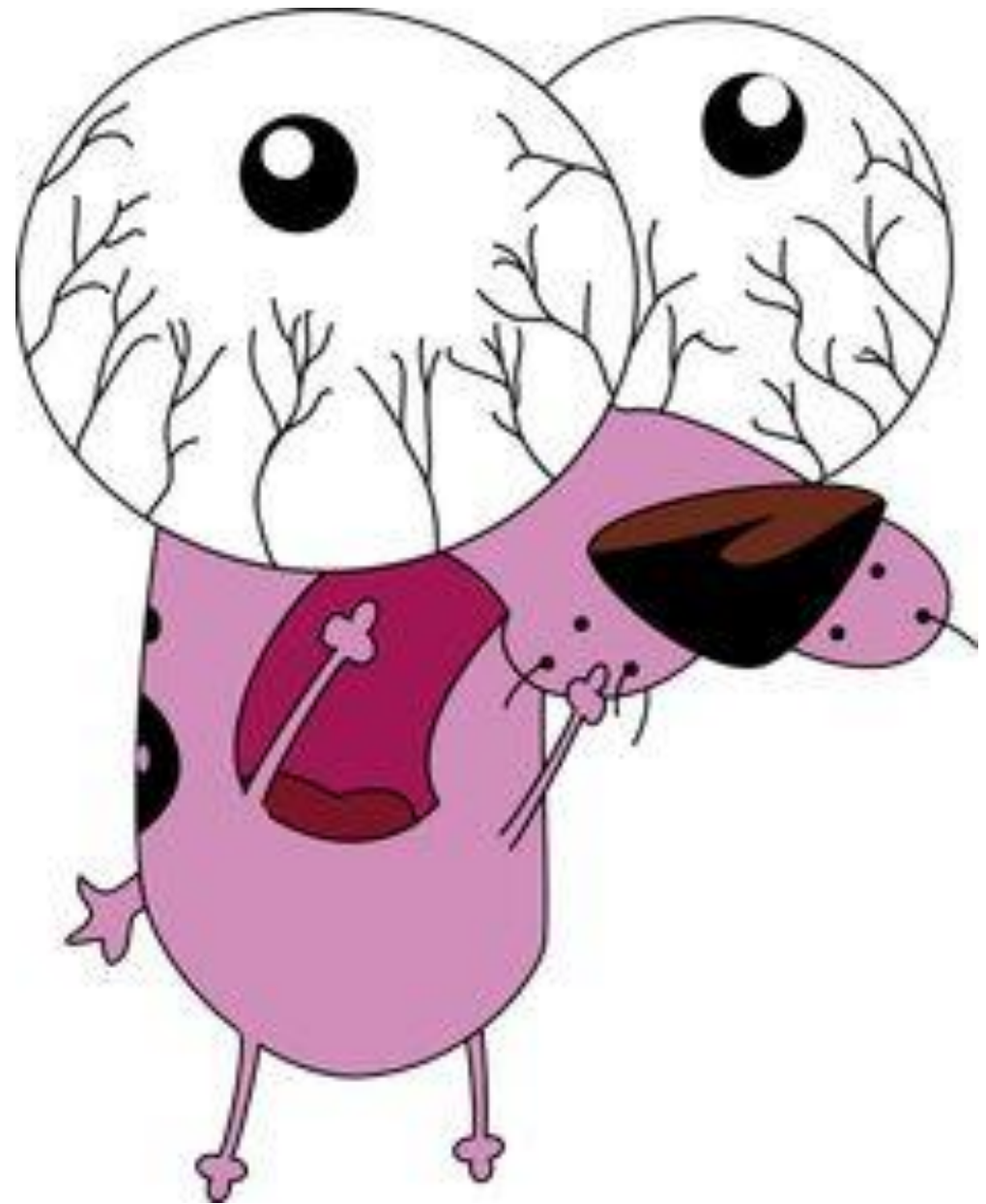


Programação Frontend Javascript

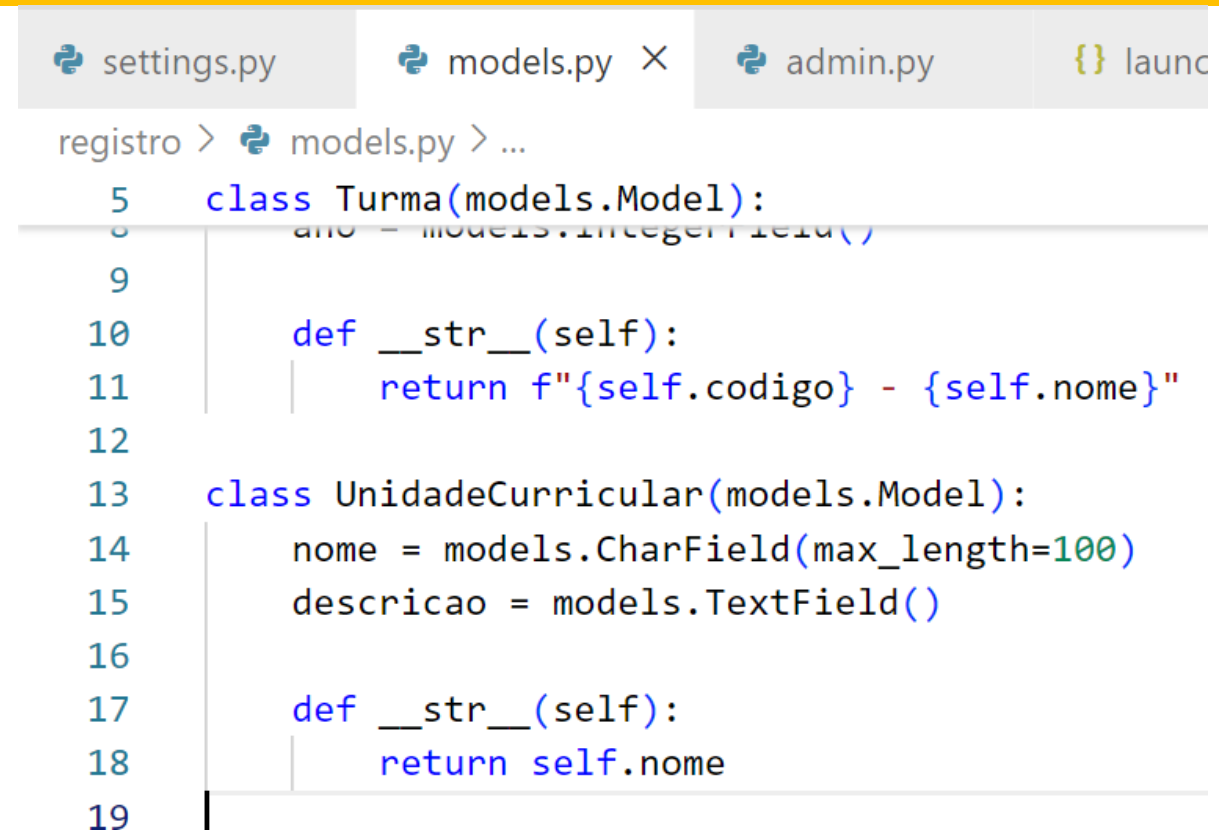
Rodrigo Attique
Desenvolvedor de Sistemas

O que é
Front-end?



Front-end

- **Traduzindo para o português quer dizer, parte da frente.**
- **Em termos práticos é a camada de sistema mais próxima dos usuários.**
- **Um front-end é sempre responsável por interagir com o usuário, conversando com o back-end.**



The screenshot shows a code editor with three tabs: settings.py, models.py (active), and admin.py. The active tab displays Python code for two Django models. The first model is 'Turma', which inherits from 'models.Model' and has a field 'ano' of type 'models.IntegerField()'. It also has a '__str__' method that returns a string formatted as '{self.codigo} - {self.nome}'. The second model is 'UnidadeCurricular', which also inherits from 'models.Model' and has two fields: 'nome' of type 'models.CharField(max_length=100)' and 'descricao' of type 'models.TextField()'. It also has a '__str__' method that returns 'self.nome'. The line numbers 5 through 19 are visible on the left side of the code editor.

```
registro > models.py > ...
5  class Turma(models.Model):
6      ano = models.IntegerField()
9
10     def __str__(self):
11         return f"{self.codigo} - {self.nome}"
12
13     class UnidadeCurricular(models.Model):
14         nome = models.CharField(max_length=100)
15         descricao = models.TextField()
16
17         def __str__(self):
18             return self.nome
19
```

Como funciona?



No geral front-ends combinam HTML (marcação) e programação (JavaScript).



Como JavaScript....

[Esta Foto](#) de Autor Desconhecido está licenciado em [CC BY](#)



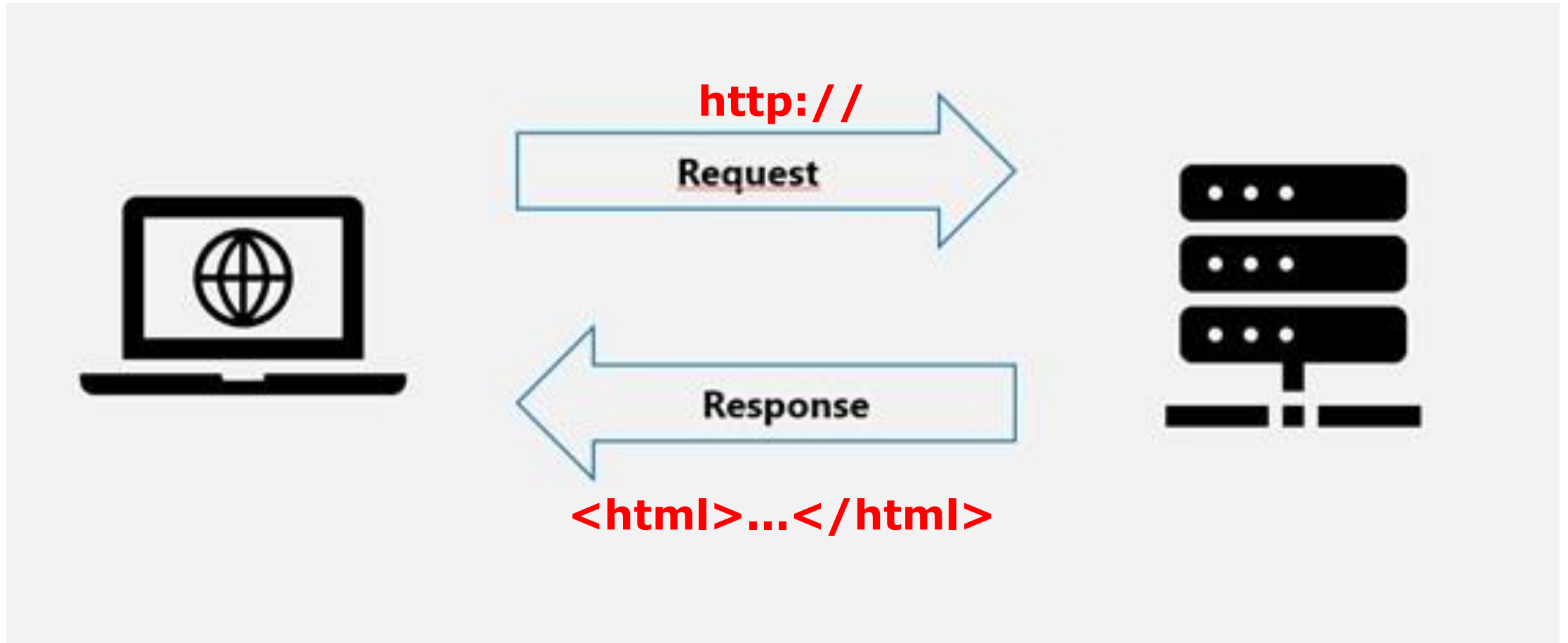
Essas linguagens são responsáveis por “gerar” o HTML que será exibido no navegador diretamente.

Front-end Vs Back-end

- Enquanto o back-end é responsável pela camada de dados.
- O front-end é o responsável pela apresentação ao usuário.
- Em resumo o enquanto o back-end usa uma linguagem de programação.
- Font-end usa linguagem de marcação HTML e JavaScript.

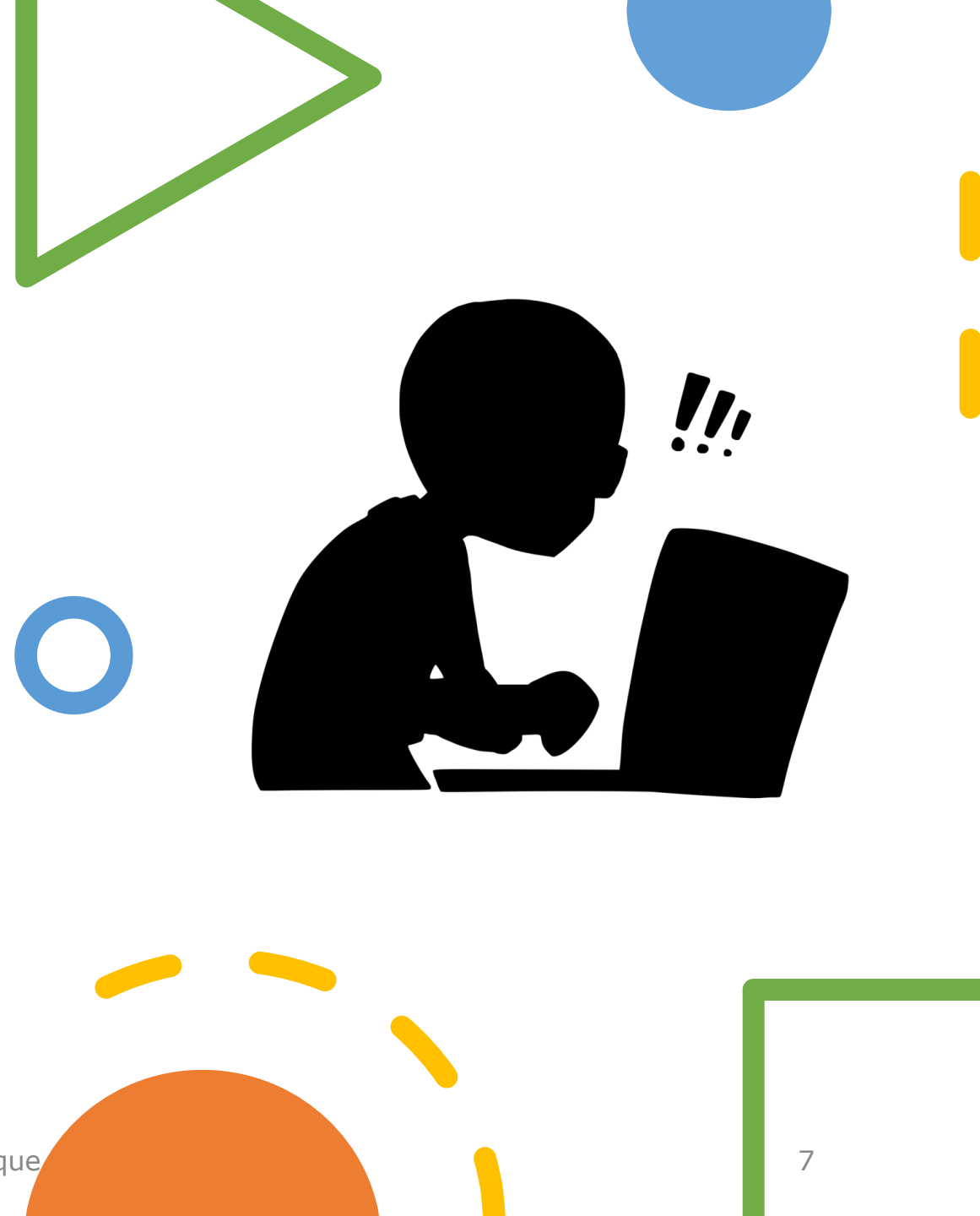


Modelo Cliente Servidor



Como funciona?

- **O cliente (navegador) faz uma requisição.**
- **O front end faz uma requisição ao backend.**
- **Que devolve apenas os dados sem processá-los.**
- **Esses dados serão processados diretamente pelo cliente.**



Mas e o JavaScript?

- **JavaScript é uma linguagem de eventos criada para automatizar páginas HTML.**
- **JavaScript não é Java, sim são duas linguagens totalmente distintas.**
- **Enquanto Java é uma linguagem de back-end como Python, JavaScript só funciona no navegador.**
- **Podemos usá-lo para fazer validações, animações, montar HTML entre outras finalidades.**



Full-Stack

- Ou pilha completa.
- São frameworks que carregam **bibliotecas** para todas as operações.
- Banco de dados
- Templates
- ETC....
- No geral é necessário somente instalar e usar o que ele oferece.



Alguns termos...

Microframeworks

- Já os microframeworks são responsáveis apenas por gerenciar as rotas.
- Com todo o resto devendo ser instalado pelo desenvolvedor.
- No geral é necessário mais conhecimento para usá-los.
- Porém dão muito mais flexibilidade ao projeto.



Bibliotecas

- São trechos de código pronto que usamos para diversos fins.
 - **Como manipulação de data e hora.**
 - **Conexão com banco de dados**
 - **Manipulação de dados e gráficos.**
 - **Etc.**



Mas e o JavaScript?

- **JavaScript está na categoria de linguagem de programação.**
- **É possível usar frameworks que nos ajudem.**
- **Como o React ou a biblioteca JQuery.**



JavaScript não é Java!

JavaScript

```
function AddOne(value) {  
  if (typeof value === 'number') {  
    return value + 1  
  } else if (typeof value === 'string') {  
    return value + '1'  
  }  
}  
  
console.log(AddOne(1)) // 2  
console.log(AddOne('1')) // '11'
```

Java

```
public void processData() {  
  do {  
    int data = getData();  
  
    if (data < 0)  
      performOperation1(data);  
    else  
      performOperation2(data);  
  } while (hasMoreData());  
}
```

JavaScript não é Java!

JavaScript

- Orientada a Eventos;
- Multiparadigma;
- Tipagem dinâmica fraca;
- Focada em client-side (front-end);
- Interpretada

Java

- Orientada a Objetos;
- Paradigma POO;
- Tipagem estática forte;
- Focada em server-side (back-end);
- Pré-compilada, JIT (Just-in-Time);

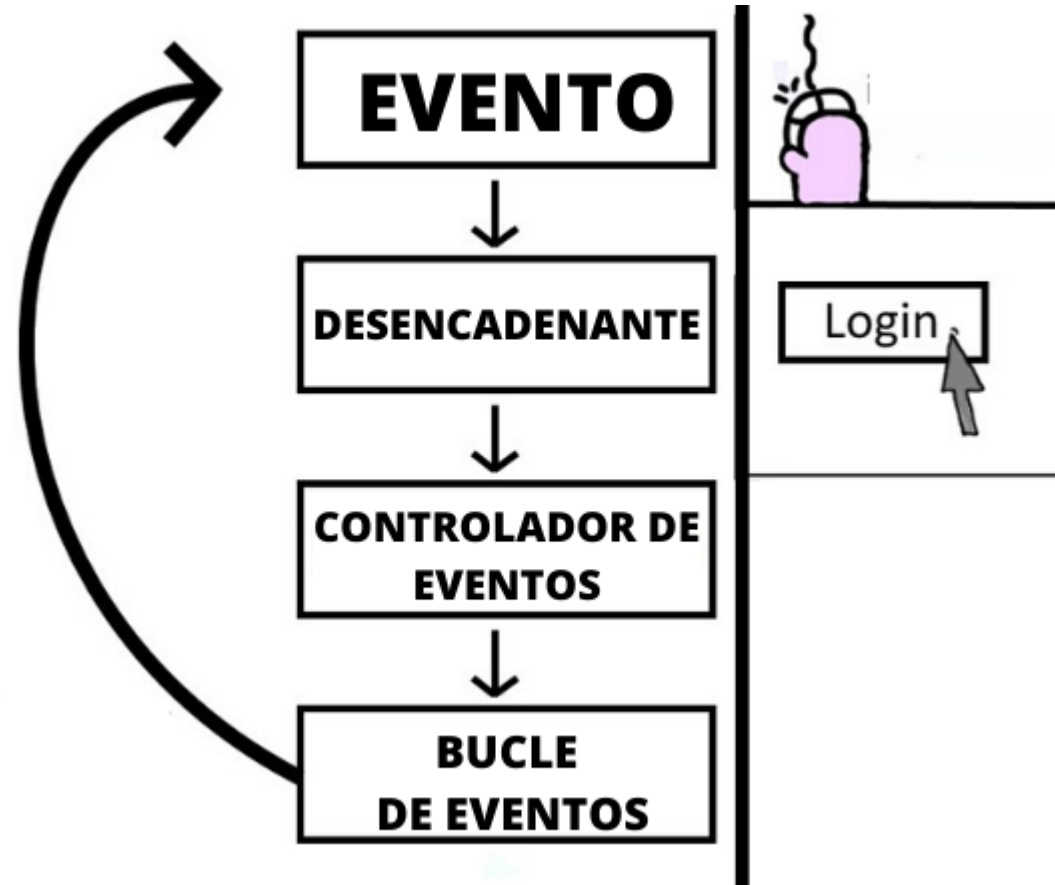
Java vs JavaScript

Vamos fazer uma pesquisa

Pesquise sobre os pontos de conflito do slide anterior, vamos criar uma discussão dos pontos fortes e fracos entre as duas linguagens.

Vamos dar exemplos práticos dessas diferenças, como é linguagem multiparadigma, como é alocação dinâmica de memória...

Orientada a Eventos



Exemplo do jogo Arcade

- Baixe o gameJS.zip
- https://github.com/attiquetecnologia/javascript_examples/tree/main

Multiparadigma

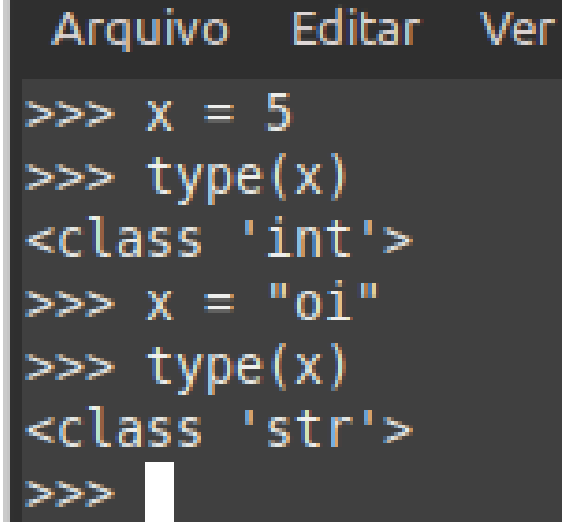
Quer dizer que é possível escrever código estruturado junto com orientação a objetos e funções.

Isso dá uma enorme produtividade porém precisa ser usado com cuidado.



Tipagem Dinâmica

- A tipagem dinâmica permite que variáveis sejam criadas e alocadas sem a necessidade de declará-las.
- O interpretador sabe o tipo da variável pelo tipo de dado como no exemplo.
- Perceba que o tipo de X muda de int para str durante a execução do programa.



A screenshot of a Python interpreter window with a dark background. The window has a menu bar at the top with three items: 'Arquivo', 'Editar', and 'Ver'. Below the menu bar, the following commands and their outputs are shown in a monospaced font: '>>> x = 5', '>>> type(x)', '<class 'int'>', '>>> x = "oi"', '>>> type(x)', '<class 'str'>', and '>>>' followed by a white cursor block.

```
>>> x = 5
>>> type(x)
<class 'int'>
>>> x = "oi"
>>> type(x)
<class 'str'>
>>> 
```

Tipagem Estática

- Na tipagem estática os tipos de dados das variáveis são pré definidos na declaração e não mudam ao longo da execução do programa.
- Logo é preciso **declarar** a variável antes de usá-la.

```
1 public class Program {  
2  
3  
4 public static void main(String[] args) {  
5  
6     int num0;  
7  
8     int num1 = 2;  
9  
10    float num2 = 3;  
11  
12    double num3 = 4;  
13  
14    String nome = "Artigo Blog";  
15 }
```

Tipagem dinâmica Fraca

- Python e JavaScript usam tipagem dinâmica, por exemplo.
- Mas enquanto Python trabalha com tipagem forte JavaScript trabalha com tipagem fraca.

Tipagem Dinâmica Forte (PYthon)

Perceba que não se pode somar um inteiro com uma string

```
>>> 1 + '1'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Tipagem dinâmica Fraca (JS)

Perceba que o exemplo anterior em JS não retorna erro mas efetua a concatenação de um inteiro com string

```
> 1 + '1' |
```

```
< '11'
```


Configurando o JavaScript

- Não é necessário instalar nada no computador.
- **Todo navegador WEB já vem com um interpretador JavaScript embutido!**

Explicando...

- **Todo navegador vem com JavaScript embutido**
- **Logo vamos usá-lo em conjunto com nossas páginas WEB.**
- **Embutido em uma tag SCRIPT ou em arquivos .js.**



Sintaxe JavaScript

- **A sintaxe de programação é o conjunto de regras que definem como escrever código em uma linguagem de programação.**
- **Toda linguagem de programação tem a sua!**

Sintaxe JavaScript



```
if condition1:
    # Code
elif condition2:
    # Code
elif condition3:
    # Code
else:
    # Code
```

```
if (condition1) {
    // Code
} else if (condition2) {
    // Code
} else if (condition3) {
    // Code
} else {
    // Code
}
```

Sintaxe JavaScript

- Enquanto Python usa indentação e dois pontos para definir um bloco de código;
- JS usa chaves com indentação opcional.



```
for i in range(n):  
    # Code
```



```
for (var i = 0; i < n; i++) {  
    // Code  
}
```

Sintaxe JavaScript



```
x = 5
```

```
var x = 5;  
let x = 5;
```

Sintaxe JavaScript

- As duas linguagens usam o sinal de igual '=' para atribuir valores a variáveis.



```
x = 5
```

```
var x = 5;  
let x = 5;
```

Sintaxe JavaScript

- Todas as linhas de um bloco JS terminal com ponto-e-vírgula `;`.



```
x = 5
```

```
var x = 5;  
let x = 5;
```


Sintaxe JavaScript

- Existem três tipos de variáveis em JS.



```
x = 5
```

```
var x = 5;  
let x = 5;
```

Variáveis em JavaScript

Variáveis em JavaScript

- Variáveis declaradas com **var** são usadas em um escopo global ou de função.
- Variáveis declaradas com **let** são usadas em um contexto de bloco.
- Variáveis declaradas com **const** tem o mesmo escopo de var porém não podem ser alteradas no programa.

Variáveis em JavaScript

```
var nome = "João"; //contexto global
```

```
let nome = "João"; //contexto local em um bloco
```

```
const PI = 3.14159; // contexto global que não  
pode ser alterado
```



Mas o que é escopo?

Escopo de variável

Escopo de uma variável

- Escopo, essencialmente, significa onde essas variáveis poderão ser utilizadas. Declarações com `var` tem escopo global ou escopo de função/local.
- **var** tem escopo de função quando é declarado dentro de uma função. Isso significa que ele está disponível e pode ser acessado somente de dentro daquela função.

Escopo de uma variável

- Escopo, essencialmente, significa onde essas variáveis poderão ser utilizadas. Declarações com `var` tem escopo global ou escopo de função/local.
- **var** tem escopo de função quando é declarado dentro de uma função. Isso significa que ele está disponível e pode ser acessado somente de dentro daquela função.

Escopo de uma variável

- Aqui, **greeter** tem um escopo global, pois existe fora de uma função, enquanto **hello** tem escopo de função.
- Por isso, não podemos acessar a variável **hello** fora de uma função. Assim, se fizermos isso:

```
var greeter = "hey hi";  
  
function newFunction() {  
    var hello = "hello";  
}
```

Escopo de uma variável

- Teremos um erro resultante do fato de **hello** não estar disponível fora da função.

```
var tester = "hey hi";

function newFunction() {
    var hello = "hello";
}

console.log(hello); // erro: hello não está definido
```

Tipos de dados

Tipos de Dados

- Toda linguagem de programação possui tipos de dados;
- Esses tipos são usados para orientar o tipo de variável.
- Imagine que você deseja somar dois números $1 + "1"$.
- Sem a definição de tipos poderíamos atribuir qualquer valor deixando todo o trabalho de "saber" o que fazer com a máquina.
- Isso exigiria um enorme processamento.

Tipos de Dados

- Separando tipos de dados a linguagem pode entender então quais operações são possíveis.
- Exemplo:
 - "Abacaxi" + 1, resultaria em "Abacaxi1", pois tudo entre aspas "" é considerado uma String ou cadeia de caracteres.
 - "1" + 1, resultaria em 11.
 - 1 + 1, em 2, pois, ambos são números inteiros e aritmeticamente calculáveis.

Tipos de Dados

- Boolean: Valores verdadeiro (true) ou falso (false)
- Null: Variável nula
- Undefined: Variável indefinida
- Number: Variável numérica, podendo ser inteira ou de ponto flutuante
- BigInt: Números inteiros de comprimento arbitrário
- String: Sequência de caracteres, que podem incluir letras, números, símbolos e espaços
- Symbol: Identificadores exclusivos
- Object: Conjunto de atributos aninhados a uma variável

Crie um programa

- Crie um programa para armazenar o nome, idade, altura e peso de três pessoas.

```
P1_nome = "Rodrigo"; //string
```

```
P1_idade = 16; // inteiro
```

```
P1_altura = 1.74; // flutuante
```

```
P1_peso = 77.9; // flutuante
```

Acrescente a tag Script no body

A princípio vamos escrever dentro de uma tag script

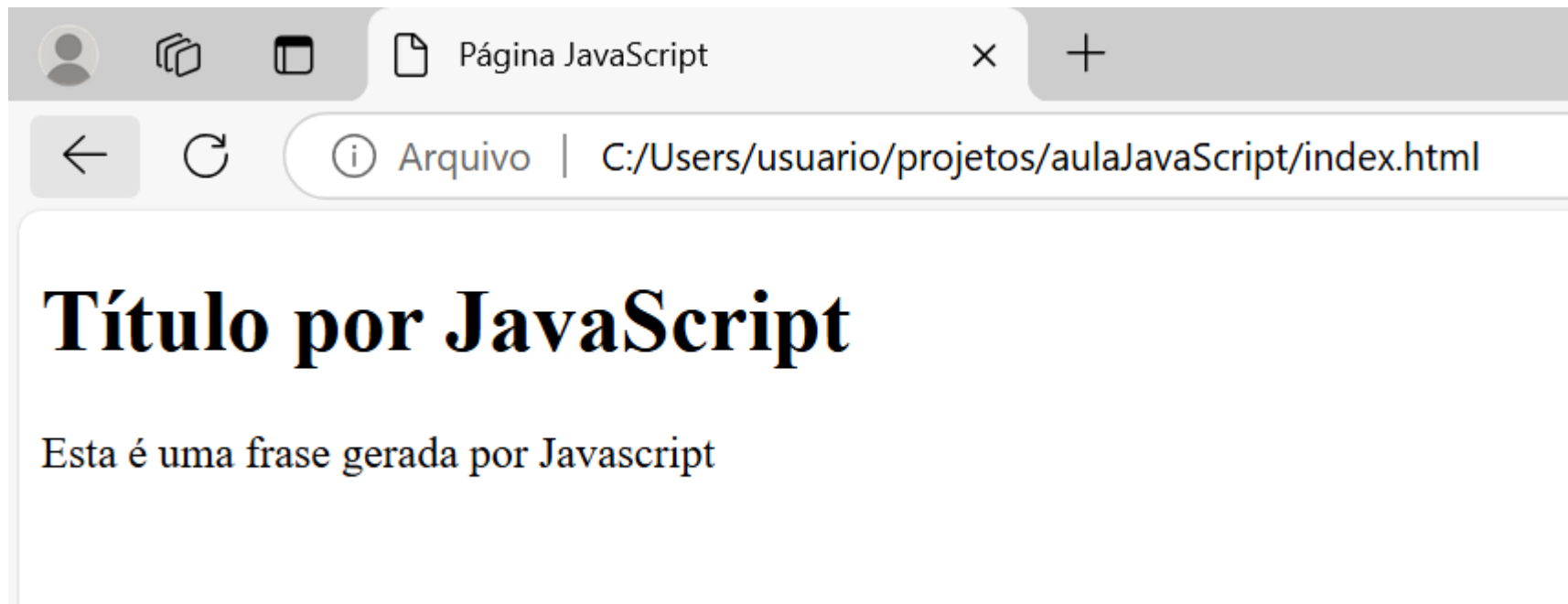
```
8   <body>
9       <script>
10          // Muda título
11          let h1 = document.createElement('h1');
12          h1.textContent = "Título por JavaScript";
13          document.body.appendChild(h1);
14          //adiciona uma frase
15          let p = document.createElement('p');
16          p.textContent = "Esta é uma frase gerada por Javascript";
17          document.body.appendChild(p);
18       </script>
19   </body>
```


Comentários

- `//` Comentário de uma linha
- `/*` Comentário de mais de uma linha `*/`

Rode o código com <F5>

- Veja o resultado!



Explicando....

- **A TAG SCRIPT nos permite “programar” em uma linguagem dentro do HTML, uma linguagem de marcação.**
- **Linha11: Declaramos uma variável h1 que cria uma tag.**
- **Linha12: Adicionamos um valor a esse elemento**
- **Linha13: Inserimos esse elemento HTML na página.**
- **Linhas Seguintes repetimos o processo para uma TAG P.**

Que confusão

- Calma.... rs
- **JavaScript é uma linguagem orientada a EVENTOS**
- **E orientada a objetos**
- **Em linguagem orientada a objetos temos recursos que a programação estrutural não nos permite.**



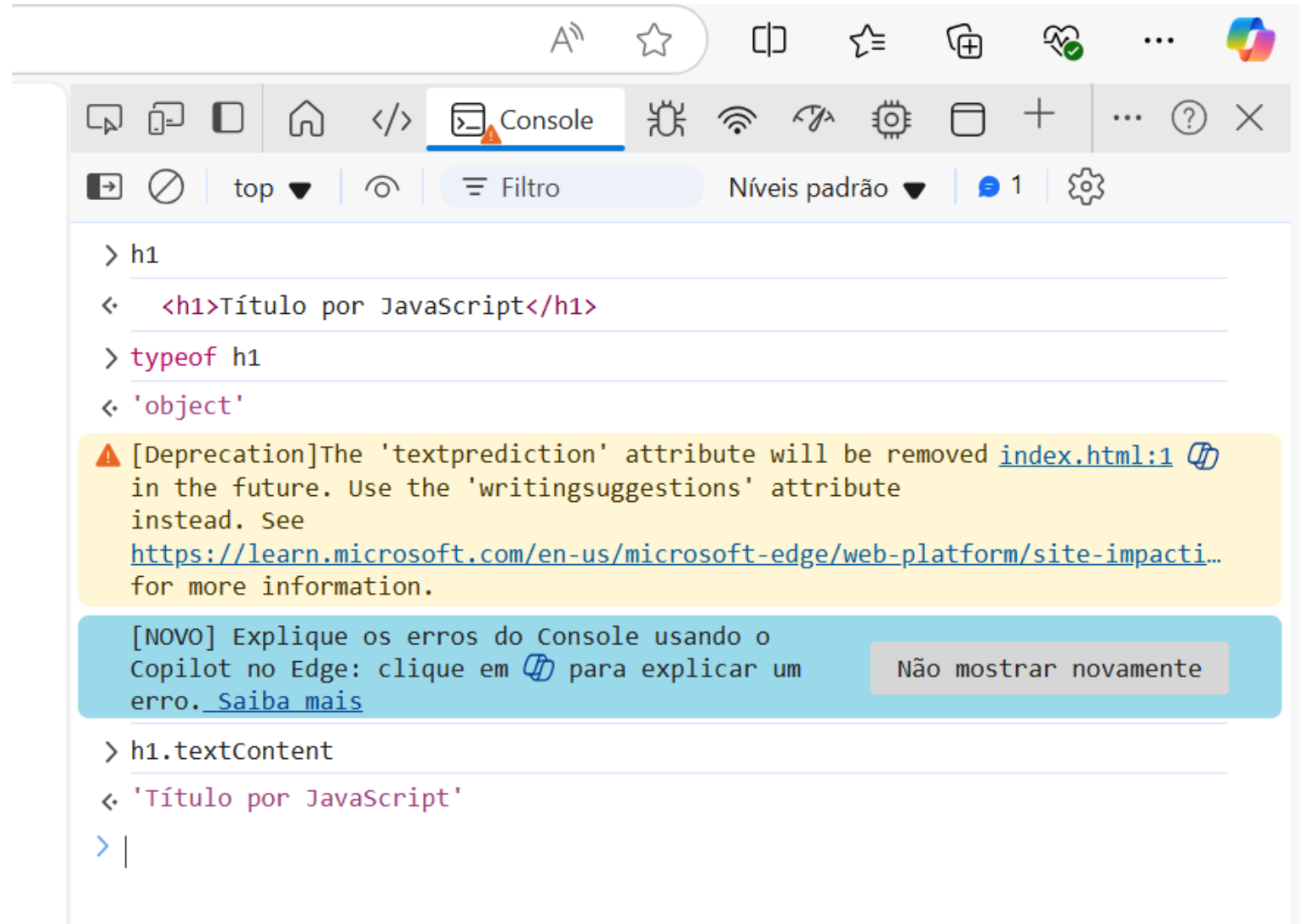
Orientação a Objetos na prática

- **Imagine que você possui uma classe chamada pessoa:**
- **Ela tem os atributos, nome, telefone e idade...**
- **Esses atributos são como variáveis embutidas nessa classe pessoa.**
- **Logo se quisermos modificar no nome dessa pessoa, precisamos invoca-la e acessar seu atributo usando `.` ponto.**
- **pessoa.nome = "Rodrigo";**

Usando Orientação a Objeto

- **Em Javascript assim como no Python, tudo é objeto.**
- **Isso quer dizer que variáveis que criamos (construímos) possuem métodos e atributos.**
- **Na Linha 11, por exemplo, instanciamos uma classe do tipo Element, que podemos alterar suas características usando `.` (ponto).**

Abra o
console de
depuração na
página, digite
os
commandos...



Alterações em execução

- **Tente digitar a instrução no console de depuração.**
- **`h1.style.color = "red";`**

Modificar o HTML

- A principal função do JavaScript é permitir fazer alterações na página em tempo de execução.
- Isso quer dizer que podemos melhorar a experiência do usuário com nossa página.
- Há uma enorme gama de funções que podemos fazer e todas estão disponíveis nas documentações oficiais.
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

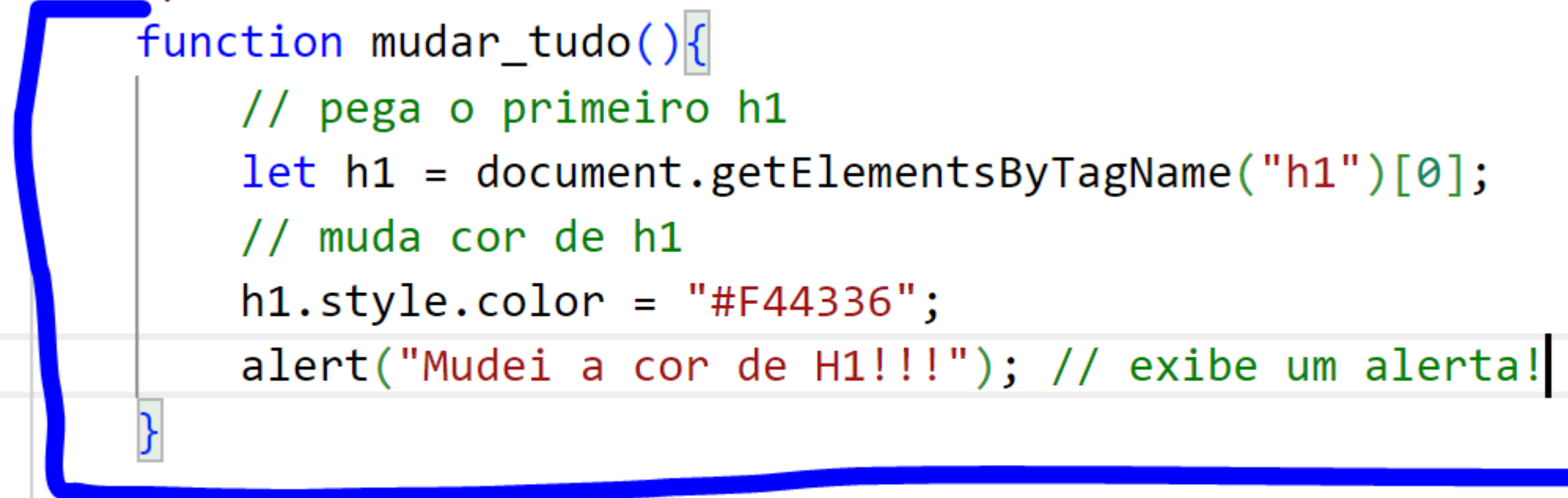
Eventos com botões

- Outra forma de deixar nossa página mais dinâmica é usando eventos e botões.
- Vamos adicionar um botão no local abaixo.

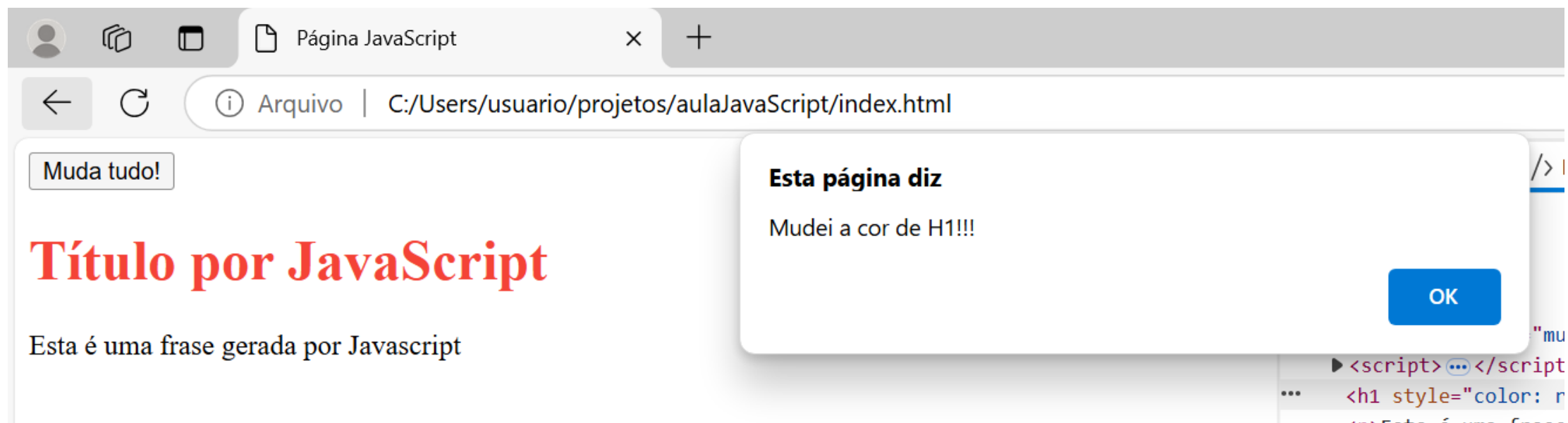
```
7      </thead>
8      <body>
9
10         <button onclick="mudar_tudo();" >Muda tudo!</button>
11
12         <script>
13             // Muda título
```

Vamos criar a função mudar_tudo()

```
12  <script>
13      function mudar_tudo(){
14          // pega o primeiro h1
15          let h1 = document.getElementsByTagName("h1")[0];
16          // muda cor de h1
17          h1.style.color = "#F44336";
18          alert("Mudei a cor de H1!!!"); // exibe um alerta!
19      }
20
21      // Muda título
22      let h1 = document.createElement('h1');
```



Atualize a página e clique no botão



Exemplo Game Coragem

- Baixe o arquivo gameJS.zip

Acessando elementos de página

Conectando JS ao HTML

- JavaScript foi criado para dinamizar páginas HTML através de eventos.
- Esses eventos podem ser feitos através de:
 - Cliques de botões;
 - Pressionando teclas;
 - Movendo o mouse;
 - ETC;
- Basicamente podemos conectar uma função a um evento provocado pelo usuário na tela.
- E assim responder o usuário com alguma ação.

getElementById

- `getElementById`: usado para conectar tags com elemento id;

- Exemplo no HTML:

```
<button id="btnSalvar">Salvar</button>
```

Exemplo no JS:

```
btnSalvar = document.getElementById("btnSalvar");
```


Estruturas de decisão

IF-Else

```
if (condicao) {  
    decisã01  
} else {  
    decisã02  
}
```

JS

```
function checkValue(a, b) {  
    if (a === 1)  
        if (b === 2)  
            console.log("a is 1 and b is 2");  
    else  
        console.log("a is not 1");  
}
```

Switch-Case

```
switch (expression) {  
  case caseExpression1:  
    statements  
  case caseExpression2:  
    statements  
  // ...  
  case caseExpressionN:  
    statements  
  default:  
    statements  
}
```

```
switch (expr) {  
  case "Oranges":  
    console.log("Oranges are $0.59 a pound.");  
    break;  
  case "Apples":  
    console.log("Apples are $0.32 a pound.");  
    break;  
  case "Bananas":  
    console.log("Bananas are $0.48 a pound.");  
    break;  
  case "Cherries":  
    console.log("Cherries are $3.00 a pound.");  
    break;  
  case "Mangoes":  
  case "Papayas":  
    console.log("Mangoes and papayas are $2.79 a pound.");  
    break;  
  default:  
    console.log(`Sorry, we are out of ${expr}.`);  
}
```

If de uma linha

Estruturas de Repetição

While

JS

```
let n = 0;  
let x = 0;  
  
while (n < 3) {  
  n++;  
  x += n;  
}
```

For

Semelhante ao While, cria um contador dentro do laço

```
1 let str = '';
2
3 for (let i = 0; i < 9; i++) {
4   str = str + i;
5 }
6
7 console.log(str);
8 // Expected output: "012345678"
9
```

Foreach

Usado para iterar com listas e objetos

JavaScript Demo: Statement - For...In

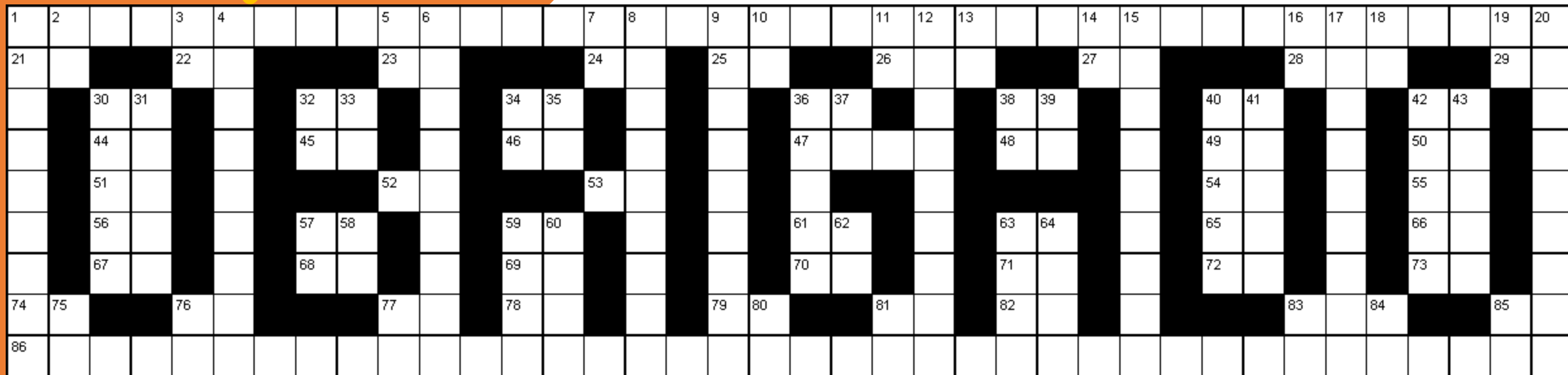
```
1  const object = { a: 1, b: 2, c: 3 };
2
3  for (const property in object) {
4      console.log(`${property}: ${object[property]}`);
5  }
6
7  // Expected output:
8  // "a: 1"
9  // "b: 2"
10 // "c: 3"
```


Pronto!

Ainda há muito o que aprender sobre JavaScript, para nos aperfeiçoar é fundamental a prática de exercícios e situações que nos permitam evoluir.

Documentação de referência

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements>



Esta Foto de Autor Desconhecido está licenciado em [CC BY-NC](#)