Calculator" and the status bar says "Don't divide by zero". The calculator has a numeric keypad from 0 to 9, arithmetic operators (+, -, \*, /), and a clear button (C). A red arrow points from the JS code snippet to the calculator's display area.

O código CSS não está comentado  
 O código JS deverá permanecer comentado  
 Linhas comentadas possuem // (duas barras)

9  
C 0  
7 8 9 +  
4 5 6 -  
1 2 3 \*  
0 . = /

141

## Página com HTML e CSS

---

- O resultado consiste dos elementos HTML, isto é, o que temos na tela e a estilização fornecida pelo CSS.
- A página está “agradável” pois agora os elementos estão posicionados adequadamente, com cores e fonte.
- Mas nada funciona ☹

# JavaScript são as ações

---

- Como não há código JS, não temos nenhuma ação.
- O JS que consiste nos “verbos” e permite executarmos ações.
  - Operações aritméticas
  - Enviar formulários
  - Fazer login
  - Jogar
  - ...

# Voltando com o JavaScript

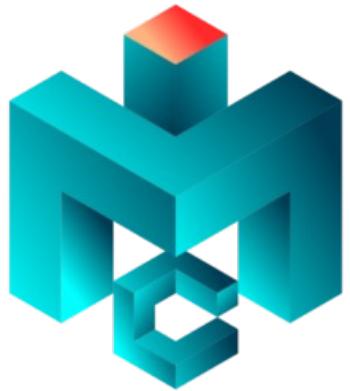
---

- Agora descomente o código JS e verá que a página voltará a funcionar normalmente.
- Assim, temos:
  - Elementos na página (botões) : HTML
  - Estilização (cores, fonte, posicionamento): CSS
  - Ações (operações aritméticas): JS

# Dúvidas?

---





# Aula – 3

## Introdução ao HTML

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ O que é HTML
- ❑ Elementos básicos
  - ❑ Parágrafos, Cabeçalhos, Listas, Imagens e âncoras
- ❑ Elementos em Bloco e Linha
- ❑ Extensão do VSCode
- ❑ Navegador para inspecionar elementos.

HTML

**HTML**





## O que é HTML

*Linguagem de Marcação* que diz quais elementos se encontram na página

**HTML diz ao navegador onde e  
quais elementos estão na tela:**

- Botões
- Formulários
- Tabelas
- Títulos
- Cabeçalhos
- Rodapé



# HTML foi elaborado pensando no compartilhamento e formatação de artigos científicos.

□ Como descreveríamos um artigo científico?

Entertainment Computing 46 (2023) 100549

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: [www.elsevier.com/locate/entcom](http://www.elsevier.com/locate/entcom)

 ELSEVIER

An analysis of DOOM level generation using Generative Adversarial Networks

Edoardo Giacomello, Pier Luca Lanzi, Daniele Loiacono \*

*Politecnico di Milano, Italy*

---

ARTICLE INFO

Keywords:

Deep learning  
Generative Adversarial Networks  
Video games  
Procedural Content Generation

ABSTRACT

Generative Adversarial Networks (GANs) learn models of data distributions that can be employed to generate synthetic data with similar characteristics. In this paper, we analyze how GANs can create levels for the iconic first-person-shooter Doom. We designed a framework to train GANs to extract regularities from human-designed levels and trained them using more than a thousand levels, taken from the most extensive online library of Doom content. We trained two GAN models: an unconditional one using only visual information about the levels; a conditional one integrating the same visual information with features capturing high-level structures of the levels. We evaluated the two models by comparing the levels they generated against the human-designed levels used for training. First, we compared the levels using topological metrics inspired by the ones used in robotics showing that the conditional model produces levels more similar to the human-designed ones. Next, we compared the levels using the high-level structural features used for the conditional network, showing that the generated levels are similar to human-designed ones when considering features describing the spatial layout. Finally, we analyzed how much the generation of levels in the conditional network can be controlled using the input features. Our results show that some input features (like the ones related to the number of rooms and the size of the walkable area) influence the generation process. In contrast, the remaining features appear to be ineffective in this respect.

---

1. Introduction

Procedural Content Generation (PCG) provides a broad family of algorithmic methods to generate *functional* content to support game mechanics and gameplay (like, for example, weapons, enemies, and levels) as well as *non-functional* content with a limited impact on actual gameplay dynamics (like for example, textures, sprites, and mod-

dubbed this family of methods Procedural Content Generation via Machine Learning (PCGML). Deep neural networks and, more specifically, Generative Adversarial Networks (GANs) [3] played a significant role in the development of PCGML approaches [4].

In this paper, we analyze how GANs generate levels for Doom [5], the iconic game that helped define the first-person shooter genres, extending the preliminary results in [6]. At first, we selected more than

# Estrutura de um artigo científico

<Título>

<Autores>

<Resumo>

The screenshot shows a scientific article page from the journal "Entertainment Computing". The page has a blue header bar with the journal title and volume information ("Entertainment Computing 46 (2023) 100549"). Below the header, there's a logo for Elsevier and links to the journal homepage and ScienceDirect. The main content area has a white background with a red dashed border around the title and author section. The title is "An analysis of DOOM level generation using Generative Adversarial Networks" by Edoardo Giacomello, Pier Luca Lanzi, and Daniele Loiacono from Politecnico di Milano, Italy. To the right of the title is a "Check for updates" button. Below the title is a "ARTICLE INFO" section with keywords: Deep learning, Generative Adversarial Networks, Video games, Procedural Content Generation. To the right of this is a "ABSTRACT" section containing a detailed description of the research. At the bottom of the page, there are two columns of text: "1. Introduction" and a summary of the paper's contribution.

Entertainment Computing 46 (2023) 100549

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: [www.elsevier.com/locate/entcom](http://www.elsevier.com/locate/entcom)

ELSEVIER

An analysis of DOOM level generation using Generative Adversarial Networks

Edoardo Giacomello, Pier Luca Lanzi, Daniele Loiacono<sup>a</sup>

<sup>a</sup> Politecnico di Milano, Italy

**ARTICLE INFO**

**Keywords:**  
Deep learning  
Generative Adversarial Networks  
Video games  
Procedural Content Generation

**ABSTRACT**

Generative Adversarial Networks (GANs) learn models of data distributions that can be employed to generate synthetic data with similar characteristics. In this paper, we analyze how GANs can create levels for the iconic first-person-shooter Doom. We designed a framework to train GANs to extract regularities from human-designed levels and trained them using more than a thousand levels, taken from the most extensive online library of Doom content. We trained two GAN models: an unconditional one using only visual information about the levels; a conditional one integrating the same visual information with features capturing high-level structures of the levels. We evaluated the two models by comparing the levels they generated against the human-designed levels used for training. First, we compared the levels using topological metrics inspired by the ones used in robotics showing that the conditional model produces levels more similar to the human-designed ones. Next, we compared the levels using the high-level structural features used for the conditional network, showing that the generated levels are similar to human-designed ones when considering features describing the spatial layout. Finally, we analyzed how much the generation of levels in the conditional network can be controlled using the input features. Our results show that some input features (like the ones related to the number of rooms and the size of the walkable area) influence the generation process. In contrast, the remaining features appear to be ineffective in this respect.

**1. Introduction**

Procedural Content Generation (PCG) provides a broad family of algorithmic methods to generate *functional* content to support game mechanics and gameplay (like, for example, weapons, enemies, and levels) as well as *non-functional* content with a limited impact on actual gameplay dynamics (like for example, textures, sprites, and mod-

dubbed this family of methods Procedural Content Generation via Machine Learning (PCGML). Deep neural networks and, more specifically, Generative Adversarial Networks (GANs) [3] played a significant role in the development of PCGML approaches [4].

In this paper, we analyze how GANs generate levels for Doom [5], the iconic game that helped define the first-person shooter genres, extending the preliminary results in [6]. At first, we selected more than

# **Removendo as marcações, temos apenas “palavras” que não deixam o texto adequadamente estruturado.**

Annotation Visualizer: A software visualization tool for code annotations

Phyllipe Lima

Federal University of Itajubá – IMC – UNIFEI

[phyllipe@unifei.edu.br](mailto:phyllipe@unifei.edu.br)

Nathalya Stefhany Pereira

National Institute for Telecommunications -- Inatel

[nathalya.stefhany@gec.inatel.br](mailto:nathalya.stefhany@gec.inatel.br)

Eduardo Guerra

Free University of Bolzano-Bolzen -- UniBZ

[eduardo.guerra@unibz.it](mailto:eduardo.guerra@unibz.it)

Paulo Meirelles

University of São Paulo -- IME-USP

[paulormm@ime.usp.br](mailto:paulormm@ime.usp.br)

## **Abstract**

The Annotation Visualizer (AVisualizer) is a software visualization tool for displaying code annotations distribution in a given target Java-based software system. Implemented as a web application, it can extract annotations usage from the target source code and display it using a hierarchical circle packing approach. Using a dedicated suite of software metrics, it can display size-related information and code responsibilities associated with annotations usage. The tool provides three different views of the analyzed system, each with different granularity. The AVISualizer is a tool that helps improve code comprehension.

**HTML utiliza etiquetas para marcar o conteúdo de uma página web.**



## Estrutura da Etiqueta

---

- As etiquetas utilizam o símbolo “chevron” para defini-las. Esse símbolo também é conhecido como “menor que” e/ou “maior que”.
- Para fechar a etiqueta (*tag*) utilizamos uma barra na abertura.
- Exemplo:
  - <ETIQUETA>conteúdo</ETIQUETA>
- Podemos ter exceções.

# Parágrafo em HTML

---

- Podemos definir um parágrafo com a tag `<p>`.

Conteúdo que será formatado

`< Olá Eu sou um parágrafo >`

*Tag de abertura*

*Tag de fechamento*

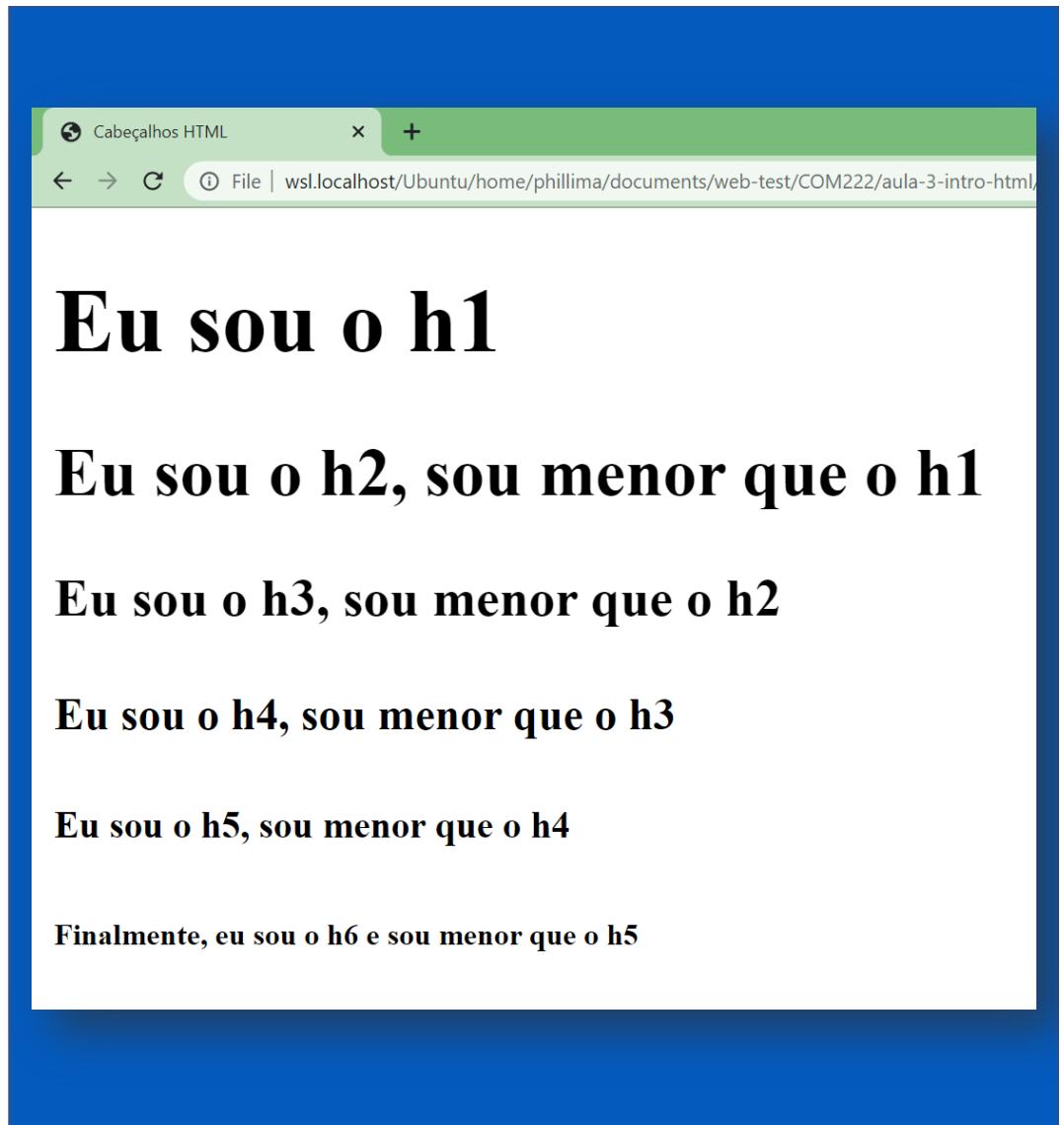
# Cabeçalhos

---

- Podemos indicar cabeçalhos para as diversas seções que queremos colocar na página. Essas tags permitem organizar os títulos das seções.
- Ao total temos 6 (seis) cabeçalhos <h1> até <h6>, e visualmente a primeira característica que notamos é a diminuição da fonte.
- A tag <h1> possui a fonte de maior tamanho.

# Código HTML com cabeçalhos

```
<h1>Eu sou o h1</h1>
<h2>Eu sou o h2, sou menor que o h1</h2>
<h3>Eu sou o h3, sou menor que o h2</h3>
<h4>Eu sou o h4, sou menor que o h3</h4>
<h5>Eu sou o h5, sou menor que o h4</h5>
<h6>Finalmente, eu sou o h6 e
sou menor que o h5</h6>
```



## Formatação rígida

---

- Observe que mesmo com quebra de linha no meio do conteúdo não fez diferença na forma como o navegador interpretou a frase:

“Finalmente, eu sou o h6 e sou menor que o h5”
- A regra de formatação do <h6> entende que todo o conteúdo deve ser formatado como um <h6>.
- Para a quebra de linha, deveríamos explicitamente ter colocado uma marcação/tag para isso.

# Cabeçalhos e a semântica

---

- Não devemos utilizar os cabeçalhos apenas para aumentar/diminuir a fonte.
- Devemos fazer o uso consiente destes, nos preocupando com o real significado do conteúdo que as *tags* irão envolver.
- Uma boa regra é utilizar apenas um único `<h1>` por página. Os demais devem ser utilizados de forma a representar conteúdo hierárquicos destes.

# Estilização é com o CSS

---

- ❑ Adicionalmente, detalhes relacionados apenas a tamanho de fonte devem ser lidados com o CSS.
- ❑ O CSS que é o recurso adequado para ajustes na estilização da página.
- ❑ Usando HTML devemos nos preocupar com “*qual o significado desse conteúdo?*” e assim utilizarmos a tag adequada.

Cabeça  
e  
Corpo

**HTML**



162

# Cabeça e Corpo HTML

---

- ❑ Existe um código mínimo que precisamos para que o navegador consiga interpretar uma página HTML.
- ❑ Vamos dividir o código HTML em duas partes: “Cabeça” e “Corpo”



# HTML5 e a tag <HTML>

---

1. `<!DOCTYPE html>`
2. `<html lang="pt-BR">`
3. `<head>`
4. `</head>`
5. `<body>`
- 6.
7. `</body>`
8. `</html>`

Informa a versão do HTML

Tag de abertura da página HTML. O atributo “lang” está informando que é uma página em português do Brasil.

Tag de fechamento da página HTML

# HTML na Mente

---

1. `<!DOCTYPE html>`
2. `<html lang="pt-BR">`
3. `<head>`
- 4.
5. `</head>`
- 6.
7. `<body>`
8. `</body>`
9. `</html>`



Tudo que se encontra entre `<head>` e `</head>` é a “cabeça”. Esse conteúdo carrega metainformações da página, além do título da aba.

# HTML BodyBuilder <body>

1. `<!DOCTYPE html>`
2. `<html lang="pt-BR">`
3. `<head>`
- 4.
- `</head>`
5. `<body>`
- 6.
7. `</body>`
8. `</html>`

Tudo que se encontra entre `<body>` e `</body>` é o “corpo”. Esse conteúdo é tudo que ficará visível na página.



Mão na  
Massa

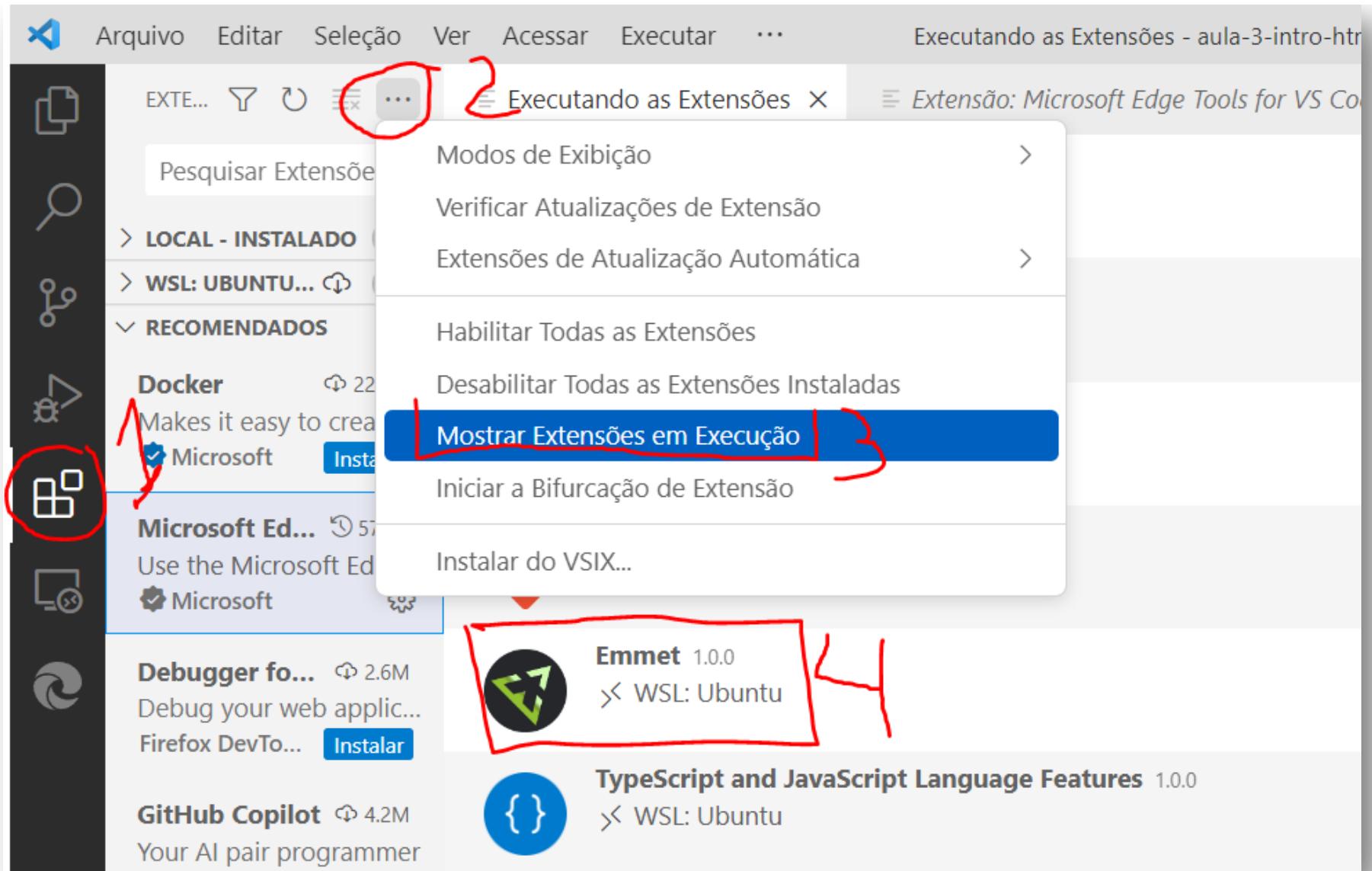
**HTML**



## Mão na Massa - Minha Primeira Página HTML

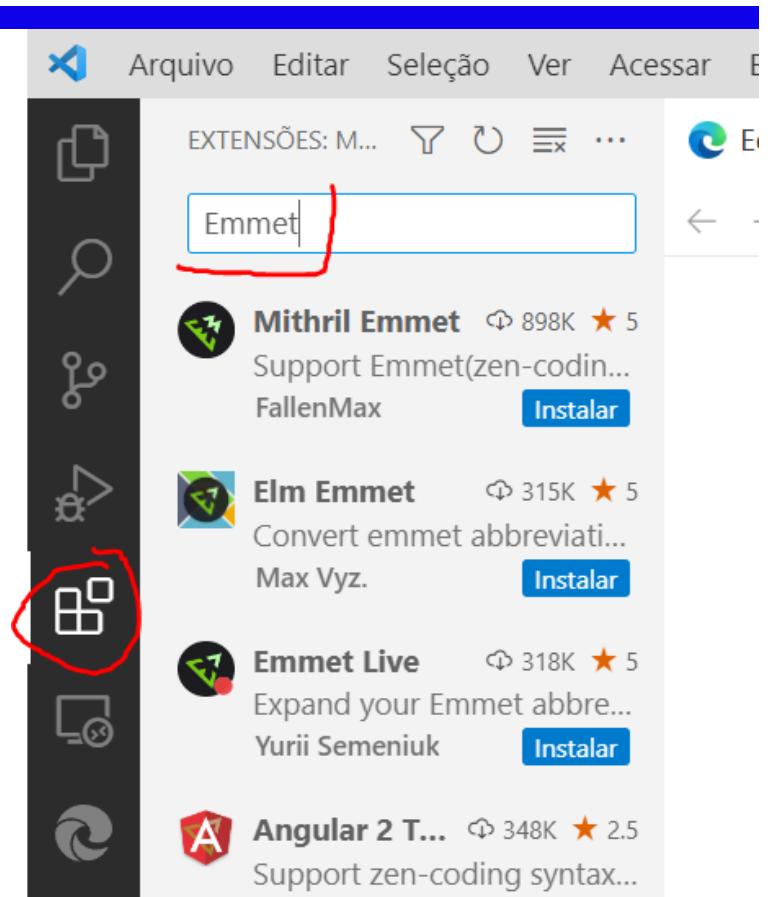
---

- Abra o Visual Studio Code
- Certifique que possui a extensão Emmet.
- Essa extensão facilita, além de outras funcionalidades, a criação do código HTML inicial.
- A extensão, normalmente, é padrão com a instalação do VSCode.



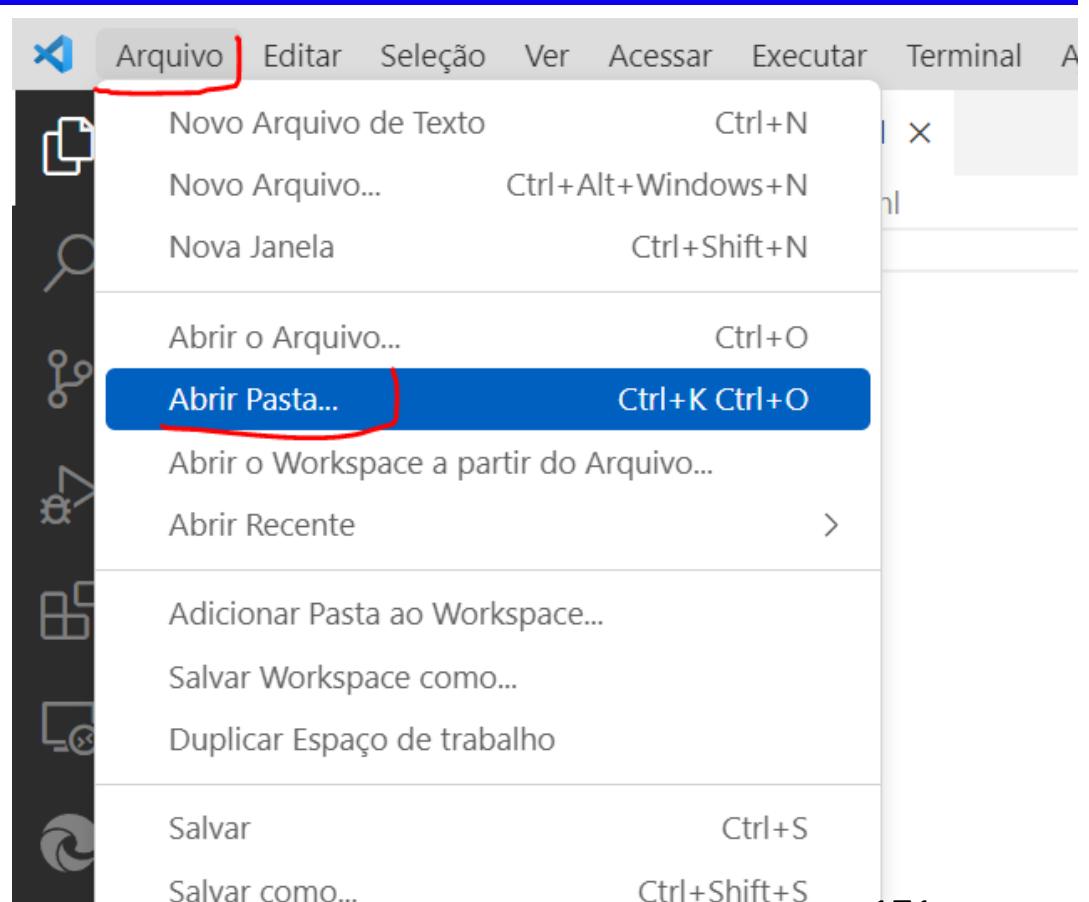
# Instalação Emmet

- Caso seja necessário, instale a extensão “Emmet”



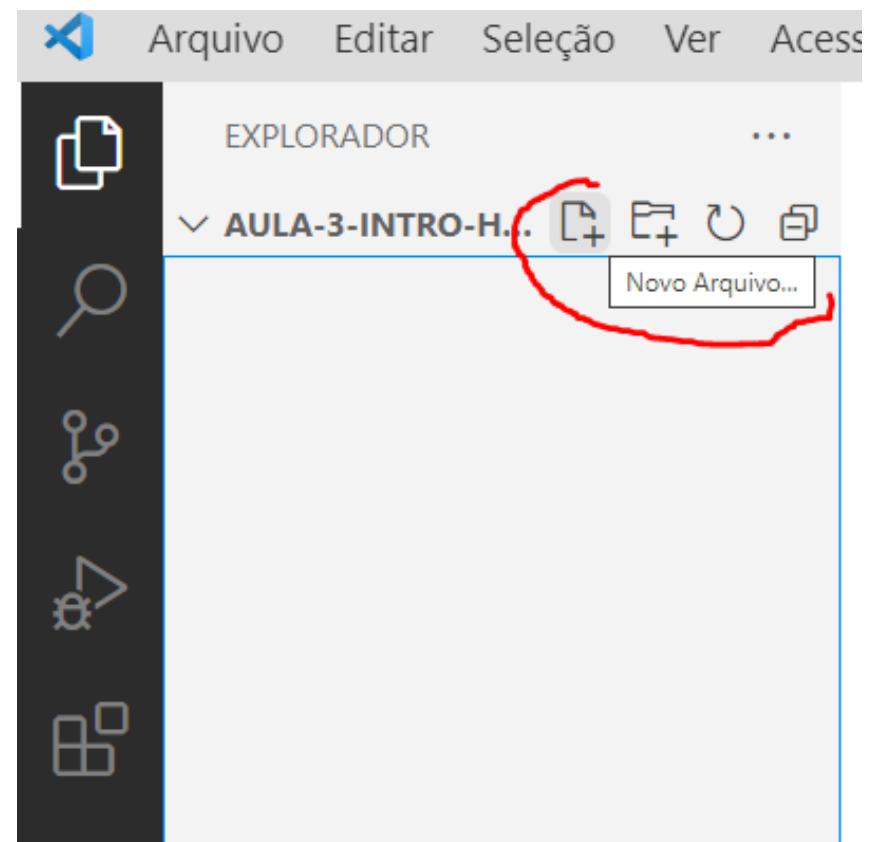
# Criando diretório no VSCode

- No VSCode crie um diretório com nome e local que julgar adequado.



# Criando Arquivos no VSCode

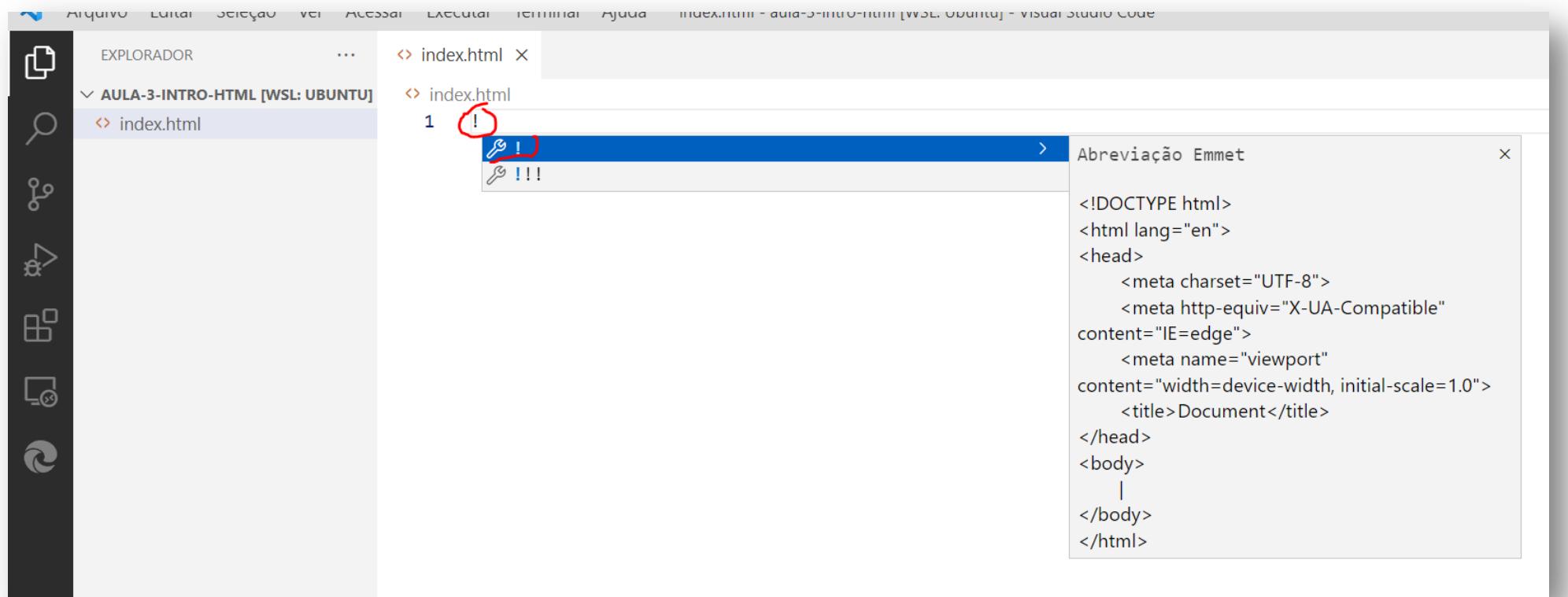
- ❑ No VSCode, dentro da pasta crie um arquivo chamado “index.html”



# O Atalho “!” para Criar Código HTML

---

- Abra o arquivo “index.html” e digite “!” (exclamação).
- Duas opções deverão surgir. Se isso não ocorrer é porque o Emmet não está instalado.
- Selecione a primeira opção como mostrado no próximo slide.



# Código Gerado pelo Emmet

---

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.   <meta charset="UTF-8">
5.   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7.   <title>Document</title>
8. </head>
9. <body>
10.
11.</body>
12.</html>
```

## Exemplo - 1

---

- ❑ Sabendo que todo o conteúdo da página se encontra no <body>, escreva um código HTML que apresente as seguintes características:
  - ❑ Página sobre as disciplinas que está cursando
  - ❑ Escreva um parágrafo descrevendo brevemente cada disciplina.
  - ❑ Como faria isso usando <h1>, <h2> e <p>?

# Exemplo - 1 - Resultado



## **Disciplinas Cursadas em 2023/1**

### **COM222 - Desenvolvimento Web**

Disciplina boa demais. Nela estamos aprendendo sobre desenvolvimento web. Começamos com HTML (inception???) e estamos criando nossa primeira página HTML agora. Uau!

### **CCO016 - Fundamentos de Programação**

Outra disciplina boa demais da conta sô. Aqui eu vi pela primeira vez como fazer um if-else, e depois travei o computador com um while mal desenhado. Loop infinito

### **COM - 937**

Melhor disciplina da UNIFEI. Game dev. Só faz a matrícula e seja feliz.

# Exemplo - 1 - Código

---

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Minhas Disciplinas</title>
</head>
<body>
    <h1>Disciplinas Cursadas em 2023/1</h1>
    <h2>COM222 - Desenvolvimento Web</h2>
    <p>Disciplina boa demais. Nela estamos aprendendo sobre desenvolvimento web. .... Uau!
    </p>
    <h2>CC0016 - Fundamentos de Programação</h2>
    <p>Outra disciplina boa demais da conta sô. Aqui eu vi pela primeira vez como fazer um if-
else, e depois travei o computador com um while mal desenhado. Loop infinito</p>
    <h2>COM - 937</h2>
    <p>Melhor disciplina da UNIFEI. Game dev. Só faz a matrícula e seja feliz.</p>
</body>
</html>
```

# Exemplo - 1 - Código (Análise)

---

- ❑ Apenas um `<h1>` no código informando o título do conteúdo “Disciplinas Cursadas em 2023/1”
- ❑ Em seguida criamos três regiões que podem ser vistas como subseções de `<h1>`, utilizando o `<h2>`. Cada `<h2>` contém a sigla e nome de uma disciplina.
- ❑ Após cada `<h2>` colocamos a descrição da disciplina como um parágrafo usando a tag `<p>`.

## Exemplo - 1 - Execução

---

- ❑ Para executar esse código e ter o navegador renderizando o conteúdo, basta pressionar Ctrl+F5 (Executar Sem Depuração).
- ❑ A cada modificação no código HTML é necessário atualizar a página com F5.
- ❑ Para quebrar o texto, marque a opção com Alt+Z

## Exemplo - 2

---

- ❑ Crie uma página HTML para apresentar uma lista de plataformas de jogos digitais
- ❑ Dicas:
  - ❑ Para preencher o texto use o atalho “lorem” como conteúdo e pressione TAB
  - ❑ Para desenhar uma barra horizontal use a tag <hr>. Essa tag não requer um fechamento.

# Exemplo - 2 - Resultado



The screenshot shows a Microsoft Edge browser window with a blue header bar. The title bar reads "Jogos Digitais". The address bar shows the URL "wsl.localhost/Ubuntu/home/phillima/documents/web-test/COM222/aula-3-intro-html/jogos.html". The page content is titled "Plataformas de Jogos Lorem". Below the title is a horizontal line. The first section is titled "PC Master Race" with the text "Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ullam qui praesentium voluptatum similique ad sed est! Eum voluptate temporibus pariatur in doloremque. Eaque adipisci quisquam preferendis et. Ratione, aperiam voluptates?". The second section is titled "Playstation 5" with the text "Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda blanditiis repudiandae soluta voluptatem dicta vitae quas aperiam at cupiditate labore velit dignissimos a, sint molestias asperiores cumque quisquam autem ex.". The third section is titled "Xbox" with the text "Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum cupiditate fugit soluta porro quod ex ullam fugiat, eligendi doloremque enim ea, necessitatibus culpa sunt quo quasi nam temporibus natus hic.". The browser interface includes standard controls like back, forward, search, and a user profile icon.

# Listas

- —
- —
- —
- —

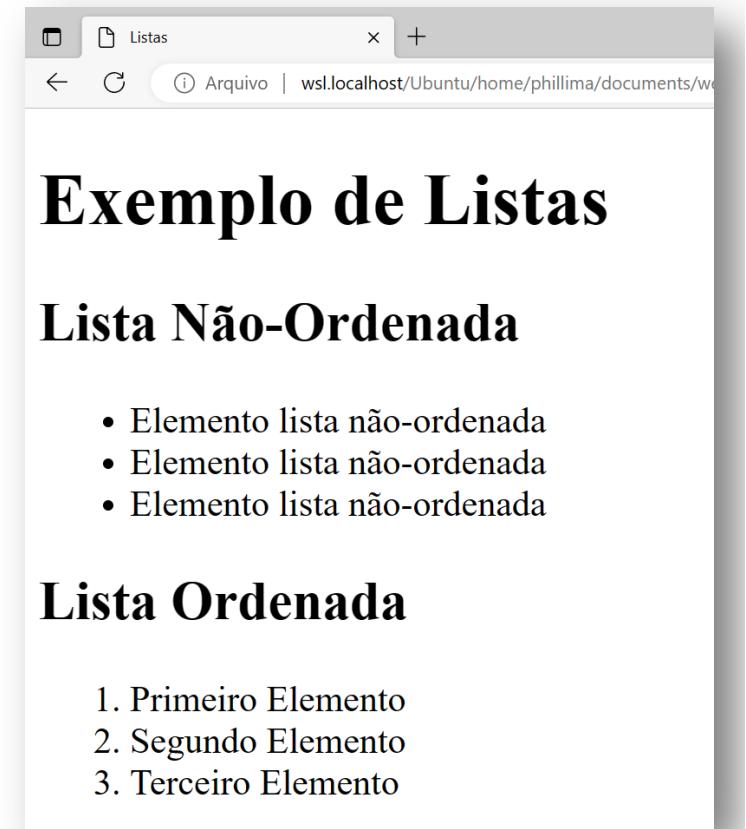
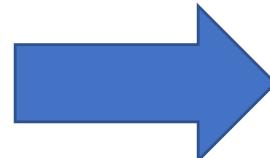
## Listas com HTML

---

- Para listas ordenadas usamos a tag <ol>
- Para listas não-ordenadas usamos a tag <ul>
- Em ambas, definimos cada elemento com a tag <li>

# Listas com HTML - Código

```
<body>
  <h1>Exemplo de Listas</h1>
  <h2>Lista Não-Ordenada</h2>
  <ul>
    <li>Elemento lista não-ordenada</li>
    <li>Elemento lista não-ordenada</li>
    <li>Elemento lista não-ordenada</li>
  </ul>
  <h2>Lista Ordenada</h2>
  <ol>
    <li>Primeiro Elemento</li>
    <li>Segundo Elemento</li>
    <li>Terceiro Elemento</li>
  </ol>
</body>
```



## Listas Aninhadas

---

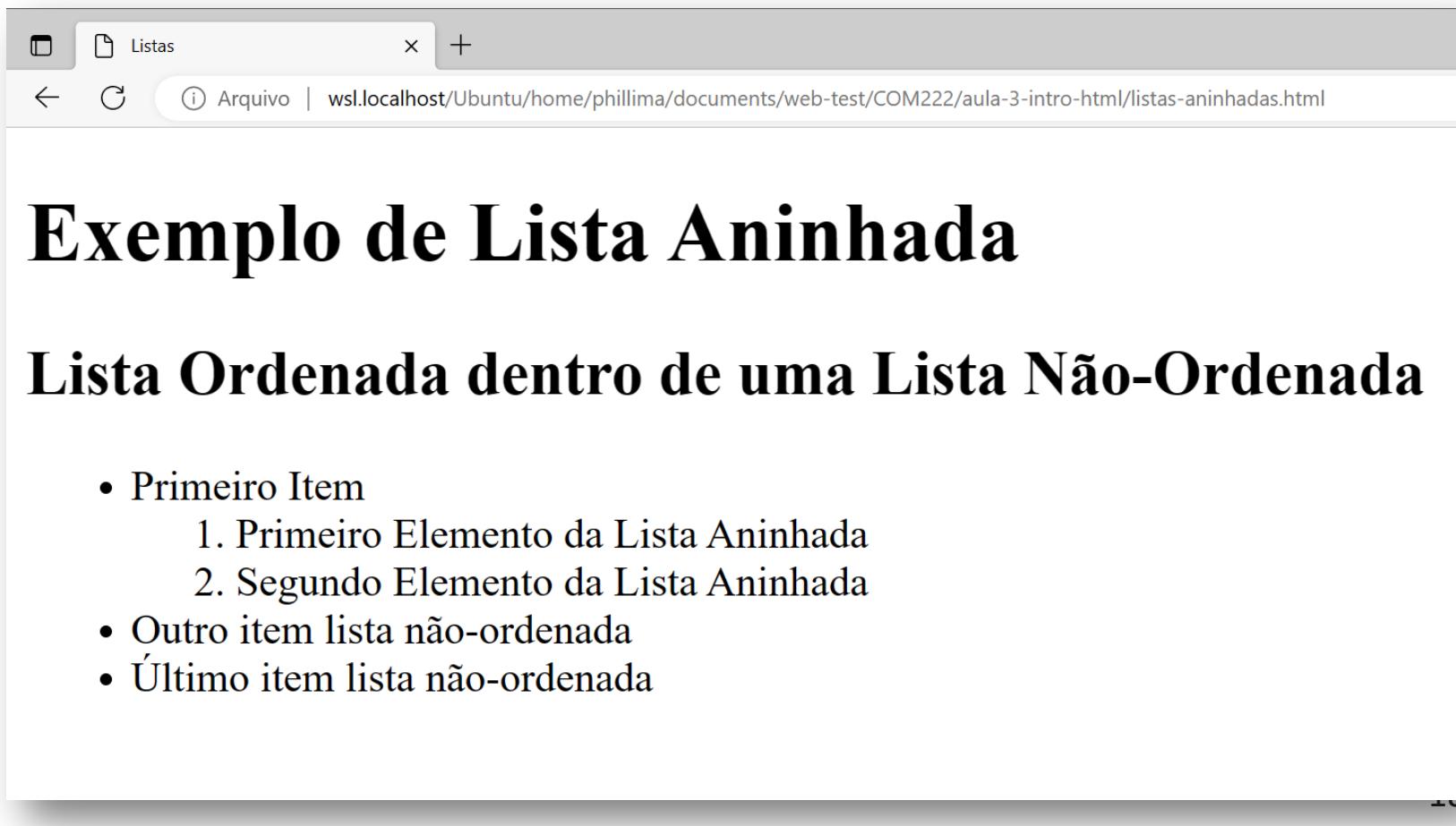
- Para aninhar uma lista, basta que o conteúdo da tag <li> seja outra lista, isto é, utilizando tag <ol> (ordenada) ou <ul> (não-ordenada).
- É possível aninhar lista ordenada dentro de uma não-ordenada e o contrário também.

# Listas Aninhadas - Código

---

```
<body>
    <h1>Exemplo de Lista Aninhada</h1>
    <h2>Lista Ordenada dentro de uma Lista Não-Ordenada</h2>
    <ul>
        <li> Primeiro Item      <!-- Abertura tag <li> -->
            <ol>
                <li>Primeiro Elemento da Lista Aninhada</li>
                <li>Segundo Elemento da Lista Aninhada</li>
            </ol>
        </li>                  <!-- Fechamento da tag <li> ficou no fim -->
        <li>Outro item lista não-ordenada</li>
        <li>Último item lista não-ordenada</li>
    </ul>
</body>
```

# Listas Aninhadas - Demonstração



The screenshot shows a web browser window titled "Listas". The address bar indicates the page is "Arquivo | wsl.localhost/Ubuntu/home/phillima/documents/web-test/COM222/aula-3-intro-html/listas-aninhadas.html". The main content area displays the following text and list:

## Exemplo de Lista Aninhada

### Lista Ordenada dentro de uma Lista Não-Ordenada

- Primeiro Item
  - 1. Primeiro Elemento da Lista Aninhada
  - 2. Segundo Elemento da Lista Aninhada
- Outro item lista não-ordenada
- Último item lista não-ordenada

In the bottom right corner of the browser window, there is a small number "108".

## Exemplo - 3 - Top Spotify

---

- Crie um código HTML que apresente ao menos quatro de sua banda favorita.
- Para cada banda informe ao menos 3 músicas em formato de lista.
- Organize as bandas em ao menos duas categorias de gênero musical

# Exemplo - 3 - Top Spotify - Resultado

---

## Bandas Favoritas do Spotify

---

### Rock

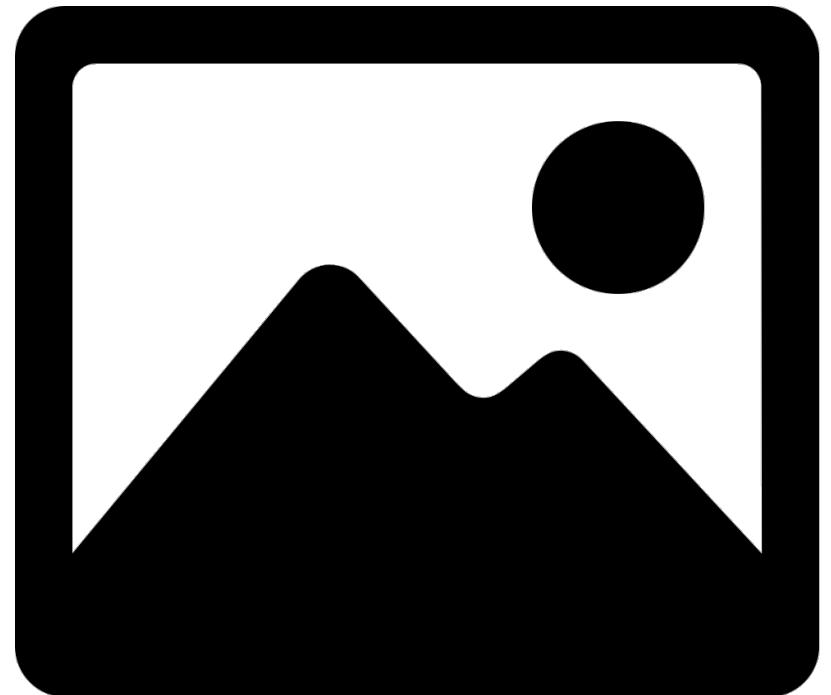
1. Legião Urbana
  - Tempo Perdido
  - Pais e Filhos
  - Faroeste Caboclo
2. Guns N' Roses
  - Don't Cry
  - November Rain

### MPB

1. Chico Buarque
  - Cotidiano
  - João e Maria
  - Roda-Viva
2. Tulipa Ruiz
  - Éfemera
  - Aqui
  - Dois Cafés

# Figuras

<img>



## Figuras com HTML

---

- Para inserirmos uma imagem na página HTML utilizamos a tag <img>.
- Ela possui um atributo chamado “src”, que significa a fonte de onde a figura se encontra. Pode ser um caminho remoto, local ou absoluto.
- A tag <img> não possui fechamento

## Figuras com HTML - A tag <img>

---

- 
- Os atributo são colocados antes do último "chevron", ou "sinal de maior".
- O conteúdo dos atributos são colocados entre aspas duplas.
- O atributo "alt" prove uma descrição para a imagem.

## Figuras com HTML - A tag <figure>

---

- Para aprimorar a experiência, o HTML5 introduziu a tag <figure> que contém também uma legenda.
- A tag <figure> possui fechamento e irá conter a tag <img>, além da tag <figcaption>

## Figuras com HTML - Código

---

```
<figure>
    
    <figcaption>Legenda</figcaption>
</figure>
```

## Exemplo - 4 - Animais

---

- ❑ Crie uma página HTML com ao menos quatro animais e uma imagem para cada.
- ❑ Para cada imagem utilize o atributo alt para fornecer uma descrição
- ❑ As figuras podem ser caminhos remotos ou locais.

# Exemplo - 4 - Animais - Resultado

---

## Página Informativa Sobre Animais

### Cachorro



Caramelo Gente Boa

### Elefante



Elefante Africano caminhando de boas

# Âncoras

<a>



## Criando hiperlinks em HTML

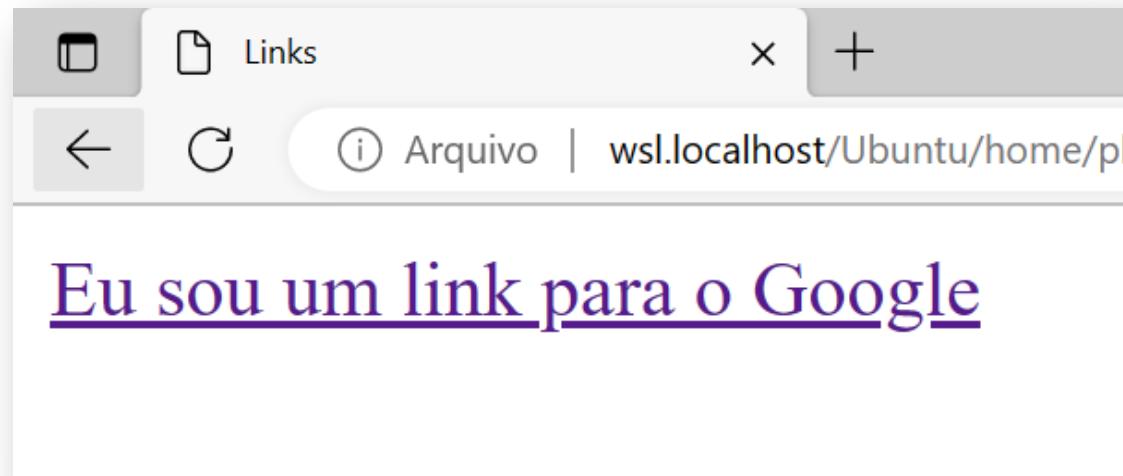
---

- ❑ Usamos a tag `<a>` para definirmos conteúdos linkáveis.
- ❑ O conteúdo pode ser um texto ou imagem.
- ❑ O “`href`” é o atributo que define a URL associada ao conteúdo.
- ❑ Lembre-se de colocar a URL completa caso seja na Web

# Criando hiperlinks em HTML - Código

---

- <a href="http://ww.google.com">Eu sou um link para o Google</a>
- Para abrir o link em outra aba, use o atributo "target" com valor "\_blank"



## Exemplo - 5 - Seleções Ganhadoras de Mundiais

---

- ❑ Crie uma página web que lista as dez seleções que mais ganharam Copa do Mundo FIFA.
- ❑ Informe a quantidade de títulos e um link para a página oficial da respetiva confederação.

# Exemplo - 5 - Seleções Ganhadoras de Mundiais - Resultado

## Seleções Ganhadoras da Copa do Mundo FIFA

### 1. Brasil

O Brasil é o maior campeão, com 5 títulos. Quem gerencia a seleção é a [Confederação Brasileira de Futebol \(CBF\)](#)

### 2. Alemanha

Em seguida temos a Alemanha com 4 títulos mundiais. Quem gerencia a seleção é a [Deutscher Fußball-Bund](#)

### 3. Itália

Empatada com a Alemanha, a Itália também possui 4 títulos mundiais. Quem gerencia a seleção é a [Federazione Italiana Gioco Calcio](#)

### 4. Argentina

Os Hermanos possuem 2 títulos mundiais. Quem gerencia a seleção é a [Asociación del Fútbol Argentino](#)

### 5. França

Para fechar o TOP 5, temos a França com 2 títulos mundiais. Quem gerencia a seleção é a [Fédération Française de Football](#)

# Elementos de Bloco e Linha



## Elementos em Bloco

---

- ❑ Elementos em bloco, ocupam um bloco completo. Isto é, é reservado área em cima e embaixo.
- ❑ Elementos como <p>, <hx>, <li>, e demais são em blocos
- ❑ Podemos utilizar a tag <div> para criar um bloco e agrupar elementos “em bloco”

## Elementos em Linha (inline)

---

- ❑ Elementos em linha, ocupam apenas o espaço necessário para serem renderizados.
- ❑ Elementos como <a> são em linhas
- ❑ Podemos utilizar a tag <span> para criar um bloco e agrupar elementos “em linha”

# Agrupadores

**<div>**

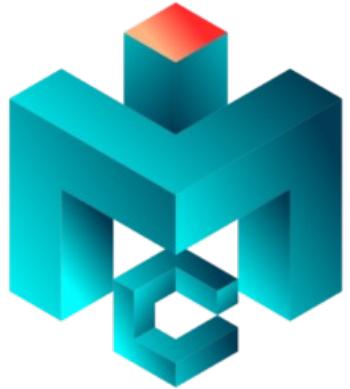
E

**<span>**

## Agrupadores

---

- ❑ Os agrupadores são tags especiais que servem apenas para “agrupar” outros elementos HTML.
- ❑ Para agrupar em blocos utilizamos <div>
- ❑ Para agrupar em linha utilizamos <span>
- ❑ Os agrupadores serão importantes quando combinados com CSS



# Aula – 4.1

## Semântica com HTML

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Semântica no HTML.
- ❑ Elementos Semânticos
- ❑ Demonstração com inspeção

# Semântica e HTML

**HTML**



210



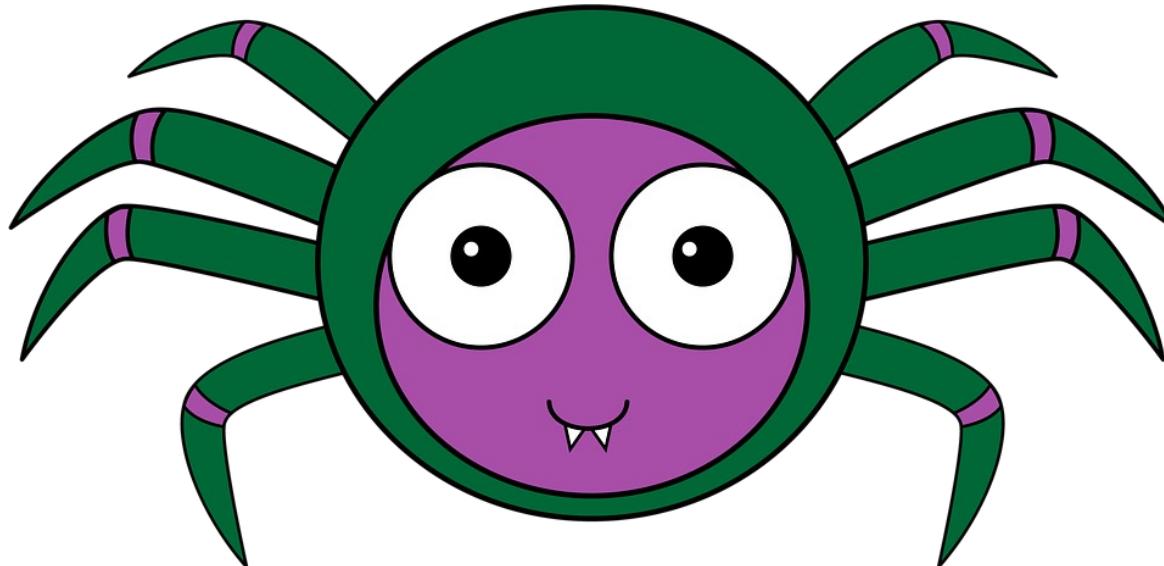
## O que é Semântica?

Em linguagens de programação, semântica trata do significado do trecho de código. O propósito do trecho.

## *Web Crawlers* e a Semântica

---

- Ao usar *tags* semânticas, *web crawlers* podem mais facilmente encontrar informações na páginas.

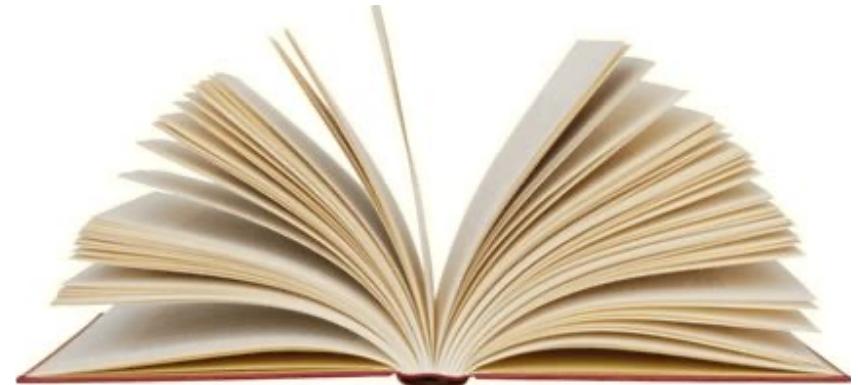


# Legibilidade e a Semântica

---

□ Ao usar *tags* semânticas, a legibilidade do código HTML pode aprimorar.

Facilitando o processo de evolução e manutenção do sistema.



# Acessibilidade e a Semântica

---

- Ao usar *tags* semânticas, leitores de tela podem localizar de forma mais precisa conteúdos que estão na página.



Um passo  
além do  
agrupador

**<div>**

# Usar agrupador <div> quando necessário

---

- É muito comum utilizar a tag <div> para criar blocos e agrupar elementos. Esse é o processo tradicional.
- Mas nem sempre pode ser o ideal. Em alguns casos pode ser interessante substituir a tag <div> por alguma tag semântica.

## A tag <div> pode comprometer o significado

---

- ❑ Ao usar apenas o <div> o significado e propósito do conteúdo sendo exibido pode ficar comprometido.
- ❑ Isso poderá levar a dificuldades para a acessibilidade, legibilidade e web crawlers

# Tags Semânticas

**HTML**



# Implementando a semântica

---

- ❑ Existem aproximadamente 100 tags criadas com objetivos semânticos.
- ❑ A princípio elas poderiam ser substituídas por <div> ou <span> que a página funcionaria da mesma forma. Mas acessibilidade, legibilidade e *crawlers* poderiam ser prejudicados.

## O prato principal – a tag <main>

---

- ❑ A tag <main> deve incorporar o conteúdo principal.
- ❑ Tudo que é essencial e está diretamente relacionado ao conteúdo que se deseja apresentar.
- ❑ Barra de navegação, menu lateral, rodapé, e outros podem não fazer parte do conteúdo principal e portanto ficam fora da tag <main>

# Exemplo com <main>

- Página do MDN apresentando a tag <main>

The screenshot shows a browser window displaying the MDN Web Docs page for the `<main>` element. The page content is highlighted with a red dashed border. The browser's developer tools are open, showing the DOM tree on the right side. The `<main>` element is selected in the tree, and its corresponding CSS styles are visible in the bottom panel.

**MDN Plus** now available in your country! Support MDN and make it your own. [Learn more](#) ☀

mdn web docs References Guides MDN Plus

English (US)

References > HTML > Elements > `<main>`

`<ins>`  
`<kbd>`  
`<keygen>`  
`<label>`  
`<legend>`  
`<li>`  
`<link>`  
`<main>`  
`<map>`  
`<mark>`  
`<marquee>`  
`<menu>`  
`<menuitem>`  
`<meta>`

**<main>**

The `<main>` HTML element represents the dominant content of the `<body>` of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.

**Try it**

HTML Demo: `<main>`

HTML	CSS	RESET
<pre>1 &lt;header&gt;Gecko facts&lt;/header&gt; 2 3 &lt;main&gt; 4   &lt;p&gt;Geckos are a group of usually small,       usually nocturnal lizards. They are found on every continent except       Australia</pre>		
		OUTPUT
		Gecko facts
		Geckos are a group of usually small, usually nocturnal lizards. They are found on every continent except Australia

In this article

- Try it
- Attributes
- Usage notes
- Example
- Accessibility concerns
- Specifications
- Browser compatibility
- See also

Elements Console Sources Network

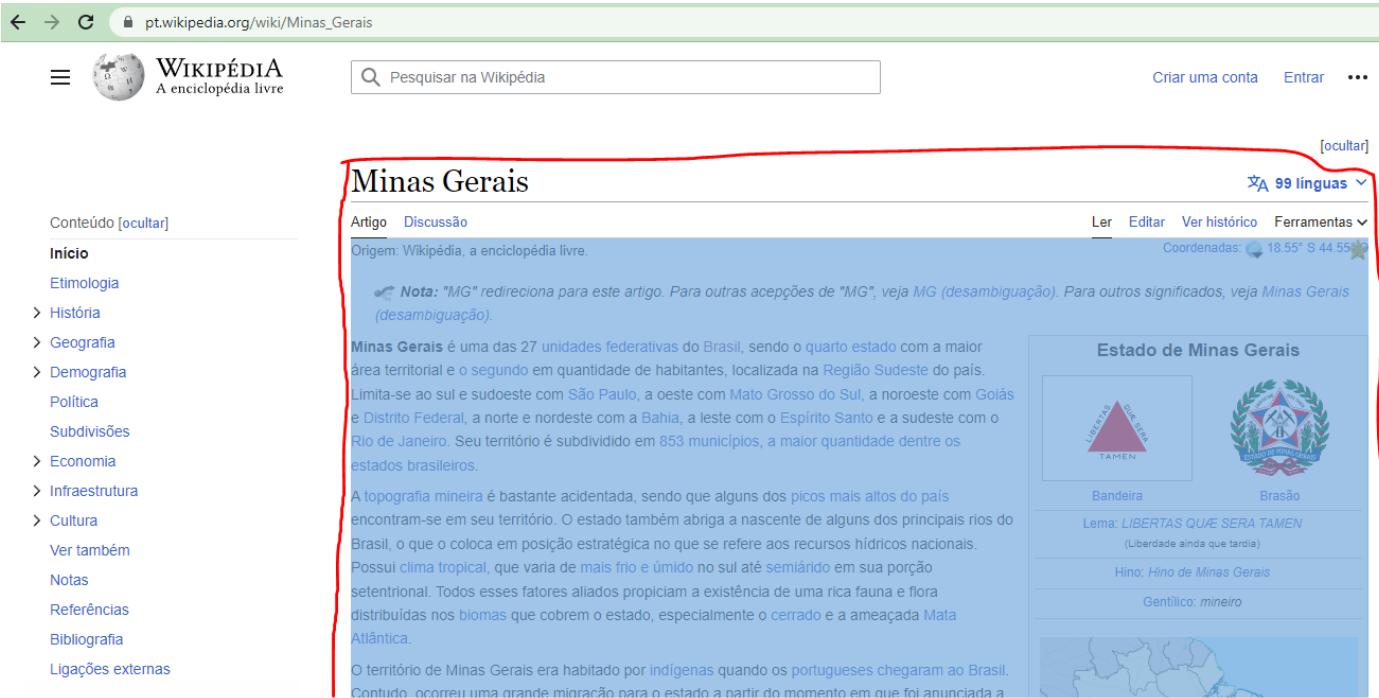
main

Filter

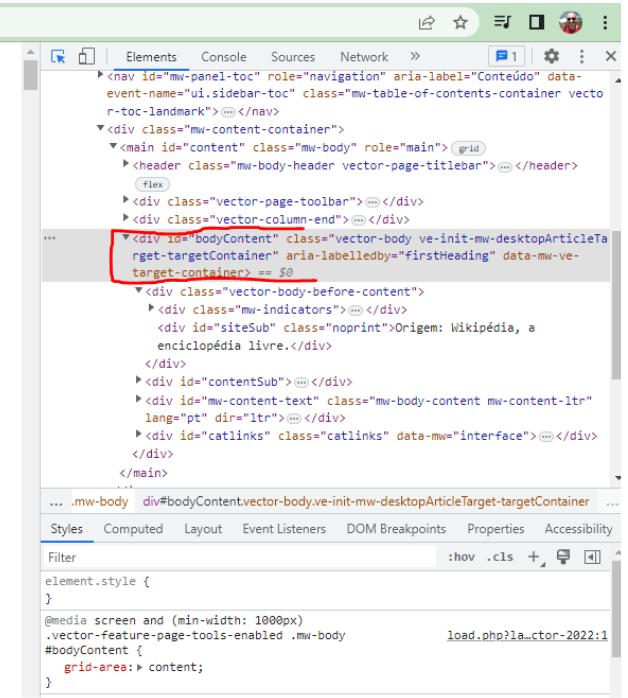
element.style {  
}  
@media screen and (min-width: 769px)  
.main-wrapper .main-content {  
 grid-area: main;  
}  
@media screen and (min-width: 769px)  
.main-wrapper .main-content, .main-wrapper .sidebar, \_document-page.scss:41

# Exemplo sem tag <main>

- ☐ Wikipédia de Minas Gerais: [https://pt.wikipedia.org/wiki/Minas\\_Gerais](https://pt.wikipedia.org/wiki/Minas_Gerais)



The screenshot shows the Wikipedia article for Minas Gerais. A large red box highlights the main content area, which includes the title "Minas Gerais", the lead section with a note about redirects, and the "Estado de Minas Gerais" sidebar. The sidebar contains the state's flag, coat of arms, motto ("LIBERTAS QUÆ SERA TAMEN"), hymn ("Hino de Minas Gerais"), and nickname ("Gentílico: mineiro"). The page also features a map of Brazil with a callout to Minas Gerais.



The developer tools window shows the HTML structure of the page. The main content area is identified by the class "mw-content-container" and the role "main". A specific element within this area, which contains the state's emblem and motto, is highlighted with a red box in the DOM tree. The CSS styles panel at the bottom shows the rule for the main content container.

```
... .mw-body { div#bodyContent{vector-body}.ve-init-mw-desktopArticleTarget-targetContainer ... }
```

```
element.style { }
```

```
@media screen and (min-width: 1000px) ,vector-feature-page-tools-enabled .mw-body { #bodyContent { grid-area: content; } }
```

## Exemplo 1 – Minhas Disciplinas 2.0

---

- ❑ Escreva um código HTML para listar as disciplinas cursadas em 2023/1
- ❑ Use a tag semântica quando adequado.

# Exemplo 1 – Código (com Lorem)

---

```
<body>
  <main>
    <h1>Minhas Disciplinas</h1>
    <hr>
    <h2>COM222 - Programação Web</h2>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Laborum  
aspernatur quasi dolor corporis iusto! Totam eveniet error maxime eaque velit  
nesciunt tempora, harum cumque tempore nobis, soluta, sapiente nemo! Nihil?  
    </p>
    <h2>CC0016 - Fundamentos de Programação</h2>
    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Quos suscipit  
nesciunt iusto eos, et, officia perspiciatis minima ex cumque vero nemo laboriosam  
voluptatibus! Dolorem quas iste totam suscipit debit is rem.</p>
  </main>
</body>
```

## Navegando pelo mar – a tag <nav>

---

- ❑ A tag <nav> deve incorporar o conteúdo relacionado a barra de navegação com links para:
  - ❑ Páginas externas
  - ❑ Conteúdo na própria página
- ❑ Normalmente fica fora do conteúdo principal.
- ❑ Menus, índices de conteúdos, etc.

# Exemplo com <nav>

Wikipédia de Minas Gerais: [https://pt.wikipedia.org/wiki/Minas\\_Gerais](https://pt.wikipedia.org/wiki/Minas_Gerais)

Screenshot of the Wikipedia page for Minas Gerais. The page title is "Minas Gerais". The sidebar on the left contains links to various topics like História, Geografia, Demografia, Política, Subdivisões, Economia, Infraestrutura, Cultura, and Ligações externas. The main content area includes a note about the article being a redirect, information about the state's location and subdivisions, a section on topography and climate, and details about its history and symbols (Bandeira and Brasão).

Screenshot of the browser developer tools showing the HTML structure of the page. A red box highlights the <nav> element with the ID "mw-panel-toc" located in the header area. The element has the attribute "aria-label='Conteúdo'" and "data-event-name='ui.sidebar-toc'". It contains a <div> with the class "vector-pinned-container" which holds the "vector-toc" and "vector-toc-pinnable-element". Below it is a <ul> with the class "vector-toc-list" containing several <li> items, each with a unique ID such as "toc-Etimologia" and "toc-História".

## Exemplo 2 – Minhas Disciplinas Navegantes

---

- Escreva um código HTML para listar as disciplinas cursadas em 2023/1
- Use a tag semântica quando adequado.
- Coloque uma barra de navegação superior para redirecionar a : Página da UNIFEI, Página do IMC, Página Sobre fictícia. Coloque um “voltar” no “Sobre”.

## Exemplo 2 – Código (com Lorem)

---

### □ Trecho com <nav>

```
<nav>
  <hr>
  <ul>
    <li><a href="https://unifei.edu.br/">UNIFEI</a></li>
    <li><a href="https://imc.unifei.edu.br/">IMC</a></li>
    <li><a href="sobre.html">Sobre</a></li>
  </ul>
</nav>
```

### □ Código completo nas anotações

## *Seções* da Tarde – a tag <section>

---

- ❑ A tag <section> representa um seção genérica de conteúdo em uma página. É usada quando não há uma tag mais específica.
- ❑ A tag <section> normalmente é acompanhada de um cabeçalho <h1>..<h6>

# Exemplo com <section>

❑ Página do IMC: <https://imc.unifei.edu.br/>

The screenshot shows a web browser displaying the IMC website (<https://imc.unifei.edu.br/>). The page features a teal header with navigation links for 'Notícias' and 'Contatos'. Below the header, there is a section titled 'Nossa missão' containing a photograph of a building and a text block about the institute's mission. The browser's developer tools are open, specifically the 'Elements' tab, which shows the HTML structure of the page. A red box highlights a specific section element in the DOM tree.

Nessa página você encontra

Notícias

Contatos

**Nossa missão**

A missão do IMC consiste na formação de recursos humanos de alto nível no que tange ao ensino de graduação e de pós-graduação por meio da produção e disseminação do conhecimento nas áreas de Matemática e Computação, bem como da promoção de ações culturais e de inserção social, do desenvolvimento de pesquisas e da extensão de serviços à comunidade. Visando a esses objetivos, o Instituto possui comprometimento constante com o desenvolvimento científico, tecnológico e social da região de Itajubá, do Estado de Minas Gerais e do Brasil."

```
light-default" data-id="1f4d6b5" data-element_type="section">> @) </section>
> <div class="elementor-element elementor-element-3713e a4 elementor-widget elementor-widget-spacer" data-id="3713e4" data-element_type="widget" data-widget_type="spacer-default"> @) </div>
> <div class="elementor-element elementor-element-2b2c8 35 elementor-widget-divider-view-line elementor-widg et elementor-widget-divider" data-id="2b2c835" data-element_type="widget" data-widget_type="divider.default"> @) </div>
> <section class="elementor-section elementor-inner-section elementor-element elementor-element-26e2c40 elem entor-section-boxed elementor-section-height-default elementor-section-height-default" data-id="26e2c40" data-element_type="section"> @) </section>
> <section class="elementor-section elementor-inner-section elementor-element elementor-element-71d0c02 elem entor-section-boxed elementor-section-height-default elementor-section-height-default" data-id="71d0c02" data-element_type="section"> ... </section> == $0
> <div class="elementor-element elementor-element-e2c3b bf elementor-widget elementor-widget-spacer" data-id="a2c3hhF" data-element_type="widget" data-wi ent.elementor-element-71d0c02.elementor-section-boxed.elementor-section-height-default ...
... Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility
Filter
element.style {
}
.elementor-widget-wrap .elementor-element {
    width: 100%;
}
.elementor-element {
    --widgets-spacing: 20px;
}
```

## Exemplo 3 – Minhas Seções Disciplinas

---

- ❑ Faça uma atualização no Exemplo 2 e coloque as disciplinas como uma seção.
- ❑ Use a tag semântica <section>
- ❑ Código nas anotações.

# Álbum de figurinhas – a tag <figure>

---

- ❑ A tag <figure> deve ser usada quando se deseja incluir conteúdo de imagem que pertence ao fluxo principal da página.
- ❑ Normalmente tem uma legenda.
- ❑ A legenda e a imagem são vistas como uma única unidade inseridas na tag <figure>

## Exemplo - <figure>

---

```
<figure>
  
  <figcaption>Legenda para a Figura</figcaption>
</figure>
```

- A tag <figure> já foi trabalhada nas aulas anteriores

## Rodando o pé – a tag <footer>

---

- A tag <footer> deve ser usada quando se deseja criar uma seção que se caracteriza como rodapé da página. Não usamos <section> pois temos uma tag mais específica.
- Normalmente fica fora da <main>, mas não é uma regra.

# Exemplo com <footer>

- ❑ Página do Integra UNIFEI: <https://integraunifei.com/>

The screenshot displays the footer area of the [Integra UNIFEI](https://integraunifei.com/) website. The footer is divided into three main sections: **REALIZAÇÃO**, **APOIO**, and **ORGANIZADORES**. The **REALIZAÇÃO** section features the logos of **Integra** and **D&E**. The **APOIO** section features the logos of **UNIFEI** and **PROEX**. The **ORGANIZADORES** section lists the names of the organizing team members. A red bracket on the left side of the footer highlights the structure of the footer element. To the right, a screenshot of the Chrome DevTools shows the footer's CSS code, illustrating the use of Flexbox for layout.

```
<!DOCTYPE html>
<html lang="en">
  <head> ...
  </head>
  <body>
    ...
    <div id="__next">
      ...
      <main class="__variable_287515 font-sans">
        ...
        <section id="sobre" class="w-screen max-w-full flex flex-col justify-between bg-gradient-to-t from-line-500 via-[#4ADE80] to-[#86EFAC] md:flex-x-row md:gap-10"> ...
      </main>
    </div>
    <script id="__NEXT_DATA__" type="application/json"> ...
    </script>
    <next-route-announcer> ...
  </body>
</html>
```

Filter: .element.style { } .bg-sky-600 { ... } .justify-center { ... } .items-center { ... } .max-w-full { ... } .w-screen { ... }

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Automatic in place pretty print The Sources panel now automatically pretty-prints minified source files in place.

Console What's New

Highlights from the Chrome 111 update

new

## Exemplo 4 – Meu Pé Rodando

---

- ❑ Adicione um <footer> na página sobre as disciplinas.
- ❑ Redirecione para alguma página profissional
- ❑ Exemplo:
  - ❑ “Feito com <3 por <sua empresa/pagina/etc>”.
  - ❑ Como colocar um coração em HTML?

Feito com ❤ por [GitHub](#)

## Entrando de cabeça – a tag <header>

---

- ❑ A tag <header> deve ser usada quando se deseja introduzir um conteúdo.
- ❑ Pode ser utilizada como descendente de <nav>, <main>, <section> e outros.
- ❑ Pode conter informações de autoria, data/hora, imagem e outros.

# Exemplo com a tag <header>

- ❑ Página do Integra UNIFEI: <https://integraunifei.com/>

The screenshot shows a web page for the EURECA! exhibition at UNIFEI. The header features the UNIFEI logo and a 'D&E' icon. Below the header is a large green banner with a blue star containing a cartoon character and the text 'EURECA! UNIFEI'. The main content area has a dark blue background with white text describing the exhibition's purpose. To the right, the browser's developer tools are open, specifically the Elements tab, which displays the HTML structure of the page. A red arrow points from the text 'Exemplo com a tag <header>' to the 'header' element in the DOM tree.

```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body>
    ... <div id="__next" style="background-color: #fff; color: #000; padding: 10px; font-family: sans-serif; font-size: 14px; margin-bottom: 20px;">
      <header class="h-20 bg-white flex justify-between">...</header>
      <div id="eureca" class="flex flex-col mobile:h-[330px] sm:h-full mobile:e-block justify-center items-center bg-green-400">...</div>
      <div id="projetosAcademicos" class="grid grid-cols-12 gradient-projetos">...</div>
      <div id="ejs" class="grid grid-cols-12 gradient-empresas">...</div>
    </div>
    <script id="__NEXT_DATA__" type="application/json">...</script>
    <next-route-announcer>...</next-route-announcer>
  </body>
</html>
```

## Exemplo 5 – Encabeçando as disciplinas

---

- Adicione um <header> na introdução do conteúdo na página sobre as disciplinas.
- Adicione uma imagem com logo da UNIFEI

## Deixa isso de lado – a tag <aside>

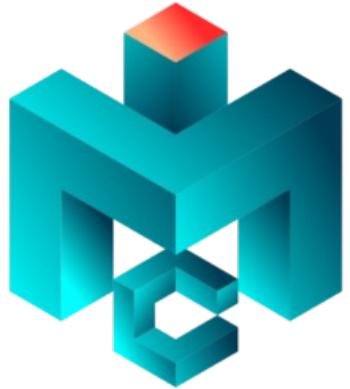
---

- ❑ A tag <aside> é utilizada para conter parte de conteúdo que está indiretamente relacionado ao conteúdo principal. Muito comumente utilizada englobando barra de navegação lateral.

## E muitas outras....

---

- ❑ Como dito, existem cerca de 100 tags semânticas.
- ❑ É necessário constantemente consultar para entender a melhor forma de utilizar
- ❑ Fonte MDN: <https://developer.mozilla.org/en-US/docs/Glossary/semantics>
- ❑ Com CSS seu uso será ainda melhor percebido.



# Aula – 4.2

## Formulário com HTML

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Formulários
- ❑ Entradas (<input>)
- ❑ Rótulos (<label>)
- ❑ Parâmetros (query parameters)

# Formulários

## <forms>





## O que são formulários?

Usamos formulários para criar uma seção que nos permite submeter informação para um servidor.

## O que faz a tag <form> ?

---

- A tag <form> é uma das mais importantes e utilizadas de forma recorrente em páginas web.
- A tag não adiciona nenhuma estilização diferente, mas se comporta como um “wrapper” que envolve os elementos do formulário.

# O que a tag <form> oferece?

---

- ❑ Com a tag <form> conseguimos usar atributos especiais que definem a ação e para onde as informações serão submetidas.
- ❑ Dentro da tag <form>, colocamos outras tags que irão, visualmente, compor o formulário como:
  - ❑ Entradas de texto, botões, rótulos, e etc.

# Exemplo tag <form>: Amazon

## ❑ Página de login da Amazon

The screenshot shows the Amazon login page. At the top is the Amazon logo and the URL 'amazon.com.br'. Below it is a light blue header with the text 'Fazer login'. Underneath is a form with a light blue background. It contains an input field labeled 'E-mail ou número de telefone celular' and a green 'Continuar' button below it. Below the button is a note about accepting terms and conditions. At the bottom left is a link 'Precisa de ajuda?' and at the bottom right is a tooltip for the form element.

form.auth-validate-form.auth-re  
al-time-validation.a-spacing-no  
ne 350 x 317.59

The screenshot shows the browser's developer tools with the 'Elements' tab selected. The DOM tree is displayed, and a red box highlights the <form> element located within a section. The form has a name of 'signIn', a method of 'post', and an action of 'https://www.amazon.com.br/ap/signin'. It is associated with classes 'auth-validate-form', 'auth-re', and 'a-spacing-no'.

# Exemplo tag <form>: SIGAA

## □ Página de login do SIGAA

The screenshot shows the SIGAA login page. At the top, there is a logo and the text "SIGAA Sistema Integrado de Gestão de Atividades Acadêmicas". Below this, a yellow banner reads: "Caro visitante, Para acessar a área de inscritos em cursos e eventos de extensão é necessário realizar o login no sistema. O enciamento será possível se inscrever nos curso e eventos abertos, bem como os já realizadas." A form with the ID "formLoginCursosEventosExtenso" is displayed, containing fields for "E-mail" and "Senha" with an "Entrar" button. Below the form are links for "Esqueci minha senha" and "Ainda não possuo cadastro!". At the bottom, there is a link "[<< voltar ao menu principal](#)".

The screenshot shows the browser's developer tools with the HTML code for the login form. The code is as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" class=" ext-strict">
<head> ...
<body id="ext-gen6" class=" ext-safari">
  <div id="container">
    <div id="container-inner">
      <div id="cabecalho"> ...
        <div id="corpo">
          <style type="text/css"> ...
            <h2>Área de Login para Acesso à Inscrição em Cursos e Eventos de Extensão</h2>
          <div class="descricaoOperacao"> ...
            <form id="formLoginCursosEventosExtenso" name="formLoginCursosEventosExtenso" method="post" action="/sigaa/public/extenso/loginCursosEventosExtenso.jsf;jsessionid=9EDDC787A9DC12D795E6C73C16AAF88A.sigaa05" enctype="application/x-www-form-urlencoded"> ...
          <br>
          <br>
          <div style="width: 80%; text-align: center; margin: 0 auto;"> ...
            <br>
            <div class="clear"></div>
          </div>
        ... div#container div#container-inner div#corpo form#formLoginCursosEventosExtenso ...
      Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility
      Filter :hov .cls + [ ]
```

# Atributos do formulário

---

- O atributo “action” determina para onde os dados do formulários serão enviados.
- O atributo “method” determina o tipo de verbo HTTP que será utilizando no envio. Isso será visto com mais detalhes em aulas posteriores.

## Atributo *action* em ação

---

- A tag <form> inicialmente se apresenta como:

<form action=""></form>

- O valor do “action” é o endereço para onde os dados do formulário serão submetidos.
- Para início vamos inspecionar páginas comerciais

## Atributo *action* em ação na página Google

- O valor de “action” é “/search”.
- Digite “html” (ou qualquer outro conteúdo) e pressione “enter”.

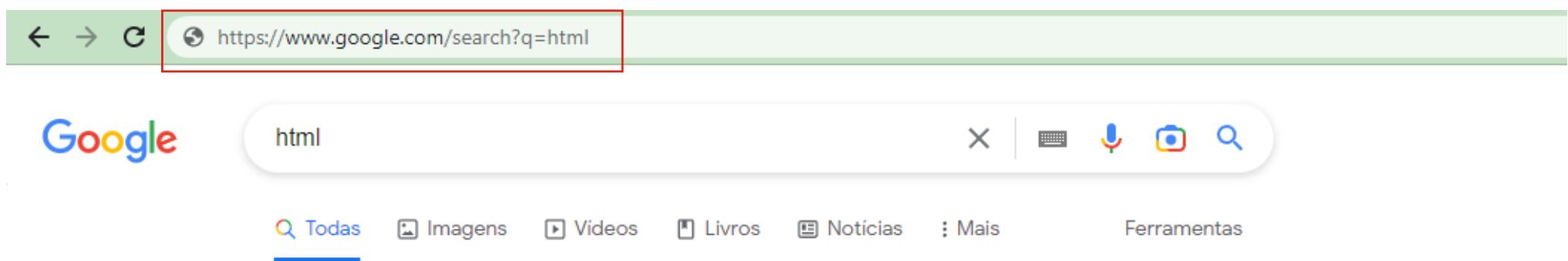


```
<div class="o3j99 ikrT4e om7nvf">
  <style data-iml="1679971187829">.om7nvf{padding:20px}</style>
  <dialog class="spch-dlg" id="spch-dlg">@@</dialog>
...
<form action="/search" autocomplete="off" method="GET" role="search">
  <div jsmodel="sj77Re P9Kqfe" jsdata="MuIEvd;_;Bx2EmE">
    <div jscontroller="cnjECF" jsmodel="VYkzu a4L2gc QwfZb BFDhle gx0hc
      b TnHsd icvlie " class="A8SBwf emcav" data-adhe="false" data-alt="f
      alse" data-hp="true" data-mlcfn="false" data-mle="false" data-mof="fa
      lse" jsdata="Lvp1cb;_;" jsaction="1XGRIid:w3Wsmc;Dkpm0b:d3sQLd;IQoav
      d:dFyQEf;XzZZPe:jI3wzf;Aghsf:AVsn1b;iHd9U:Q7Cnrc;f5hEH:G0jgYd;vmxU
      b:j3bJnb;HpoZje:FrsEXe;R2c50LuRugf;qiCkJd:ANldic;htMNz:SNIJTd;Noe9
      L:Hlg3;uGoIkdk:epUokb;zLdLw:eagBS;H9muVd:J4e6lb;hBEIVb:nUZ9le;rcuQ6
      b:npt2md">
      <style data-iml="1679971187829">@@</style>
      <style data-iml="1679971187829">
        .CKb9sd{background:none;display:flex;flex:0 0 auto}</style>
    <div class="RNnxgb" jsname="RNnxgb"> flex
      <div class="SDKEP"> flex
        <div class="ib1pc" jsname="uFMOof">@@</div> flex
        <div jscontroller="vZr2rb" class="a4BIC" data-mle="false"
          data-mnr="10" jsname="gLfvf" jsaction="h5M12e;input:d3s0ld;blu
          ...
        </div>
      </div>
    </div>
  </div>
</form>
```

Resultado da busca no Google preenchendo o formulário

---

- A URL contém "/search" ao final da página.
- O símbolo "?q=html" indica o parâmetro que foi passado com a submissão do formulário



## Analisando o valor do atributo “action”

---

- ❑ O valor de “action” é um caminho relativo. Isto é, “search” é uma rota que se encontra na raiz do servidor da página [www.google.com](http://www.google.com)
- ❑ Ao final da URL, temos os “parâmetros” passados como chave-valor. Onde, nesse exemplo, a chave é “q” e o valor é “html”.
- ❑ Podemos passar vários parâmetros pela URL. Mas em alguns casos, como login, não é interessante passar dados pessoais abertos na URL. Veremos uma forma melhor de fazer isso.

## Usando a URL para fazer a busca

---

- ❑ Seria possível ignorar o formulário e a página, indo diretamente na URL e fornecendo os parâmetros de busca.
- ❑ Exemplo: Buscar pela palavra “computador”.
  - ❑ Podemos digitar diretamente a URL
  - ❑ [www.google.com/search?q=computador](http://www.google.com/search?q=computador)
  - ❑ Pois a página do Google usa a action como “search” e o nome do atributo é “q”.
  - ❑ Veremos o atributo “name”

## Página com <form>

---

- Ao criamos uma página HTML com o seguinte formulário básico:

```
<form action=""></form>
```

- Nada irá aparecer na página, pois precisamos definir as entradas

# Entradas

## <input>

## Entradas com a tag <input>

---

- ❑ A tag <input> permite uma grande variedade de tipos de entradas dentro de um formulário.
- ❑ Ao modificar o parâmetro “type”, o elemento se adapta para representar o tipo atribuído.

## Entrada de texto com <input>

---

- O padrão é “entrada de texto”
- Pode ser colocado de forma explícita ao marcar “type” como “text”

<input type="text">



## Adicionando um *placeholder* (espaço reservado)

---

- Com o atributo placeholder, podemos deixar instruções para auxiliar o que deve ser colocado na entrada.

```
<input type="text" placeholder="nome de usuário">
```



nome de usuário

## Adaptando para entrada de senhas

---

- ❑ Modificando o atributo type para “password” o navegador irá ocultar o conteúdo sendo digitado.
- ❑ Isso pode auxiliar na proteção de senhas

```
<input type="password" placeholder="senha">
```

The image displays two rectangular input fields side-by-side. The top input field contains the word "senha" in a light blue, semi-transparent font, which is the standard placeholder text for a password input field. The bottom input field contains six black dots (".....") followed by a vertical cursor bar, illustrating how a password is visually represented when the type attribute is set to "password".

## Escolhendo cores

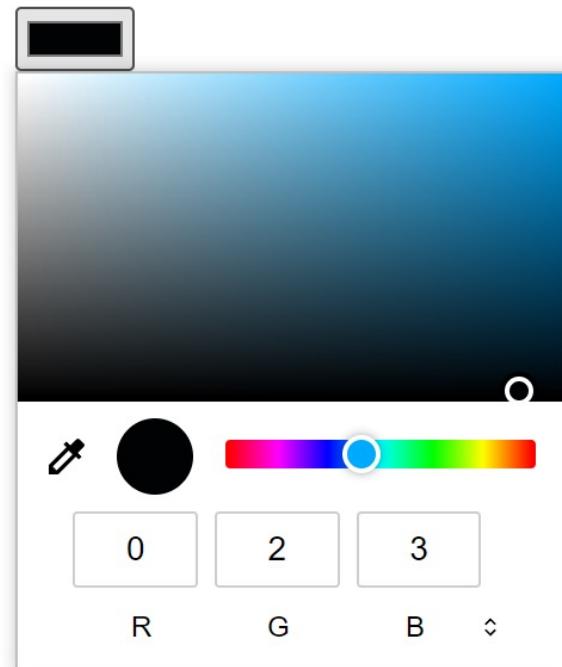
---

- Modificando o atributo type para “color” o navegador apresenta a entrada como um “color picker”. Permite visualmente buscar uma cor.
- Também há possibilidade de modificar a cor, inserindo valores numéricos das caixas RGB
- É possível modificar para hexadecimal.

## Escolhendo cores - navegador

---

```
<input type="color" placeholder="senha">
```



## Rótulos com a tag <label>

---

- ❑ A tag <label> aprimora a legibilidade e acessibilidade de entradas em um formulário. Com ele é possível associar um rótulo a uma entrada.
- ❑ Para isso precisamos usar dois novos atributos
  - ❑ id no elemento <input>
  - ❑ for no elemento <label>

## Associando rótulos as entradas

---

```
<label for="usuario">Nome de Usuário</label>
<input type="text" placeholder="usuário" id="usuario">
```

- ❑ Usando o atributo for em “label” especificamos “qual” entrada esse rótulo está associado.
- ❑ Na entrada, <input>, precisamos definir valor do atributo id como o mesmo colocador no “for”
- ❑ O valor de id precisa ser único na página.

## Exemplo 1 – Cadastro de Vagas em Rep

---

- ❑ Crie um formulário visando auxiliar colegas que acabaram de chegar em Itajubá e buscam moradias.
- ❑ Nesse formulário solicite os seguintes dados:
  - ❑ Nome, curso, idade, cidade de origem
- ❑ Como não há botão o formulário não executará nenhuma ação.
- ❑ Coloque os elementos alinhados verticalmente usando os elementos de bloco.

## Exemplo 1 – Cadastro de Vagas em Rep - Resultado

---

Nome Completo

Curso

Idade

Cidade de Origem

# Submetendo formulário com botões

---

- ❑ Para submeter informações de um formulário precisamos preencher o atributo action no <form> e adicionar um botão dentro do <form>
- ❑ A princípio, qualquer botão dentro do forma irá submeter os dados.

## Exemplo 2 – Cadastro de Vagas em Rep

---

- ❑ Utilizando o exemplo 1. Adicione uma action com valor “cadastro”. (isso não é real, apenas demonstrativo)
- ❑ Adicione um botão com conteúdo “Cadastrar”
- ❑ O type do botão pode ser omitido, pois por padrão será “submit”

## Exemplo 2 – Resultado

---

- Ao preencher o formulário e clicar no botão “Cadastrar” a página será redirecionada, e provavelmente teremos um erro.
- Não existe a rota “/cadastro”
- Mas o formulário foi submetido
- Código nas anotações.

# Nomeando os parâmetros

---

- ❑ Ao submeter o formulário, apareceu no URL:
- ❑ “cadastro?”
- ❑ Cadastro é a página que mostraria o resultado
- ❑ “?” a interrogação representa que estava esperando parâmetros serem passados. Mas precisamos dar nome a esses.

## Exemplo 3 – Cadastro de Vagas em Rep

---

- ❑ Na tag <input>, coloque um atributo name e dê um valor significativo. O id pode ser um bom começo.
- ❑ Em seguida preencha e submeta o formulário

## Exemplo 3 – Resultado

---

- Ao submeter o formulário, observe o final da URL

cadastro?nome=Jaum&curso=Ciéncia+da+Computaçoo&idade=20&cidade=Santa+Rita+do+Sapucaí

- `cadastro?` => indica o início dos parâmetros no formato “chave-valor”
- `nome=Jaum` (1º parâmetro)
- `&` => indica um separador para o próximo
- E segue para os demais parâmetros

# Tratando os Parâmetros

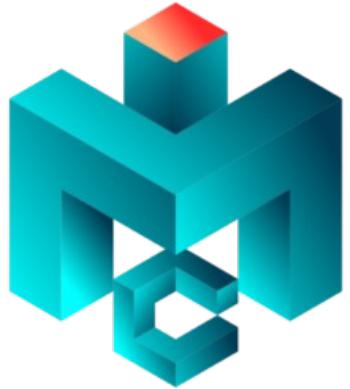
---

- Para tratarmos esses parâmetros, precisamos de um pouco de JavaScript que será visto logo adiante.

## Exemplo 4 (Desafio)

---

- ❑ Com o que vimos, descubra quais os parâmetros de busca do Youtube e crie um formulário que redirecione.
- ❑ Use o inspetor de elementos do navegador para auxiliar a buscar a URL correta e o atributo name



# Aula – 5

## Introdução ao CSS

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Definição e Importância do CSS
- ❑ Sintaxe Básica
- ❑ Formas de Estilizar
- ❑ Seletores
- ❑ Propriedade Iniciais: Cores, Fontes e Textos.

CSS

CSS





## O que é CSS?

Linguagem de estilização utilizada para definir a apresentação de uma página web



## Por que é importante?

Permite separar o conteúdo estrutural do design visual.

# Sintaxe Básica



## Sintaxe Básica

---

- A sintaxe básica do CSS é composta por um seletor (i), uma propriedade (ii) e um valor (iii). Essas três informações combinadas formam uma regra de estilização que será aplicada ao elemento HTML.

# Sintaxe Básica

---

```
seletor {  
    propriedade: valor;  
}
```

## Exemplo Sintaxe – Seletor Elemento

---

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

# Tipos de Estilização

---

- Podemos aplicar a estilização de três formas com o CSS: inline, interna e externa.

# Estilização Inline

---

- ❑ Inserir a estilização diretamente como um atributo do elemento HTML. Não é flexível, não permite reuso e complica a legibilidade.
- ❑ Não é recomendado

## Estilização Inline - Exemplo

---

```
<p style="color:red;">Texto em Vermelho</p>
```

## Estilização Interna

---

- ❑ A estilização é aplicada no “head” da página e ficará disponível para todos os elementos da página.
- ❑ É problemática pois dificulta reuso por páginas HTML diferentes.

# Estilização Interna - Exemplo

---

```
<head>
    <title>Somos todos vermelho</title>
    <style>
        p{
            color: red;
        }
    </style>
</head>
<body>
    <p>Eu sou vermelho</p>
    <p>Eu também serei vermelho</p>
</body>
```

# Estilização Externa

---

- ❑ A mais utilizada e, por regra, a melhor opção. As propriedades CSS são definidas em um arquivo separado, geralmente com a extensão ".css".
- ❑ Para vincular esse arquivo CSS externo a uma página HTML, é usado o elemento <link> no <head> do HTML.

# Estilização Externa – Exemplo Parte 1

---

- Primeiro devemos criar um arquivo externo “.css” e em seguida passar o caminho para o atributo “href”
- Considere um arquivo chamada styles.css
  - <head>
  - <link rel="stylesheet" href="styles.css">
  - </head>

# Estilização Externa – Exemplo Parte 2

---

## □ Arquivo de estilização styles.css

```
p {  
    color: red;  
    font-size: 18px;  
}  
  
h1 {  
    font-size: 32px;  
    text-align: center;  
    color: blue;  
}
```

# Seletores



# Seletores

---

- Os seletores são usados para selecionar os elementos HTML que se deseja estilizar com CSS. Existem vários tipos de seletores que podemos usar, alguns exemplos são:
  - Elemento, classe, identificador e atributo

# Seletor de Elemento

---

- Seleciona todos os elementos do tipo especificado.  
O seletor é o nome da tag HTML.
- É o tipo de seletor que vimos nos exemplos anteriores.

## Seletor de Classe

---

- Seleciona elementos com base na classe atribuída.

Pode-se usar o seletor de classe para estilizar um grupo específico de elementos em sua página.

- Na sua definição é necessário colocar o caractere ponto (.)

## Seletor de Classe – Exemplo definindo a classe

---

```
.minha-classe{  
    text-align: center;  
    color: blue;  
    font-size: 22px;  
}
```

# Seletor de Classe – Exemplo usando a classe

---

```
<section class="minha-classe">
```

Eu sou seção e possuo o atributo class com valor "minha-classe". Isso significa que todas regras de estilização definidas em ".minha-classe" serão aplicadas em mim

```
</section>
```

```
<p class="minha-classe">Eu não sou uma seção, mas possuo o atributo class com valor "minha-classe". Isso significa que todas regras de estilização definidas em ".minha-classe" também serão aplicadas em mim. Não é necessário sermos o mesmo elemento HTML, mas termos o atributo da classe. Talvez isso não seja adequado e possa gerar confusão, mas é possível.</p>
```

## Seletor de ID (Identificador)

---

- ❑ No HTML, id é um atributo utilizado para definir um valor exclusivo para determinado elemento.
- ❑ O CSS pode utilizar esse valor exclusivo como um identificador. Se utiliza o “#” para fazer a seleção.
- ❑ Não devemos ter mais de um elemento HTML com o mesmo valor de id.

## Seletor de ID (Identificador) – Exemplo CSS

---

```
#exemplo {  
    color: green;  
    background-color: black;  
}
```

# Seletor de ID (Identificador) – Exemplo HTML

---

```
<body>
    <h1>Eu sou um H1 e não tenho id.</h1>
    <p id="exemplo">Eu sou um parágrafo e
tenho um id. Lembrando que esse id não
pode ser duplicado. Só eu posso possuir
esse
valor</p>
</body>
```

## Seletor de Atributo

---

□ Seleciona elementos com base em um atributo específico. Podemos usar o seletor de atributo para estilizar elementos que possuem um determinado atributo ou valor de atributo.

# Seletor de Atributo – Exemplo 1

---

- Estilizar ancoras <a> que redirecionam para o GitHub.

```
a[href="https://www.github.com"]{  
    color:green;  
}
```

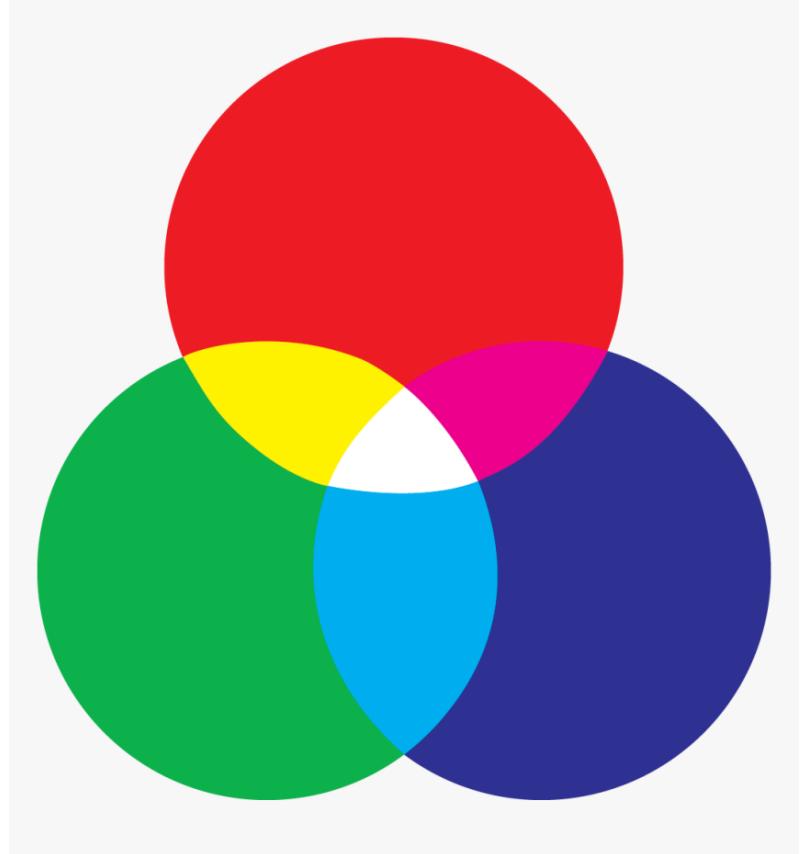
## Seletor de Atributo – Exemplo 2

---

- Estilizar parágrafos <p> que tenham um id

```
p[id]{  
    color:green;  
}
```

# Propriedades Iniciais



# Color

---

- ❑ A propriedade cor modifica a cor do texto e pode receber valores em forma de:
  - ❑ Nome
  - ❑ RGB
  - ❑ RGBA
  - ❑ Hexadecimal

# Background-color

---

- A propriedade background-cor modifica a cor do fundo e pode receber valores em forma de:
  - Nome
  - RGB
  - RGBA
  - Hexadecimal

# Text-align

---

- Descreve como o texto é alinhado no elemento

Pode assumir valores como:

- left

- right

- center

- ...

## Font-weight

---

- ❑ Controla o peso da fonte. Pode receber valores nominais e numéricos.
- ❑ Pode variar conforme a fonte
  - ❑ normal
  - ❑ bold
  - ❑ 100

# Text-decoration

---

- ❑ Pode ser quebrada em três propriedades:
- ❑ text-decoration-line
- ❑ text-decoration-style
- ❑ text-decoration-color

## Text-decoration-line

---

- ❑ Define um tipo de linha para decorar o texto
  - ❑ none
  - ❑ underline
  - ❑ overline
  - ❑ line-through

## Text-decoration-style

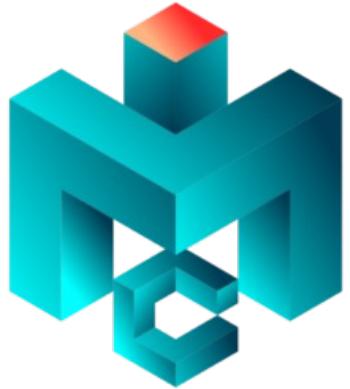
---

- ❑ Define um estilo para a linha decorando o texto
  - ❑ solid
  - ❑ double
  - ❑ dotted
  - ❑ wavy

## Text-decoration-color

---

- ❑ Define a cor para a linha decorando o texto. Pode receber valores em forma de:
  - ❑ nome
  - ❑ RGB
  - ❑ RGBA
  - ❑ Hexa



# Aula – 6

## Modelo de Caixa

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Modelo de Caixa
- ❑ Propriedades do modelo de caixa
- ❑ Unidades absolutas e relativas
- ❑ Display
- ❑ Posicionamento

Modelo  
Caixa

**CAIXA**



## O que é modelo caixa?

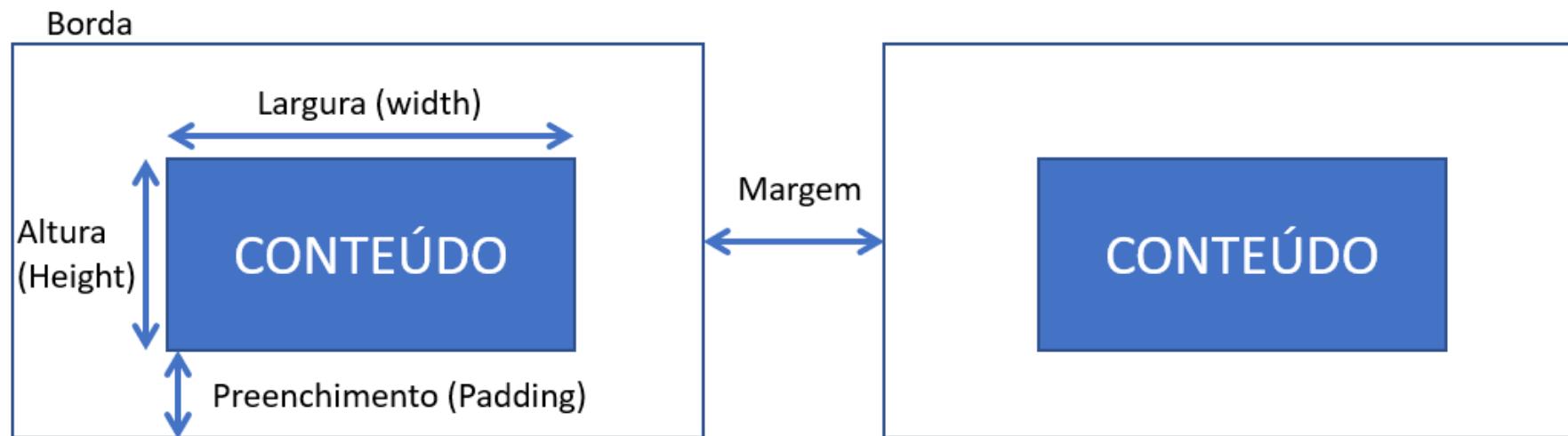
Conceito básico usado pelo CSS para representar elementos HTML na tela. Cada elemento HTML é considerado como uma caixa retangular



## O que cada caixa possui?

Preenchimento (padding), borda (border)  
margem (margin), largura (width) e altura  
(height)

# Modelo Caixa



## Usando Inspeção para analisar elemento

---

- Vamos utilizar o inspetor do navegador para analisar modelo caixa de um elemento <p>.
- Mão na massa!

# Propriedades Básicas

## Largura e Altura

---

- Podemos definir a altura (height) e largura (width) de um elemento atribuindo valores diretamente nessas propriedades
- Essas medidas irão definir a área retangular com o conteúdo. É o retângulo mais interno.

## Largura e Altura - Exemplo

---

```
p{  
    width:  
    200px;  
    height:  
    200px;  
}
```

# Preenchimento

---

- Valor que será adicionado entre o conteúdo e a borda, isto é, o preenchimento fica dentro do elemento HTML

```
padding-top: ;  
padding-right: ;  
padding-bottom: ;  
padding-left: ;  
padding: ;
```

# Margem

---

- ❑ Representa a distância externa a borda. É ela quem irá determinar qual distante o elemento ficará dos demais.
- ❑ Para controlar a margem usamos a propriedade "margin".

# Borda

---

- ❑ Representa o retângulo mais externo do elemento HTML.
- ❑ A princípio a borda fica invisível e precisa ser modificada para que se torne presente/visível.
- ❑ Pode ser estilizada de várias maneiras, incluindo largura, cor e estilo.

# Exemplo 1

- Criar uma borda sólida, azul com largura 5px

**L**orem, ipsum dolor sit amet  
consectetur adipisicing elit.  
**I**llum deserunt quam  
repellendus debit is nisi veniam  
minus quo, eligendi corrupti  
beatae libero voluptatum sit  
officia fugiat numquam  
nostrum totam! Dolorum, vel.

## Exemplo 2

---

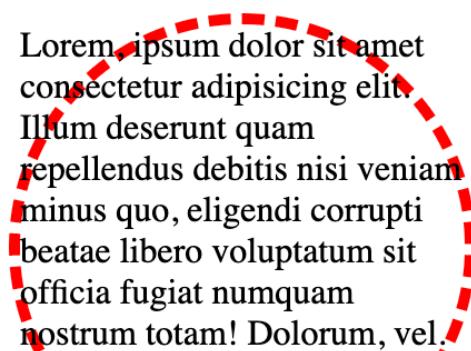
- Criar uma borda arredondada com 10px

  Lorem, ipsum dolor sit amet  
  consectetur adipisicing elit.  
  Illum deserunt quam  
  repellendus debitis nisi veniam  
  minus quo, eligendi corrupti  
  beatae libero voluptatum sit  
  officia fugiat numquam  
  nostrum totam! Dolorum, vel.

## Exemplo 3

---

- Criar uma borda tracejada, vermelha que se apresente como um círculo.



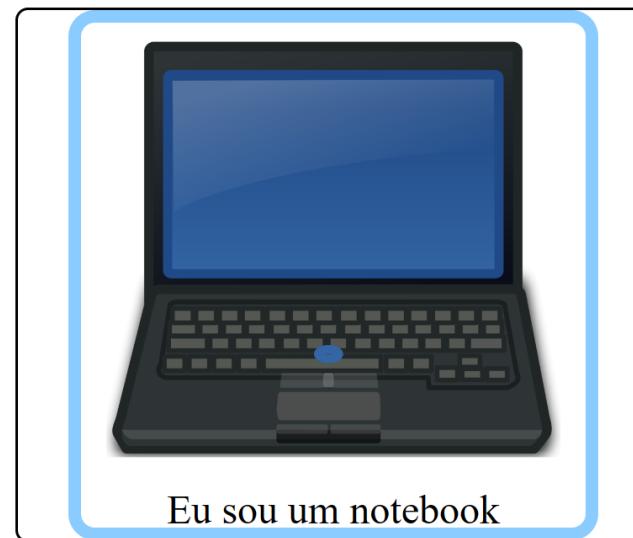
A red dashed circle is centered on the page, containing the following placeholder text:

  Lorem ipsum dolor sit amet  
  consectetur adipisicing elit  
  Illum deserunt quam  
  repellendus debitis nisi veniam  
  minus quo, eligendi corrupti  
  beatae libero voluptatum sit  
  officia fugiat numquam  
  nostrum totam! Dolorum, vel.

## Exemplo 4

---

- Escreva um código HTML/CSS para reproduzir o conteúdo abaixo.



# Medidas



# Medidas Absolutas

---

□ PX (CSS pixel): Unidade absoluta mais utilizada. A confusão feita é que 1px não é necessariamente 1 pixel no monitor, definido como a menor unidade endereçável. Não é recomendado para sites responsivos.

## Medidas Relativas - Porcentagem

---

□ % (porcentagem): Unidade relativa que ocupa o espaço em relação a uma porcentagem do elemento “pai”.

# Medidas Relativas - EM

---

- EM: Unidade relativa ao tamanho da fonte aplicado no elemento “**pai**”. Se o elemento pai possui uma fonte de 16px, e o filho uma fonte de “1em”, este será 16px. Mas se for “2em” este será 32px.
- Conforme os objetos são aninhados, o valor de EM pode se alterar

## Medidas Relativas - REM

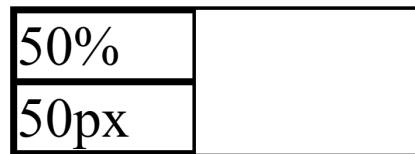
---

- REM: Unidade relativa ao tamanho da fonte aplicado no elemento “**raiz**”. Se o elemento raiz possui uma fonte de 16px, e o filho uma fonte de “1rem”, este será 16px. Mas se for “2rem” este será 32px. Conforme os objetos são aninhados, o valor de REM não se altera.

## Exemplo 5

---

- ❑ Comparação entre PX e %.
- ❑ Fonte: MDN



# Exemplo 6

---

□ Comparação entre EM e REM.

□ Fonte: MDN

Eu sou 1EM  
Eu sou 1REM

Eu sou 2EM

Eu sou 2REM

Eu sou 1EM  
Eu sou 1REM

**Eu sou 2EM**

Eu sou 2REM

# Display



## A propriedade Display - Definição

---

- ❑ Usada para controlar a forma que um elemento HTML é exibido na página.
- ❑ Pode ser usada para mudar o comportamento padrão de um elemento e torná-lo mais flexível

# A propriedade Display - Valores

---

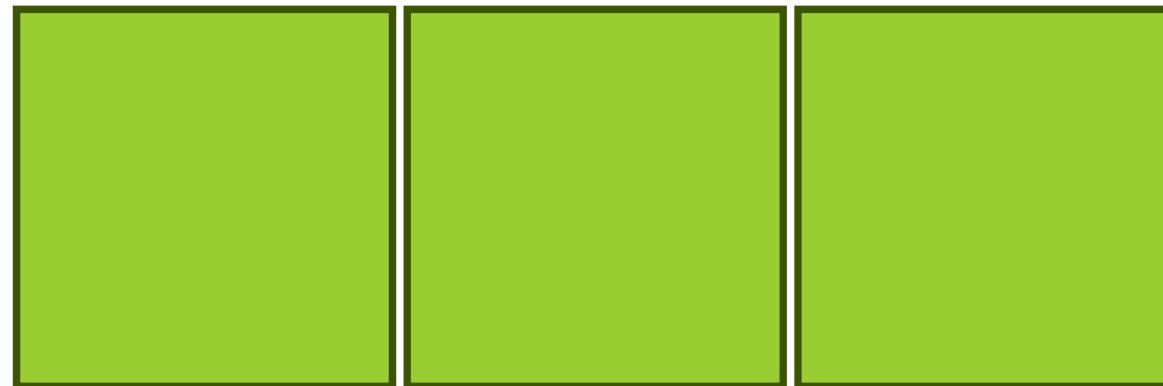
```
display: block ;  
display: inline;  
display: inline-  
block;  
display: none;  
display: flex;
```

## Exemplo 7

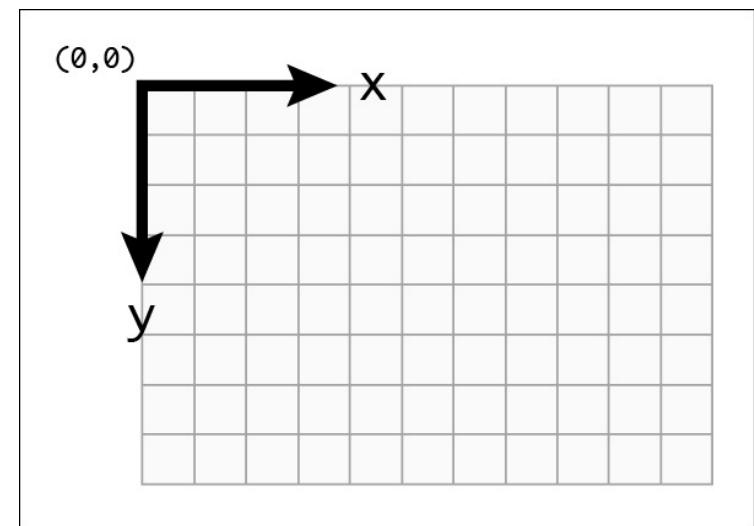
---

- Crie um código para exibir 3 (três) quadrados lado a lado. E felizes

**Três Quadrados Felizes Lado a Lado**



# Posicionamiento



## A propriedade *Position* - Definição

---

- ❑ Permite definir como um elemento deve ser posicionado na página em relação aos outros elementos.

# A propriedade *Position* - Valores

---

**position: static;**

**position: relative;**

**position: absolute;**

**position: fixed;**

**position: sticky;**

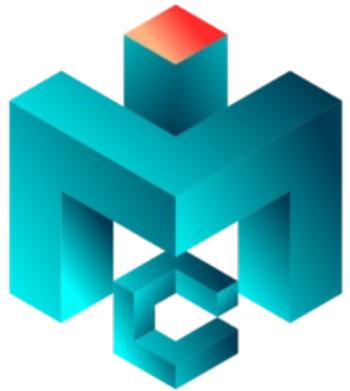
## Exemplo 8

---

- Crie um código HTML/CSS para criar uma barra de navegação que fique fixa no “header” da página

[GitHub](#) [UNIFEI](#) [IMC](#)

Loreum ipsum dolor sit amet consectetur adipisicing elit. Sint ipsum officia nobis necessitatibus, eligendi inventore ad tempore animi placeat delectus id voluptates beatae



# Aula – 7

## Flexbox e Media Queries

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ Flexbox
- ❑ Eixos
- ❑ Algumas Propriedades
- ❑ Media Queries
- ❑ Exemplo Barra de Navegação

# Modelo Flexbox



## O que é modelo Flexbox?

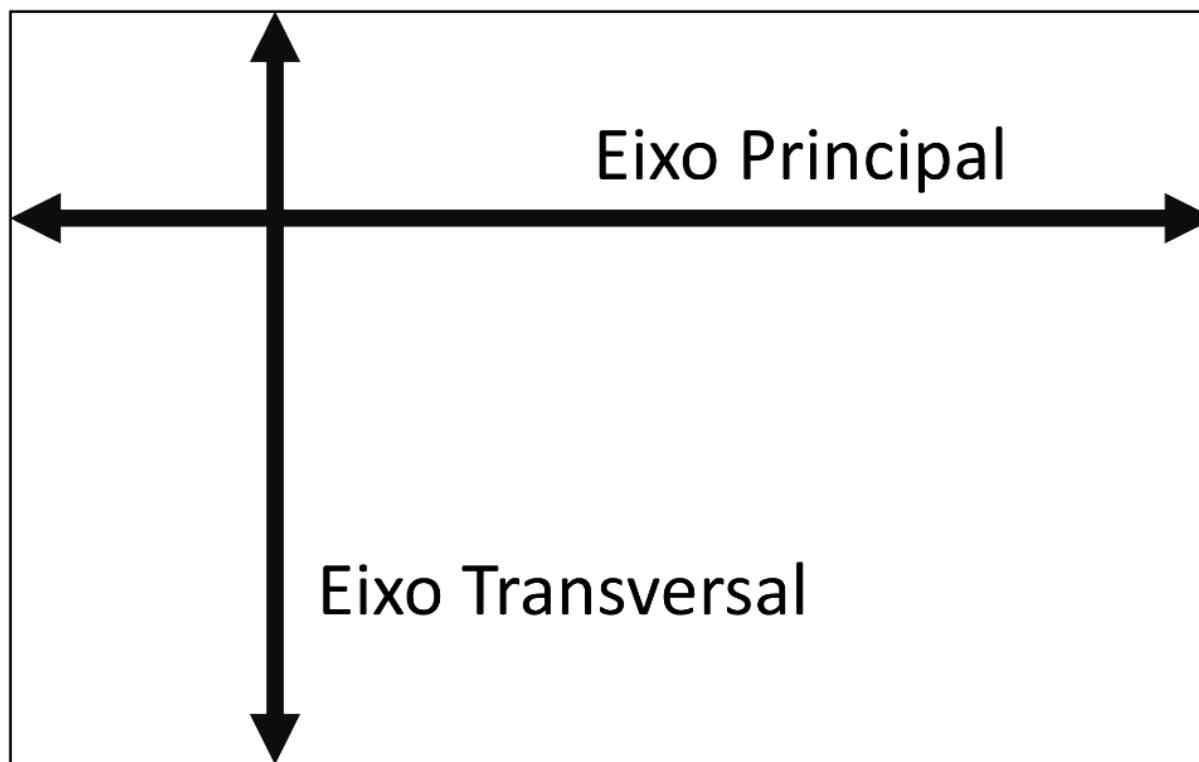
Técnica de layout muito que permite posicionar e organizar elementos em um contêiner de forma flexível e responsiva.



## Como começar?

Definindo um contêiner cuja a propriedade **display**, do elemento pai, será **flex**.

# Modelo Flexbox



## Eixos – Principal

---

- ❑ Ao declararmos um elemento com a propriedade “flex”, por padrão, o conteúdo é posicionado em linha, da esquerda para a direita.
- ❑ A propriedade utilizada para esse controle é a “**flex-direction**”.

# Eixos – Principal

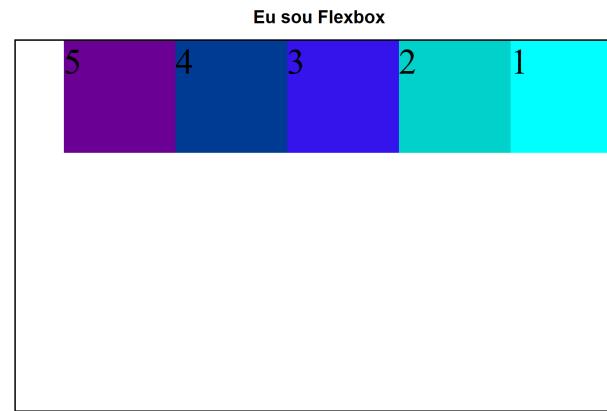
---

```
/* Valor padrão. O eixo principal é horizontal, da esquerda para a  
direita */  
flex-direction: row;  
  
/* O eixo principal é horizontal, da direita para a esquerda */  
flex-direction: row-reverse;  
  
/* O eixo principal é vertical (coluna), de cima para baixo */  
flex-direction: column;  
  
/* O eixo principal é vertical (coluna), de baixo para cima */  
flex-direction: column-reverse;
```

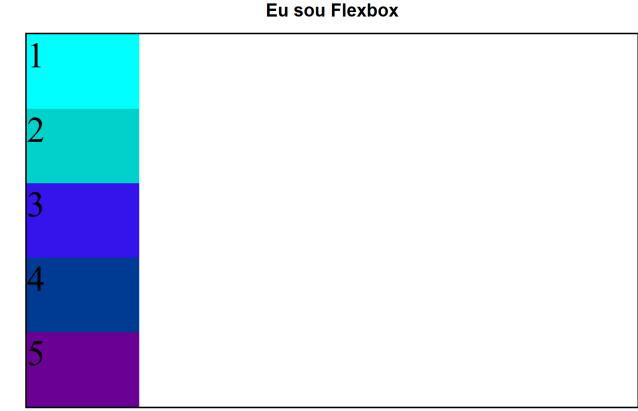
# Eixos – Principal - Exemplos



`flex-direction: row;`



`flex-direction: row-reverse;`



`flex-direction: column;`

# Alinhamento e Propriedades

# Alinhamento no Eixo Principal

---

- ❑ Para alinhar os elementos no eixo principal, usamos a propriedade `justify-content`.
- ❑ Seu comportamento irá depender de como a propriedade `flex-direction`

# Alinhamento no Eixo Principal : justify-content

---

```
/* Alinhamento padrão, com os elementos posicionados de acordo com o
início do eixo principal */
justify-content: flex-start;

/* Elementos posicionados de acordo com o fim do eixo principal */
justify-content: flex-end;

/* Elementos centralizados no eixo principal */
justify-content: center;

/* Elementos com o espaço distribuído ao redor */
justify-content: space-around;

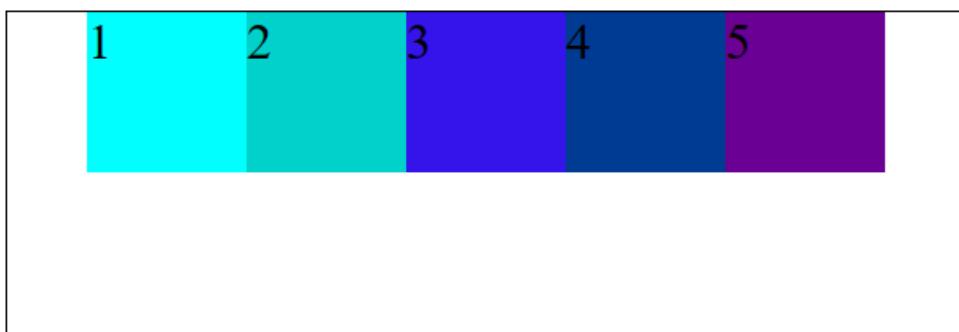
/* Elementos com o espaço distribuído entre os elementos */
justify-content: space-between;

/* Elementos com o espaço distribuído igualmente entre os elementos */
justify-content: space-evenly;
```

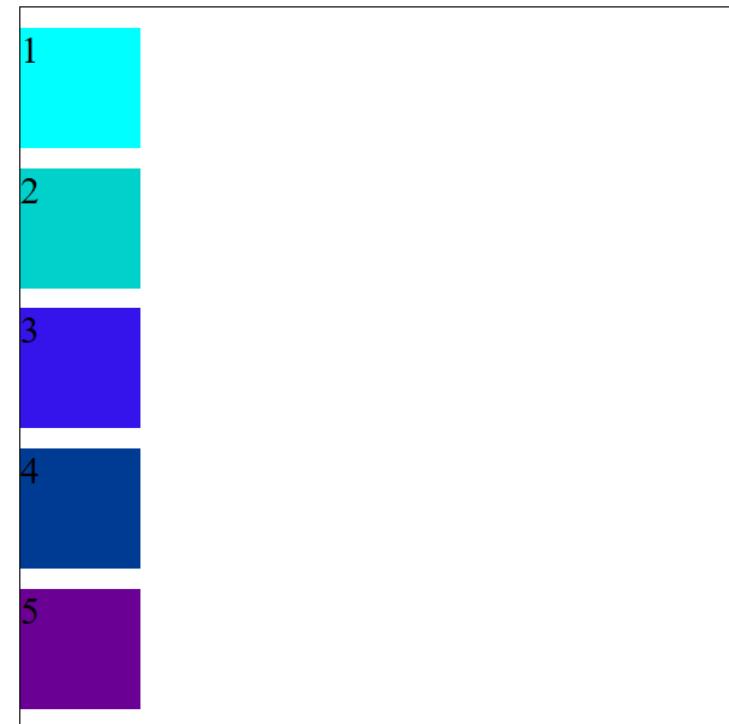
# Alinhamento no Eixo Principal : justify-content

---

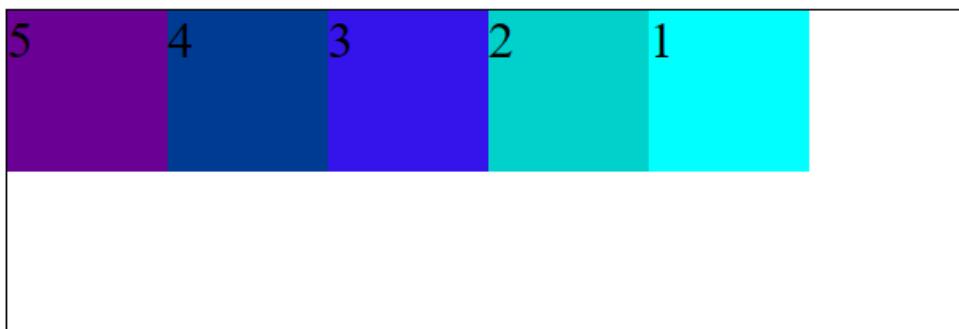
**Row e Center**



**Column e Space Evenly**



**Row Reverse e Flex End**



# Wrap do Elementos

---

- ❑ A propriedade **flex-wrap** define se os elementos são forçados a ficarem na mesma linha ou se podem ser quebradas em várias linhas.
- ❑ O sentido que os elementos serão “quebrados” segue o eixo transversal.

# Wrap do Elementos : flex-wrap

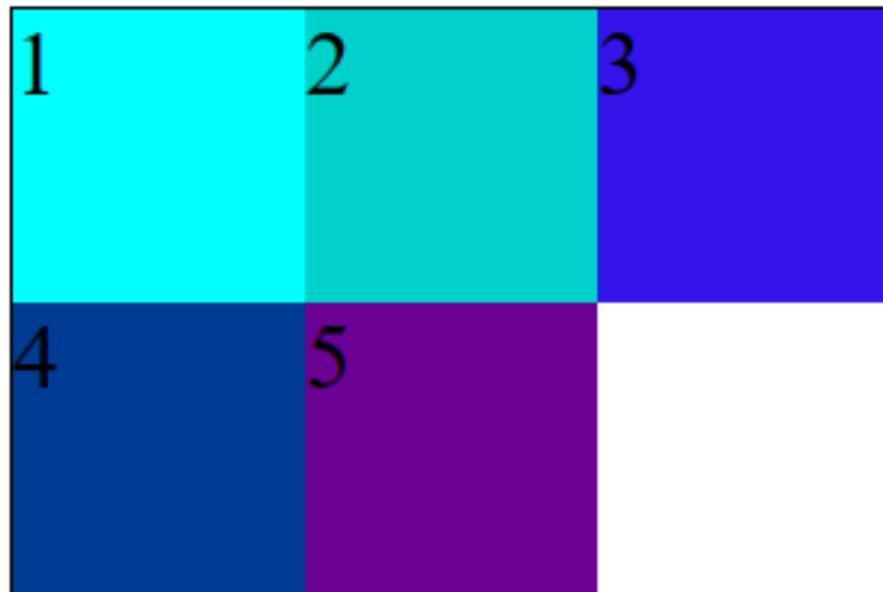
---

```
/* Valor padrão. Os elementos não passam para proxima linha/coluna */
flex-wrap: nowrap ;  
  
/* Na mesma direção do eixo transversal */
flex-wrap: wrap;  
  
/* Na direção contrária ao eixo transversal
flex-wrap: wrap-reverse;
```

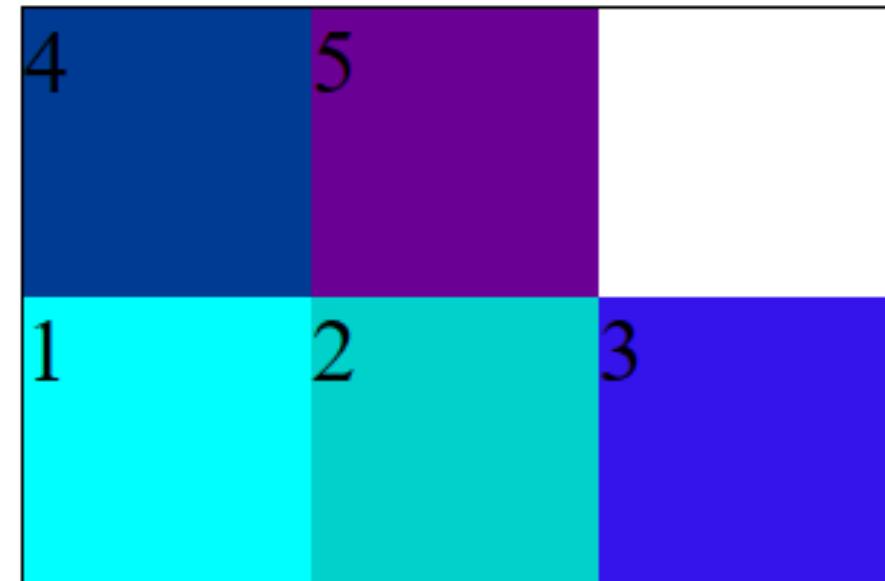
# Wrap do Elementos : flex-wrap

---

**Row e Wrap**



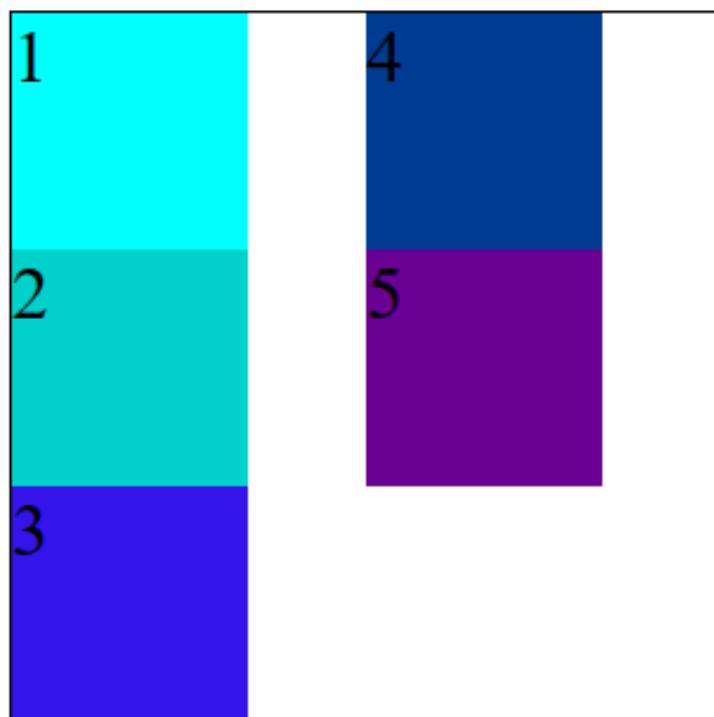
**Row e Wrap-Reverse**



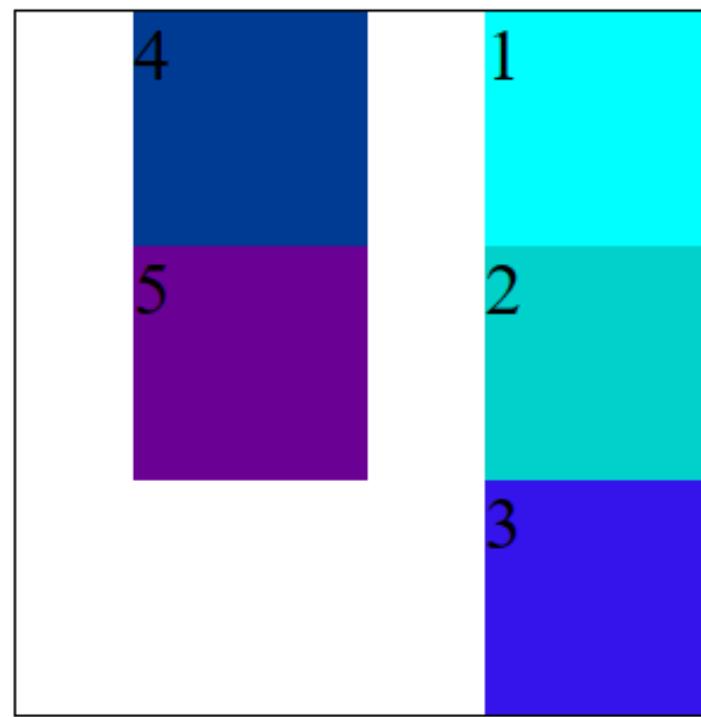
# Wrap do Elementos : flex-wrap

---

**Column e Wrap**



**Column e Wrap-Reverse**



# Alinhamento no Eixo Transversal

---

- ❑ Para alinhar os elementos de um “flex” container, ao longo do eixo transversal, usamos a propriedade align-items.
- ❑ A propriedade justify-content, alinha e espaça os elementos ao longo do eixo principal.

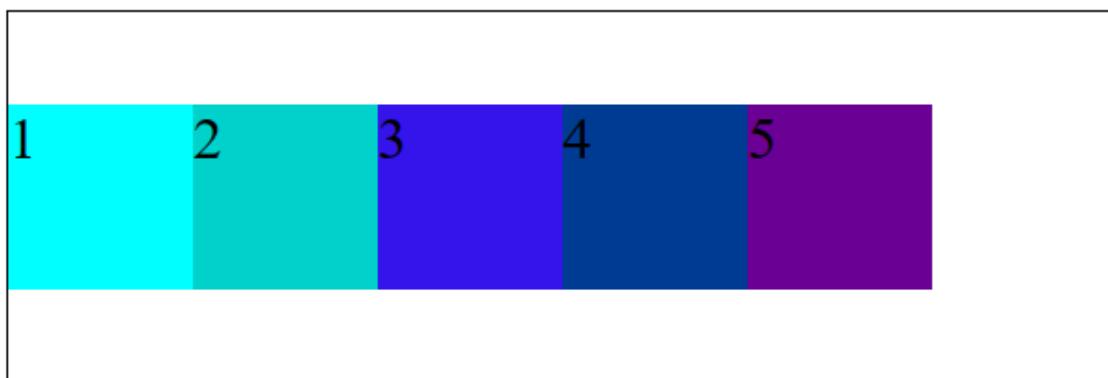
# Alinhamento no Eixo Transversal: align-items

---

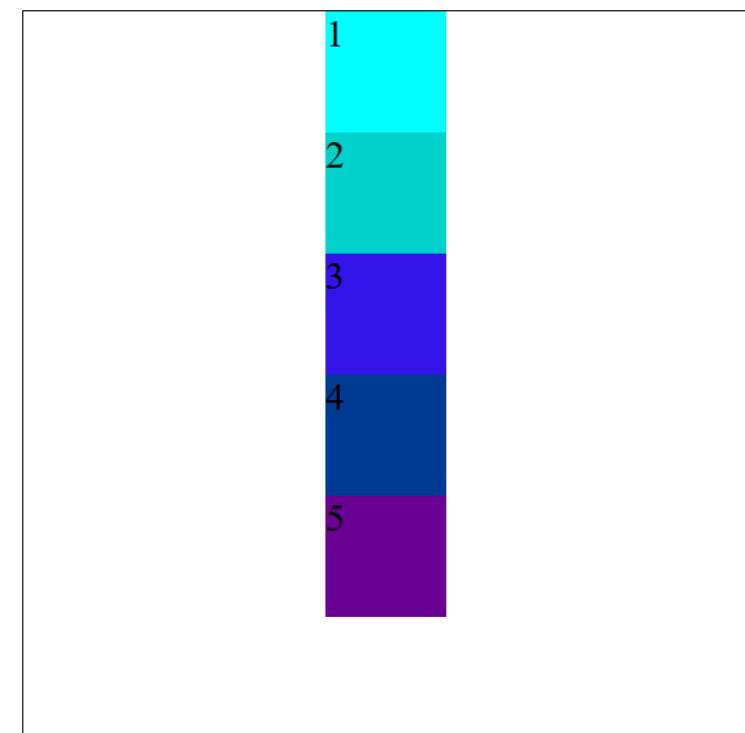
```
/* Valor padrão. Alinha os elementos a partir do início do eixo  
transversal */  
align-items: flex-start;  
  
/* Alinha os elementos a partir do fim do eixo transversal */  
align-items: flex-end;  
  
/* Centraliza os elementos no eixo transversal */  
align-items: center;  
  
/* Alinha os elementos no eixo transversal de acordo com a base do  
conteúdo */  
align-items: baseline;
```

# Alinhamento no Eixo Transversal

**Row e Center-Transversal**



**Column e Center-Transversal**



# Alinhamento com Múltiplas Linhas/Colunas

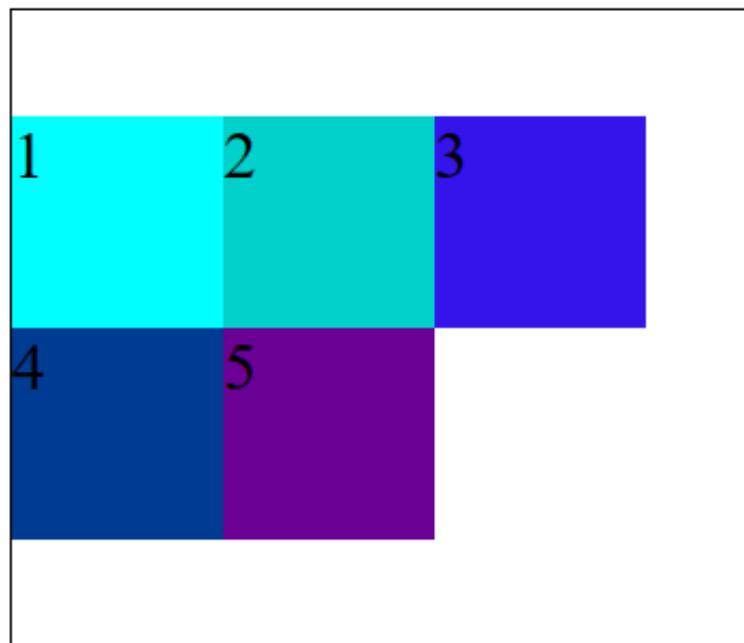
---

- Quando se tem múltiplas colunas e/ou linhas usamos a propriedade `align-content` para controlar o espaçamento.
- É necessário ter algum tipo de “wrap” ativado.

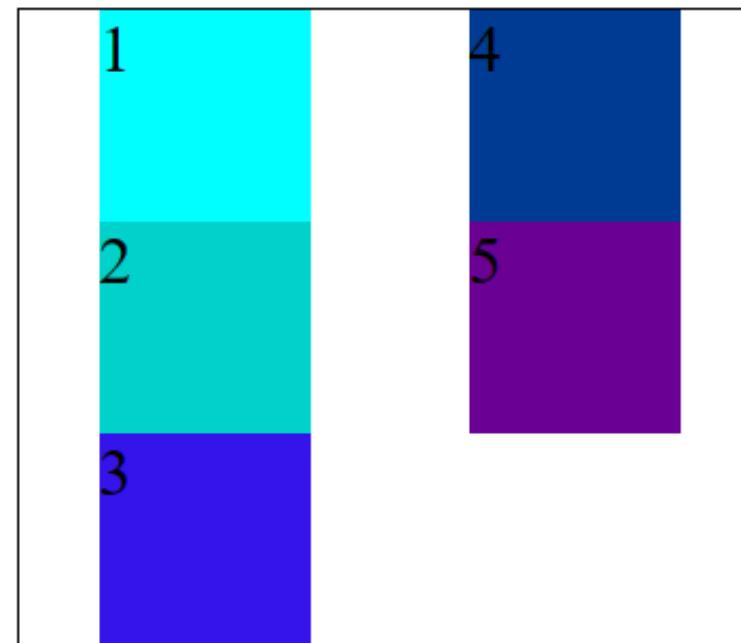
# Alinhamento com Múltiplas Linhas/Colunas

---

**Row, Wrap e Align-Content: Center**



**Column, Wrap e Align-Content: Space-Around**



# Media Queries

# Layout Responsivo

---

- ❑ Permite modificar a estilização dependendo de características como tamanho de tela e tipo de dispositivo.
- ❑ Podemos especificar uma largura, orientação, e outras condições para alterarmos o layout.

# Media Queries – Exemplo 1

---

- Modificar a cor do elemento <h1> quando a tela estiver **exatamente** com 800px:

```
@media (width: 800px){  
    h1{  
        color: purple;  
    }  
}
```

# Media Queries – Exemplo 2

---

- Modificar a cor do elemento <h1> enquanto a tela tiver pelo menos 800px;

```
@media (min-width: 800px){  
    h1{  
        color: purple;  
    }  
}
```

# Media Queries – Exemplo 3

---

- Modificar a cor do elemento <h1> enquanto a tela tiver entre 600px e 800px;

```
@media (min-width: 600px) and (max-width:800px){  
    h1{  
        color: purple;  
    }  
}
```

# Exemplo Prático

---

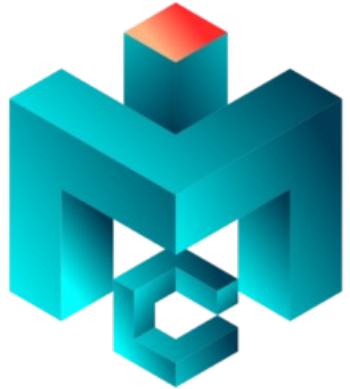
- Criando uma barra de navegação responsiva.

[Home](#) [Aprenda Mais](#) [Sobre](#) [Contato](#) [Entrar](#)

# Página Responsiva

[Home](#)  
[Aprenda Mais](#)  
[Sobre](#)  
[Contato](#)  
[Entrar](#)

# Página Responsiva



# Aula – 8

# Introdução ao JavaScript

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

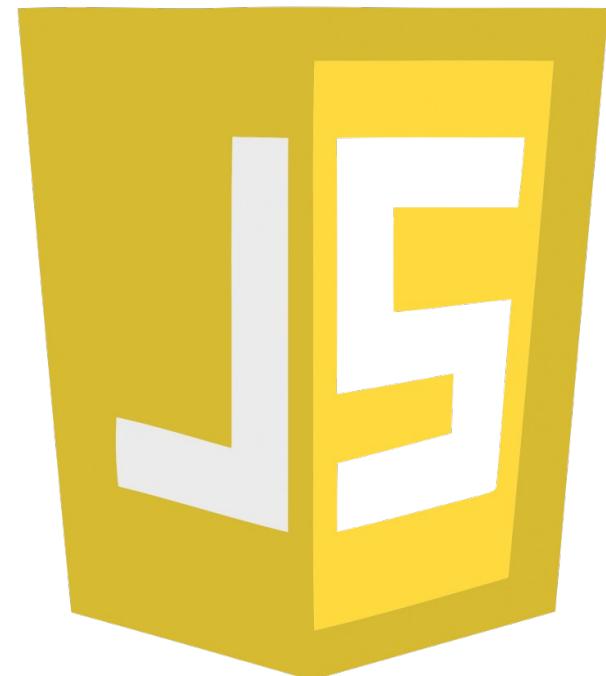
---



- ❑ Variáveis e Constantes
- ❑ Tipos Primitivos
- ❑ Condicionais
- ❑ Arrays
- ❑ Repetição

# JavaScript

# JavaScript





## O que é JavaScript?

Linguagem de programação utilizada **principalmente** para adicionar interatividade em páginas web



Quais suas características?

Multi-paradigma, de tipagem dinâmica e  
do tipo função *first-class*

# Variáveis e Constantes

# Variáveis com *Let* - Definição

---

- Para criar variáveis em JS utilizamos a palavra-chave “let”.
- Dado que JS é uma linguagem de tipagem dinâmica, o tipo só será definido em tempo de execução.

# Variáveis com *Let* - Exemplo

---

```
let x; //declarando a variável cujo identificador é x  
x = 3; //atribuindo valor após a definição  
console.log(x); // 3  
  
let y = 4; //atribuindo valor no momento da definição  
console.log(y); // 4
```

# Constantes com *const* - Definição

---

- Área de memória que não pode ser modificada após ter o seu valor atribuído.
- Em JS usamos a palavra-chave “*const*”.
- Uma das principais razões pelas quais é importante usar “*const*” é para evitar a reatribuição acidental de valores de variáveis e referências.

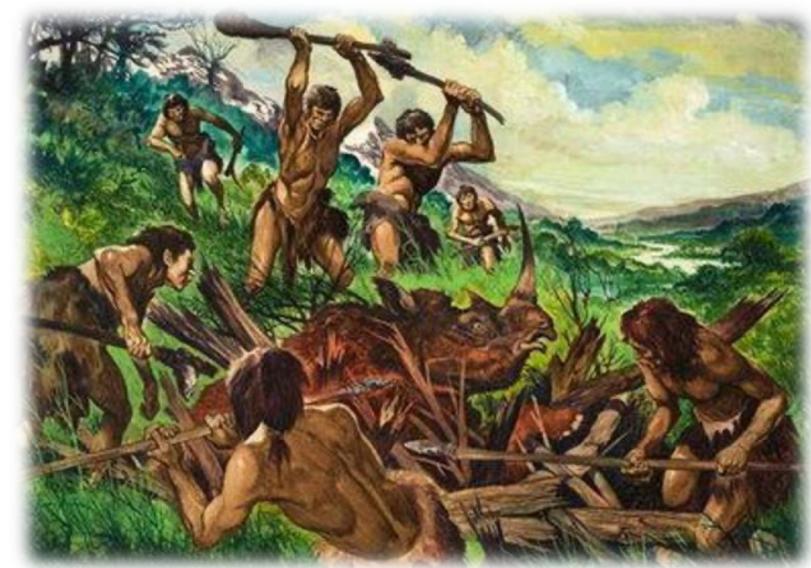
## Constantes com *const* - Exemplo

---

- O uso de constantes aprimora a legibilidade e ajuda a criar um código mais seguro.

```
const x = 7;  
x = 8; //ERRO
```

# Tipos Primitivos



# Tipos Primitivos em JS

---

- ❑ Number
- ❑ Boolean
- ❑ String
- ❑ Null
- ❑ Undefined

# Numbers - Definição

---

- ❑ A linguagem JS possui um único tipo de dado para número. Esse tipo é capaz de armazenar números inteiros e valores reais.
- ❑ Essa abordagem é diferente de outras linguagens populares como C e Java. Pois estas utilizam tipos diferentes, como “int” e “float”.

## Numbers – Precisão/Capacidade

---

- Formato IEEE 754 64bits com dupla precisão.
- Sua precisão pode variar de 15 a 17 casas decimais
- Existem diversos métodos/propriedades que nos informam valores máximos e mínimos para o tipo "number".

# Numbers – Precisão/Capacidade

---

```
//Maior valor  
console.log(Number.MAX_VALUE); //1.7976931348623157e+308  
  
//Menor valor  
console.log(Number.MIN_VALUE); //5e-324  
  
//Maior valor seguro para cálculos sem perder precisão  
console.log(Number.MAX_SAFE_INTEGER); //9007199254740991  
  
//Menor valor seguro para cálculos sem perder precisão  
console.log(Number.MIN_SAFE_INTEGER); //-9007199254740991
```

# Numbers – Exemplos

---

```
let idade = 34;  
console.log(typeof(idade));//number
```

```
let telefone = 988998899;  
console.log(typeof(telefone));//number
```

```
let peso = 84.56;  
console.log(typeof(peso));//number
```

```
let temp = -14.673561;  
console.log(typeof(temp));//number
```

# Numbers – Operações Matemáticas

---

- Podemos executar operações matemáticas básicas com valores do tipo “number” tais como:
  - Soma (+), Subtração (-), Multiplicação (\*).
  - Divisão (/), Resto (%), Exponenciação (\*\*).
  - Outros..

# Numbers – NaN (Not a Number)

---

- Algumas operações matemáticas podem resultar em NaN (Not a Number).
- Exemplo:  $0/0$  irá resultar em NaN
- O valor NaN é do tipo “number” e representa algo que não é um valor numérico.

# Numbers – Funções em *Math*

---

```
let x = 34.56;
let y = -78.12;

//remove a parte fracionária
console.log(Math.floor(x)); //34

//arredonda para cima
console.log(Math.ceil(x)); //35

//valor absoluto
console.log(Math.abs(y)); //78.12

//exponenciação da base 2 pelo expoente 3
console.log(Math.pow(2,3)); //8

//constante PI
console.log(Math.PI); //3.141592653589793
```

# Numbers – Incremento/Decremento

---

```
let x = 6;  
console.log(x++); //6  
console.log(x); //7  
console.log(++x); //8  
console.log(--x); //7
```

# Numbers – Números Aleatórios

---

- Geramos números aleatórios usando a função `Math.random()` que retorna um valor real aleatório entre 0 e 1 (exclusivo).
- É possível obter números inteiros e outras faixas de valores combinando com algumas funções matemáticas.

# Numbers – Números Aleatórios - Exemplo

---

```
//Valor aleatório entre 0 e 9
let x = Math.floor(Math.random() * 10);

console.log(x);
//Valor aleatório entre 1 e 10

x = Math.floor(Math.random() * 10) + 1;
console.log(x);

//Valor aleatório entre 2 e 6
x = Math.floor(Math.random() * 5) + 2;
console.log(x);
```

## Boolean - Definição

---

- ❑ Existem situações que podem apresentar apenas dois estados.
- ❑ Uma porta pode se encontrar `aberta` ou `fechada`.
- ❑ Uma lâmpada pode estar `acesa` ou `apagada`.
- ❑ Um valor numérico qualquer pode ser `maior` ou `menor` que uma dada referência.

## Boolean - Definição

---

- ❑ Para representar tais situações, a linguagem JS oferece o tipo primitivo “Boolean”.
- ❑ Variáveis desse tipo podem ter o valor “true” ou “false”.
- ❑ Importante notar que estamos falando do valor true/false e não do literal “true” ou “false”.

# Boolean - Exemplo

---

```
let x = true;
let y = false;
let z = "true"; //diferente de atribuir true ou false sem aspas
console.log(typeof(x)); //Boolean
console.log(typeof(y)); //Boolean
console.log(typeof(z)); //string
```

## Boolean - Exemplo

---

- Como a tipagem em JS é dinâmica, nada impede de atribuirmos um valor de outro tipo a está variável.

```
let x = true;  
console.log(typeof(x)); //boolean  
x = 1;  
console.log(typeof(x)); //number
```

## Boolean – Por que usar?

---

- Ao invés de usar números como 1/0, o uso de Boolean aprimora a legibilidade do código e evita a possibilidade de usar valores fora do padrão (exemplo: “3” para verdadeiro e “-6” para falso).
- Esse cenário é conhecido como “magic numbers”

# String - Definição

---

- ❑ Representa uma sequência de caracteres. É utilizada para armazenar informações textuais e precisam ser envolvidas por aspas duplas/simples.
- ❑ Podem ser envolvidas pelo acento grave (`) para criar *template strings*.

# String - Exemplo

---

```
let nomeUsuario = "mestreDosMagos";
let nome = 'Maria';
let cidade = `Itajubá`;
let endereco = `Moro em ${cidade}, no estado de Minas Gerais.`;

console.log(typeof(nomeUsuario)); //string
console.log(typeof(nome)); //string
console.log(typeof(cidade)); //string
```

# String – Comprimento e Concatenação

---

```
let cidade = 'Itajubá';
console.log(cidade.length); //7
```

```
let nome = "João";
let sobrenome = " da Silva";
let nomeCompleto = nome + " " + sobrenome;
console.log(nomeCompleto); //João da Silva
```

# String – Concatenação com Números

---

```
let num = 2; //sou number
num += 1;
console.log(num); //3
let literal = "2"; //sou string
literal += 1; //Literal + Number = Literal
console.log(literal) //"21"
literal++ //???????
```

# String – Métodos Auxiliares

---

```
let nome = "    João    ";
let ret;

nome = nome.trim(); //remove espaços em branco no início e fim
ret = nome.toLowerCase(); //todas as letras ficam minúsculas
ret = nome.toUpperCase(); //todas as letras ficam maiúsculas
ret = nome.startsWith("J"); //retorna true se nome se iniciar com "J"
ret = nome.startsWith("o",2); //true se nome se iniciar com "o" no
índice 2
ret = nome.endsWith("o"); //retorna true se nome terminar com "o"
```

# String – Métodos Auxiliares com Argumentos

---

```
let frase = "Eu quero tomar café";
//índice onde se encontra a primeira ocorrência de "E"
frase.indexOf('E'));

//índice onde se encontra a primeira ocorrência de "quero"
frase.indexOf('quero');//3

//sequencia entre os caracteres na posição 0 (inclusivo) e 2 (exclusivo)
frase.slice(0,2); //"Eu"

//sequencia de caracteres a partir da posição 3
frase.slice(3); //"quero tomar café"

//sequencia de 4 caracteres a partir do final da string
frase.slice(-4); //"café"

frase.replace('café', 'suco');
```

# String – Template Literals

---

- Sequencias de caracteres que permitem  
interpolação e embutir expressões que serão  
calculadas e substituídas em tempo de execução  
em seus *placeholders*.

# String – Template Literals Exemplo

---

```
let precoCafe = 4.50;
let precoCoxinha = 6.00;
const msg = `O café ${precoCafe} e coxinha ${precoCoxinha} resultam no total
de ${precoCafe + precoCoxinha}.`;
console.log(msg);
//O café 4.5 e coxinha 6 resultam no total de $10.5.
```

## Null e Undefined – Definição

---

- ❑ O tipo *null* se refere a ausência proposital de valor e, portanto, é necessária a operação de atribuição do valor null.
- ❑ O tipo *undefined* ocorre quando alguma variável não teve nenhum valor atribuído e apenas foi definida.

## Null e Undefined – Exemplo

---

```
let usuarioLogado = null;  
console.log(typeof(usuarioLogado)); //null  
let x;  
console.log(x); //undefined
```

# Condicionais

# Condicionais – Definição

---

- ❑ Estrutura que permite executar um trecho de código se uma dada condição for verdadeira ou falsa.
- ❑ Se nenhuma condição é testada, o código executa de forma sequencial.

# Condicionais – Operadores de Comparaçõ

---

- Podemos comparar valores e verificar se resulta em um valor verdadeiro ou falso

>	maior que
<	menor que
>=	maior ou igual
<=	menor ou igual
==	igual
!=	diferente
==	estritamente igual
!=	estritamente diferente

# Condicionais – Operadores de Comparaçõ

---

- Diferença entre a igualdade (==) e estritamente igual (====)

```
5 == '5'; //true  
5 === '5'; //false  
0 == false; // true  
0 === false; //false  
undefined == null; //true  
undefined === null; //false
```

# Condicionais – Estrutura If-Else

---

- Estrutura tradicional presente em diversas linguagens para testar condições.

```
if(condição){  
    //executa se a condição é verdadeira  
}else{  
    //caso contrário  
}
```

# Condicionais – Estrutura If-Else

---

- Encadear diversos *else-if* com novas condições

```
if(condição1){  
    //executa se a condição1 é verdadeira  
}else if(condição2){  
    //executa se a condição2 é verdadeira  
}else if(condição3){  
    //executa se a condição2 é verdadeira  
}else{  
    //executa se todas as condições forem falsas  
}
```

# Condicionais – Operadores Lógicos

---

- ❑ Operadores binários cujos operandos são expressões lógicas.
- ❑ O resultado é um valor booleano

AND (E)	&&
OR (OU)	
NOT (NEGAÇÃO)	!

## Condicionais – Exemplo

---

□ Escreva um programa para solicitar que o(a) usuário(a) digite sua temperatura corporal em Celsius. Se for menor ou igual a 36.5 imprima "normal". Acima de 36.5 e menor ou igual a 37.5 imprima "estado febril". Acima de 37.5 imprima "febre".

# Condicionais – Exemplo

---

```
const temp = prompt("Digite a temperatura em graus Celsius");
```

.....



# Condicionais – Valores Booleanos Internos

---

- Todo valor em JS, isoladamente, retorna um valor verdadeiro com exceção dos seguintes:
- false
- 0
- "" (string vazia)
- Null, undefined e NaN

# Condicionais – Estrutura Switch

---

- ❑ Estrutura alternativa de controle condicional
- ❑ Se usa uma chave, e executa a comparação estritamente igual com os diversos casos.
- ❑ O primeiro que resultar em verdadeiro é executado e nenhuma outra comparação será feita.
- ❑ A estrutura executará até encontrar um *break*

# Condicionais – Estrutura Switch

---

```
switch (valor testado) {  
    case valor1:  
        break;  
    case valor2:  
        break;  
    default:  
        break;  
}
```

# Arrays



# Arrays - Definição

---

- ❑ Coleções heterogêneas que armazenam dados de forma indexada. O seu tamanho pode se modificar durante a execução.
- ❑ Mesmo que seja possível, pode ser confuso criar Arrays cujos elementos são de tipos diferentes.

# Arrays – Exemplo Inicialização

---

```
//array vazio
let discentes = []

//array de strings
let cores = ['red','green','blue'];

//array de numbers
let versoesWindows = [95,98,7,8,8.1,10,11];

//array com tipos diferentes
let coisas = [true, 18,'café',null];

console.log(cores[1]);//green
console.log(versoesWindows[3]);//8
console.log(versoesWindows.length);//7
```

# Arrays – Exemplo Atualizando Valores

---

```
let cores = ['red', 'green', 'blue'];
console.log(cores[1]); //green
cores[0] = 'yellow'; //Modificando o valor da posição 0 para yellow
console.log(cores[0]); //green
```

# Arrays – Função Push/Pop

---

```
let num = [1,2,3];
console.log(num.length); // 3
num.push(4);
num.push(5,6);
console.log(num); // [1, 2, 3, 4, 5, 6]
console.log(num.length); // 6
num.pop();
console.log(num); // [1, 2, 3, 4, 5]
console.log(num.length); // 5
```

# Arrays – Função Shift/Unshift

---

```
let num = [1,2,3];
console.log(num.length); // 3
num.unshift(4);
num.unshift(5,6);
console.log(num); // [5, 6, 4, 1, 2, 3]
console.log(num.length); // 6
num.shift();
console.log(num); // [6, 4, 1, 2, 3]
console.log(num.length); // 5
```

# Arrays – Função Concat/Includes/IndexOf

---

```
const arr1 = ['a','b','c'];
const arr2 = ['d','e','f'];
const arr3 = arr1.concat(arr2);
console.log(arr3); //['a', 'b', 'c', 'd', 'e', 'f']
//includes verifica se um valor está presente
console.log(arr3.includes('b'));//true
console.log(arr3.includes('g'));//false
//indexOf verifica a posição de um valor presente.
console.log(arr3.indexOf('c'));//2
//Caso o valor não esteja presente, a função retorna -1
console.log(arr3.indexOf('g'));//-1
```

# Arrays – Função Reverse

---

```
const arr1 = ['a','b','c'];
const arr2 = ['d','e','f'];
const arr3 = arr1.concat(arr2);
console.log(arr3); //['a', 'b', 'c', 'd', 'e', 'f']
//Reverse irá sobrescrever o array com os valores reversos
arr3.reverse();
console.log(arr3); //['f', 'e', 'd', 'c', 'b', 'a']
```

# Arrays – Função Slice

---

```
const arr1 = ['a','b','c'];
const arr2 = ['d','e','f'];
const arr3 = arr1.concat(arr2);
console.log(arr3); //['a', 'b', 'c', 'd', 'e', 'f']
//Slice irá retornar uma cópia de uma porção do array
//de acordo com os índices passados.
const arr4 = arr3.slice(1);
console.log(arr4); //['b', 'c', 'd', 'e', 'f']

const arr5 = arr3.slice(2,4);
console.log(arr5); //['c', 'd']
```

# Repetição

## Repetição – Definição

---

- ❑ Estrutura de controle que executa um trecho de código *enquanto* uma dada condição for verdadeira.
- ❑ JS conta com as tradicionais estruturas *for* e *while*.

# Repetição – *For*

---

- ❑ For tradicional utilizando um contador

```
let num = [1,2,3,4,5]

for(let i = 0; i < num.length; i++)
    console.log(num[i]);
```

## Repetição – *For - in*

---

❑ For utilizado para percorrer todas as propriedades de um objeto.

```
const carro = {  
    nome: "Jaum",  
    idade: 32,  
};  
  
for(const i in carro)  
    console.log(i, carro[i]);
```

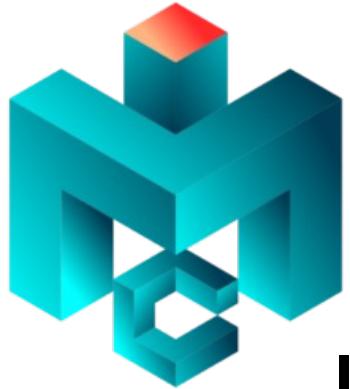
## Repetição – *For - of*

---

- ❑ For utilizado para percorrer coleções

```
let num = [1,2,3];
```

```
for(const valores of num)
    console.log(valores);
```



# Aula – 9

## Introdução ao DOM (Document Object Model)



**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Agenda

---



- ❑ O que é o DOM
- ❑ Buscando elementos no DOM
- ❑ Buscando conteúdo dos elementos
- ❑ Modificando estilização
- ❑ Adicionando Elementos
- ❑ Removendo Elementos

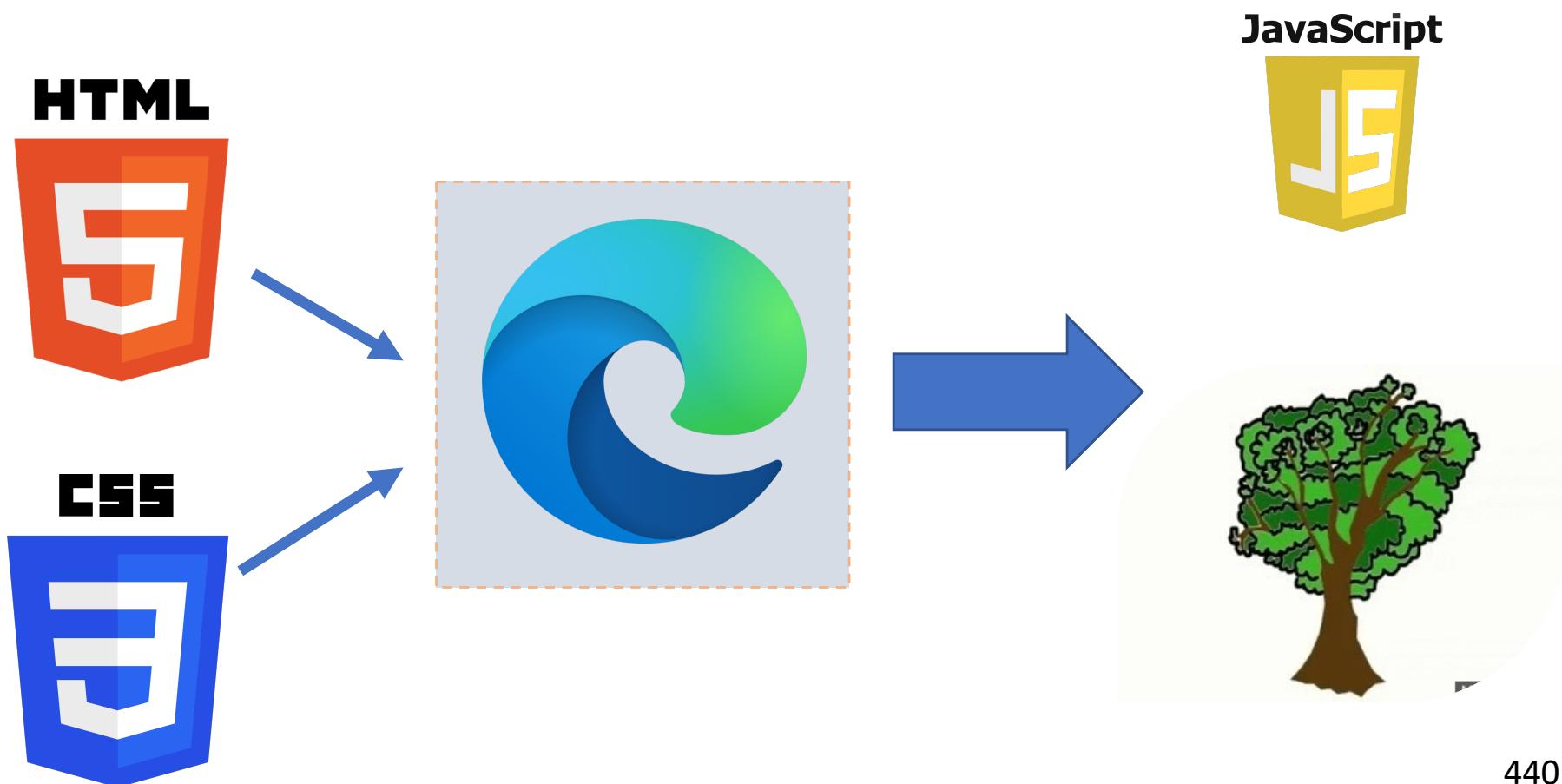
# Navegador Cria o DOM

---

- Durante o processo de renderizar e carregar a página WEB, o navegador transforma cada elemento em um objeto JS.
- Esses objetos são armazenados em uma estrutura de árvore.
- Essa árvore é o DOM.

# Navegador Cria o DOM

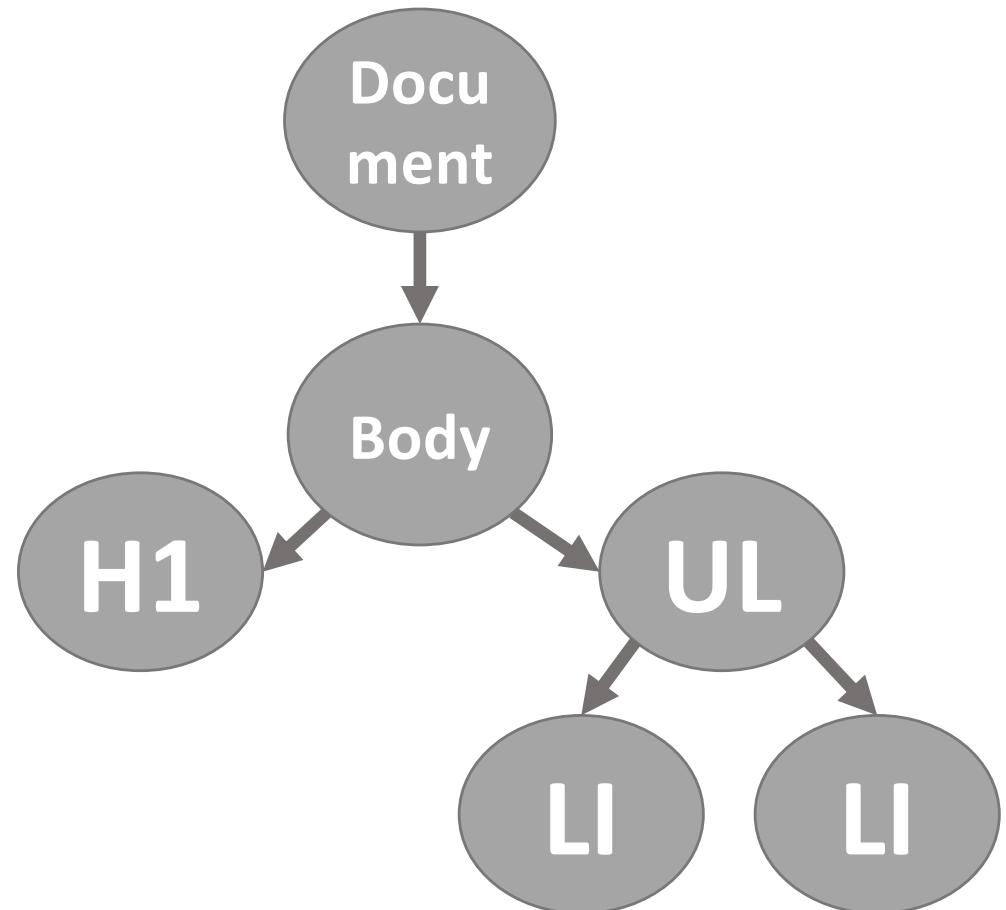
---



# DOM - Árvore

---

```
<body>
  <h1>Tarefas da semana</h1>
  <ul>
    <li>Estudar Web</li>
    <li>Estudar C++</li>
  </ul>
</body>
```



# Visualizar o DOM

---

- É possível invocar o objeto DOM diretamente no JS com *document*.
- O comando `console.dir(document)` apresenta toda a estrutura da página em formato objeto Javascript

# Exemplo DOM Wikipedia

---

```
> console.dir(document) VM145:1
  ▼ #document ⓘ
    ► jQuery361062609839005971591: {events: {...}, focusin: 1, handle: f}
    ► location: Location {ancestorOrigins: DOMStringList, href: 'https://pt.wiki
      write: (...)  
writeln: (...)  
URL: "https://pt.wikipedia.org/wiki/Minas_Gerais"
    ► activeElement: body.skin-vector.skin-vector-search-vue.mediawiki.ltr.site
    ► adoptedStyleSheets: Proxy(Array) {}
      alinkColor: ""
    ► all: HTMLAllCollection(8967) [html.client-js.vector-feature-language-in-t
    ► anchors: HTMLCollection []
    ► applets: HTMLCollection []
      baseURI: "https://pt.wikipedia.org/wiki/Minas_Gerais"
      bgColor: ""
    ► body: body.skin-vector.skin-vector-search-vue.mediawiki.ltr.sitedir-ltr.n
      characterSet: "UTF-8"
      charset: "UTF-8"
      childElementCount: 1
    ► childNodes: NodeList(2) [<!DOCTYPE html>, html.client-js.vector-feature-1
    ► children: HTMLCollection [html.client-js.vector-feature-language-in-head
      compatMode: "CSS1Compat"
```

# Seleção de Elementos





## Por que selecionar elementos?

Em muitos cenários desejamos mudar informações de elementos HTML específicos, por isso precisamos selecionar.



## Como fazer a seleção?

Originalmente, essa seleção poderia ser feita por *id*, *class*, *tag* e *name* (o atributo)



## O que é retornado?

O retorno é um objeto JavaScript com as propriedades do elemento HTML

## Selecionando pelo ID

---

- ❑ `Document.getElementById();`
- ❑ Esse método devolve o elemento com a ID especificada.
- ❑ Caso nenhum elemento seja encontrado, o valor atribuído é *null*.

# Selecionando pelo ID – Exemplo

---

```
const cabecalhoNull = document.getElementById('cabecalho');
//null
const cabecalhoCorreto = document.getElementById('heading');
//O id correto, nesse exemplo, é heading
//console.log irá imprimir o HTML
//<h1 id="heading">Calculadora Moderna Ultra Megazorde</h1>
```

## Selecionando pela *Class*

---

- `Document.getElementsByClassName();`
- Esse método devolve um objeto do tipo *HTMLCollection*, que pode ser visto como Array com os elementos HTML.
- Apesar de ser indexada e permitir iteração não é um Array convencional com as funções da API Array

# Selecionando pela Class – Exemplo 1

---

- Selecionando todos os botões de operação da calculadora

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
    console.log(botao); //os elementos HTML serão impressos
```

## Selecionando pela Class – Exemplo 2

---

- Imprimindo todos os valores do atributo “type” para cada elemento.

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
    console.log(botao.type); //o valor “button” será impresso
```

## Selecionando pela Class – Exemplo 3

---

- Transformando todos os botões de operação em soma (+)

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
    botao.innerText = '+';//innerText acessa o conteúdo visivel
```

# Selecionando por *Tags*

---

- ❑ `Document.getElementsByTagName.`
- ❑ Retorna uma *HTMLCollection* com todos os elementos da *tag* selecionada.
- ❑ Se nenhum elemento for encontrado, é retornada a coleção vazia, ao invés de *null*.

# Selecionando por *Tags* - Exemplo

---

- Selecionando todos os botões.

```
const botoes = document.getElementsByTagName('button');

for(let botao of botoes)
    console.log(botao); //todos os botões serão impressos
```



Tem como selecionar  
semelhante ao CSS?

Sim! A forma mais utilizada e moderna  
para buscar elementos no DOM é com  
*querySelector*.

## *querySelector - Definição*

---

- ❑ Com o avanço do JS, um novo método foi adicionado ao DOM.
- ❑ Este permite fazer a seleção da mesma forma que é feita no CSS.
- ❑ É possível buscar por id, class, atributos, e todas as outras combinações usadas no CSS.

## *querySelector - Exemplos*

---

- ❑ `document.querySelector('#id');`
- ❑ `document.querySelector('.class');`
- ❑ `document.querySelector('nome-tag');`
- ❑ Em todos esses casos, o retorno é apenas um elemento. Para retornar uma lista, se usa o método `querySelectorAll()`.

## *querySelector* – Exemplos ID

---

- Retornando elemento H1 por ID.

```
const h = document.querySelector("#heading");
console.log(h);
```

## *querySelector* – Exemplos Class

---

- Retornando o primeiro botão com a classe especificada

```
const btn = document.querySelector(".calculadora-tecla-operador");
```

## *querySelector* – Exemplos Tag

---

- Retornando o primeiro botão buscando pela *tag*.

```
const elemento = document.querySelector("button");
```

## *querySelectorAll* – Exemplos

---

- Retornando todos os botões com a classe especificada.

```
const btnS = document.querySelectorAll('.calculadora-tecla-operador');
console.log(btnS.length); //4
```

# Conteúdo Textual



# Conteúdo Textual

---

- ❑ Podemos manipular o conteúdo textual dentro dos elementos HTML com as seguintes propriedades:
  - ❑ innerHTML
  - ❑ innerText
  - ❑ textContent

# Conteúdo Textual - innerText

---

- Retorna o texto visível que se encontra no elemento HTML.
- Se algum texto estiver com a condição 'hidden' não será mostrado

## Conteúdo Textual - textContent

---

- Retorna o texto completo que se encontra no elemento HTML, mesmo que esteja “Hidden”.

# Conteúdo Textual – Exemplo innerText e textContent

---

## □ HTML Exemplo:

```
<p id="paragrafo">  
    Eu sou um paragrafo.  
    <span style="display: none;">Eu estou  
    hidden.</span>  
</p>
```

## □ JavaScript:

```
//Eu sou um paragrafo.  
document.querySelector('#paragrafo').innerText;  
  
//Eu sou um paragrafo. Eu estou hidden.  
document.querySelector('#paragrafo').textContent
```

# Conteúdo Textual - innerHTML

---

- Retorna o texto completo que se encontra no elemento HTML, incluindo HTML interno, isto é, elementos HTML descendentes.

# Conteúdo Textual – innerHTML Exemplo

---

```
<p id="paragrafo">  
    Eu sou um paragrafo.  
    <span style="display: none;">Eu estou  
hidden.</span>  
</p>
```

```
//Eu sou um paragrafo. <span style="display: none;">Eu estou hidden.</span>  
document.querySelector('#paragrafo').innerHTML;
```

# Conteúdo Textual - Modificações

---

- Com essas propriedades é possível também alterar o conteúdo e com isso sobrescrever utilizando o operador de atribuição.

```
document.querySelector('#paragrafo').innerText = 'Novo Texto';
document.querySelector('#paragrafo').innerText += ' Concatenando';
```

# Acessando os Atributos



# Acessando Atributos

---

- ❑ Os atributos dos elementos HTML podem ser acessando como propriedades do objeto JS. Pode-se utilizar o operador ponto (.) para acesso.
- ❑ Esse cenário é interessante quando se está criando os elementos, e deseja preencher os valores dos atributos.

## Acessando Atributos - Métodos

---

- ❑ É possível também utilizar dois métodos para acessar e modificar os atributos.
- ❑ `getAttribute()` para acessar
- ❑ `setAttribute()` para modificar

# Acessando Atributos - Exemplo

---

- Modificar a descrição da imagem, isto é, o atributo 'alt'. Usando as duas formas.

```
const mapaMG = document.querySelector('#minas-gerais img');

mapaMG.alt = 'O mapa de minas gerais';

mapaMG.setAttribute('alt', 'O mapa de minas gerais');
```

# Estilizando

css



JavaScript



# Modificando as propriedades do CSS - inline

---

- As propriedades de estilização podem ser modificadas diretamente no objeto JS, mas estas estão escritas em formato *Camel Case*.
- As propriedades inseridas via JS serão *inline* e pode não ser conveniente adicionar nesse formato.
- A propriedade *style* é acessada primeiro.

## Modificando as propriedades do CSS - Exemplo

---

- Cada propriedade é acessada separadamente

```
const pp = document.querySelector('#paragrafo');
pp.style.fontSize = '5em';
pp.style.color = 'blue';
```

# Acessa as propriedades do CSS

---

- As estilizações que não são inline, não conseguem ser lidas diretamente pelo propriedade *style*
- É preciso acessar o objeto global *window* e o método *getComputedStyle()* passando como parâmetro o objeto JS.
- Com isso é possível acessar as propriedades.

# Acessa as propriedades do CSS - Exemplo

---

```
const pp = document.querySelector('#paragrafo');
pp.style.color = 'red';

window.getComputedStyle(pp).color;//rgb(255, 0, 0)

window.getComputedStyle(pp).fontSize;// 16px
```

# Exercício 1

---

- Crie um código HTML/JS que escreva a frase RAINBOW com diferentes cores. Coloque o texto em um H1 e cada letra com uma *span*
- Como sugestão usem as seguintes cores:

```
const cores = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
```

**R A I N B O W**

# Exercício 1 - HTML

---

- Use o HTML fornecido

```
<body>
    <h1>
        <span>R</span>
        <span>A</span>
        <span>I</span>
        <span>N</span>
        <span>B</span>
        <span>O</span>
        <span>W</span>
    </h1>
</body>
```

## Modificando com *classList*

---

- Uma forma mais interessante de estilizar a partir do JS é manipulando as classes CSS.
- Para isso usamos a propriedade *classList*
- Esta apresenta uma API que permite adicionar, remover e ler as classes utilizadas no elemento HTML.

## Modificando com *classList*

---

- `classList.add()`
- `classList.remove()`
- `classList.toggle() // desliga/liga a classe`

```
const pp = document.querySelector('#paragrafo');
pp.classList.add('caixa-p');
```

# Percorrendo a Árvore DOM

---

- ❑ parentElement
  - ❑ Propriedade que devolve o elemento pai.
- ❑ childElementCount
  - ❑ Retorna o número de descendentes.
- ❑ children (HTMLCollection)
  - ❑ Retorna uma coleção com os descendentes.

# Adicionando Elementos

---

- ❑ createElement("nome tag")
- ❑ append("elemento ou texto")
- ❑ appendChild("apenas elementos")
- ❑ prepend("elemento ou texto") //add no início

# Adicionando Elementos

---

- ❑ `insertAdjacentElement(posição, elemento)`
  - ❑ Posição -> 'beforebegin' //Antes do parent
  - ❑ Posição -> 'afterbegin' // primeiro descendente
  - ❑ Posição -> 'beforeend' //último descendente
  - ❑ Posição -> 'afterend' //Após o parent

# Adicionando Elementos

---

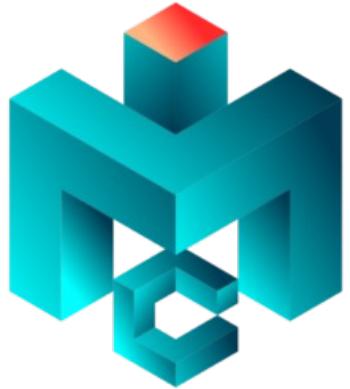
- ❑ insertAdjacentElement()//adiciona adjacente
- ❑ after()//Adiciona após

# Removendo Elementos

---

- Element.remove();

```
const img = document.querySelector('img');
img.remove();
```



# Aula – 9 – PT2

## Eventos com DOM

**Disciplina:** COM222/XDES03 – Programação Web/Sistemas Web

Prof: Phyllipe Lima  
*phyllipe@unifei.edu.br*

Universidade Federal de Itajubá – UNIFEI  
IMC – Instituto de Matemática e Computação

# Adicionando *Listeners*

---

- ❑ Elemento.addEventListener('evento', callback)
- ❑ Evento -> 'click' // clique no elemento
- ❑ Evento -> 'change' // mudança no elemento
- ❑ Evento -> 'input' // novo valor inserido no elemento

# Adicionando *Listeners* – Exemplo Botão

---

## Exemplo 1:

```
const btn = document.querySelector('#btn');

btn.addEventListener('click', () => {
    console.log('Ocorreu o clique');
});
```

## Exemplo 2

```
const btn = document.querySelector('#btn');

//Se necessário ter controle de qual elemento
//disparou, podemos usar o parametro "evento" (evt)
btn.addEventListener('click', (evt) => {
    console.log('Ocorreu o clique');
    console.log(evt.target); //Elemento
    console.log(evt.target.innerText); //Texto visivel do botão
});
```

# Adicionando *Listeners* – Exemplo Input

---

```
const inputText = document.querySelector('#input');

//Retorna o valor que se encontra no campo input
//Esse valor pode ser acessado dentro de listeners
let conteudo = inputText.value;
```