1: HTML e CSS e JS

segunda-feira, 5 de dezembro de 2022 23:27

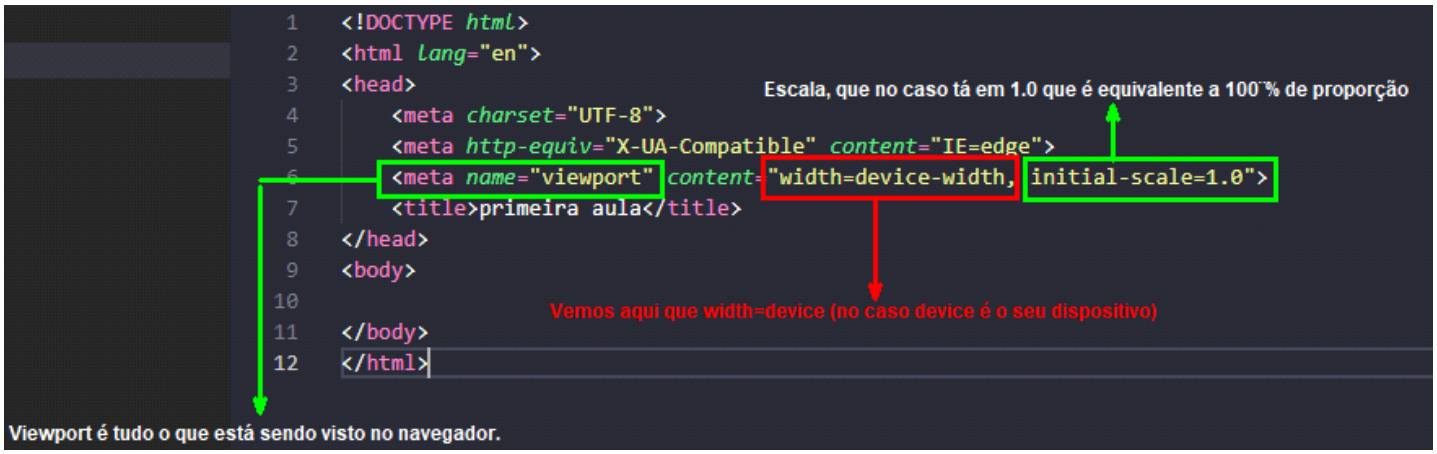
1.1 Meta viewport

segunda-feira, 5 de dezembro de 2022 23:15

Vejamos abaixo algumas considerações:

Podemos ter como base um mapeamento das aulas iniciais considerando estas anotações.

Sobre viewport, width e initial scale.



1.2 Principais comandos HTML

segunda-feira, 5 de dezembro de 2022 23:27

A princípio estes comandos abaixos são os comandos principais do HTML

Basicamente são essas:

- → Títulos --> <h1> - <h6>
- → Parágrafos e Span --> e
- → Listas -->
- → Links/ âncoras --> a
- → Imagens
- → Quebras

H1 até o H6 na verdade se trata de uma hierarquia de titulos.

- → **Paragrafos e SPAN.**

Para utilizarmos o Parágrafo, basta utilizarmos a tag , ela é a tag responsável por mantermos um texto longo dentro dela, por exemplo.

Para a tag é engraçado porque ela não quebra a linha, diferentemente da tag .
Por que usamos o afinal?
Simples, porque se você meter uma tag dentro de outra tag , simplesmente ele vai quebrar a linha e não vai dar para estilizar.
Por conta disso, utilizamos a tag span.

Veja exemplo abaixo:

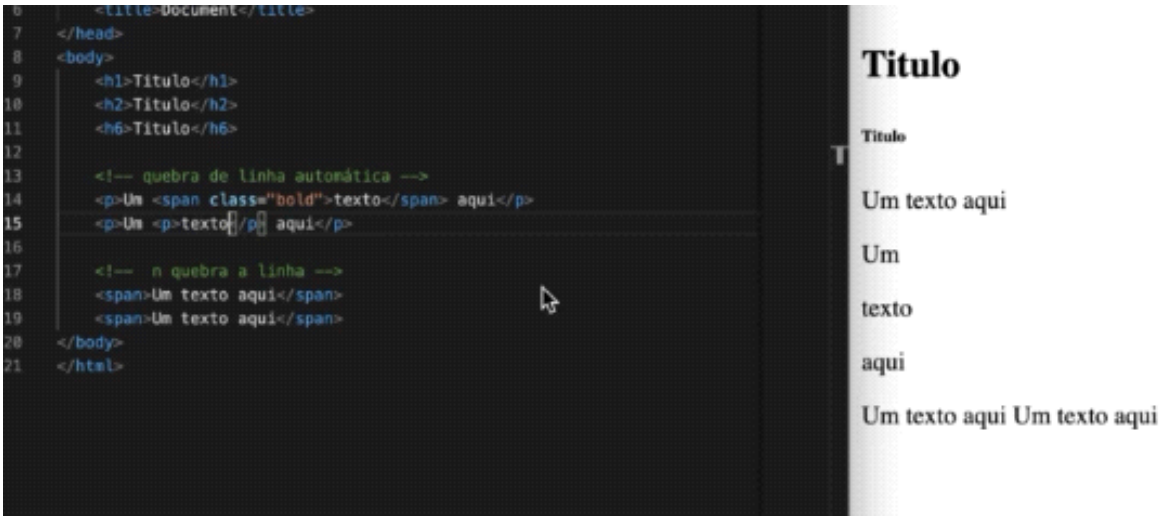
Código:

```
<style>

    span {font-size: large;
    color: blue ;}

    body {background-color: aqua;}
</style>
<body>

    <h1>Olá Mundo</h1>
    <p>Lorem ipsum dolor <br>sit amet</br> consectetur adipisicing elit. Beatae, error similique
    minus harum<span> voluptatum</span> maxime repellendus voluptates id perferendis repellat a
    labore odit ex et corporis <br>doloribus accusantium in</br> explicabo.</p>
    <h2>Seguimos aqui com uma especia de hierarquia de títulos</h2>
    <p>Não sei como continuar porém adoro HTML</p>

</body>
</html>
```

Resultado:

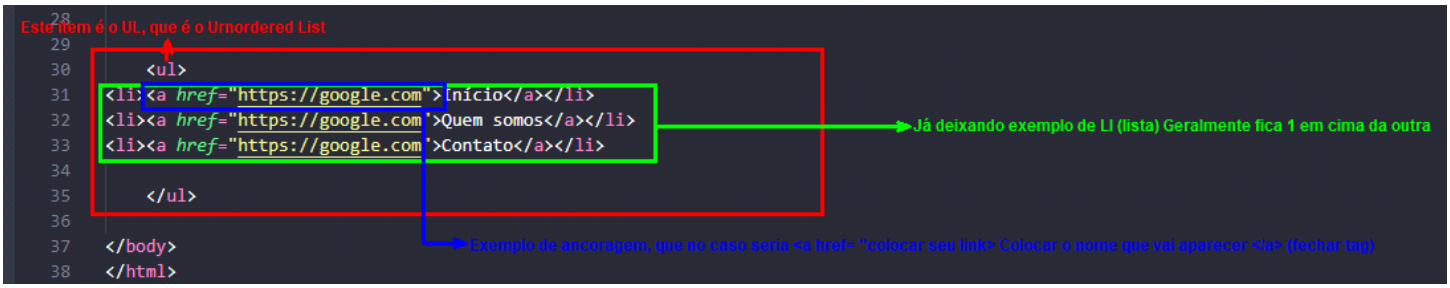
Diferente do

Vemos que o texto realmente quebra um paragrafo.



→ **UL e LI (Urnordered List e List)**

 O elemento HTML <ul> (ou elemento HTML de Lista desordenada) representa uma lista de itens sem ordem rígida, isto é, uma coleção de itens que não trazem uma ordenação numérica e as suas posições, nessa lista, são irrelevantes.



A tag img por exemplo:

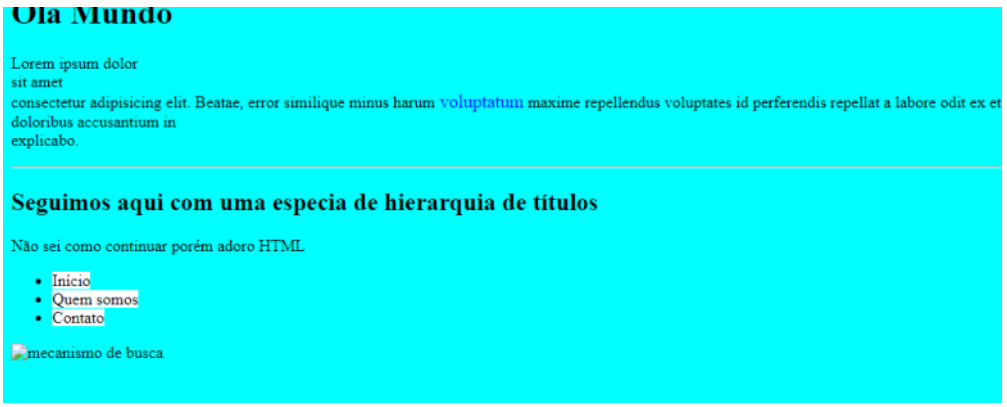
```
<!-- <IMG> não possui tag de fechamento veja exemplo abaixo-->
<img src="nome_do_caminho/diretório" alt="mecanismo de busca">
```

E por fim a tag <hr>

```
<h1>Olá Mundo</h1>
<p>Lorem ipsum dolor <br>sit amet</br> consectetur adipisicing elit. Beatae
repellendus voluptates id perferendis repellat a labore odit ex et corporis

<hr>
<h2>Seguimos aqui com uma especia de hierarquia de títulos</h2>
<p>Não sei como continuar porém adoro HTML</p>
```

Nada mais é que usar uma linha na horizontal para separar conteúdo (muito útil)

Veja exemplo abaixo:
Seria no caso essa linha cinza.



1.3 CSS Introdução

segunda-feira, 5 de dezembro de 2022 23:59

→ O que é CSS

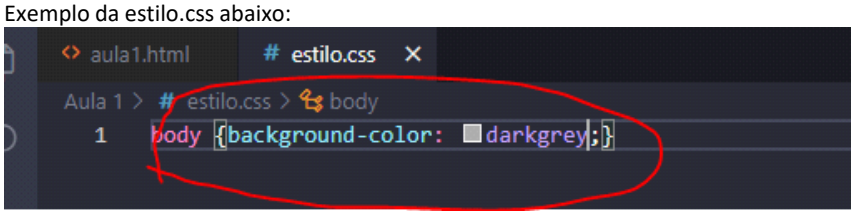
CSS (Folha de Estilo em Cascata) é o código que você usa para dar estilo à sua página Web. CSS básico apresenta tudo que você precisa para começar. Responderemos a perguntas do tipo: Como mudo meu texto para preto ou vermelho? Como faço para que meu conteúdo apareça em determinados lugares na tela? Como decoro minha página com imagens e cores de fundo?

→ Como importar o CSS

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="estilo.css">
<title>primeira aula</title>
/head>
```

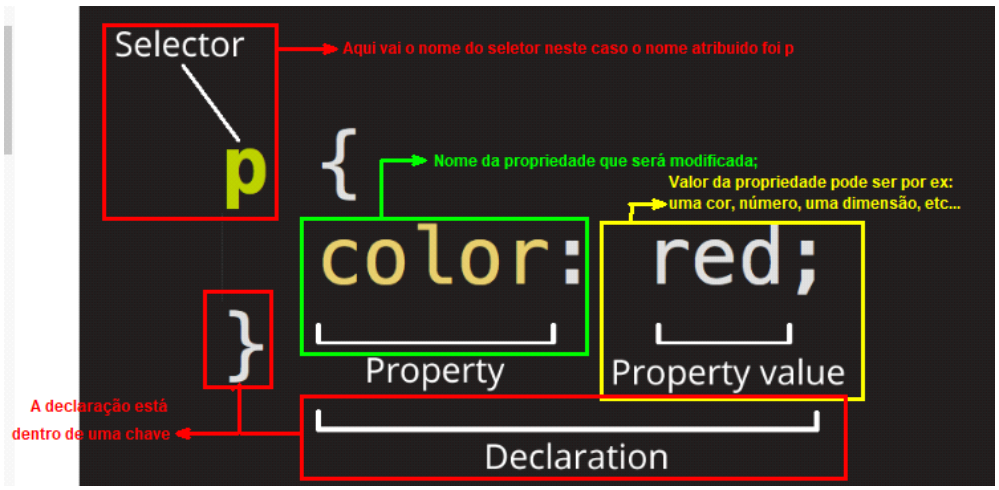
É assim que se chama uma folha de estilo

Exemplo da estilo.css abaixo:



→ Aplicando regras a um elemento (como escrever)

Veja a anatomia de uma declaração:

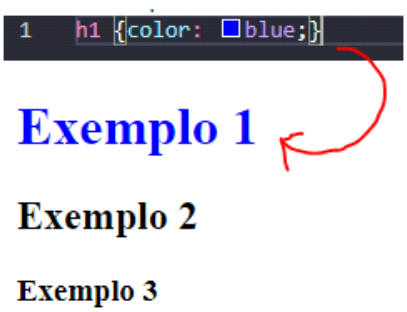


→ Selecionando multiplos elementos por exemplo: Vemos abaixo que pudemos selecionar as tags mais usadas da html, sem precisar colocar um ponto ou uma hashtag antes do nome

```
p, li, h1 {
  color: red;
}
```

Veja na pratica:

Primeiro exemplo só com o h1 modificado na cor azul:



Agora vemos com varios estilos modificados:

```
1   h1, h2, h3 {color: blue;}
```

Exemplo 1

Exemplo 2

Exemplo 3

→ Tipos de seletores

Bom, neste caso, existem varios tipos de seletores no CSS, que se faz necessário aprender antes de sair usando adoidado.

Tag HTML.

```
Para as tags normais, basta colocar conforme exemplo abaixo

p {  }
```

Detalhe que para as tags normais, não precisamos de .p nem #p, basta inserir a tag e abrir e fechar as chaves para inserir os parâmetros.

Tag "id" basicamente é um identificador de um elemento.
Para JS vamos utilizar muito id.

x

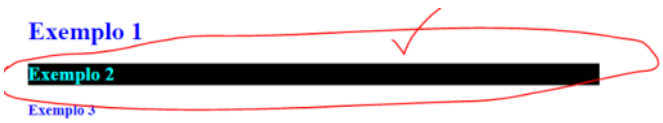
Veja o exemplo

```
                                Veja que eu declarei o h2 como id="teste", diferente das outras tags
<h1>Exemplo 1</h1>
<h2 id="teste1">Exemplo 2</h2>
<h3>Exemplo 3</h3>
```

No CSS

```
#teste1 {background-color: black; color: aqua;}
```

Resultado:



E por fim, class:

Para o seletor class, utilizamos da seguinte maneira:

```
<div class="container-basico">
    <h1>Exemplo 1</h1>
    <h2 id="teste1">Exemplo 2</h2>
    <h3>Exemplo 3</h3>
</div>
```

CSS

```
.container-basico {background-color: red;}
```

Resultado:

Toda vez que utilizamos a CLASS, utilizamos o .nome-class para chamar no css.

NOTA: Utilize id quando você quiser identificar apenas um elemento no html e utilize class quando quiser se referir a mais de um elemento.

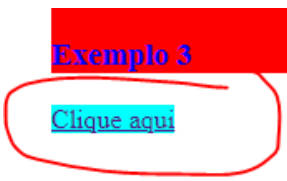
Pseudoclasses:

Pseudo-classes permitem que você aplique um estilo a um elemento não apenas em relação ao conteúdo da árvore do documento, mas também em relação a fatores externos como o histórico de navegação (:visited, por exemplo), o status do seu conteúdo (como :checked em certos elementos de um formulário), ou a posição do mouse (como :hover, que permite saber se o mouse está sobre um elemento ou não).

Veja um exemplo de uma pseudo classe utilizando :hover

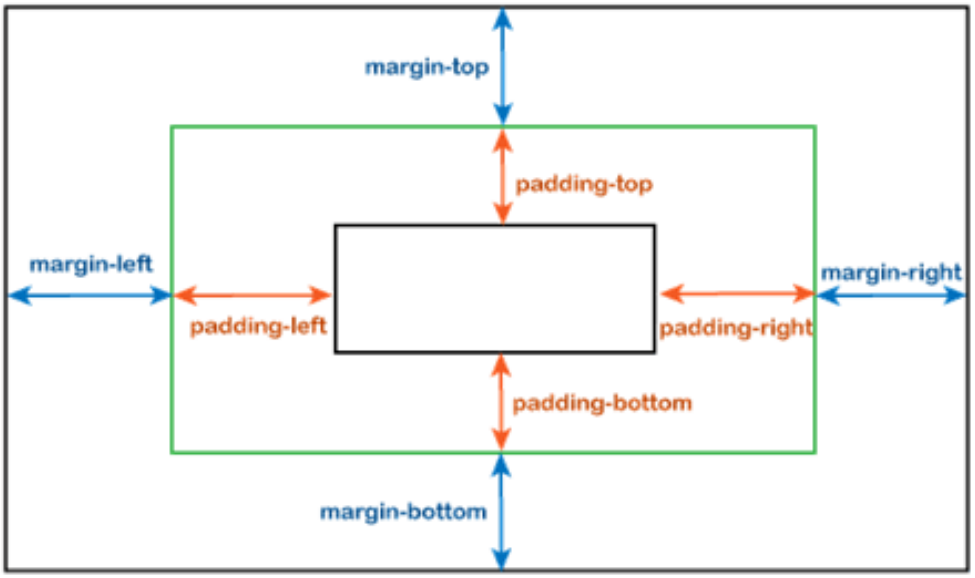
```
<div>
    <a id="clique-aqui" href="#">Clique aqui</a>


</div>
```

```
#clique-aqui:hover {background-color: aqua;}
```



Esse fundo azul no "Clique aqui" por exemplo, só apareceu pq eu passei o mouse em cima (efeito de :hover).

→ Padding, margin, border.



Exemplo 1

Exemplo 2

Exemplo 3

Quando formos mexer na Margin, do clique-aqui por exemplo, estaremos alteranda entre esses dois.

Clique aqui

1.4 - Importando Google Fonts

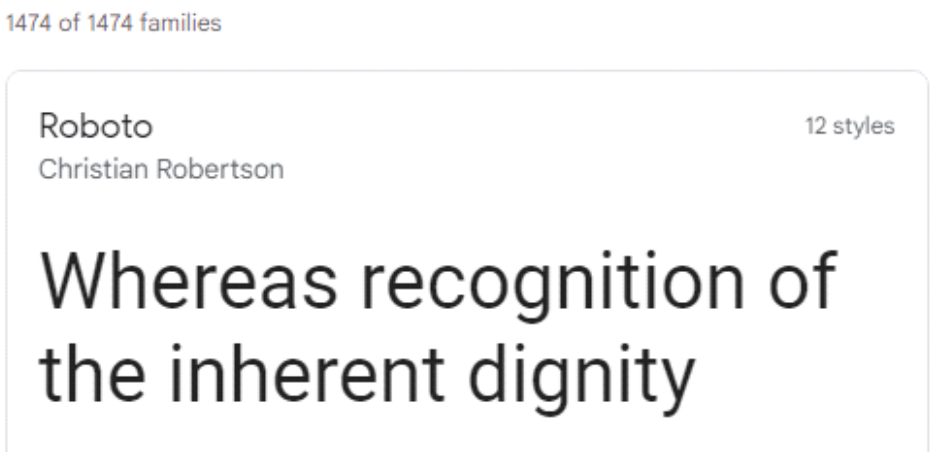
sexta-feira, 9 de dezembro de 2022 05:44

Primeiro precisamos acessar o google fonts.

Em http://fonts.google.com podemos encontrar diversas fontes gratuitas.

A principio vamos mostrar na pratica, veja bem.

Abaixo, clicamos na fonte "Roboto" como exemplo.



No menu da fonte, você pode optar por usar na web (html) ou importar só pro CSS.

Eu particularmente opto por usar diretamente no CSS por ser menos trabalhoso.

Para tal, basta você copiar e colar o código no seu CSS, esses são os dois passos:

Passo 1: Copiar

```
<style>
@import url('https://fonts.googlea
pis.com/css2?family=Roboto+Condens
ed&display=swap');
</style>
```

Passo 2: Colar no CSS

```
@import url('https://fonts.googleapis.com/css2?family=Roboto+Condensed&display=swap');
```

E aí para poder especificar no CSS a fonte que irá aplicar nas marcações, basta copiar e colar as regras de especificações

Copiar @import

```
<style>
@import url('https://fonts.googleapis.com/css2?family=Rubik+Vinyl&display=swap');
</style>
```

Copiar regras

CSS rules to specify families

```
font-family: 'Rubik Vinyl', cursive;
```

Colar

```
@import url('https://fonts.googleapis.com/css2?family=Rubik+Vinyl&display=swap');


h1, h2,
h3 {font-family: 'Rubik Vinyl', cursive;
    color: blue;
}
```

Resultado

1.5 - Importar ícones do feather icons

sexta-feira, 9 de dezembro de 2022 05:55

Bibliotecas:

http://feathericons.com
https://fontawesome.com/

Primeiramente vamos acessar o http://feathericons.com

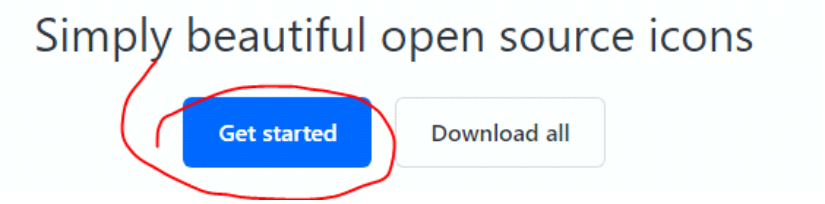
A princípio, trata-se de uma biblioteca de ícones, os quais podemos customizar os icones e etc.

Os ícones da biblioteca por exemplo, são bem mais leves do que um ícone com imagem por exemplo.

Vamos por a mão na massa.

No Feathericon, que é oque vamos usar no exemplo, seguimos o seguinte passo.

Clicamos no icone botão Get Started



Irá nos redirecionar para o github da feathericon.

Depois, precisamos copiar o <script> da biblioteca

```
<!DOCTYPE html>
<html lang="en">
  <title></title>
  <script src="https://unpkg.com/feather-icons"></script>
  <body>

    <!-- example icon -->
    <i data-feather="circle"></i>

    <script>
      feather.replace()
    </script>
  </body>
</html>
```

Depois, colamos o <script> dentro do <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">
    <link rel="stylesheet" href="style.css">
    <script src="https://unpkg.com/feather-icons"></script>
    <title>Document</title>
</head>
<body>
```

Depois, vamos copiar o <script> feather.replace()</script> e vamos colocar antes de fechar o <body>

```
<script>
  feather.replace()
</script>
</body>
```

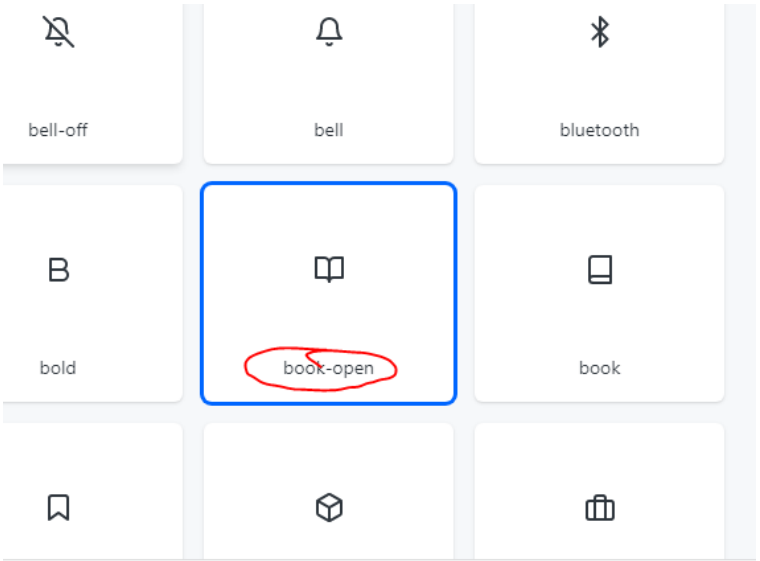
```
    <script>
        feather.replace()
    </script>
</body>
</html>
```

E por fim, a tag que vc vai utilizar para poder usar o ícone é a seguinte:

```
<!-- example icon -->
<i data-feather="circle"></i>
```

Note que data-feather está sublinhado em vermelho.
Esse é o nome do ícone padrão, porém vamos alterar para o nome do ícone da biblioteca
Tem que estar escrito identico a biblioteca.

Veja só, abaixo o nome do ícone que iremos usar:




```
<ul>
<li><i data-feather="book-open"></i></li>
<li><i data-feather="book-open"></i></li>
<li><i data-feather="book-open"></i></li>
<li><i data-feather="book-open"></i></li>
</ul>
```

Resultado:

- 
- 
- 
- 

Agora com outros ícones para você ver:

- 
- 
- 
- 

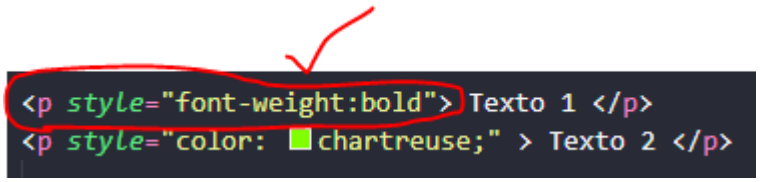
```
<ul>
<li><i data-feather="book-open"></i></li>
<li><i data-feather="bell"></i></li>
<li><i data-feather="bluetooth"></i></li>
<li><i data-feather="book"></i></li>
</ul>
```

Nota: Algumas bibliotecas permitem que você estilize.

1.6 - Inline style

sexta-feira, 9 de dezembro de 2022 06:14

O inline style nada mais é que aplicar o estilo dentro da própria tag, veja o exemplo abaixo:

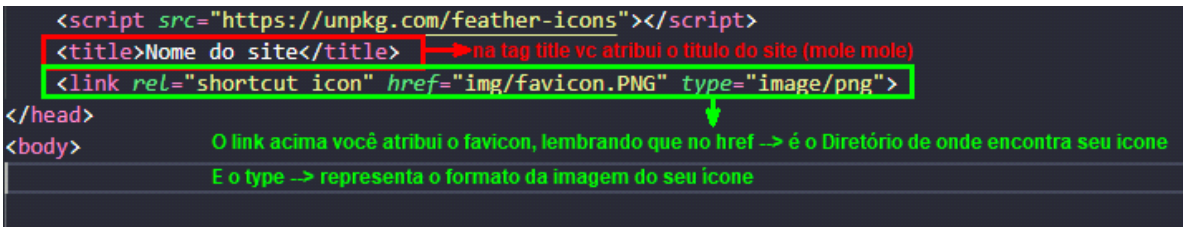


Nota: O CSS inline é usado para dar **estilo a um elemento HTML específico**. Para este estilo de CSS, você somente vai precisar adicionar o atributo style para cada tag HTML, sem usar os seletores.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual.

1.7 - Favicon e Titulo do site

sexta-feira, 9 de dezembro de 2022 06:20



Nota: O favicon aparece ao lado do título da página na guia do navegador. Ele permite que os usuários identifiquem e naveguem até o site com facilidade, mesmo que tenham várias guias abertas. Os tamanhos de resolução permitidos para favicons são (px): 24 x 24, 36 x 36, 48 x 48.

1.8 - Width e Height

sexta-feira, 9 de dezembro de 2022 08:23

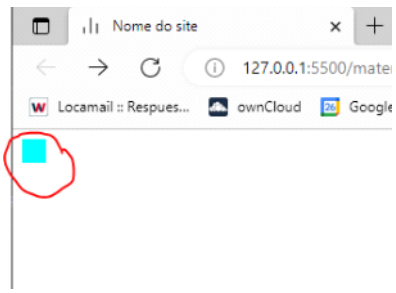
Vamos dar uma analisada no width e o height na pratica.

Veja por exemplo o width

Temos o código abaixo:

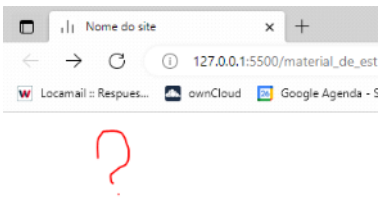
```
.container {background-color: aqua;
width: 20px;
height: 20px;}
```

E temos também o seguinte resultado, sim um quadrado de 20px por 20px



E se eu mudar a taxa de proporção para 100% de width e height?

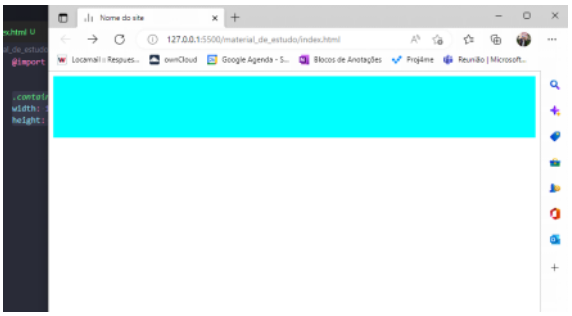
```
.container {background-color: aqua;
width: 100%;
height: 100%;}
```



Sim, desapareceu foi tudo. No entanto, se eu alterar o parâmetro de height para 100px por exemplo, o meu display passa a ser mostrado.

```
.container {background-color: aqua;
width: 100%;
height: 100px;}
```

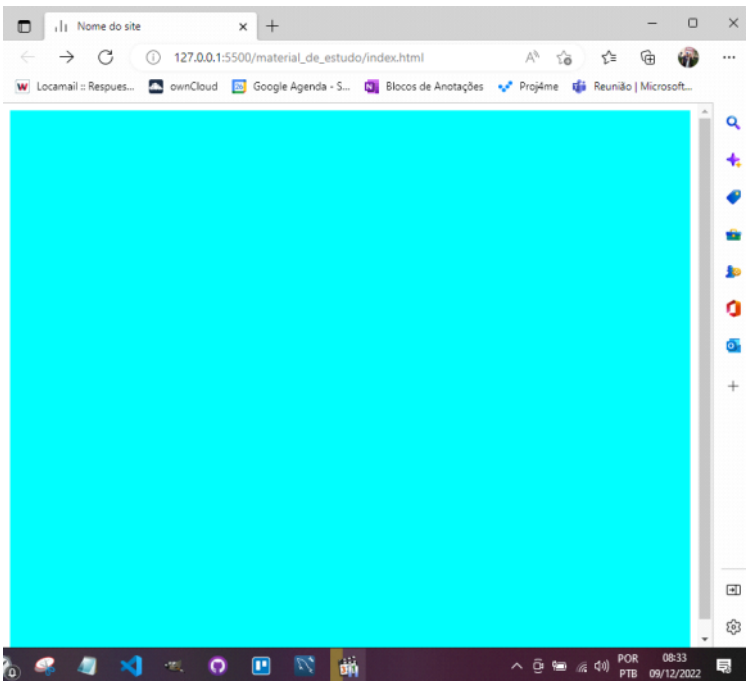
Veja só:



Pois bem, isso acontece porque, por alguma razão, se colocarmos o height com o parâmetro de xx%, não temos um resultado.

Agora veja o que acontece se utilizarmos o height: 100vh

```
.container {background-color: aqua;
width: 100%;
height: 100vh;}
```



Veja só, agora a div.container ficou com 100% de largura e 100vh (view height) (100% da altura da TELA)
E passou a ocupar a tela toda.

Ou seja, se quisermos que o height cubra toda a altura da tela, temos que usar 100vh.

```
.container {background-color: aqua;
width: 100%;
height: 10vh;
/* veja abaixo que, definimos um teto de largura em 300px, ou seja, não vai
estrapolar este tamanho
geralmente o máximo é 900 e 1200 px */
max-width: 300px;
/* margin auto pega e centraliza o elemento, pois ela bota uma margem
automatica nos lados
assim, não importa o tamanho da tela, seu conteúdo vai ficar sempre estático
ali */
margin: auto;
```

```
/* utilizamos o min e max, para limitar o tamanho do componente
as vezes com botão por exemplo, se você que ele tenha um valor, mas se ele ir
aumentando até um valor máximo você consegue por um teto e piso de valor de
configuração*/
min-width: 200px;
min-height: 100px;
}
```

1.9 Background color

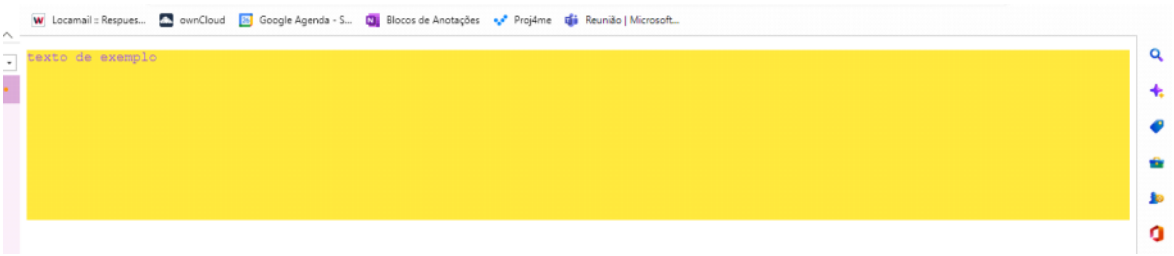
sexta-feira, 9 de dezembro de 2022 08:46

Cuidado, não confunda background-color
Com color.

Background-color muda o fundo do seu elemento.
Enquanto color sempre será pertinente a cor do texto/ fonte;

```
.container {width: 100%;
    height: 200px;
    background-color: rgb(255, 226, 60);
    color: blueviolet;
font-family: 'Courier New', Courier, monospace;}
```

Resultado:



1.10 - Display

sexta-feira, 9 de dezembro de 2022 21:49

A princípio deixarei algumas considerações abaixo:

Display

Os elementos html se dividem em 2 grandes categorias: os elementos do **tipo block** e do **tipo inline**.

Block

- Elementos como o `div`, `p`, `h1`, `h2`, ... `h6`,
- Sempre **ocupam toda a largura possível de tela** e **começam em uma nova linha**
- Pode manipular o `width` e `height`

Inline

- Elementos como `span`, `img`, `a`
- Usam **somente o espaço necessário** e não precisam começar em uma nova linha
- Elemento aparece próximo de outro elemento e na mesma linha
- Não consegue controlar o `height` e `width`
- Desconsidera `margin` e `padding` top e bottom

Inline-Block

- Muito usado para criar menus
- Responde a `width` e `height`

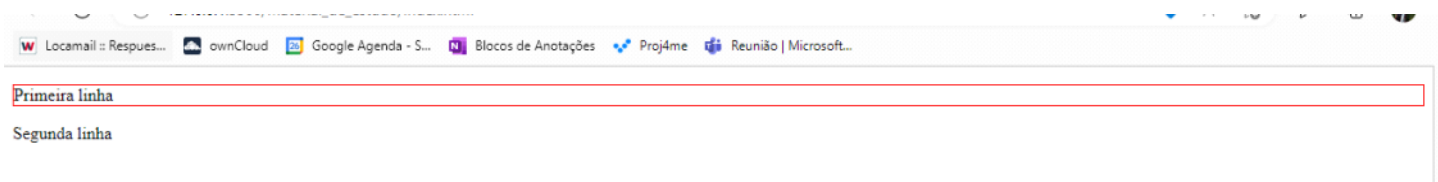
Qual é o display padrão CSS?
display é a propriedade mais importante do CSS para controlar o layout. Cada elemento tem um valor padrão para o display dependendo de seu tipo. O valor padrão na maioria dos elementos é normalmente block ou inline.

Dê uma olhada no exemplo abaixo:

```
#p1 {border: 1px solid red;}
```

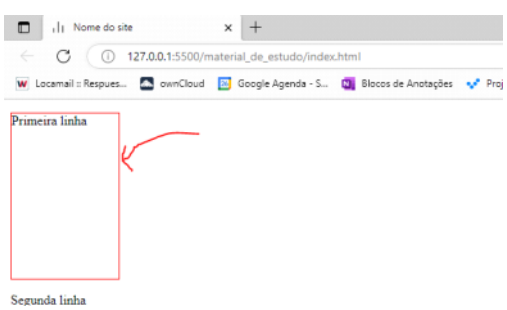
Colocamos uma borda para dar exemplo, e note que a borda pegou uma caixa que foi até o final da tela, mesmo o texto "primeira linha" sendo curto.

Veja o exemplo do resultado:



E mais importante, veja que eu posso manipular o tamanho e a largura do block:

```
#p1 {border: 1px solid red;
width: 10%;
height: 200px;}
```

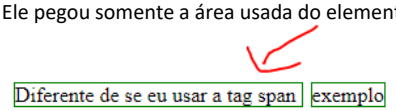


Utilizamos o estilo:

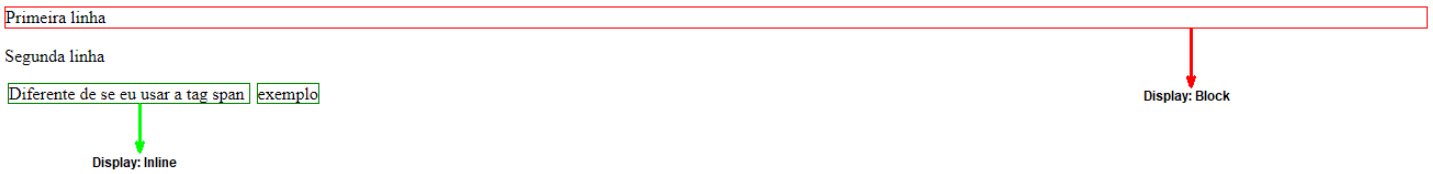
```
span {border: 1px solid green;
margin: 3px;}
```

E obtivemos o seguinte resultado:

Lembrando que a borda, ao inves de pegar a largura da tela toda, por exemplo.
Ele pegou somente a área usada do elemento.



Veja a diferença entre os dois:



Lembrando que nos elementos que são inline, eu não consigo mexer nas propriedades height e width por exemplo.

Display None;

Mas afinal, se o elemento some, pra que serve display none?

```
.exe1 {background-color: blueviolet;
width: 100px;
height: 100px;
display: none;}
```

display:none retira o elemento do layout da página. Mas você ainda pode continuar manipulando ele no DOM.
visibility:hidden deixa de mostrar o elemento, ou seja, ele deixa de ser visível na página mas seu espaço continua ocupado, ou seja, o layout da página não é alterado por causa disto

1.10.1 - Display None vs Visibility

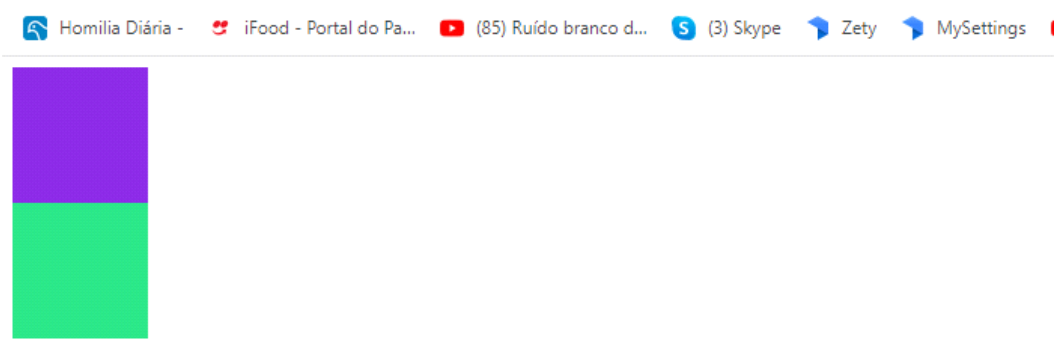
sexta-feira, 9 de dezembro de 2022 22:53

Quando temos o display none, passando o mouse em cima, ele fica praticamente INEXISTENTE.

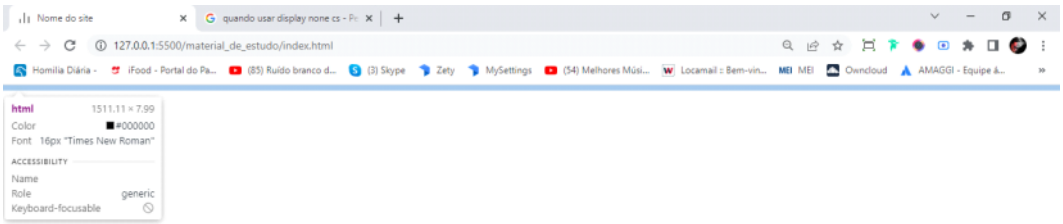
Analise o css abaixo:

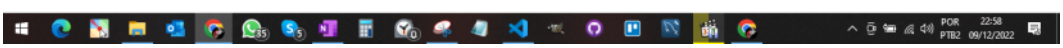
```
.exe2 {
    background-color: blueviolet;
    width: 100px;
    height: 100px;

}
.exe3 {
    background-color: rgb(43, 226, 134);
    width: 100px;
    height: 100px;
```

Resultado:

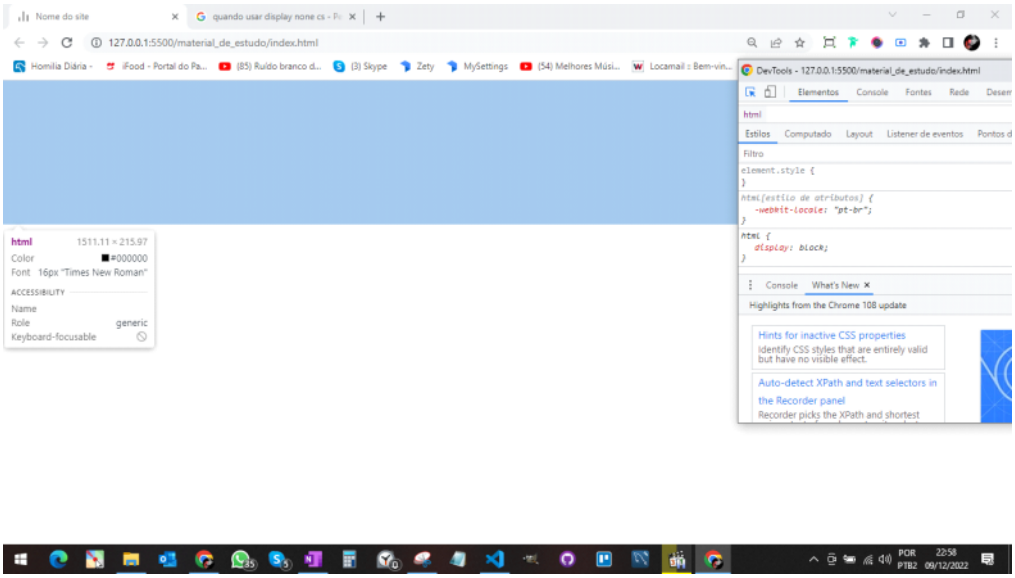


Veja que no dev, praticamente sumiu o elemento:





Agora veja se eu coloco a visibility como hidden:



1.11 - Position

sexta-feira, 9 de dezembro de 2022 23:10

A propriedade position especifica o tipo de método de posicionamento usado para um elemento (estático, relativo, fixo, absoluto ou fixo).

Existem cinco valores de posição diferentes:

→ static
→ relative
→ fixed
→ absolute
→ Sticky

Onde,

Static:

Os elementos HTML **são posicionados estáticos por padrão.**

Os elementos posicionados estáticos não são afetados pelas propriedades superior, inferior, esquerda e direita.
Um elemento com position: static por exemplo, não é posicionado de nenhuma maneira especial; está sempre posicionado de acordo com o fluxo normal da página:
Este elemento tem a posição: static;

Exemplo de css:

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

Relative: Um elemento com position relative, é posicionado em relação à sua posição normal.

Definir as propriedades superior, direita, inferior e esquerda de um elemento relativamente posicionado fará com que ele seja ajustado para longe de sua posição normal. Outros conteúdos não serão ajustados para caber em qualquer lacuna deixada pelo elemento.

Este elemento tem posição: relativo;

Exemplo de css:

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

Temos 3 quadrados alinhados na vertical.

Teste 1

Teste 3

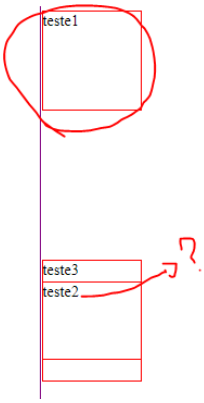
Teste 2

Isso aconteceu porque mudamos a configuração de posição da div teste 2.

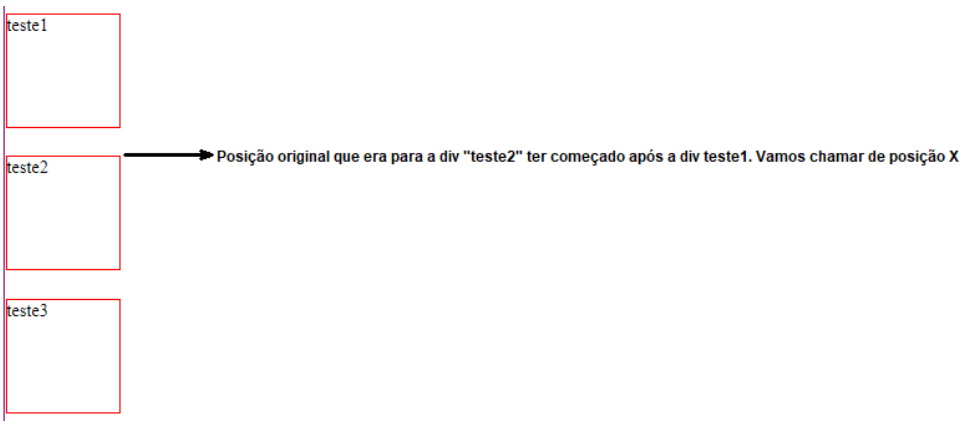
Veja exemplo no código:

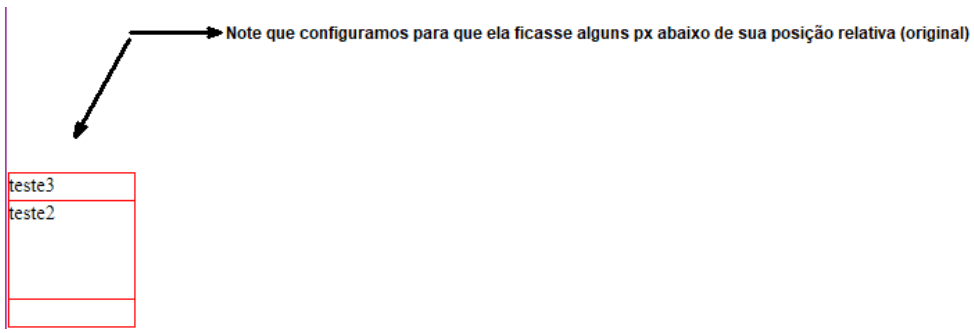
```
div {border: 1px solid red; margin-top: 25px;
width: 100px;
height: 100px;}

#teste2 {position: relative;
top: 150px}
```

Exemplo abaixo:



Ou seja:





Fixed

Um elemento com position: fixed, ele é posicionado em relação à viewport, o que significa que ele sempre permanece

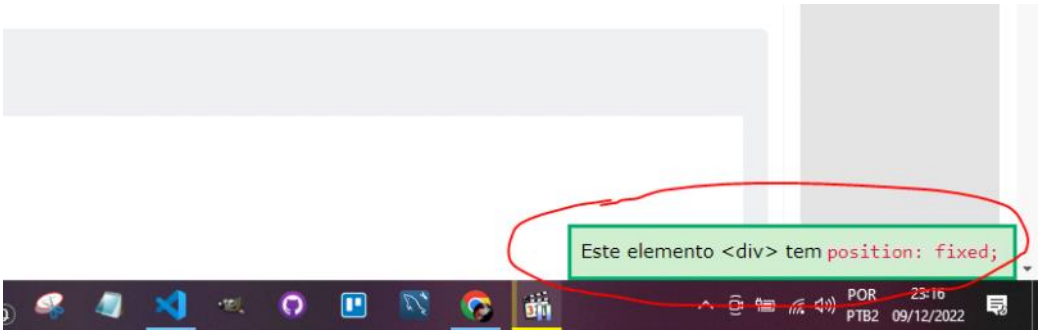
no mesmo lugar, mesmo que a página seja rolada. As propriedades superior, direita, inferior e esquerda são usadas para posicionar o elemento.

Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado.

Código:

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

Exemplo:



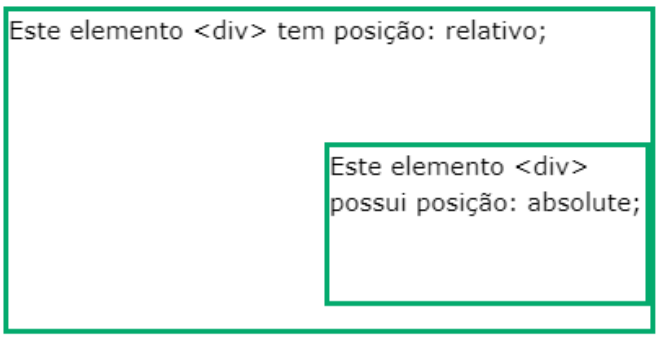
Position: absolute

Elemento com absolute, é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à viewport, como fixo).

No entanto; se um elemento posicionado absoluto não tiver ancestrais posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página.

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

Aqui está um exemplo simples:



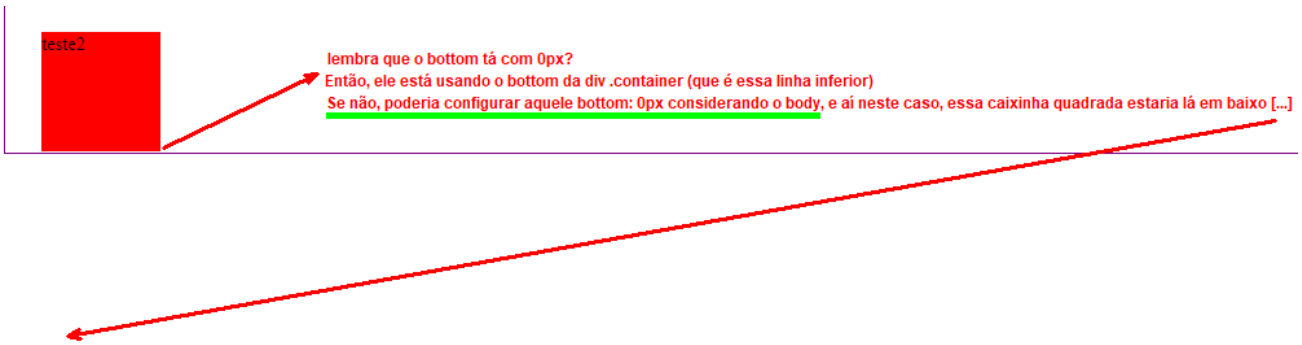
```css
.container {
    width: 100%;
    height: 500px;
    border: 1px solid purple;
    position: fixed;
}
```

A partir de então, essa é a div pai da div abaixo, pois a div teste2 está dentro da div .container. Além disso, ela não está com a position atribuida como static.

```css
#teste2 {
    position: absolute;
    bottom: 0px;
    left: 30px;
    background: red;
}
```

Essa div é a div filha, com a posição de absolute.
pode se dizer que a posição dela, irá usar como ponto de referência a div pai, que no caso é a .container

O bottom que configura a posição a partir da parte inferior do elemento, está com 0px
Logo, a posição de 0px irá ser considerada com base na parte inferior da div .container veja no exemplo:



Então, o legal do absolute, é que você consegue dizer a qual elemento ele vai ser relativo.

O position Absolute é um grande quebra galho no CSS. Com ele você pode posicionar qualquer elemento de acordo com o elemento pai que tenha um position diferente de static.

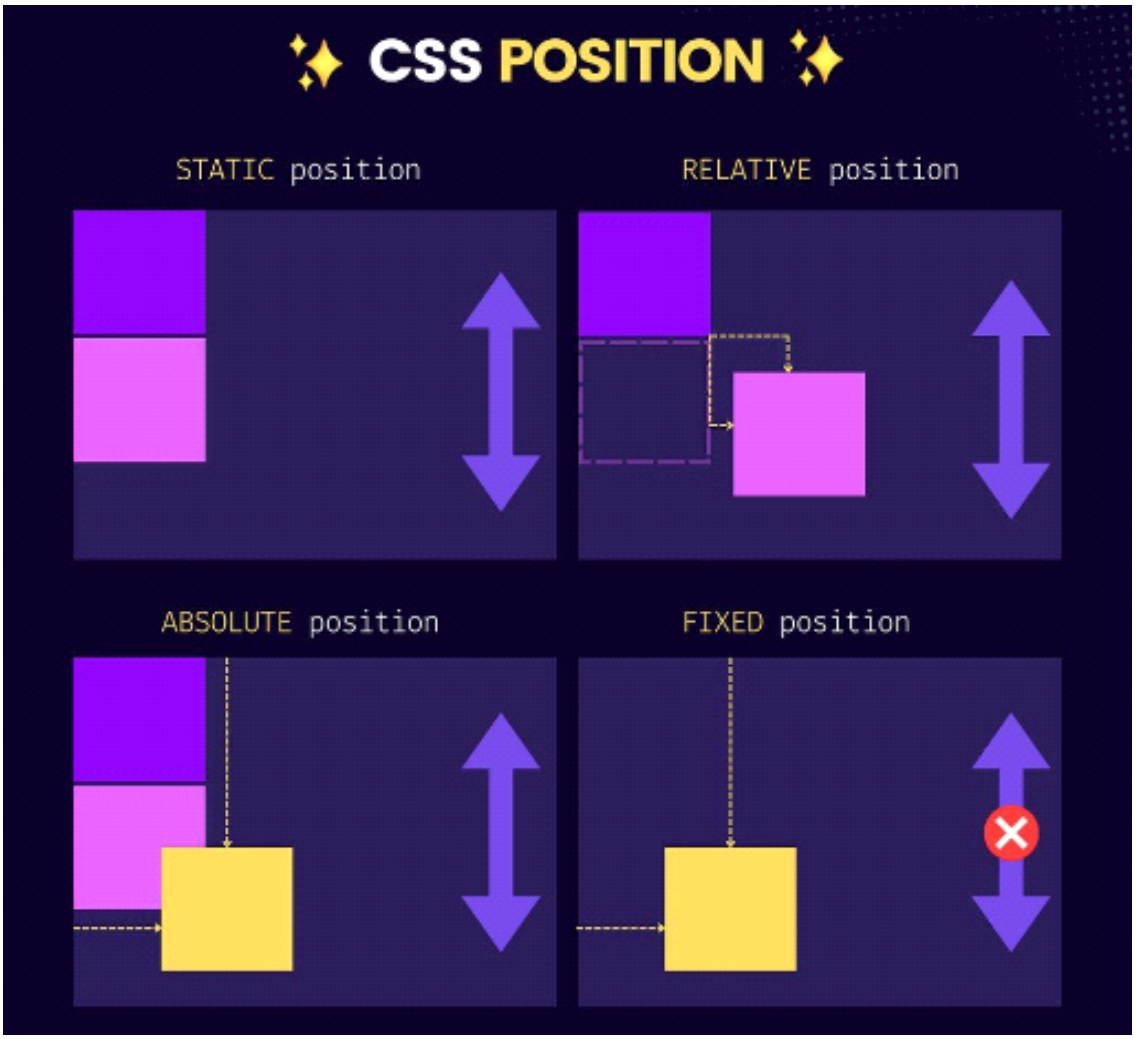
Sticky

Para a função de position sticky você precisa colocar o local que ele vai parar e quando atingir esse local, automaticamente ele vai virar fixed.

No caso, colocamos o top: 0;

Então quando essa div, chegar ao top 0 do scroll do mouse, ela muda pra fixed.
Legal né brother?

```css
header {
    background-color: aquamarine;
    position: sticky;
    top: 0;
    width: 100%;
    height: 20vh;
```


CSS POSITION
STATIC position
RELATIVE position
ABSOLUTE position
FIXED position

1.12 - Z Index

sábado, 10 de dezembro de 2022 00:47

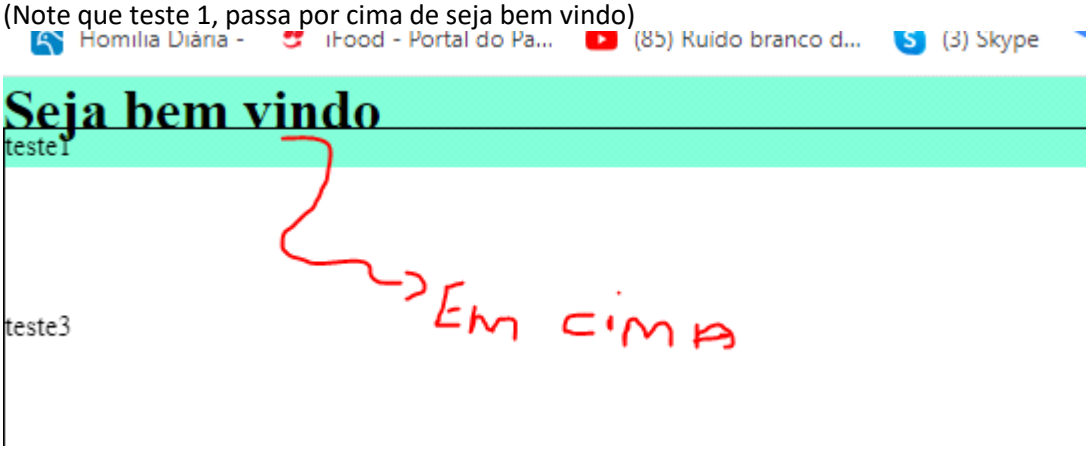
O que é o z-index?

Para deixar uma à frente da outra precisamos alterar o eixo Z dela. É nesse momento que entra a propriedade z-index. Uma coisa interessante sobre essa propriedade é que ela só funciona se a propriedade position do elemento tiver um valor diferente de static.

Código sem z-index:

```
header {
    background-color: aquamarine;
    position: sticky;
    top: 0;
    width: 100%;
    height: 50px;


}
```

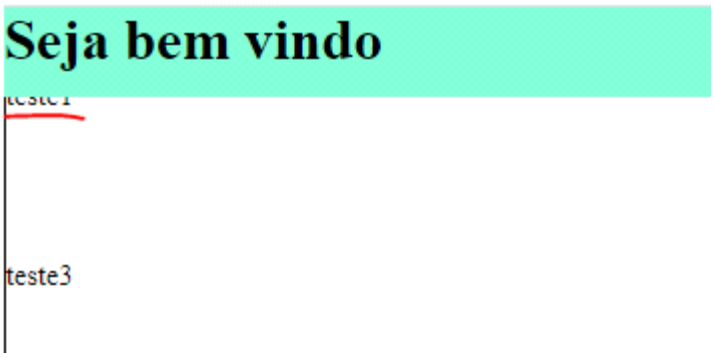
Resultado:
(Note que teste 1, passa por cima de seja bem vindo)



Com z-index:

```
header {
    background-color: aquamarine;
    position: sticky;
    top: 0;
    width: 100%;
    height: 50px;
    z-index: 1;

}
```

Note que o teste 1 agora ta passando por trás do seja bem vindo.

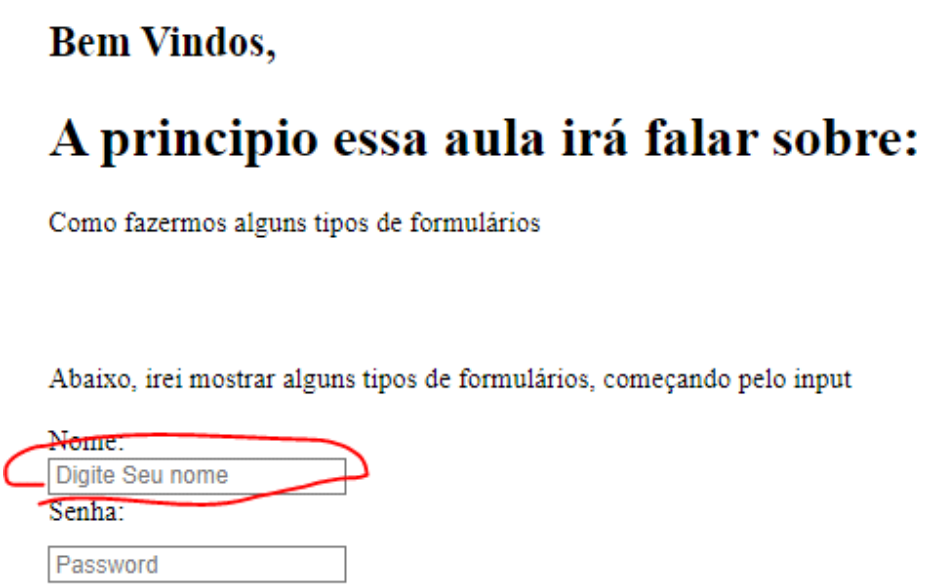

Seja bem vindo
teste3

1.13 - Formulários HTML

sábado, 10 de dezembro de 2022 19:33

Bem-vindo meus consagrados, hoje vamos falar dos formulários.

Exemplo de formulário:



<label></label>

Vou começar explicando alguns conceitos, o que é <label> e pra que serve a <label>:

O elemento label:

Um elemento **HTML <label> **representa uma legenda para um item em uma interface de usuário. Ele pode estar associado com um elemento de controle, colocando este dentro do elemento label , ou usando o atributo for . Tal controle é chamado o controle etiquetado do elemento etiqueta.

Exemplo de Label:

```
1 <label>Nome:</label>
2 <input type="text" name="Name"/>
```

A tag <label> define um rótulo para os seguintes elementos:

→ <button>
→ <input>
→ <meter>
→ <output>
→ <progress>
→ <select>
→ <textarea>

Benefícios de rótulos:

O uso adequado de rótulos com os elementos acima beneficiará:

Usuários com qualquer tipo de dificuldade visual (o rótulo será lido em voz alta quando o usuário estiver focado no elemento);
Usuários que tem dificuldade em clicar em regiões muito pequenas (como caixas de seleção), pois quando um usuário clica no texto dentro do <label> ele alterna a entrada (aumentando a área de acerto).

<input></input>

O elemento HTML <input> é usado para criar controles interativos para formulários baseados na web para receber dados do usuário. A semântica de um <input> varia consideravelmente dependendo do valor de seu atributo type.

Atributos

Este elemento inclui os atributos globais.

type

O tipo de controle a ser exibido. O tipo padrão é **text**, se este atributo não for especificado. Os valores possíveis são:

- → **button**: Um botão sem comportamento padrão.
- → **checkbox**: Uma caixa de marcação. Você deve usar o atributo **value** para definir o valor enviado por este item. Use o atributo **checked** para indicar se o item está selecionado por padrão. Você também pode usar o atributo **indeterminate** para indicar que a caixa de marcação está em um estado indeterminado (na maioria das plataformas, isso desenha uma linha horizontal cortando a caixa).
- → **color**: HTML5 Um controle para especificar cores. A interface de um seletor de cores não tem nenhuma funcionalidade obrigatória a não ser aceitar cores simples em texto.
- → **date**: HTML5 Um controle para inserir uma data (ano, mês e dia, sem horário).
- → **datetime**: HTML5 Um controle para inserir data e horário (hora, minuto, segundo e fração de segundo) baseado no fuso horário UTC.
- → **datetime-local**: HTML5 Um controle para inserir data e horário, sem fuso horário.
- → **email**: HTML5 Um campo para editar um endereço de e-mail. O valor do campo é validado para estar vazio ou ter um único endereço de e-mail válido antes de ser enviado. As pseudoclasses CSS :valid e :invalid são aplicadas apropriadamente.
- → **file**: Um controle que permite ao usuário selecionar um arquivo. Use o atributo**accept** para definir os tipos de arquivo que o controle pode selecionar.
- → **hidden**: Um controle que não é exibido mas cujo valor é enviado ao servidor.
- → **image**: Um botão gráfico para enviar o formulário. Você deve usar o atributo **src** para definir a fonte da imagem e o atributo **alt** para definir um texto alternativo. Você pode usar os atributos **height** e **width** para definir o tamanho da imagem em pixels.
- → **month**: HTML5 Um controle para inserir mês e ano, sem fuso horário.
- → **number**: HTML5 Um controle para inserir um número de ponto flutuante.
- → **password**: Um campo de texto com uma só linha cujo valor é obscurecido. Use o atributo **maxlength** para especificar o comprimento máximo do valor que pode ser inserido.
- → **radio**: Um botão de escolha. Você deve usar o atributo **value** para definir o valor a ser enviado por este item. Use o atributo **checked** para indicar se este item deve estar selecionado por padrão. Botões de escolha que têm o mesmo valor para o atributo **name** estão no mesmo "grupo de botões de escolha"; apenas um botão de escolha no grupo pode estar selecionado de cada vez..
- → **range**: HTML5 Um controle para inserir um número cujo valor exato não é importante. Este tipo de controle usa os seguintes valores padrão se os atributos correspondentes não forem especificados:
  - min: 0
  - max: 100
  - value: min + (max-min)/2, ou min se max for menos que min
  - step: 1
- → **reset**: Um botão que faz o conteúdo do formulário voltar a ter seus valores padrão.
- → **search**: HTML5 Um campo de texto com uma só linha para digitar termos de busca; quebras de linha são automaticamente removidas do valor entrado.
- → **submit**: Um botão que envia o formulário.
- → **tel**: HTML5 Um controle para inserir um número de telefone; quebras de linha são automaticamente removidas do valor entrado, mas nenhuma outra sintaxe é imposta. Você pode usar atributos como **pattern** e **maxlength** para restringir os valores inseridos no controle. As pseudoclasses CSS :valid e :invalid são aplicadas apropriadamente.
- → **text**: Um campo de texto com uma só linha; quebras de linha são automaticamente removidas do valor

entrado.

→ **time**: HTML5 Um controle para inserir um horário sem fuso horário.

→ **url**: HTML5 Um campo para editar uma URL. O valor inserido é validado para ser vazio ou uma URL absoluta válida antes de ser enviado. Quebras de linha e espaços em branco antes e após o valor inserido são automaticamente removidos. Você pode usar atributos como **pattern** e **maxlength** para restringir os valores inseridos no controle. As pseudoclasses CSS :valid e :invalid são aplicadas apropriadamente.

→ **week**: HTML5 Um controle para inserir uma data consistindo de ano da semana e número da semana sem fuso horário.

→ **accept**: Se o valor do atributo **type** for file, este atributo indica quais tipos de arquivo o servidor aceita; caso contrário, este atributo é ignorado. O valor deve ser uma lista de especificadores de tipo únicos separados por vírgula.

1.14 - Box-Sizing

domingo, 11 de dezembro de 2022 10:46

Entenda

box-sizing

A propriedade CSS box-sizing é utilizada para alterar a propriedade padrão da box model, usada para calcular larguras (widths) e alturas (heights) dos elementos. É possível usar essa propriedade para emular o comportamento dos navegadores (browsers) que não suportam corretamente a especificação da propriedade CSS box model.

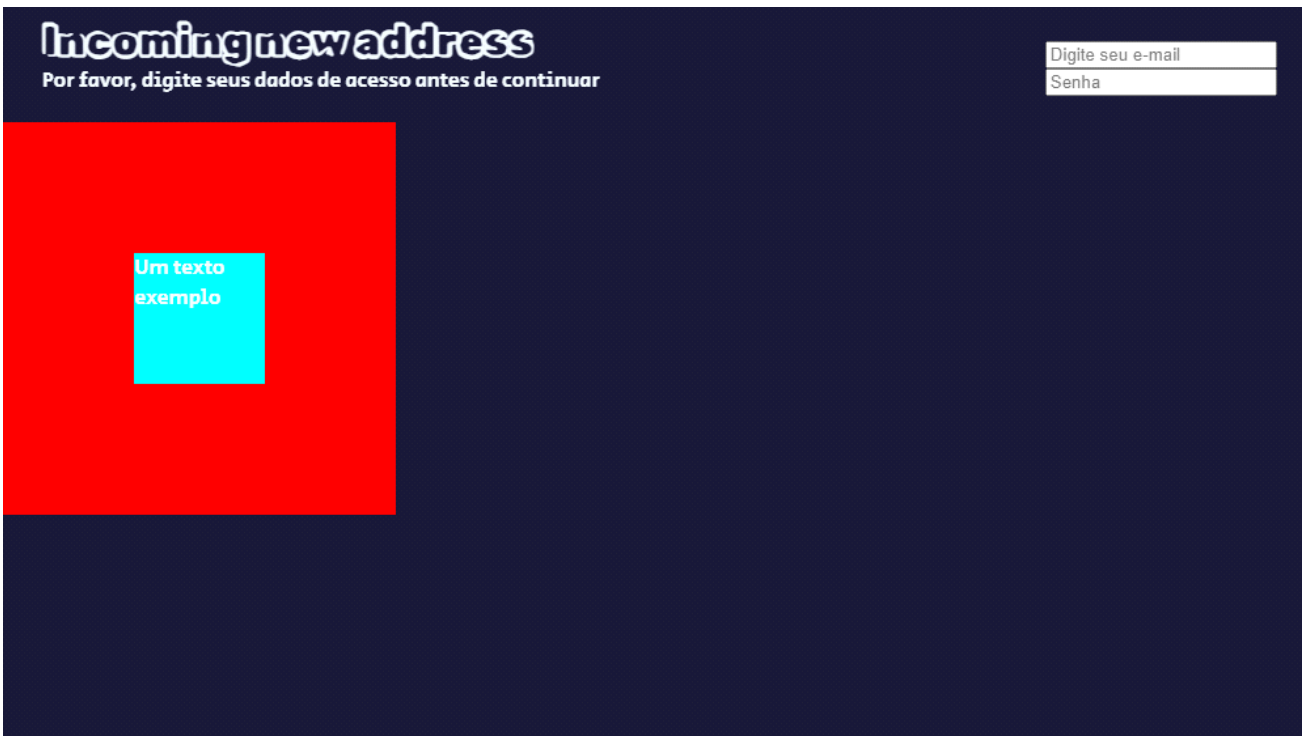
```
Sintaxe formal:

  box-sizing =
    content-box
    border-box
```

border-box

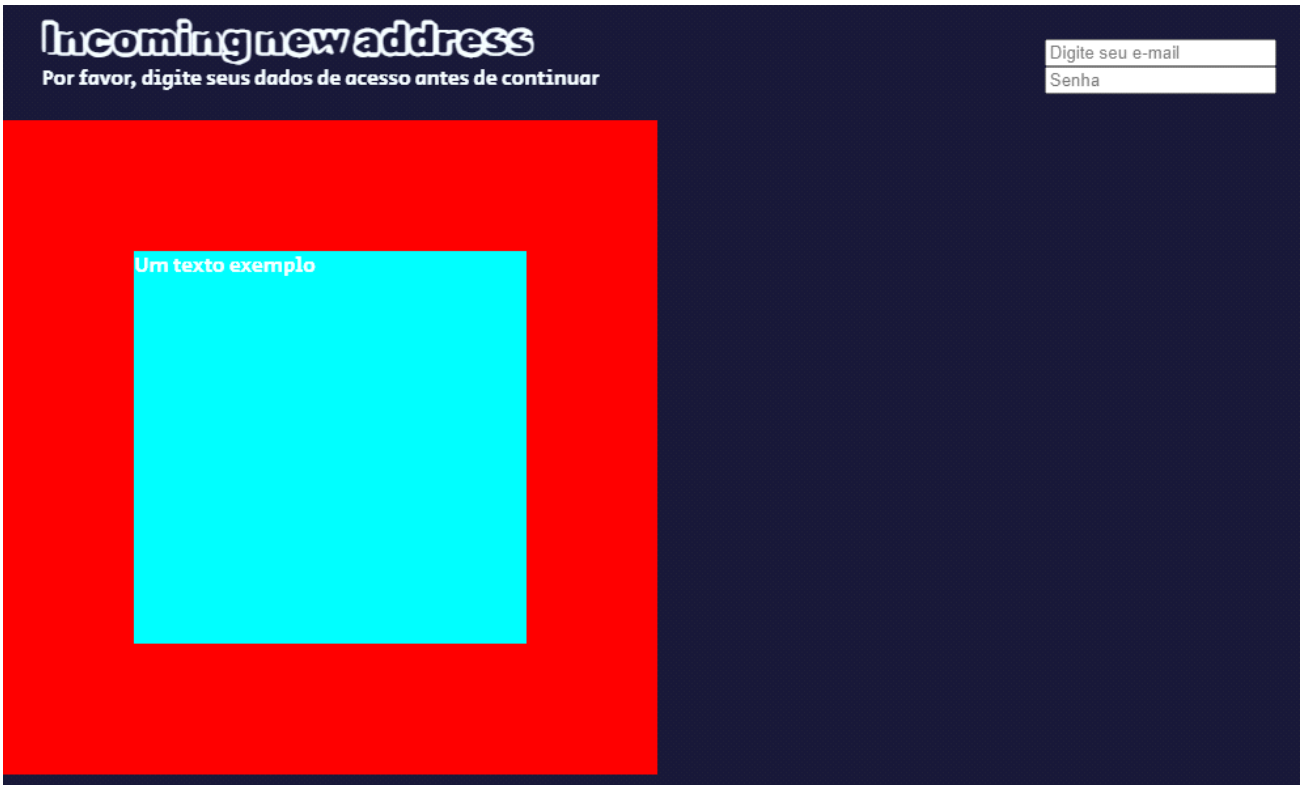
As propriedades de largura (width) e de altura (height) **incluem o tamanho padding size e a propriedade border**, mas não incluem a propriedade margin.

Veja exemplo abaixo, repare no TAMANHO da que está com a configuração box-sizing: border-box
E depois veja sem o border-box

Com box-sizing: border-box



Sem border box:



Essa diferença é dada ao fato de que as propriedades de largura (width) e de altura (height) passaram a **incluir o tamanho padding size e a propriedade border**.

1.15 - [IMPORTANTE] Resetando o CSS

domingo, 11 de dezembro de 2022 12:22

Resetar o CSS é uma parte do processo que irá facilitar todo o posterior processo de desenvolvimento.

Temos aqui, duas configurações que resetam o CSS
Na configuração global (*) usar somente box-sizing: border-box, enquanto na
Tag body e html podemos usar muito bem o margin: 0

Outros exemplos:

```
* {box-sizing: border-box;
    }

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video{
   margin: 0;
   padding: 0;
   border: 0;
   font-size: 100%;
   font: inherit;
   vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section{
   display: block;
}
body{
   line-height: 1;
}
ol, ul{
   list-style: none;
}
blockquote, q{
   quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after{
   content: '';
   content: none;
}
table{
   border-collapse: collapse;
```

```
    border-spacing: 0;
}
```

Veja que ele divide as tags em grupos que devem receber valores genéricos, como **margin**: 0 , **border**: 0 , **padding**: 0 e outras que precisam de valores mais específicos, como

```
ol, ul{
    list-style: none;
}
```

em que ele retira o estilo padrão da tag que utiliza círculos nos destaque dos itens de lista não ordenada. Há formas simplificadas também que, particularmente, utilizo mais em meus projetos, como o uso de **seletores universais**:

```
*,
*:after,
*:before{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    text-decoration: none;
}
body{
    font-size: 100%;
    list-style-type: none;
}
```

1.16 - Seletores CSS

domingo, 11 de dezembro de 2022 18:21

Nome do seletor	Exemplo	Descrição do exemplo
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements
element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects the first <p> element that is placed immediately after elements
element1~element2	p ~ ul	Selects every element that is preceded by a <p> element
[attribute]	[target]	Selects all elements with a target attribute
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[attribute|=value]	[lang|=en]	Selects all elements with a lang attribute value equal to "en" or starting with "en-"
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[attribute$=value]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[attribute=value]*	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"
:active	a:active	Selects the active link
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
:checked	input:checke	Selects every checked <input> element

	d	
:default	input:default	Selects the default <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the input element which has focus
:fullscreen	:fullscreen	Selects the element that is in full-screen mode
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects input elements with a value within a specified range
:indeterminate	input:indeterminate	Selects input elements that are in an indeterminate state
:invalid	input:invalid	Selects all input elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
::marker	::marker	Selects the markers of list items
:not(*selector*)	:not(p)	Selects every element that is not a <p> element
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optiona	Selects input elements with no "required"

	l	attribute
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range
::placeholder	input::placeholder	Selects input elements with the "placeholder" attribute specified
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified
:required	input:required	Selects input elements with the "required" attribute specified
:root	:root	Selects the document's root element
::selection	::selection	Selects the portion of an element that is selected by a user
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all input elements with a valid value
:visited	a:visited	Selects all visited links

De <https://www.w3schools.com/cssref/css_selectors.php>

1.17 - Display: Flex e flexbox

domingo, 11 de dezembro de 2022 18:48

Pra começar, no início eu sempre tinha dúvida de onde usar o elemento flex.

Eu saia tacando flex em tudo.

Mas então, uma grande professora me ensinou que, o flex deve ser aplicado sempre no elemento PAI!!!

Por exemplo

Se eu tenho uma div.pai e uma div.filho dentro dela.
Eu devo aplicar no CSS da seguinte maneira

.container { display: flex}

Atente-se ao HTML e ao CSS, e depois o resultado

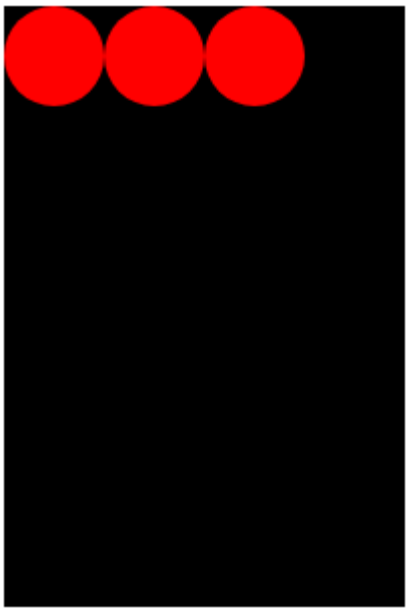
HTML

```html
<div class="pai">
    <div class="filho"></div>
    <div class="filho"></div>
    <div class="filho"></div>
</div>
```

CSS

```css
.pai {
    width: 200px;
    height: 300px;
    background-color: black;
    display: flex;
}

.filho {
    width: 50px;
    height: 50px;
    border-radius: 50%;
    background: red;
}
```

Resultado:



Por padrão, as colunas atendem a um default de row, que é o exemplo acima.

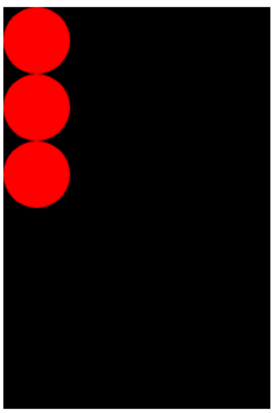
Mas também podemos mudar para column.

E obtermos as divs empilhadas em uma coluna.

Veja o exemplo prático com o CSS.

```
.pai {
    width: 200px;
    height: 300px;
    background-color: black;
    display: flex;
    flex-direction: column;
}
```

Resultado obtido:



```css
.pai {
    width: 200px;
    height: 300px;
    background-color: black;
    border-radius: 20px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

Alinha o conteudo no centro, numa direção vertical

alinha os itens no centro, considerando uma direção horizontal

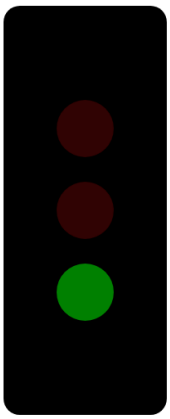
Segue abaixo algumas considerações:

Como funciona:
flex-direction: row;
align-items // alinhamento vertical
justify-content // alinhamento horizontal

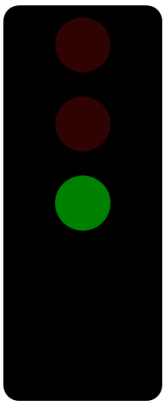
flex-direction: column;
align-items // alinhamento horizontal
justify-content // alinhamento vertical

Abaixo, algumas opções de alinhamento, lembrando que cada bolinha, representa uma div.filho dentro de um container que denominamos como div.pai

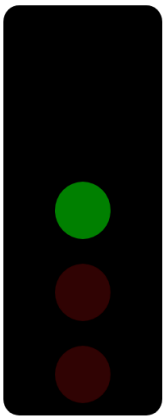
Center



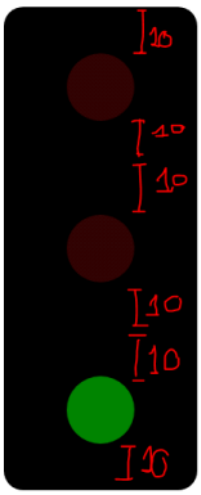
Flex-start



Flex-end

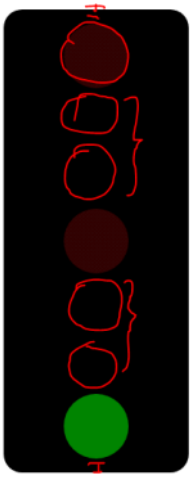


Space-around (utilizei 10px como exemplo)



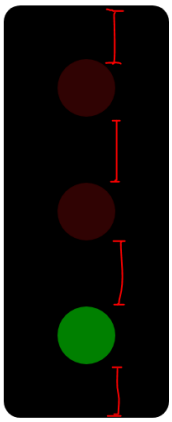
space-between

Espaço entre (somente no meio)
Ele divide a div entre as divs filhas porém aplica só no meio dos elementos.



space-evenly: espaço igual em todos os lados.

Ou seja, será distribuido igualmente para todos os elementos



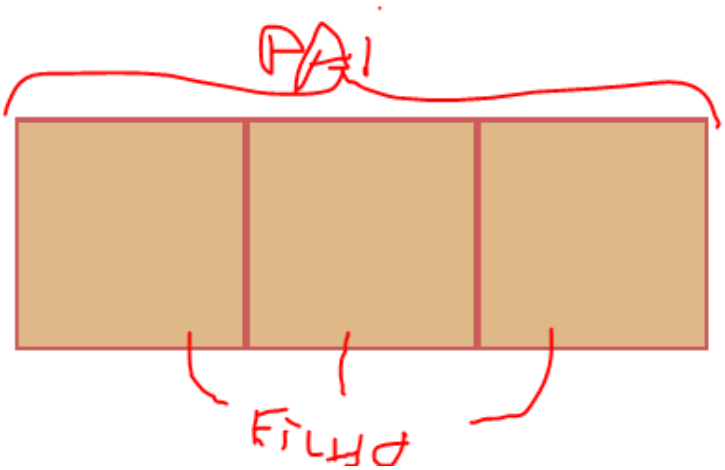
1.18 - Alinhando elementos horizontalmente

terça-feira, 13 de dezembro de 2022 23:54

Analisamos o código abaixo:

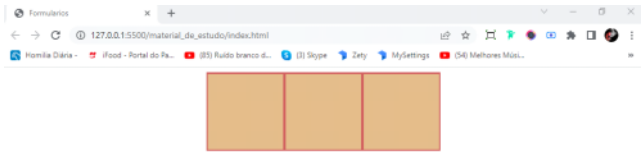
```css
.pai {
    width: fit-content;
    height: 20vh;
    background-color: indianred;
    display: flex;
    align-items: center;
    justify-content: flex-start;
    flex-direction: row;
}
```

Quer dizer que, a largura da div pai, irá se ajustar, com base na largura do div filho, veja exemplo:



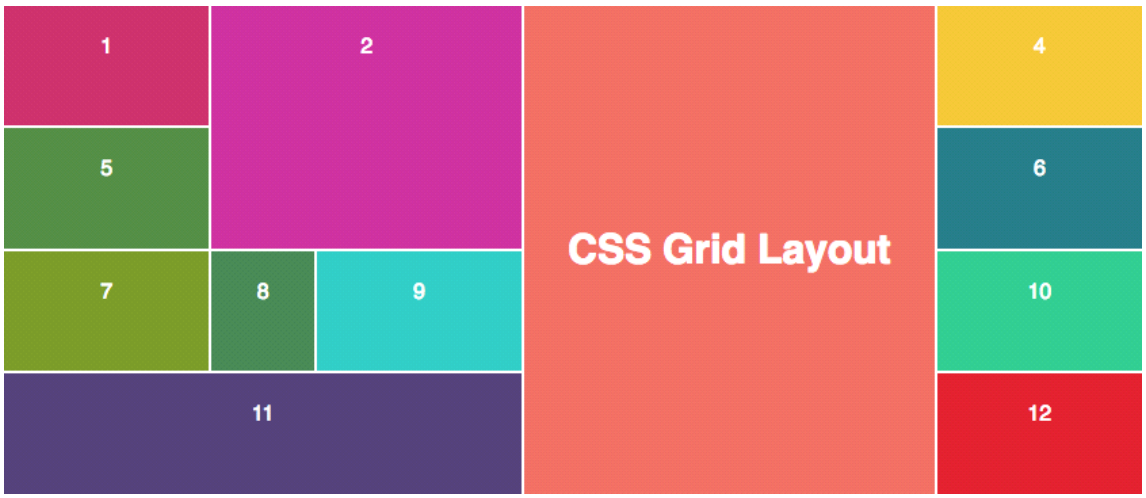
Além de tudo, se a gente colocar uma margin: auto.
E aí a gente consegue alinhar esse conteúdo na horizontal na página como o exemplo abaixo:

```css
.pai {
    width: fit-content;
    height: 20vh;
    background-color: indianred;
    display: flex;
    align-items: center;
    justify-content: flex-start;
    flex-direction: row;
    margin: auto;
}
```



GRID:

Antes de mais nada, um exemplo de grid, que é diferente de flexbox



O Flexbox é UNIDIMENSIONAL.

Já o Grid, é tipo um mosaico, então é multidirecional.

FLEXBOX vs Grid

Primeiro precisamos saber quais situações devemos usar grid ou flexbox, para tirarmos vantagens dos benefícios de cada um.

FLEXBOX

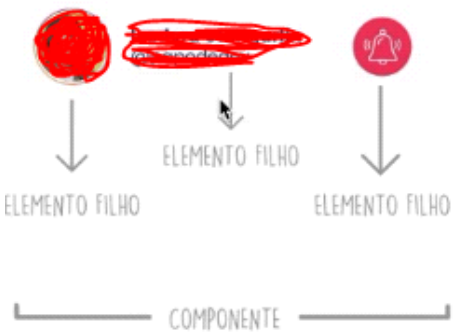
É mais usado quando estamos trabalhando em elementos em apenas 1 dimensão: ou linha ou coluna.

GRID

É ótimo quando precisamos trabalhar com 2 dimensões (linha e coluna)

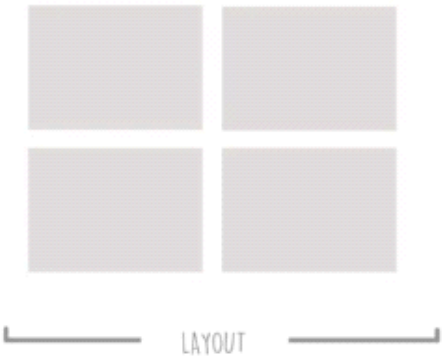
FLEXBOX

A abordagem do flexbox é mais voltada para alinhamento de elementos em componentes



GRID

A abordagem do grid é mais usada pensando no layout da página como um todo



1.19 - Display Grid e Grid Área

quarta-feira, 14 de dezembro de 2022 20:18

A princípio, os parametros mais utilizados para grid na div pai são estes:

```
grid-tempplate-columns
grid-template-rows
```

Define o número total de colunas que serão criadas no grid.
Alguns exemplos:

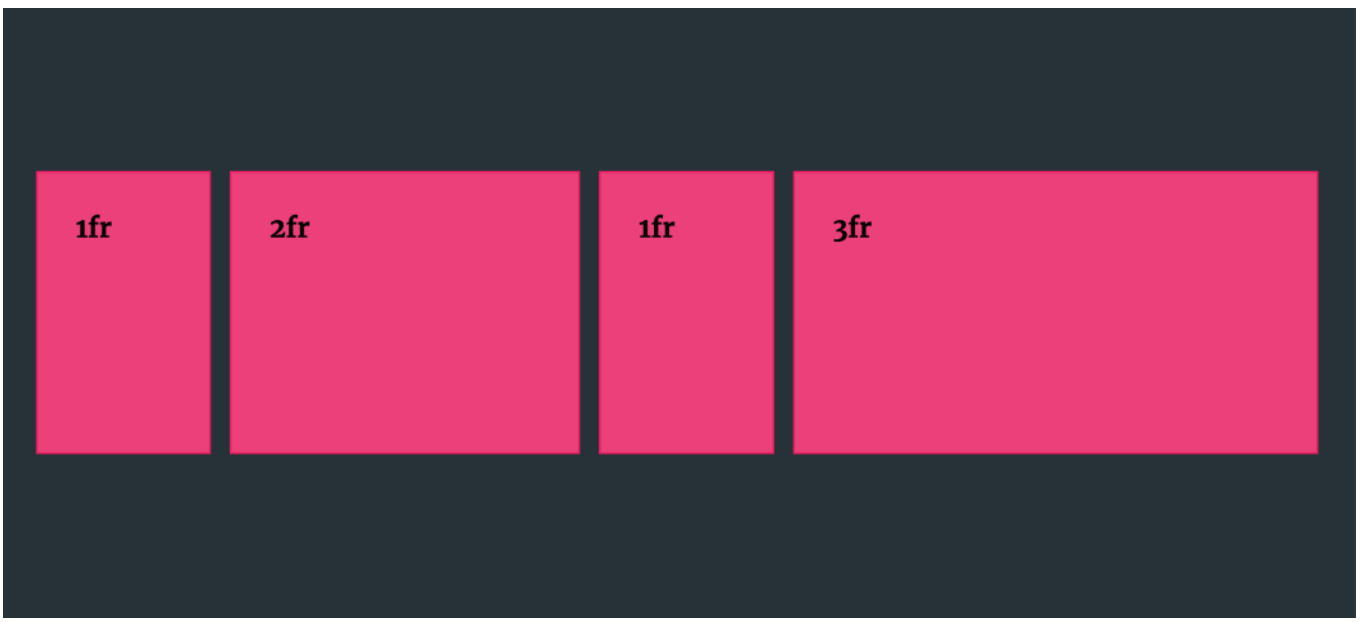
- → grid-template-columns: 100px 100px 100px 100px;// Quatro colunas de 100px de largura são criadas
- → grid-template-columns: 1fr 2fr;// Duas colunas são criadas, sendo a segunda com o dobro do tamanho da primeira. fr é uma unidade fracional. O tamanho do conteúdo é respeitado, ou seja, se o conteúdo na primeira coluna for maior que o da segunda, a primeira será maior.
- → grid-template-columns: minmax(200px, 1fr) 1fr 1fr;// Três colunas são criadas, a primeira terá no mínimo 200px de largura e no máximo 1fr(isso significa que após 200px ela se expande da mesma forma que as outras colunas). As outras duas colunas vão ter 1fr.
- → grid-template-columns: repeat(3, 1fr);// Cria 3 colunas com 1fr de tamanho. O repeat seria a mesma coisa que escrever 1fr 1fr 1fr.
- → grid-template-columns: repeat(auto-fit, minmax(100px, auto));// Cria automaticamente um total de colunas que acomode itens com no mínimo 100px de largura.

Veja o exemplo abaixo

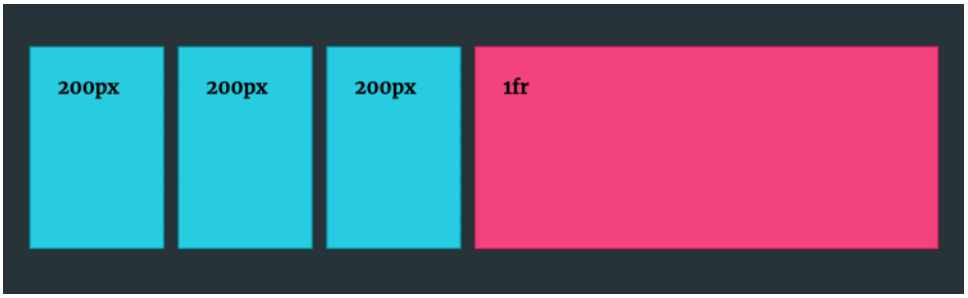
CSS:

```
.grid{
display:grid;
grid-template-columns:repeat(4,1fr);
column-gap:20px;
}
```

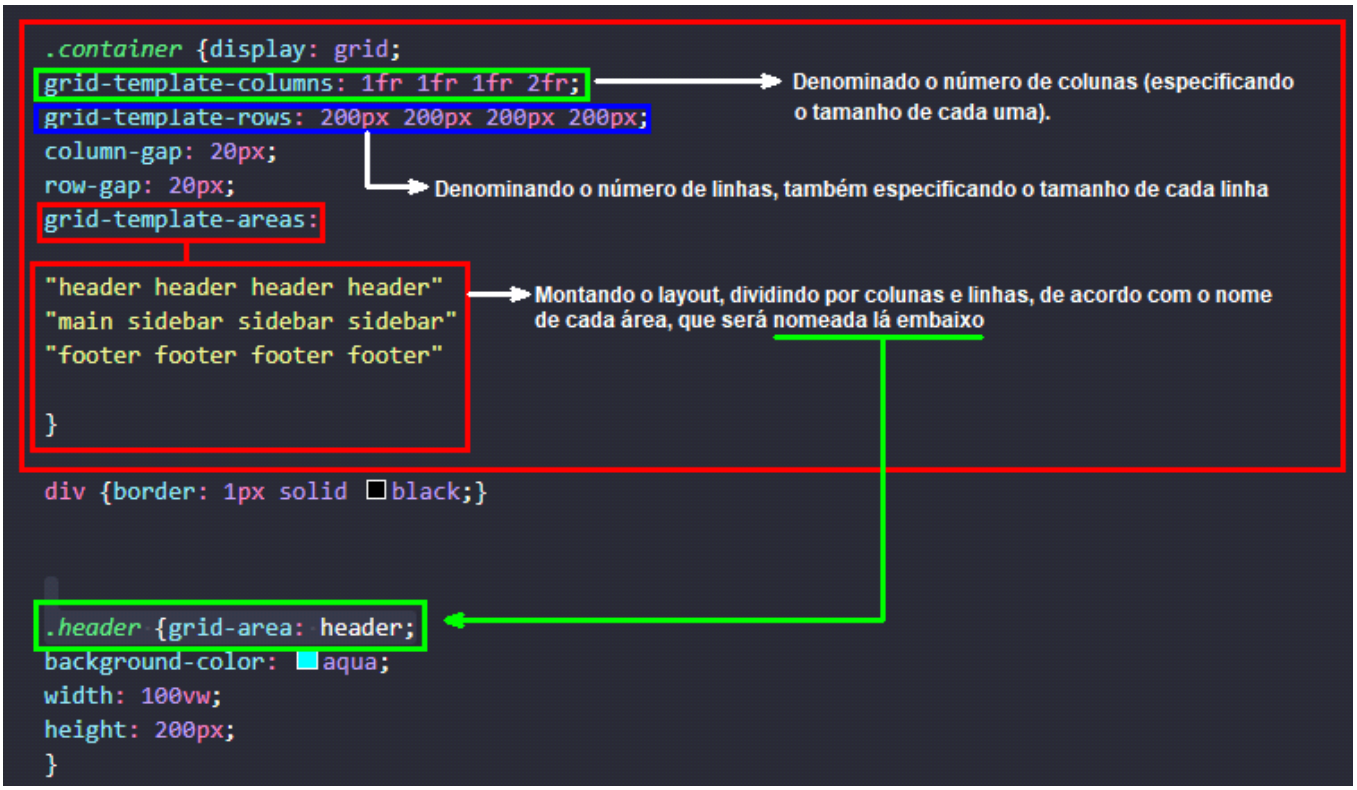
Resultado esperado



```
.grid{
display:grid;
grid-template-columns:repeat(3,200px)1fr;
column-gap:20px;
}
```



Seliga só:



Veja só no exemplo prático:

Imagine que a tela foi dividida, e cada quadrado é preenchido pelas expressões abaixo.

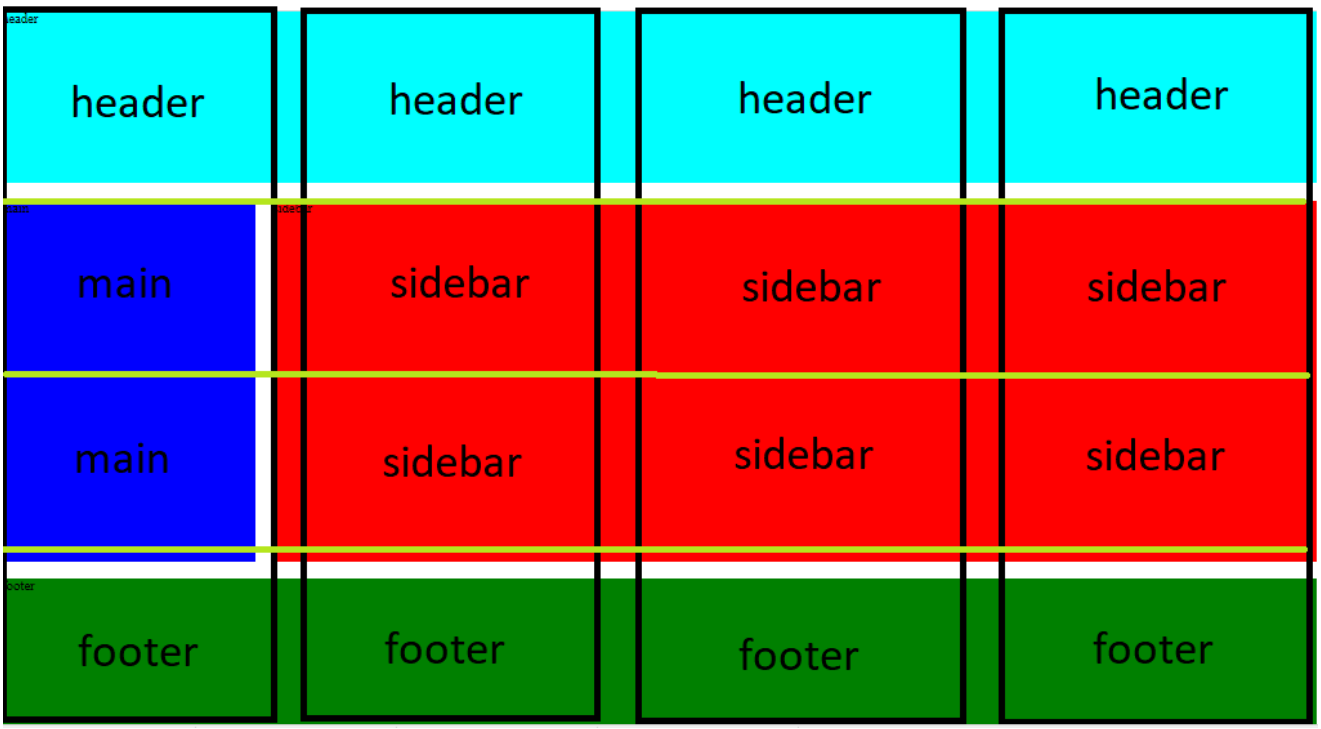
```
.container {display: grid;
grid-template-columns: 1fr 1fr 1fr 2fr; ---> especifico colunas
grid-template-rows: 200px 200px 200px 200px; ---> especifico linhas
column-gap: 20px;---> tamanho do vão de cada coluna
row-gap: 20px;---> tamanho do vão de cada linha
grid-template-areas: ---> espaço para estilizar
"header header header header"
"main sidebar sidebar sidebar"
"main sidebar sidebar sidebar"
"footer footer footer footer"
}
```

```
.header {grid-area: header;---> dentro de cada div filho, preciso especificar
o grid-area
background-color: aqua;
width: 100vw;
height: 200px;
}
.main {grid-area: main;---> dentro de cada div filho, preciso especificar o
grid-area
    background-color: blue;
width: 100%;
height: 100%;}
.sidebar {
    grid-area:sidebar;---> dentro de cada div filho, preciso especificar o
grid-area
     background-color: red;
width: 100%;
height: 100%;}
.footer {

    grid-area: footer;---> dentro de cada div filho, preciso especificar o
grid-area
    background-color: green;}
```

Vemos que header aparece 4x ("header header header header") então ele terá uma área de 4 quadrados e 1 linha.

"header header header header"
"main sidebar sidebar sidebar"
"main sidebar sidebar sidebar"
"footer footer footer footer"

Segue abaixo em representação prática, de exatamente o que está digitado ali em cima



2 - Javascript

quinta-feira, 15 de dezembro de 2022 21:34

O que é e para que serve o JavaScript?

JavaScript é uma das linguagens de programação que foram desenvolvidas para facilitar a criação de páginas web que precisam conter itens mais complexos, focados na interface visual.

Então, é uma maneira de programar e desenvolver websites que permite ao desenvolvedor uma dinâmica maior nos elementos apresentados e pode ajudar muito a melhorar a experiência do usuário dentro de um site.

Pode ser aplicado em todo tipo de construção de páginas, tanto para apresentar conteúdo estático como textos e imagens, quanto para elementos dinâmicos, como slides, vídeos, funcionamento de botões, menus e etc…

Estudando teorias

Neste repositório deixarei por escrito algumas questões teóricas que são chatas, porém necessárias para a evolução profissional

JS

- Como funciona o JS
- O que é o JS
- Hoisting
- Scope
- Nested Scopes
- Variables
- ES6 Features
- Pure Functions
- Closure
- Currying
- Higher-Order Functions

Como funciona o JS

O google chrome usa a engine v8 (open source escrita em c++) . A engine v8 serve para interpretar um código javascript. A v8 foi projetada para aumentar a perfomance de execução do JS dentro de navegadores, ele compila código JS em código de maquina ao invés de usar um interpretador. Ele compila de js para código de máquina em tempo de execução, implementando um compilador JIT (just in time).

JS => c++ => Assembly => Machine Code

O que é o JS

Javascript é como nós chamamos a linguaguem (mas isso é o trademark da Oracle), o nome oficial da linguagem é ECMAScript (ES) é a abreviação.

ES6 === ECMAScript 6 === ES2015, é simplesmente a versão *mais nova da linguagem (entre aspas)*.

O que aconteceu é que o ES5, não é de 2014.. mas sim de 1999. Isso mesmo, ficamos 16 anos sem updates. Basicamente porque não era tão bom e existiam algumas alternativas, a mais famosa delas: o Flash. Em 2008 o google Chrome foi lançado, e aclamado por sua perfomance na execução de javascript, o que iniciou uma guerra pelo desenvolvimento de interpretadores cada vez mais rápidos, o que deixou o chrome para trás. Até que foi lançado em 2013, a nova versão do chrome usando a V8, tornando-o novamente o mais rápido na execução de JS. Dois anos após.. tivemos o ES6.

Suporte ES:

- O **ECMAScript 3** é totalmente **suportado** em **todos os navegadores**.
- O **ECMAScript 5 (2009)** é totalmente **suportado** em **todos os navegadores modernos** . **Observação**: O **Internet Explorer 9 não suporta ECMAScript 5** "use strict".
- O **ECMAScript 6 (ECMAScript 2015)** é totalmente **suportado** em **todos os navegadores modernos, menos no Internet Explorer.**
- O **ECMAScript 7 (ECMAScript 2016)** é suportado **apenas no Chrome e Opera.**

NOTA: Observe sempre as **versões do browser** e quais **métodos** são **compatíveis** no **mesmo**.

Se nem todos os browsers suportam o ES6 como fazemos? por isso usamos transpilers como o BABEL. Ele transforma seu código ES6 em ES5 (que a maioria dos browsers dá suporte)

fontes:

- https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f
- https://pt.wikipedia.org/wiki/Interpretador_de_JavaScript
- https://pt.stackoverflow.com/questions/383174/o-ecmascript-6--%C3%A9-suportado-pelos-browsers-atuais

Hoisting

Hoisting é o içamento de funções e variáveis para o topo do código, isso declara as variáveis e funções em memória e permite que você use uma função/variável antes mesmo de declara-la.

```
sayHello();// a função foi chamada antes de ser declaradafunctionSayHello(){console.log("Say Hello");}
```

O mesmo é acontece para variáveis, as quais podem ser inicializadas antes de serem declaradas. Porém, o js eleva somente a declaração não a inicialização.

```
console.log(num);// undefined -> nesse caso num é undefined porque só foi declaradanum= 6;console.log(num);// 6 -> agora num já foi inicializadavarnum;
```

Scope

Escopo é a acessibilidade de objetos, variáveis e funções em diferentes partes do código.

- Escopo Global
    - Uma variável global é definida quando declaramos uma variável fora de qualquer função, assim ela torna **acessível a qualquer parte** da nossa aplicação ou site, podendo ser lida e alterada.
- Escopo Local
    - Uma variável se torna local quando ela é declarada dentro de uma função, de tal maneira a qual ela somente estará **acessível dentro dessa função**.
- Escopo de bloco
    - Não existia no JS escopo de bloco. Ou seja, for whiles e ifs não tinham escopo próprio. Porém com o ECMAScript 6 foi possível criar escopos de bloco usando as variáveis **let** e **const**, que são **acessíveis somente dentro do bloco.**

Nested Scopes

Todo o escopo é fechado para acessos externos, de forma que escopos superiores não conseguem acessar escopos internos, mas o contrário é permitido.

functionfoo(){functionbar(){}}
Quando criamos outra função dentro da função foo, estamos colocando outra caixa dentro do escopo da função, criando o que é chamado de “nested scopes”, ou escopos aninhados.

Variables (var, let e const)

- Var
    - é içada
    - tem escopo abrangente → se for declarada dentro de um bloco → vaza do escopo
    - escopo global e função → n tem escopo de bloco
    - Praticamente não são mais usadas em aplicações modernas devidos aos problemas de escopo → **substituídas por const e lets**

functionfoo(a){varname='Lucas'functionbar(){varage=23console.log(name)// Lucaconsole.log(age)// 23}bar()// Lucas - 23console.log(name)// Lucasconsole.log(age)// age is not defined}
if(true){varglobal=2;// vaza de dentro do bloco}functionteste(){varglobal=4;console.log(global);//4}
console.log(global);//2 -> acessa a que vazou do if

- Let e Const
    - Tem escopo de **bloco** e de função
    - Sofrem hoisting (são elevadas) para o topo do bloco que foram definidas → porém não é atribuido o valor de undefined como acontece com var → continuam não inicializadas e dão erro caso sejam chamadas antes de suas declarações.
    - A grande diferença entre as duas é que consts não podem ser reatribuídas enquanto lets sim.

functionname(){console.log(name);// ✕ retorna erro porque ainda não foi inicializadaletname='isadora';console.log(name);// 👍 isadoraname='isadora 2';// 👍 pode ser reatruída}
constnum=6;num=8;// ✕ Não pode ser reatribuída porque é const

ES6 Features

- Declaração de variáveis
- Default Parameters
- Rest parameters
- Programação funcional => arrow functions
- Destructing
- Classes (constructor, get/setters, herança (extends))
- Es6 Modules (import, export)

fonte: https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f

Pure Functions

- Dada a mesma entrada, vai sempre retornar a mesma saída
- Não produz nenhum efeito colateral

functionspureFunction(a,b){returna+b;}pureFunction(1,2)// retorna 3

- Retorna sempre o mesmo valor baseado na entrada e não manipula nenhuma variável de fora.
- Porque usar?
  - Mais fáceis de implementar e testar
  - Código mais limpo, prático e de simples manutenção

Currying

Currying é o nome dado à técnica de dividimos uma função que recebe vários argumentos numa série de funções cada uma lidando com **um** argumento da função inicial.

- Forma de injetar os parametros de forma parcial.
- É uma função normal
- Faz parte de programação funcional
- Transforma uma função com múltiplos argumentos em uma sequencia de nested functions (função aninhada). Ela retorna uma função que espera os próximos argumentos.

functionsomap(a){returnfunction(b){returna+b}}// Uso:varsomadois=somap(2);somadois(3);// 5

também podemos executa-lá de uma vez fazendo: somap(2)(3) // retorna 5

também pode ser escrita usando arrow functions:

constmyFunction=valor1=>valor2=>valor3=>{returnvalor1+valor2+valor3;}
console.log(myFunction(1))// retorna a função esperando o segundo parametroconsole.log(myFunction(2)(4)(3))// retorna 8

Higher-Order Functions

- É uma função que **recebe como parâmetro** outra função e/ou que **retorna** uma função
- Ex: map, reduce, filter..

// calculate recebe como parametro uma função// calculate retorna uma função// Higher-Order functionvarcalculate=function(fn,x,y){returnfn(x,y);};varsum=function(x,y){returnx+y;};varmult=function(x,y){returnx*y;};calculate(sum,2,5);// 7calculate(mult,2,5);// 10

usando ES6:

constcalculate=(fn,x,y)=>fn(x,y);constsum=(x,y)=>x+y;constmult=(x,y)=>x*y;calculate(sum,2,5);// 7calculate(mult,2,5);// 10

Closure

- Closure é a forma de fazer com que as variáveis dentro de uma função

sejam **privadas** e **persistentes**.

- Se refere à forma como funções definidas dentro de um "contexto léxico" (i.e. o corpo de uma função, um bloco, um arquivo fonte) acessam variáveis definidas nesse contexto.

```
functionpai(){varx=1;functionfilho(){console.log(x);x++;}returnfilho;}varcontador=pai();contador();// 1
contador();// 2  contador();// 3
```

A função filho possui uma referência ao escopo da função pai, e a essa referência nós damos o nome de closure.

- Módulos são estruturas de código que fazem bom uso das *closures*, vamos a um exemplo que apresenta bem seu funcionamento:

```
 functionModuloMatematico(){varx=0;functionsomaUm(){x++;console.log(x);}functionsubtraiUm()
{x--;console.log(x);}return{somaUm: somaUm,subtraiUm: subtraiUm};}
varteste=ModuloMatematico();teste.somaUm();// 1   teste.somaUm();// 2  teste.somaUm();// 3
teste.subtraiUm();// 2
```

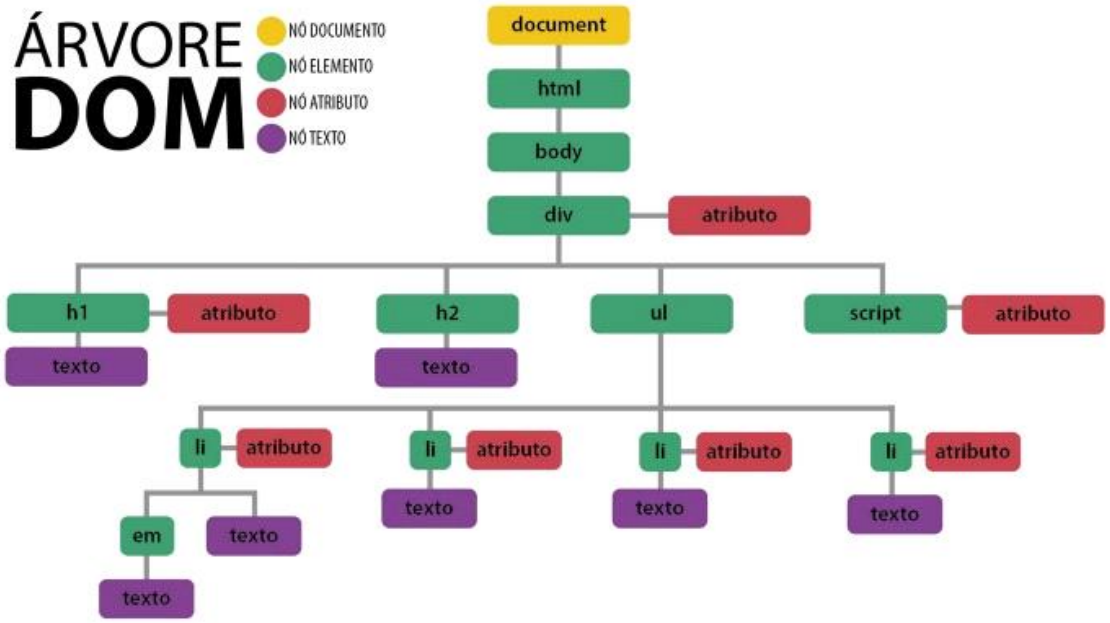
Nesse exemplo, não seria possível acessar diretamente X, porém usando closures é possível fazer isso.

De <https://github.com/isadorastan/estudos#nested-scopes>

2.1 - O que é DOM

quinta-feira, 15 de dezembro de 2022 22:28

Vamos começar mostrando um exemplo de árvore **DOM (Document Object Model)**.



Vamos ver quais são os DOM do nosso documento criando no VS Code

Dê uma olhada lá na linha do body

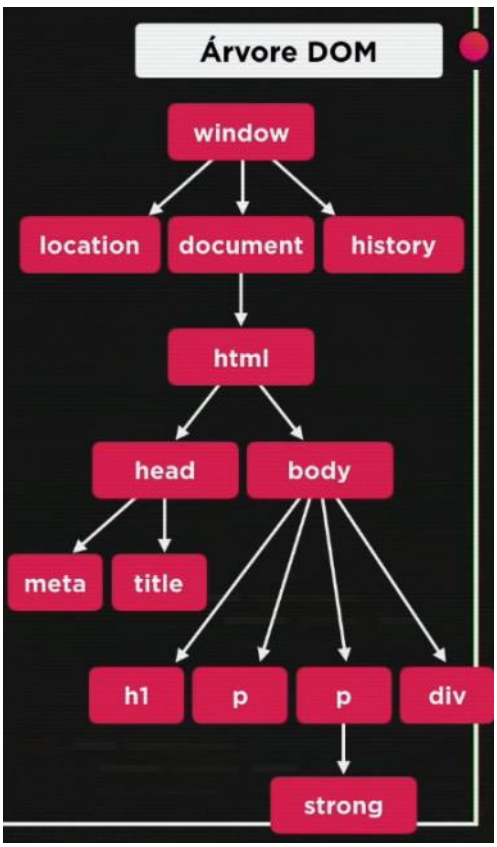
Dentro do <body> temos:
<h1>
<p>
<p>

Ou seja, como DOM, esses objetos (<h1> <p>) são childs de <body> e o <body> é o Parent (pai) deles.

Assim como a tag que está dentro de <p> é uma filha da tag <p>

Fique com as imagens abaixo para ter uma base.

```html
<!DOCTYPE html>
<html lang="pt-br">
<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>Exercicio</title>
</head>

<style>

body {background-color: rgb(69, 111, 226);
color: aliceblue;
font: normal 18pt arial;}

</style>


<body>
    <h1>Iniciando estudos Dom</h1>

    <p>Aqui vai o resultado</p>
    <p>Aprendendo a usar <strong>DOM</strong> em Javascript</p>
    <div>Clique em mim</div>

    <script></script>


</body>
</html>
```



Existem várias maneiras de navegar entre os elementos, lembrando que os elementos é qualquer coisa que aparecer na arvore DOM.

Veja que é possível selecionar os elementos (5 métodos)
Por tag, por id, nome, classe e o seletor.

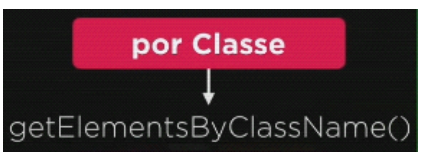
Veja só um exemplo:

Lembrando que utilizamos o window.document.getElementsByTagName('nome da tag')[número da tag selecionado]

```
<body>
    <h1>Iniciando estudos Dom</h1>

    <p>Aqui vai o resultado</p>
    <p>Aprendendo a usar <strong>DOM</strong> em Javascript</p>
    <div>Clique em mim</div>

    <script>


    var p1 = window.document.getElementsByTagName('p')[0] // declarei a variável p1 com dom window. document. trazer documento pela tag ('nome
    da tag') [número da tag que quero selecionar] (lembrando que começa pelo 0)
    window.document.write(`note que eu trouxe dentro do marcador p1 o valor dele como ${p1.innerText}`) // coloquei para escrever no documento
    (document.write) (`texto` e o marcador com a variável e o inner text, que é o texto escrito dentro da variável)


    </script>
```

Esse primeiro <p> é o [0]

São eles:



```
<body>
    <h1>Iniciando estudos Dom</h1>

    <p>Aqui vai o resultado</p>
    <p>Aprendendo a usar <strong>DOM</strong> em Javascript</p>
    <div class="divzeira">Clique em mim</div>

    <script>


  var divzeira = window.document.getElementsByClassName('divzeira')[0]
  window.document.write(`Aqui, estou te apresentando como divzeira, exemplo: ${divzeira.innerHTML}.`)


    </script>
```

Lembrando que o query selector é um método mais recente do ecmascript.

```
<body>
    <h1>Iniciando estudos Dom</h1>

    <p>Aqui vai o resultado</p>
    <p>Aprendendo a usar <strong>DOM</strong> em Javascript</p>
    <div class="divzeira">Se esta aparecendo este texto, deu certo o que você queria fazer kkk</div>

    <script>
                                        Para trazer a tag estilizada, podemos colocar para buscar via querySelector pois funciona que uma beleza também

  var divzeira = window.document.querySelector('div.divzeira')      só preste a atenção, pois como estamos trazendo uma classe, utilizamos o . após a div sendo 'iv.nome da classe'
  window.document.write(`Aqui, estou te apresentando como divzeira, exemplo: ${divzeira.innerHTML}.`)


    </script>
```

2.1.1 - Eventos DOM

quinta-feira, 5 de janeiro de 2023 23:09

Já vou deixar uma listinha marota com 6 eventos de mouse sob uma só pra se ter uma idéia.

Mas para que tenha uma noção da grandeza, irei deixar o link de referências de DOM, da Mozilla, para que todos tenham acesso.

https://developer.mozilla.org/en-US/docs/Web/Events

Ou Scaneie o QR Code abaixo para acessar o link, caso tenha impresso essas anotações.

Agora vamos voltar ao assunto principal.

É inviável listar neste documento, todos os eventos possíveis de DOM.
Até porque, nem todos os elementos podem conseguir todos os DOMs.

Então, antes de seguirmos com a aplicação de um evento DOM,

Precisamos entender o que é uma **FUNÇÃO**.

Resumidamente, é um conjunto de linhas, que serão executadas somente quando o evento ocorrer.

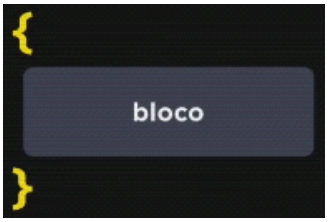
Então, entenda o seguinte:

Uma função em java é um conjunto de códigos (linhas), que vão ser executadas só quando o evento ocorrer.

Exemplo: Irei programar 10 linhas (que vamos chamar de bloco)

Essas 10 linhas não serão executadas automaticamente.
Agora, esse bloco só irá carregar quando (por exemplo: clicar dentro da div)

Para que eu execute essas 10 linhas, o primeiro passo é coloca-las dentro de um bloco.
Um bloco em javascript é delimitado entre os sinais de chaves {} coloca o código dentro da chave.

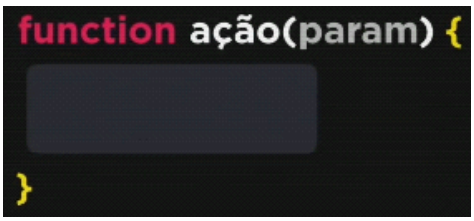


Esse bloco, precisa ser nomeado, com uma function (que quer dizer função antes do bloco)
No JS, existe função anônima, que é uma função que não tem nome.

No entanto, para que o método funcione, precisaremos nomear a função.
Geralmente os nomes das funções são ações que podemos fazer.

Geralmente funções de evento.

São nomes de ação. Iremos colocar o nome da ação que vai acontecer e abre e fecha o parente.
Dentro dos parentes vão ser inseridos alguns parâmetros. (veremos parâmetros mais pra frente).



De uma olhadinha abaixo:

Ou para facilitar, eu posso simplesmente manter método abaixo, do
nome_da_variável.addEventListener ('nome_da_ação', nome_escolhido_para_a_função)

```
<script>
var a = window.document.getElementById('area')
a.addEventListener ('onclick', clicar)
a.addEventListener ('onmouseenter', mouseenter)
a.addEventListener ('onmouseout', saindo)

// clicar ----------------------------------------------------
function clicar()
{
a.innerText= 'Clicou!'
a.style.background ='blue'
}

// passar o mouse  -------------------------------------------------
function mouseenter()
{
a.innerText= 'Mantenha o mouse dentro desta área'
a.style.background='green'
}

// tirar o mouse ---------------------------------------------------
function saindo()
{
    a.innerText= 'O mouse saiu'
    a.style.background ='red'
}
</script>
```

Por fim, veja o que conseguimos fazer aqui em baixo:

```
<body>

<h1>Somando Valores</h1>
<input type="number" name="txtn1" id="txtn1">
<input type="number" name="txtn2" id="txtn2">
<input type="button" value="somar" onclick="somar()">
<div id="res">Resultado</div>

<script>

function somar() {

var tn1 = window.document.getElementById('txtn1')
var tn2 = window.document.getElementById('txtn2')
var res = window.document.getElementById('res')
var n1 = Number(tn1.value)
var n2 = Number(tn2.value)
var s = n1 + n2
res.innerHTML = `A soma destes dois números é igual a: ${s}`
}


</script>
</body>
```

Exercício:

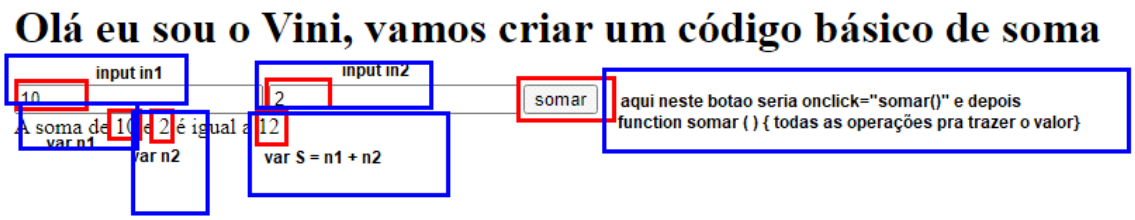
```html
<h1>Olá eu sou o Vini, vamos criar um código básico de soma</h1>

<input type="number" name="in1" id="in1">
<input type="number" name="in2" id="in2">
<input type="button" value="somar" onclick="somar()">
<div id="resultado">Aguardando Resultado da soma</div>

<script>
function somar ()
{
    var in1 = document.getElementById('in1')
    var in2 = document.getElementById('in2')
    var resultado = document.getElementById('resultado')
    var n1 = Number(in1.value)
    var n2 = Number(in2.value)
    var s = n1 + n2
    resultado.innerHTML = `A soma de ${n1} e ${n2} é igual a ${s}`

}
```

No input button eu coloquei a função de somar dentro do onclick

Veja no retângulo azul, os códigos que irão retornar, após eu clicar (onclick) no botão (button) note que estão todos dentro de uma chave { }

como a var in1 recebe um valor em texto precisei converter por Number (nome_var.value)

Resultado do exercício:



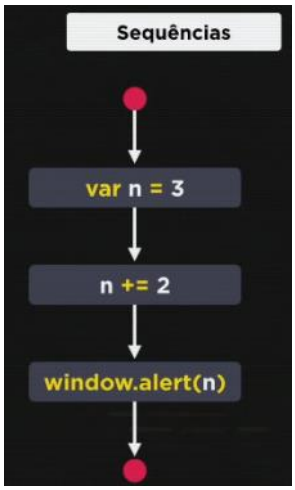
2.2 - Condições em Javascript

sexta-feira, 6 de janeiro de 2023 01:26

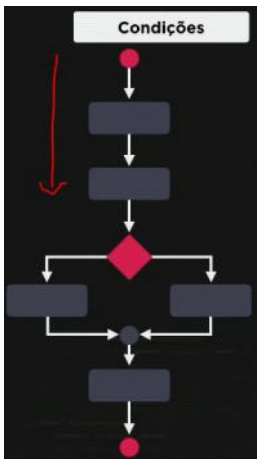
IF e ELSE...

Antes de começarmos a falar de condições, é bom começarmos a colocar na cabeça um pouco de sequências.

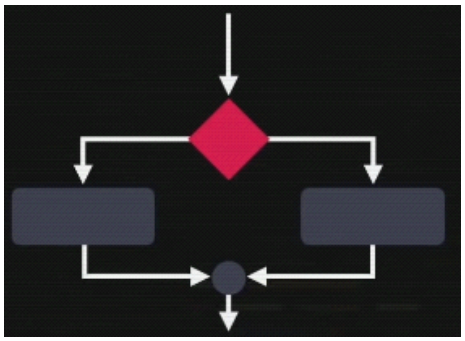
Como por exemplo:



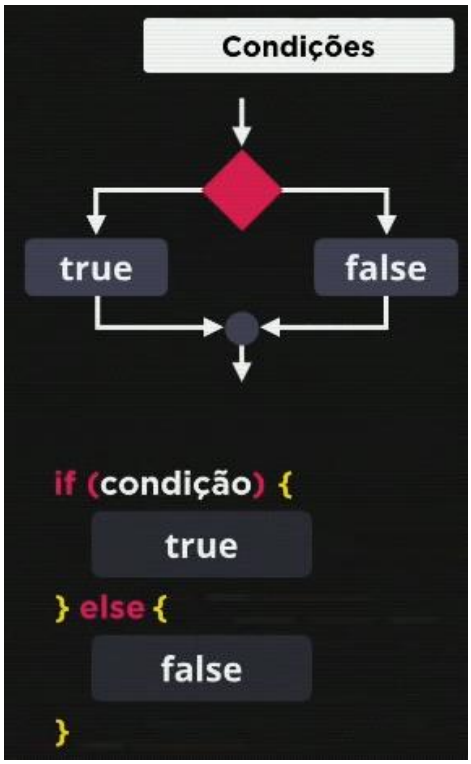
Até o presente momento, utilizamos nossa programação considerando uma sequência linear.



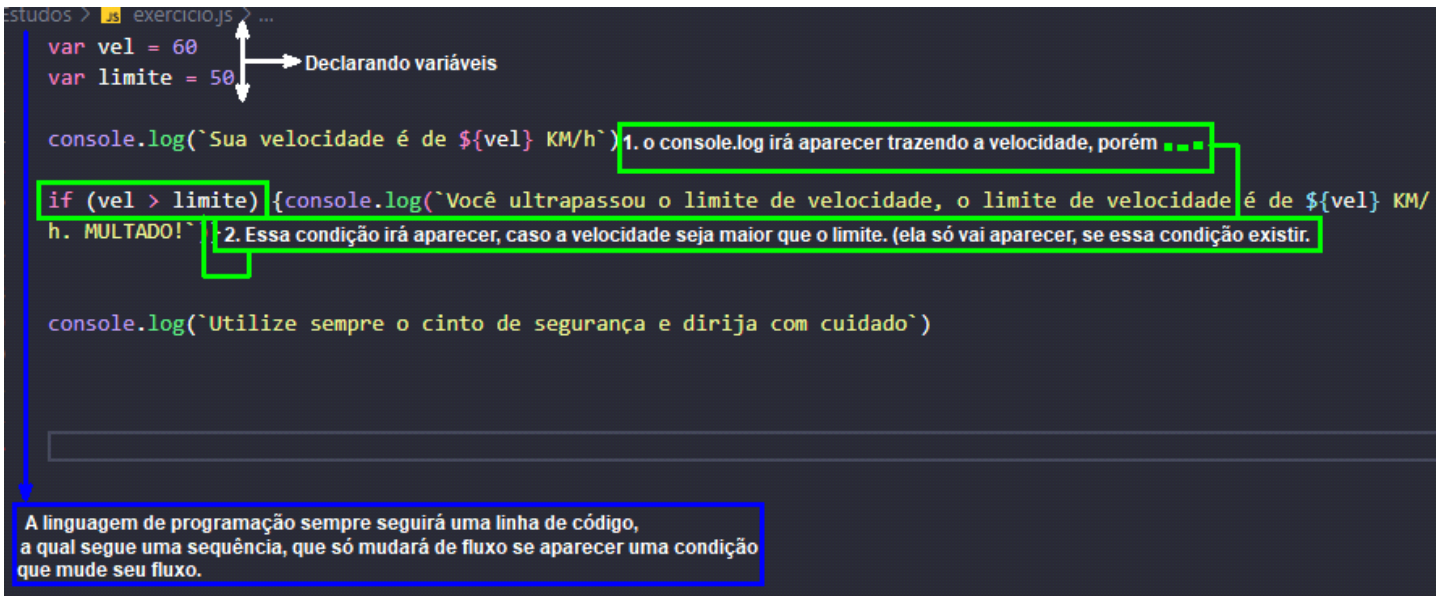
Vamos começar pela que nos interessa



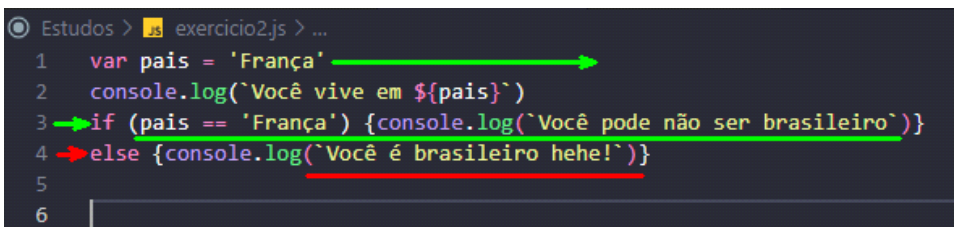
Temos dois meios de criar uma estrutura de decisão,
No caso abaixo, consideramos If e Else.



Vamos colocar a mão na massa.



Fazendo outro exemplo, só que agora com países (cara isso é muito divertido)



```
    <h1>Sistema de multas de SP!</h1>

    <p>Digite a velocidade do seu carro: </p><input type="number" name="txtvel" id="txtvel">
    <input type="button" value="verificar" onclick="calcular()">
    <div id="alerta">Aguardando você inputar sua velocidade!</div>


</body>

<script>

function calcular ()

{

var txtvel = document.getElementById('txtvel')
var vel = Number(txtvel.value)
var alerta = document.getElementById('alerta')
alerta.innerHTML = `<p>Sua velocidade é de ${vel} Km/h</p>`
if (vel > 60) {alerta.innerHTML +=`<p>Seu animal, você está correndo mais do que o permitido</p>
    <p>Você será <strong>Multado</strong> por excesso de velocidade!</p>`}

alerta.innerHTML += `<p>Dirija sempre usando cinto de segurança.</p>`
}
```

Próximo exercício:

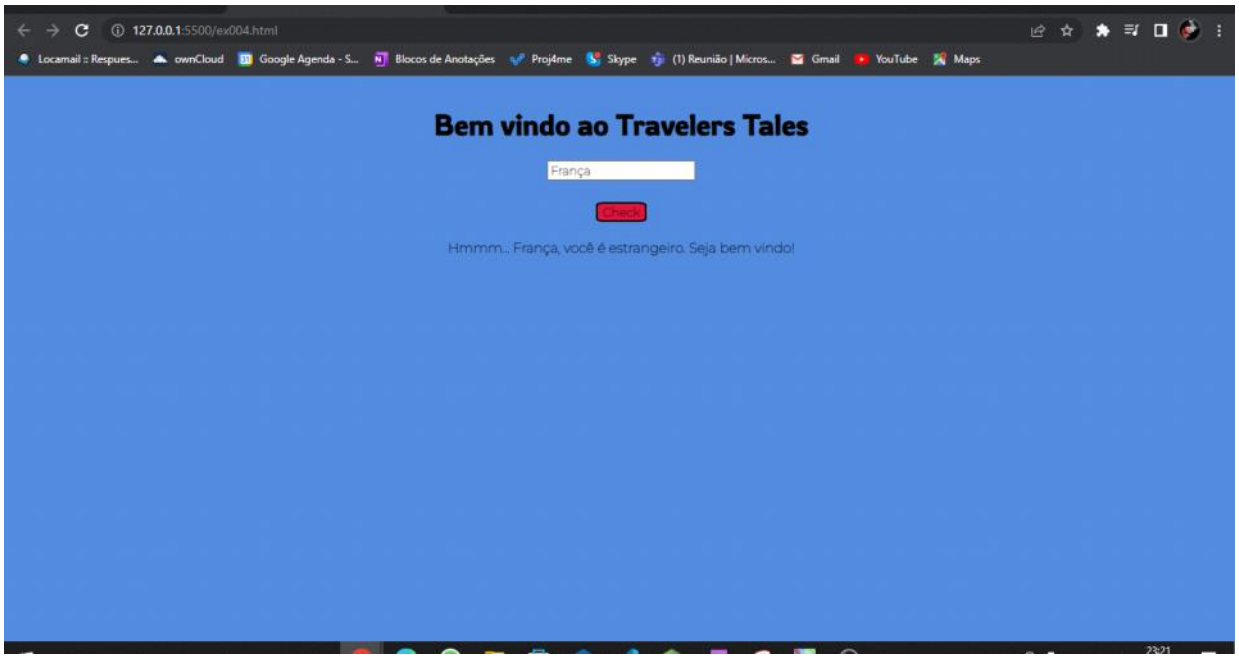
A ideia do exercicio era digitar o país de onde você vem, e se for diferente de Brasil, trazer um texto informando que somos estrangeiros.

Foi utilizado o script abaixo:

```
<body>

<div class="container">
    <h1>Bem vindo ao Travelers Tales</h1>

    <input type="text" name="txtpais" value="De onde vêm?" id="txtpais">
    <input type="button" id="botao1" value="Check" onclick="check()">
    <div id="confirm">E aí, de onde você é, humano?</div>
</div>

<script>

function check()

{
var txtpais = document.getElementById('txtpais')
var origem = 'Brasil'
var pais = (txtpais.value)
var confirm = document.getElementById('confirm')
confirm.innerHTML = `<p>Pelo visto você vem do país ${pais}</p>`
if (origem != pais) {confirm.innerHTML = `<p>Hmmm... ${pais}, você é estrangeiro. Seja bem vindo!</p>`}
else {confirm.innerHTML = `<p>Vem do ${pais}, e aí Brazukaaaa!</p>`}
}

</script>
```

Resultado:


Bem vindo ao Travelers Tales
França
Check
Hmmm... França, você é estrangeiro. Seja bem vindo!


127.0.0.1:5500/ex004.html
Locamail :: Respues...
ownCloud
Google Agenda - S...
Blocos de Anotações
Proj4me
Skype
(1) Reunião | Micros...
Gmail
YouTube
Maps
Bem vindo ao Travelers Tales
França
Check
Hmmm... França, você é estrangeiro. Seja bem vindo!

2.2.1 - Condições Aninhadas

sábado, 7 de janeiro de 2023 23:28

If Aninhado – Estrutura de Decisão em Javascript

O Javascript permite declarações if aninhados. Isto faz com que o código fique mais legível.

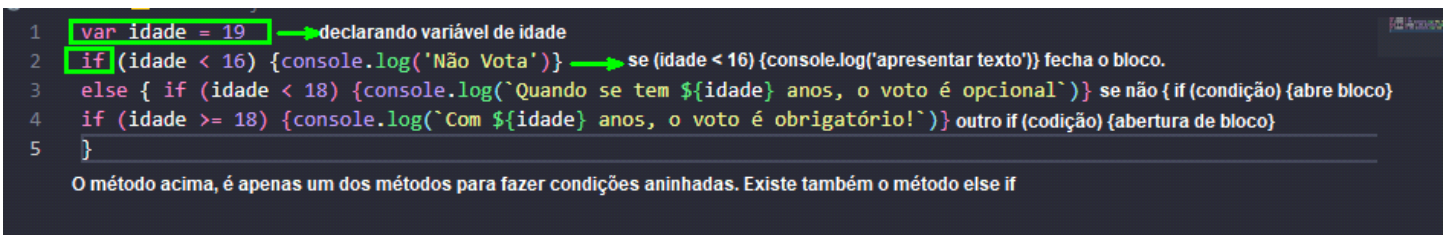
Podemos usar vários if...else um dentro do outro, dependendo de quantas opções temos disponíveis.

Sintaxe

```
if (condição)
{
if (condição)
 {
//código que será executado quando a condição for verdadeira
 }
}
```

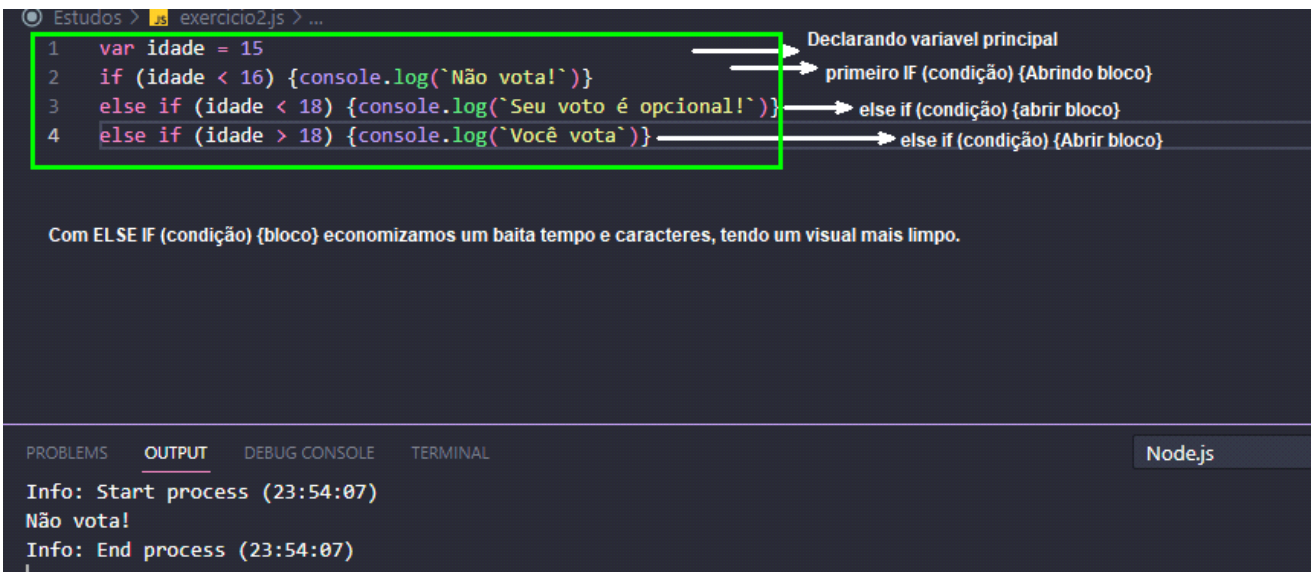
Vamos Ver Um Exemplo?

```
var i = 10;

if (i > 0)
{
 if (i <= 100)
 {
 console.log("i é um número positivo menor que 100");
 }
 else
 {
 console.log("i é um número positivo maior que 100");
 }
}
```



Com Else if tudo fica mais fácil!!!

Veja no exemplo abaixo:



Outro exemplo de condição aninhada, utilizando funções interessantes para horas.

Vale a pena anotar.

```
1  var agora = new Date() // definindo a var agora como new date () que é uma função
2  var hora = agora.getHours() // definindo var hora com a funçao getHours()
3  var minutes = agora.getMinutes() // definindo a var minutes puxando a var agora.getMinutes()
4  console.log(`Agora são ${hora} horas e ${minutes} minutos.`)//console log que vai concatenar a var hora e var
   minutes
5  if (hora > 12 && hora < 18) {console.log(`Está tarde`)} // primeira condição
6  else if (hora < 18) {console.log(`Ainda está cedo`)} // segunda condição
7  else if (hora > 18) {console.log(`Ufa! Já é de noite`)} // terceira condição
8
9
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                   Node.js

Info: Start process (00:10:46)
Agora são 0 horas e 10 minutos.
Ainda está cedo
Info: End process (00:10:46)
```

2.2.2 - Condições múltiplas

domingo, 8 de janeiro de 2023 00:15

Condição Múltipla

Existe um comando dentro do Javascript, chamado comando SWITCH
Switch (expressão) {}

Note que dentro do switch não temos uma (condição) e sim uma (expressão)

Switch (expressão) {bloco}

Existem várias possibilidades de valor.
Pra cada um, será colocado um CASE, se for outro valor, será colocado outro.

Também tem a default que é como se fosse o else do switch. (pra caso um valor não tenha sido atendido);

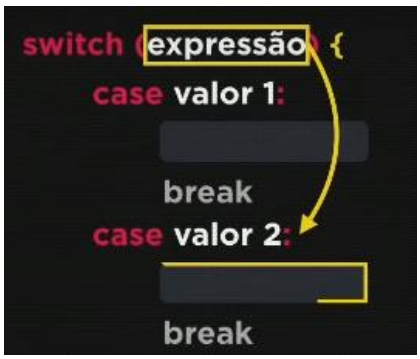
Será colocado um case pra cada valor, e por fim, podemos colocar um default (padrão).

Porém: dentro da estrutura switch, (vem desde linguagem C), dentro de cada bloco, é preciso colocar um comando break.

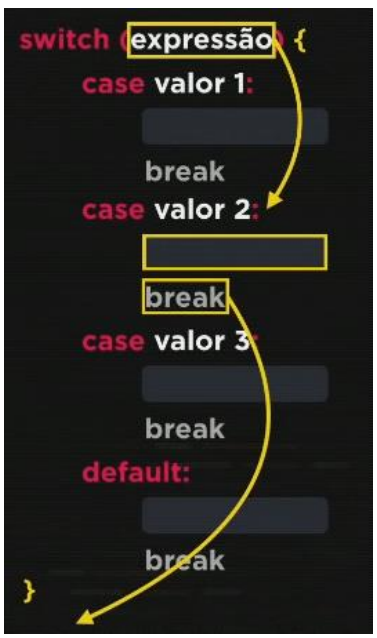
Esse break é OBRIGATÓRIO.

Como funciona:

A expressão 1 por exemplo, será testada.
Suponha que essa expressão não resulte em um valor 1 e sim, resulte num valor 2, então ela desviará automaticamente para o case valor 2, conforme seta amarela na imagem abaixo:

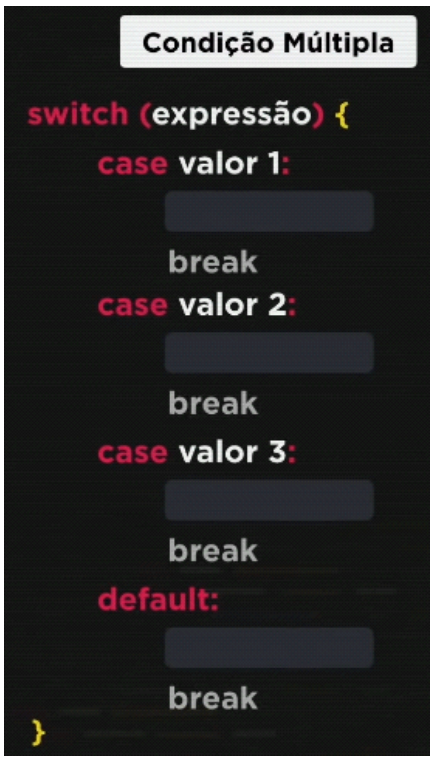


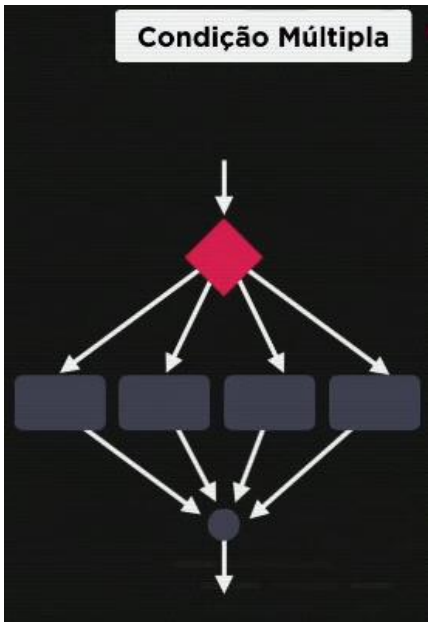
E no final, ele irá bater no break.
Quando bate no break, ele é desviado lá pra baixo (acompanhando o fluxo da seta amarela).



Se por acaso esse break não existir, vai dar problema porque ele vai executar todos os comandos até achar um break.

Exemplo:

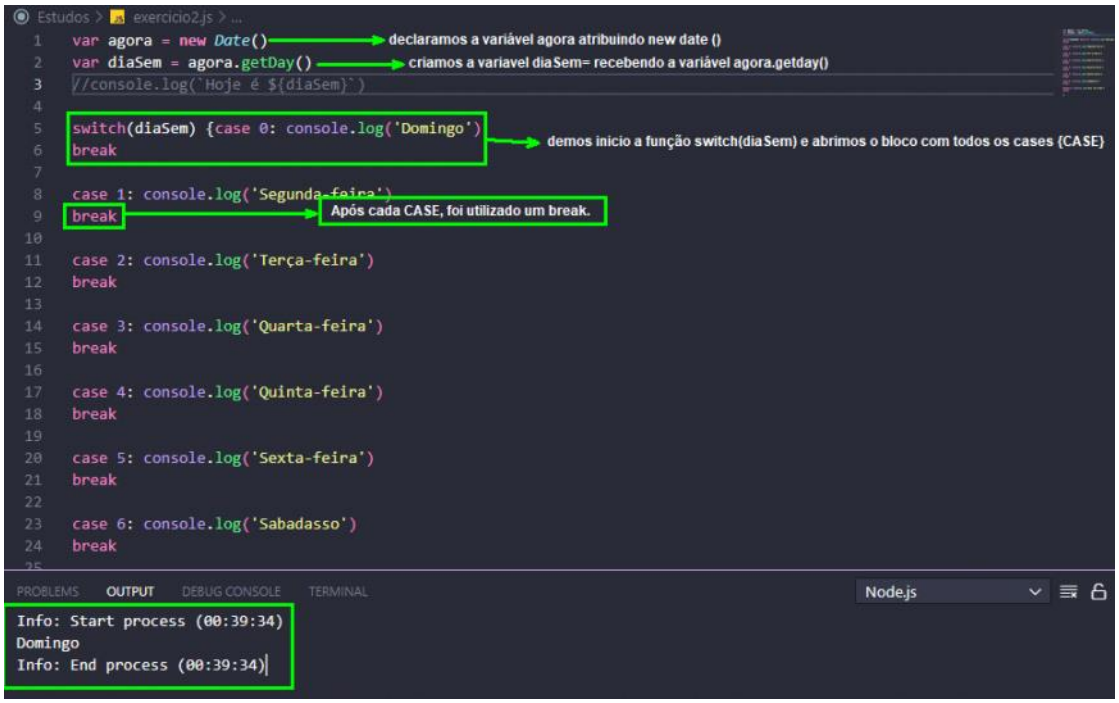




Nesse exemplo abaixo, basicamente era assim.

A variável diaSem, iria retornar o dia que estou hoje (que no caso é domingo)

Se não for a domingo, que no caso é o primeiro case (case 0:)
Ele irá procurar o próximo valor, pra cada vez que ele procurar um valor, temos que colocar um break.


Estudos > exercicio2.js > ...
var agora = new Date()
declaramos a variável agora atribuindo new date ()
var diaSem = agora.getDay()
criamos a variavel diaSem= recebendo a variável agora.getday()
//console.log(`Hoje é ${diaSem}`)
switch(diaSem) {case 0: console.log('Domingo')
break
demos inicio a função switch(diaSem) e abrimos o bloco com todos os cases {CASE}
case 1: console.log('Segunda-feira')
break
Após cada CASE, foi utilizado um break.
case 2: console.log('Terça-feira')
break
case 3: console.log('Quarta-feira')
break
case 4: console.log('Quinta-feira')
break
case 5: console.log('Sexta-feira')
break
case 6: console.log('Sabadasso')
break
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Node.js
Info: Start process (00:39:34)
Domingo
Info: End process (00:39:34)

Array

quinta-feira, 15 de dezembro de 2022 23:22

O que é um array?

Um array, que também pode ser chamado de matriz, pode ser composto por diversos tipos de dados como por exemplo nomes, idades, imagens, endereços entre outros, isso depende da pessoa desenvolvedora que está utilizando o array.
O maior objetivo de um array é armazenar informações de mesmo tipo juntos, sendo necessário então a criação de uma constante para a sua utilização.

Para entendermos melhor, imagine uma sala de aula, no qual a pessoa educadora possui 20 alunos e deseja saber quantas meninas e quantos meninos se encontram dentro da sala. Para isso, não existe a necessidade de ela passar em cada aluno para realizar essa identificação, sendo possível ela criar um único array e de acordo com a verificação setar uma cor para menino e outra cor para menina.

Qual a sintaxe de arrays em Javascript?

A sintaxe de um array tem diferentes formatos. **A declaração direta ou representação literal é aquela em que o array é representado pelos símbolos de colchetes**, que podem ou não conter elementos. Veja a sintaxe:

```
var novaArray = []
```

Nesse caso, o array novaArray é criado sem nenhum elemento atribuído. Essa mesma representação pode ser feita com a atribuição dos elementos, conforme a sintaxe abaixo:

```
var novaArray = [elemento0, elemento1, elemento2, ..., elementoN]
```

Outra sintaxe para a declaração de um array é por meio da **utilização do construtor new**. Veja abaixo:

```
var novaArray = new Array()
var novaArray = new Array(elemento0, elemento1, elemento2, ... , elementoN)
```

No primeiro caso, será criado um array vazio, ou seja, com a propriedade length, que significa tamanho, igual a zero. Já na segunda opção será criado um array com a quantidade de elementos informadas. Há, ainda, uma outra sintaxe:

```
var novaArray = new Array(numeroDeElementos)
```

Em que o numeroDeElementos corresponde ao tamanho do array. Ao criar um array apenas com a informação do número de elementos será definido o tamanho do array, ou seja, não

há nenhum conteúdo atribuído a eles, mas há a alocação de memória conforme o tamanho determinado.

É importante dizer que, apesar de existirem formas diferentes de declarar um array, **a recomendação de utilização é para a declaração do tipo literal,** uma vez que essa forma é mais prática, tem maior legibilidade e velocidade de execução.

Como criar um array

2.3 - JS na prática #1

sexta-feira, 16 de dezembro de 2022 00:12

Veja um exemplo na prática:

```
<script>
/* 1) - Declarei uma variável name sendo var name = "vininho"*/
var name = "vininho";

/*lancei uma função foo() abri a chave pra colocar dentro da função os parametros que vão trazer
console.log(nome da variável) que no caso o nome da váriavel foi name*/

function foo(){console.log(name)

}

/*Agora eu chamo a função*/

foo()

</script>
```

2.4 - JS com Gustavo Guanabara

sábado, 17 de dezembro de 2022 18:22

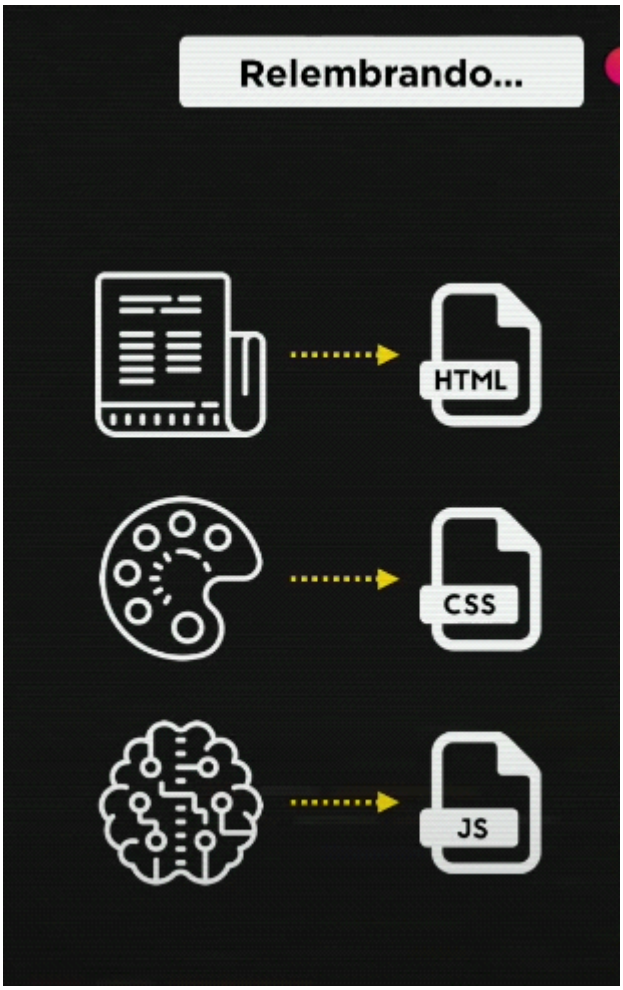
Começamos a primeira Aula, fazendo algumas análises histórica da aula.
Como por exemplo a linguagem ecmascript, que na verdade é a Javascript e o nome java veio muito por cota de Marketing;

Iniciamos o curso instalando o Node.Js
Porém agora vamos fazer algumas brincadeiras,
A primeira dela foi meter um alerta no prompt do navegador.

Veja só

```
<body>

    <div class="container">
        <h1>Olá, Mundo!</h1>
    </div>

    <script>
        window.alert("Seja bem-vindo")
    </script>

</body>
</html>
```

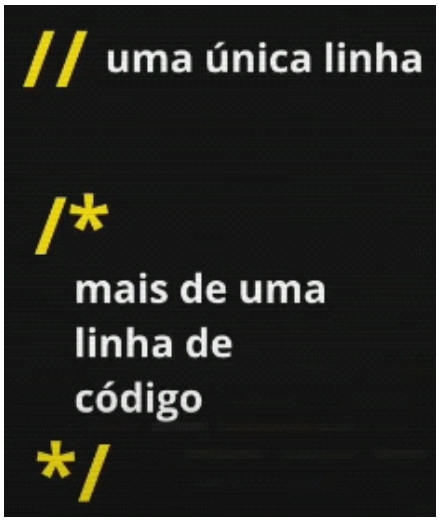
Um lembrete:


Relembrando...
HTML
CSS
JS

2.5 - Comandos básicos JS

sábado, 31 de dezembro de 2022 00:21

Bem do basico mesmo, vou te mostrar como fazer comentários no seu JS:
Os comentários são uteis para aqueles que gostam de deixar os códigos organizados.



Veja o exemplo abaixo na prática:

Existe o comentário que pode mostrar uma string toda

```
 <script>
/* Aqui, por exemplo, estou inserindo um comentário, sinalizando que os alertas abaixo irão ser disparados
logo quando eu iniciar o navegador*/
alert('Olá')
confirm('curtindo a aula?')
prompt('quer digitar algo?')
 </script>
```

E Também é possivel comentar cada linha de código (o que não é necessário, mas pode ser um dia).

```
 <script>
/* Aqui, por exemplo, estou inserindo um comentário, sinalizando que os alertas abaixo irão ser disparados
logo quando eu iniciar o navegador*/
alert('Olá') //duas barras legal pra sinalizar uma linha de código especifica
confirm('curtindo a aula?') //irá fazer uma pergunta
prompt('quer digitar algo?') //irá solicitar que digite algo!!
 </script>
```

2.5.1 - Variáveis

quinta-feira, 29 de dezembro de 2022 20:21

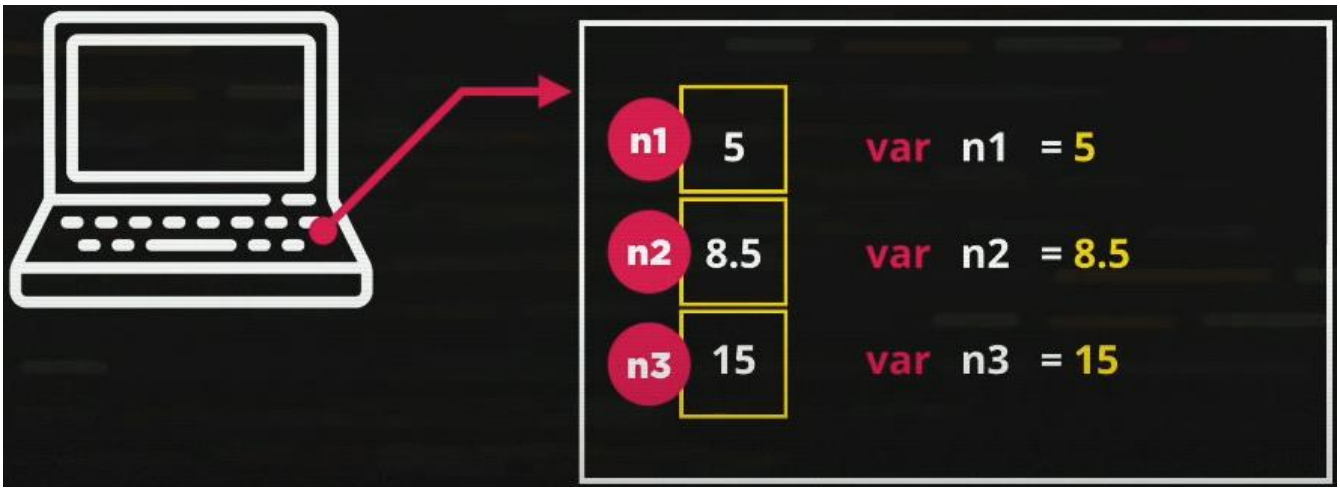
Vamos começar falando das variáveis.

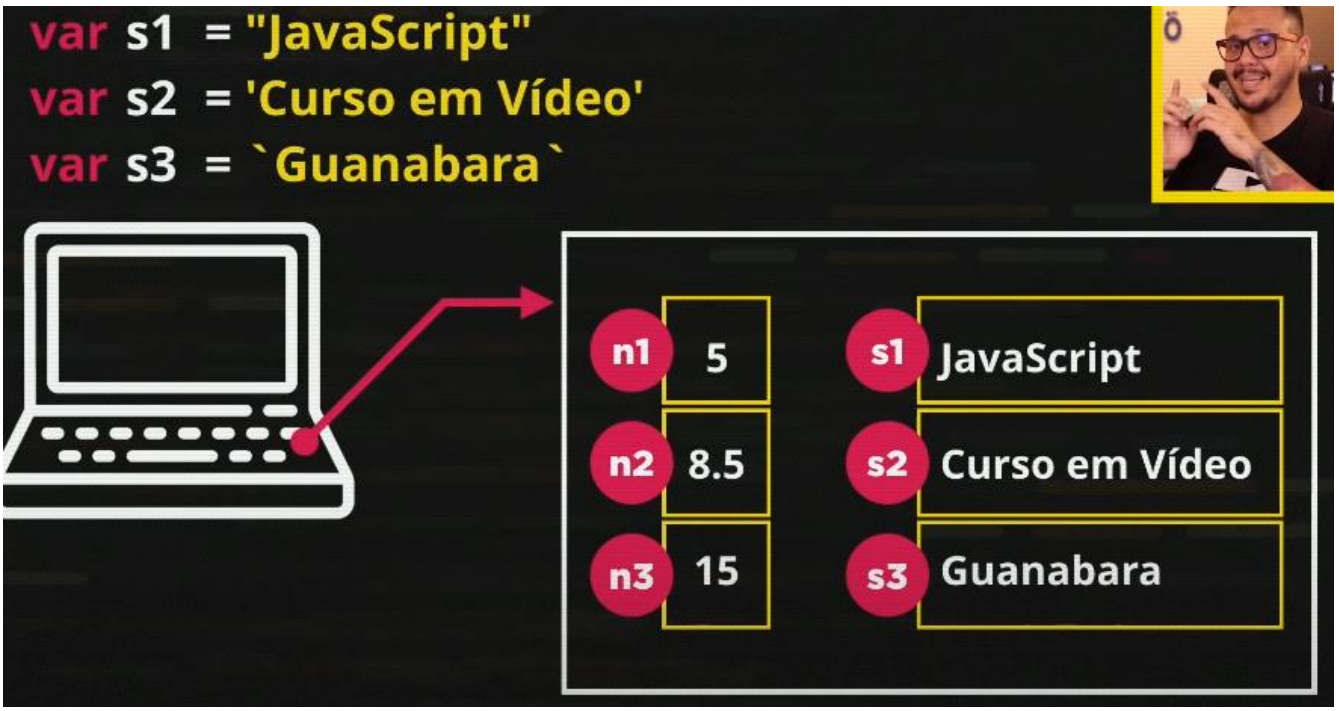
Para que servem as variáveis?
Guardar dados em palavras, note que existem varios tipos de dados para variáveis.
Sendo algumas palavras, outros números inteiros, decimais, reais, binários, true, false, entre outros.

É um dos principais comandos da programação e muito provavelmente será aplicado em tudo.

Note que no exemplo abaixo, a var n1 recebe o valor 5.
Ele não está dentro de aspas, por exemplo porque é um número inteiro.

No entanto, após a imagem das variáveis numéricas, verá que para as strings que tiverem textos, precisarão estar dentro de aspas ou apóstrofes.





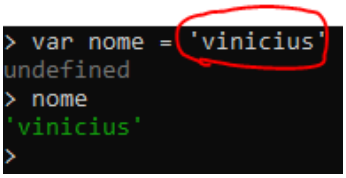
O nome de cada variável se chama de **identificador**.

→ A primeira regra é que pode começar com uma letra, um cifrão ($) ou um underline (_)
→ Não podem começar com números
→ É possível usar letras ou números

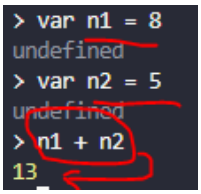
→ É possível usar acentos e símbolos.
→ Não podem conter espaços dentro do identificador
→ E por fim, não podem ser palavras reservadas (palavras que o próprio javascript usa por exemplo).

Agora vamos trabalhar com o node.js

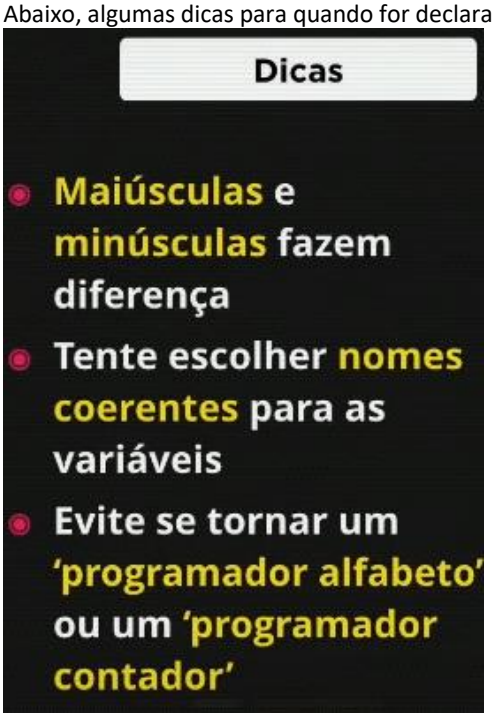
Var nome = 'vinicius'
Veja a imagem abaixo;



É notável que declaramos a variavel nome utilizando 'vinicius'
E assim que foi digitado nome sem aspas, o node retornou o valor declarado na variável.



Abaixo, algumas dicas para quando for declarar o nome das variáveis



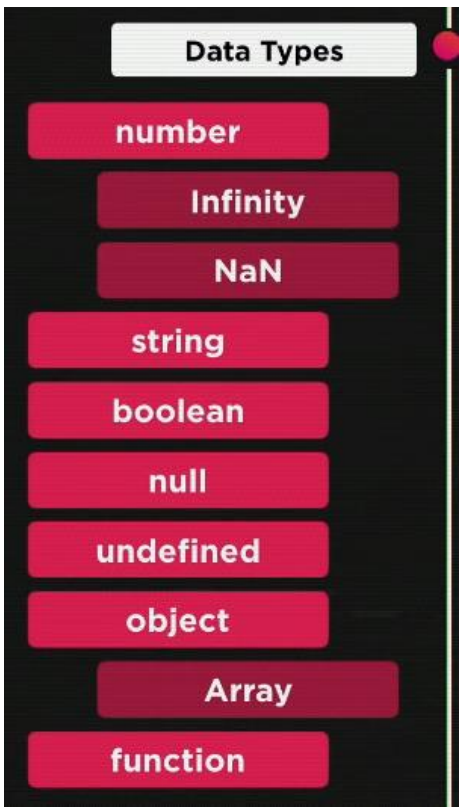
2.5.2 - Tratamento de dados

sábado, 31 de dezembro de 2022 01:04

Olá pessoal, identificamos que estes itens abaixo, são considerados data types.

Data Types:

- → Number
  - infinity
  - NaN
- → String
- → Boolean
- → Null
- → Undefined
- → Object
  - Array
- → Function



Existem 3 tipos primitivos principais, sendo eles: number, string e boolean.

Agora vamos lá pro VS Code, e dê uma olhada na prática.

```
<script>
var nome = window.prompt('Digite seu nome')
window.alert('é um prazer conhecê-lo, ' + nome +'!')
</script>
```

Utilizando o exemplo acima dentro da tag script, podemos ver que
A variável nome, recebe o valor digitado no comando prompt.
E depois, emite um alerta com um texto um sinal de + que concatena, e a variável.

```
<script>
    var nome = window.prompt('Digite seu nome')//imagem 1
    window.alert('é um prazer conhecê-lo, ' + nome +'!')//imagem 2
</script>
```

Trazendo o seguinte resultado:

Imagem 1:

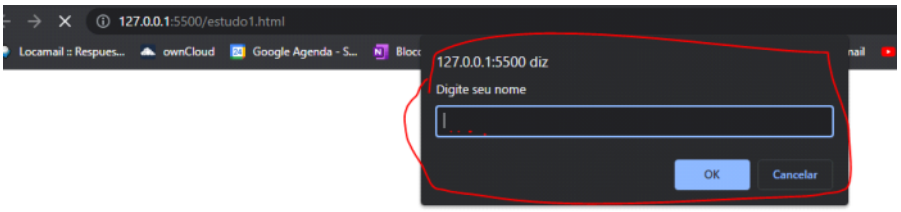
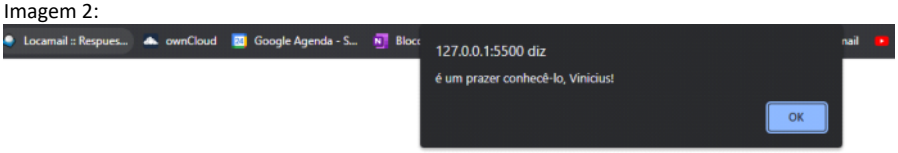


Imagem 2:



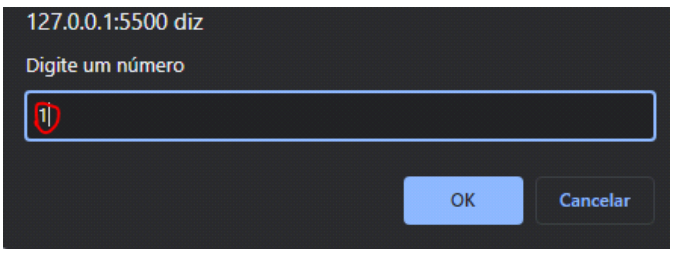
Primeiro breve desafio,

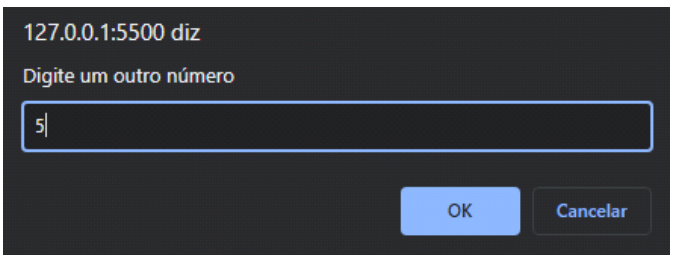
Vamos fazer uma soma aqui no VS code.

Utilizamos a linha de código a seguir:

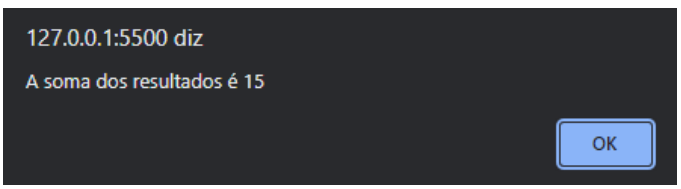
```
<script>
   var n1 = window.prompt('Digite um número')
   var n2 = window.prompt ('Digite um outro número')
   var soma = n1 + n2
   window.alert ('A soma dos resultados é ' + soma)
</script>
```

Veja o exemplo:





Surpreendentemente obtivemos o seguinte resultado abaixo:



O que quer dizer que o seguinte comando `var soma = n1 + n2` não necessariamente executou uma operação aritmética de soma, uma vez que o sinal de adição, também direciona o JS para fazer uma concatenação.

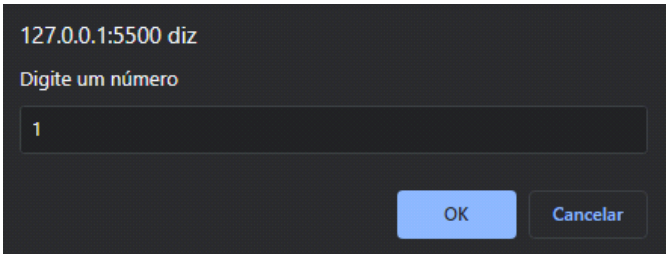
E para solucionar isso, precisamos fazer uma conversão nestes dados.

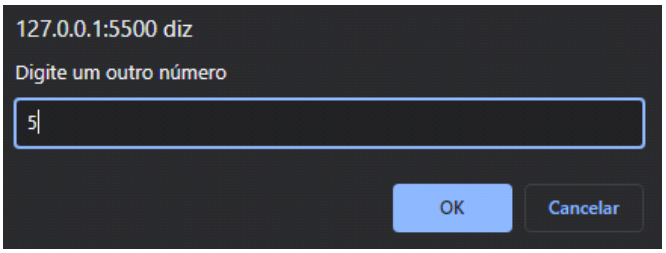
Para tal, utilizaremos o comando Number.parseInt

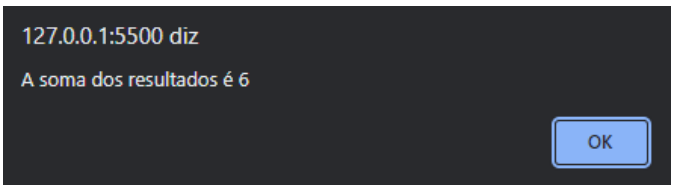
Veja:

```
<script>
   var n1 = Number.parseInt(window.prompt('Digite um número'))
   var n2 = Number.parseInt(window.prompt ('Digite um outro número'))
   var soma = n1 + n2
   window.alert ('A soma dos resultados é ' + soma)
</script>
```

Veja que agora que convertemos o Number.parseInt(conteúdo do prompt) nos trouxe novo resultado, vamos utilizar os mesmos números para exemplificar.







E agora, houve uma atualização, que nos isenta de precisar digitar o Number.parseInt.

Basta digitar antes da string, o comando Number, conforme abaixo:

```
<script>
   var n1 = Number(window.prompt('Digite um número'))
   var n2 = Number(window.prompt ('Digite um outro número'))
   var soma = n1 + n2
   window.alert ('A soma dos resultados é ' + soma)
</script>
```

a

É possível também, converter um número em string, como no exemplo abaixo:

```
<script>
   var n1 = Number(window.prompt('Digite um número'))
   var n2 = Number(window.prompt ('Digite um outro número'))
   var soma = n1 + n2
   window.alert ('A soma dos resultados é ' + String(soma))
</script>
```

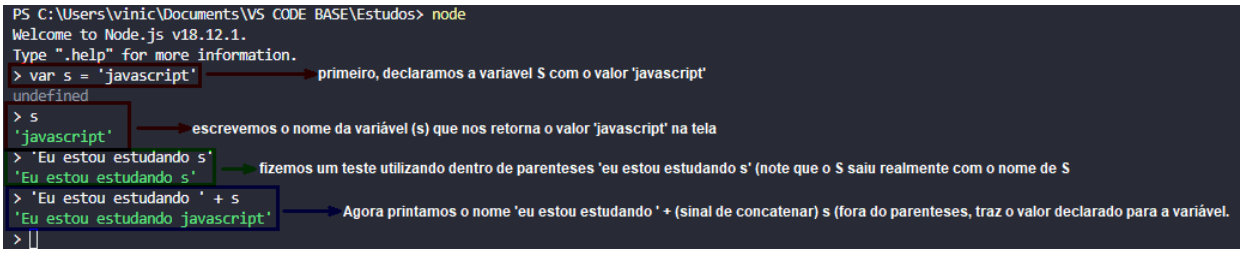
Veja que o código String(soma) é digitado após a concatenação.

Ou seja, o Number(dados) converte para número
E o String(dados) converte para string.

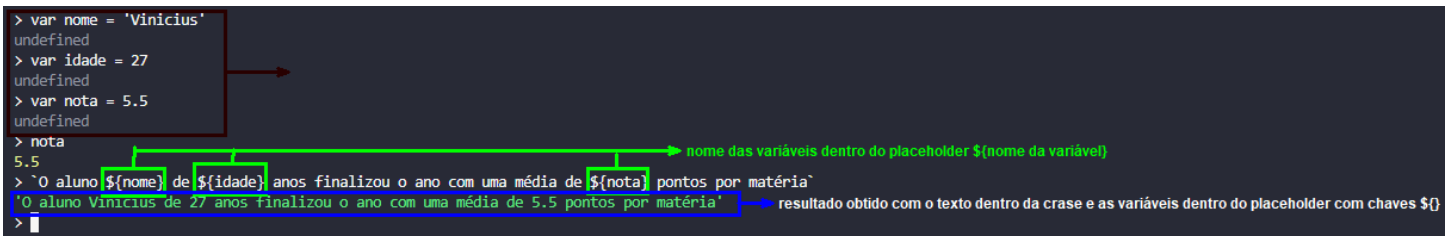
(number + number) para adição.
(string + string) para concatenação.

```
// (number + number) para adição
// (string + string) para concatenação
```

Como fazer formatação de novas strings



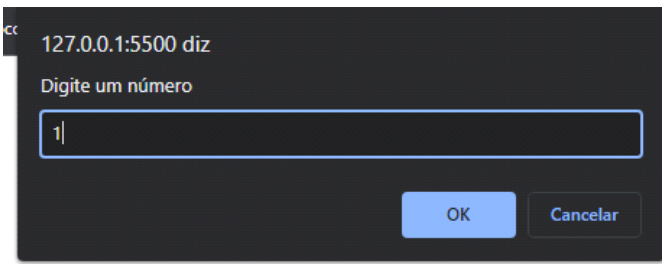
Veja a maneira mais pratica de concatenar:

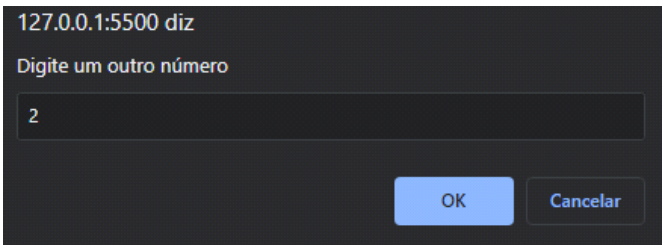


E agora veja como utilizei 'template string' no VS Code e como é muito mais rapido:

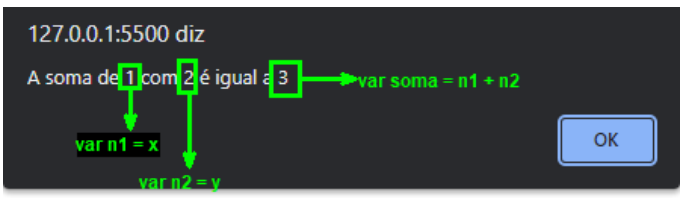
```
<script>
   var n1 = Number(window.prompt('Digite um número'))
   var n2 = Number(window.prompt ('Digite um outro número'))
   var soma = n1 + n2
   window.alert (`A soma de ${n1} com ${n2} é igual a ${soma}`)
</script>
```

Imagens do resultado obtido:

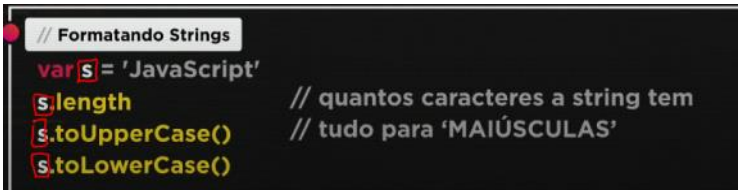




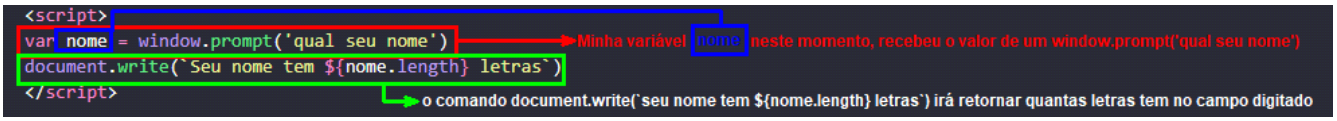

```
<script>
   var n1 = Number(window.prompt('Digite um número'))
   var n2 = Number(window.prompt ('Digite um outro número'))
   var soma = n1 + n2
   window.alert (`A soma de ${n1} com ${n2} é igual a ${soma}`)
</script>
```



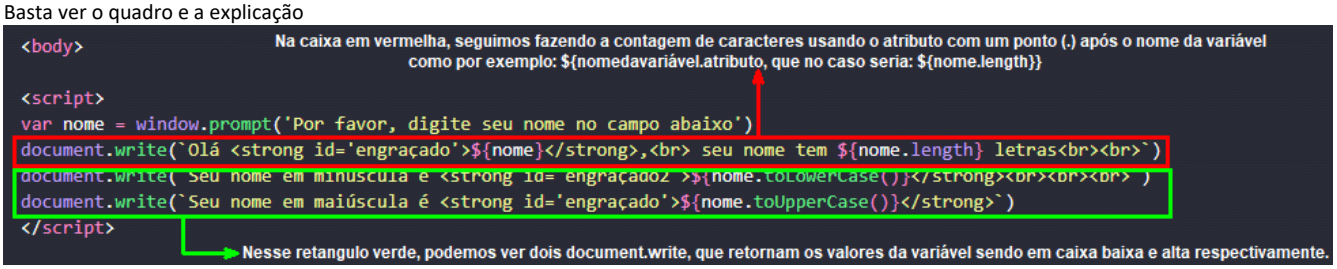
Algumas observações:



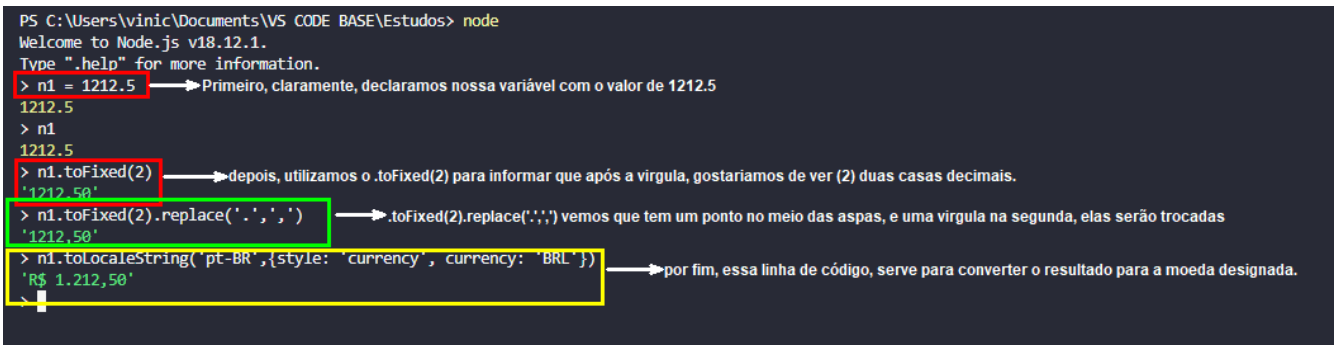
Agora, veja que interessante este exercício:



Ou ainda, complementando com maiores informações.
Basta ver o quadro e a explicação



Agora e se eu quiser utilizar uma variável númerica pra trazer um resultado do tipo moeda?



Abaixo: os operadores

```
> n1 = 1212.5
1212.5
> n1
1212.5
> n1.toFixed(2)
'1212.50'
> n1.toFixed(2).replace('.',',')
'1212,50'
> n1.toLocaleString('pt-BR',{style: 'currency', currency: 'BRL'})
'R$ 1.212,50'
```

2.5.3 - Operadores JS

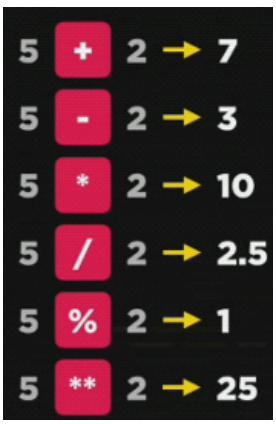
terça-feira, 3 de janeiro de 2023 22:25

Sendo aritméticos, atribuição, relacionais, lógicos e ternários.
Além disso, existem outros que da pra se aprofundar.

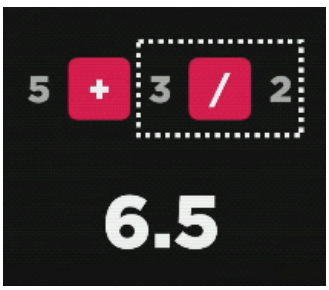
No entanto, seguiremos com os aritméticos e de atribuição.

Abaixo, um exemplo de operadores aritméticos do JS.

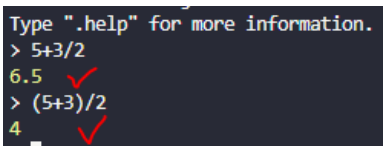
Irei dar o ênfase no 5%2 --> Esse esquema de número, retorna o resto da divisão.
Já o item 5 ** 2 --> retorna 5 ao quadrado, ou seja, é voltado para potencias.



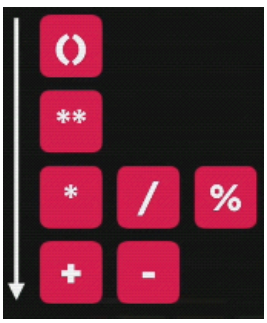
Lembrando sempre que, conforme a imagem abaixo, uma expressão númerica, o JS segue a mesma regra da matemática, onde no caso, a divisão, vem antes da adição e subtração.



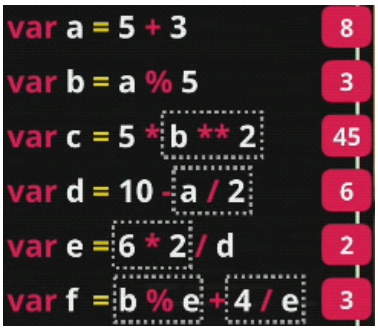
Exemplo prático:



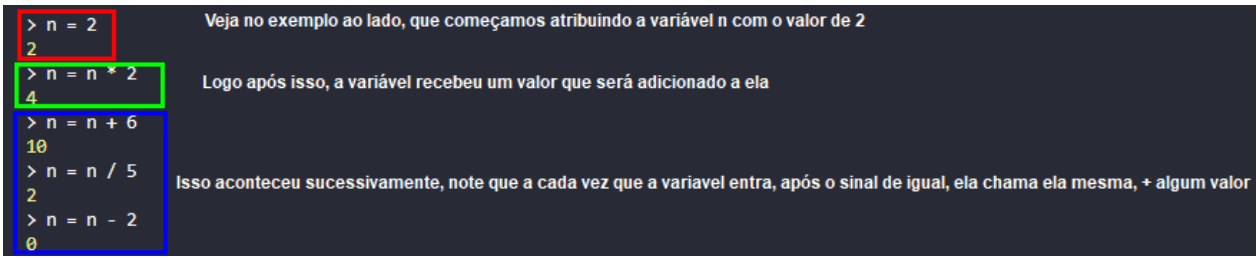
Ordem de precedência dos operadores aritméticos:



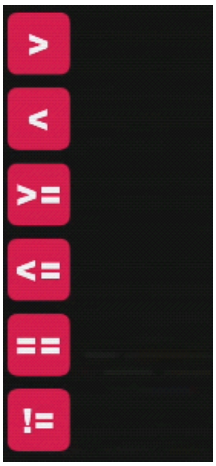
Outros exemplos de atribuições de operadores aritméticos simples:



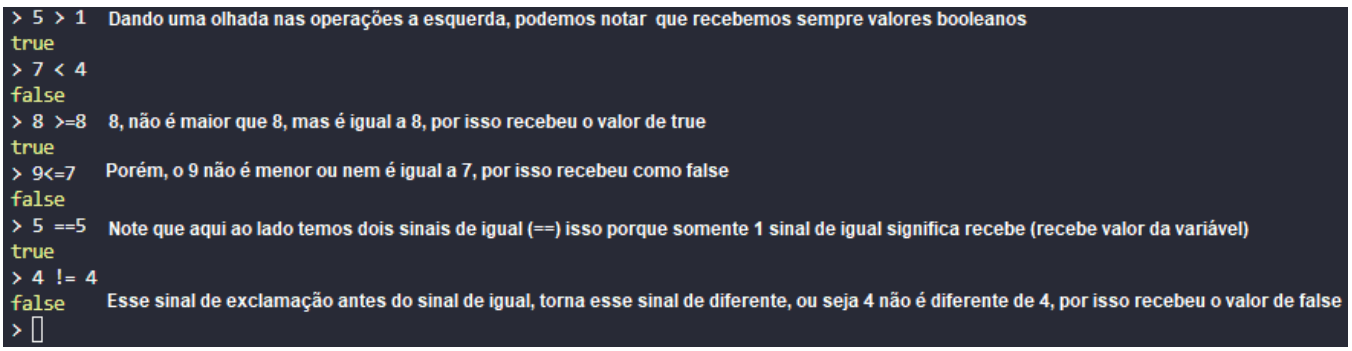
Agora, preste a atenção na explicação abaixo, que fala sobre as atribuições da variável.



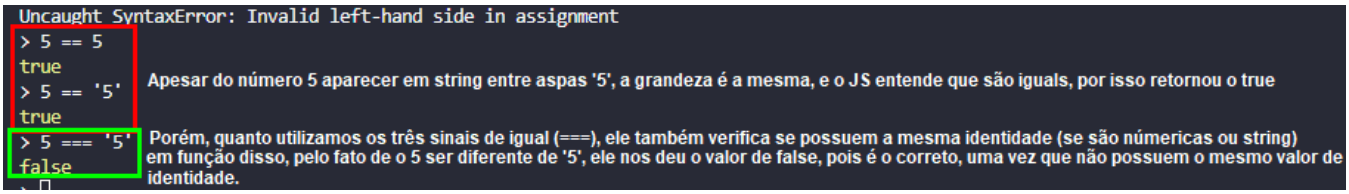
Operadores Relacionais:



Veja na pratica:

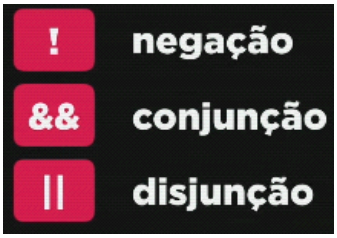


Agora, observe atentamente as marcações abaixo, sobre identificação de valores.



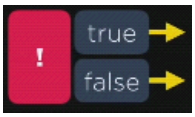
Operadores Lógicos.

Quem são os operadores lógicos?
Bom, dê uma olhada aqui abaixo para você saber quais são.

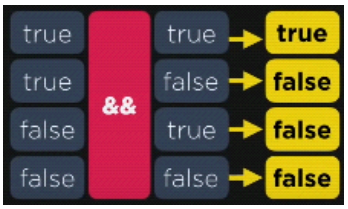


Vamos entender um pouco mais:

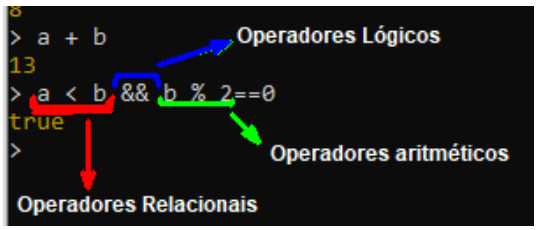
Exclamação ou você coloca true ou false (ou vc coloca uma expressão que retorne esses valores).
Simples assim.



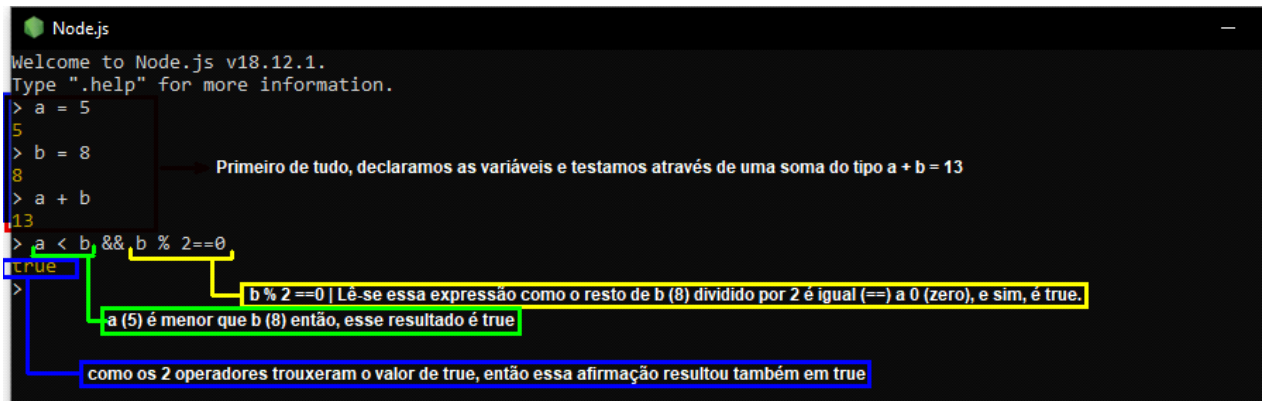
Para a conjunção, só atende se as duas atribuições forem verdadeiras.



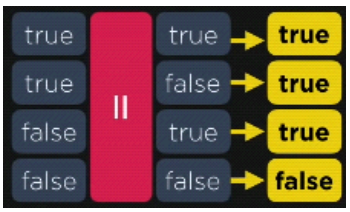
Quando se tem operadores aritméticos, relacionais e lógicos na mesma expressão
Primeiro é feito os aritméticos, relacionais e por fim lógicos, nesta mesma ordem.



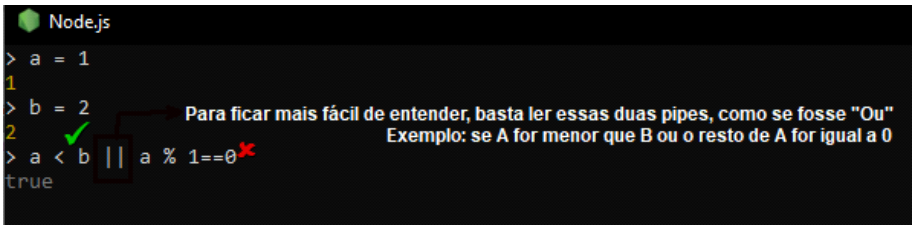
Vamos ver na prática:

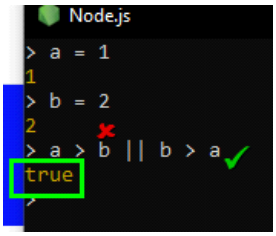


Já a **disjunção**, são dois pipes, também é um operador binário com 2 valores de cada lado.
Basta que um deles seja verdadeiro, para que o resultado retorne o valor de verdadeiro.
Veja a tabela abaixo:



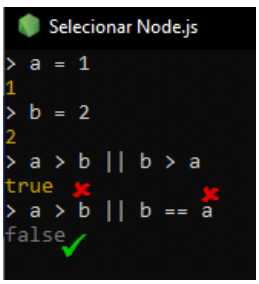
Exemplo prático:



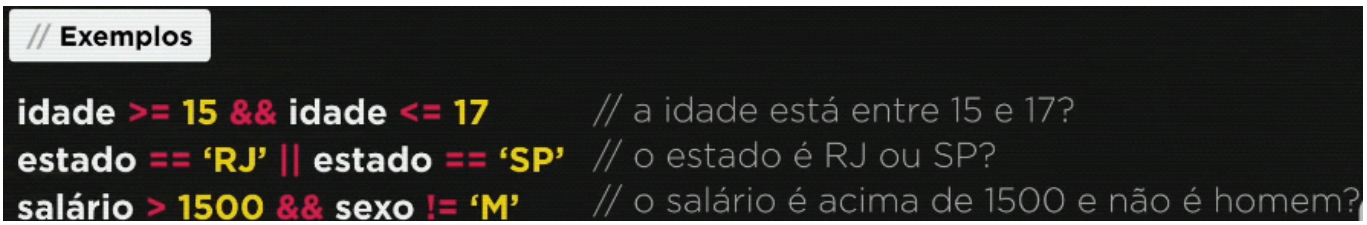


Somente se as duas condições forem falsas, ele trará o valor de False!
Conforme a imagem abaixo diz

O A não é maior que B |ou| o B não é igual ao A

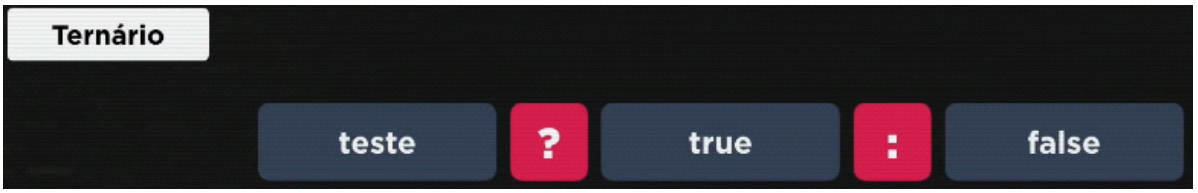


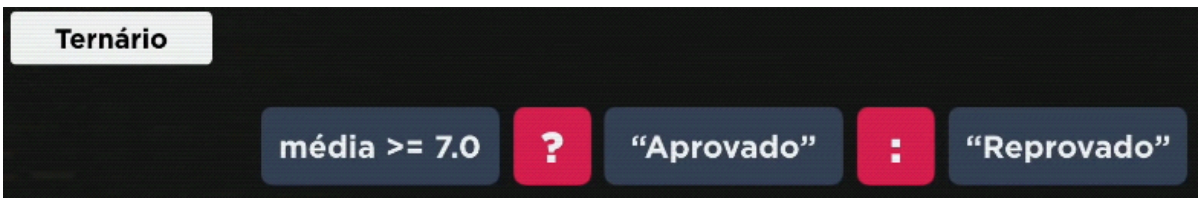
Agora, veja uma explicação básica que vai te orientar bem melhor:



Segue abaixo, um print muito importante da precedência:

Por fim, temos o ternário:





3.1 Lógica de programação e Algoritmos

quinta-feira, 29 de dezembro de 2022 20:03