



Técnico em Desenvolvimento de Sistemas

Professor Thiago Suzuqui



Introdução ao **JavaScript**



JavaScript

- Ao carregar uma página na internet, seu computador faz um download de uma pasta de arquivos, incluindo um arquivo **HTML**;
- O **HTML** é capaz de carregar **JavaScript** com uma tag chamada **script**, que pode chamar um script da mesma pasta ou de outro lugar;



index.html code

```
<html>
<head>
  <title>Meu Site</title>
</head>
<body>
  <script></script>
</body>
</html>
```

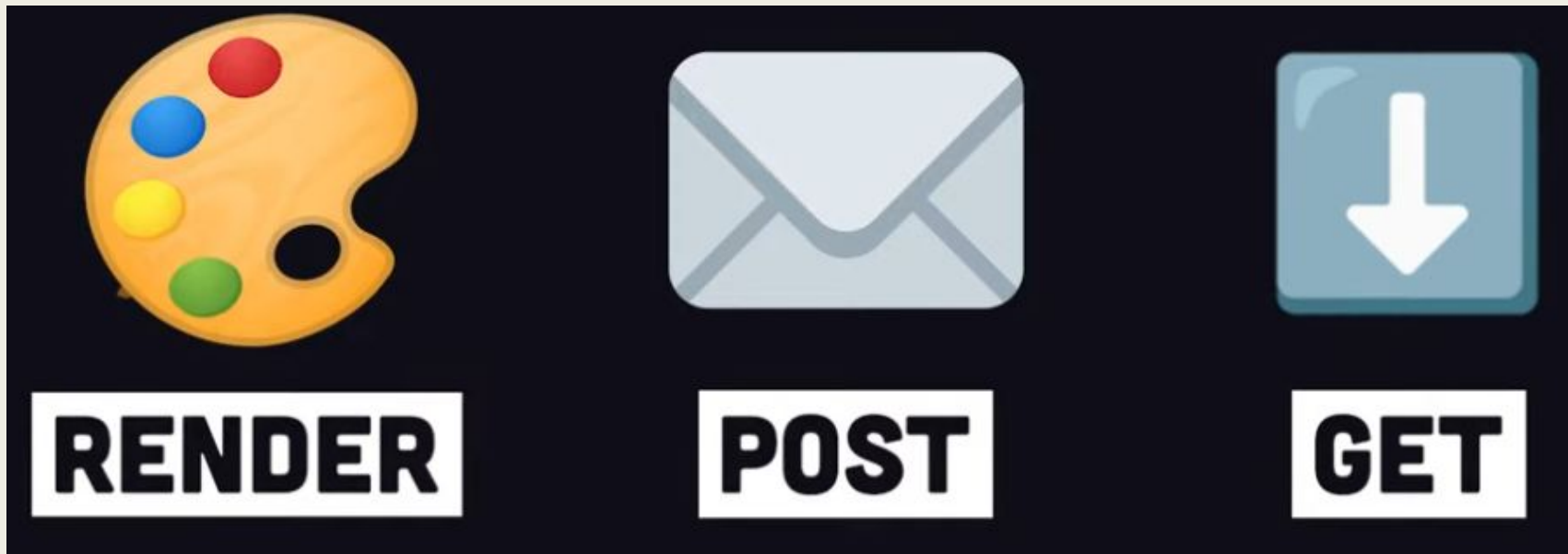
JavaScript

- Abreviado como **JS**, o JavaScript é uma linguagem **Client-Side**;
- Ou seja, para executar, é necessário um cliente e só após o código estar baixado completamente no computador ele é executado;
- Código **Client-Side** pode mudar a aparência dos itens na tela, enviar dados ou buscar por mais dados no servidor;
- OBS.: é uma **linguagem de programação interpretada**.



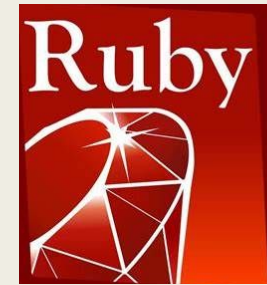
Client-Side

- O processamento das coisas acontece no lado do cliente;
- Exemplo de processos comuns que podem ser feitas no lado do cliente:



Linguagem Interpretada

- O código é lido linha por linha pelo interpretador e executado;
- Antigamente, eram significativamente mais lentas do que as linguagens compiladas;
- Com o desenvolvimento da compilação **just-in-time(JIT)** linguagens interpretadas vem se tornando cada vez mais rápidas;
- Vantagens:
 - tendem a ser mais flexíveis;
 - código não depende da plataforma
 - tamanho reduzido de programa



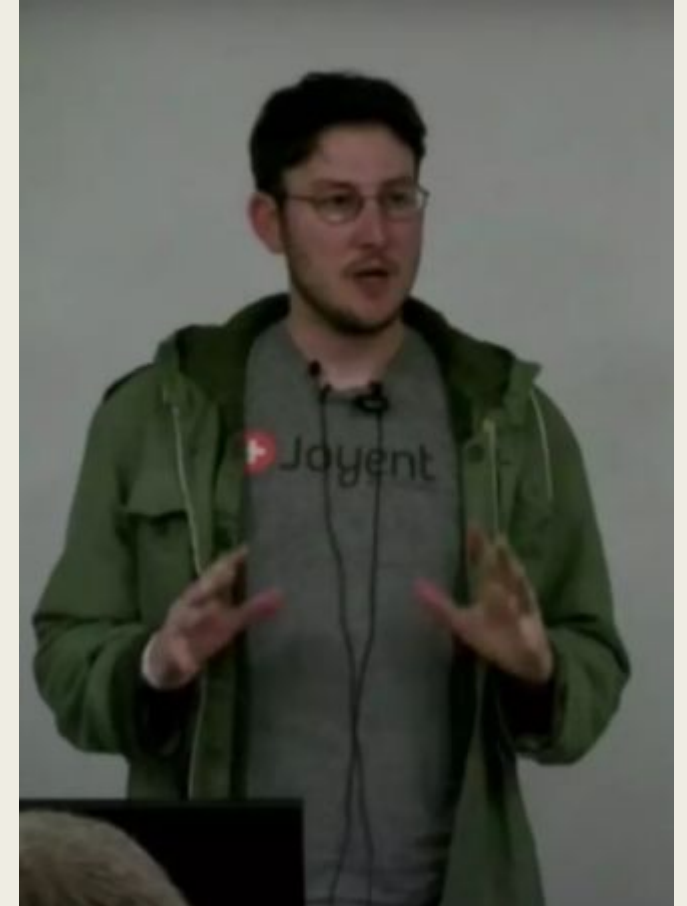
Casos de uso

- Atualmente, o JavaScript é capaz de realizar uma ampla gama de funções, exemplos:
 - desenvolvimento de servidores;
 - aprendizado de máquina;
 - desenvolvimento de Jogos;
- Uma das coisas que possibilita isso é um cliente chamado:



NodeJS


- Criado por Ryan Dahl;
- NodeJS é um ambiente de execução para o JavaScript escrito em C++;
- Permite execução nos 3 principais SO's:
 - Linux
 - Windows
 - Mac
- Seu interpretador é o V8, originalmente desenvolvida pelo Google e utilizado no Chrome;
- Proporciona um ambiente de alta performance para desenvolvimento de soluções Back-End

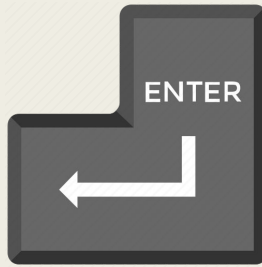


Mão no **código**




Hello World

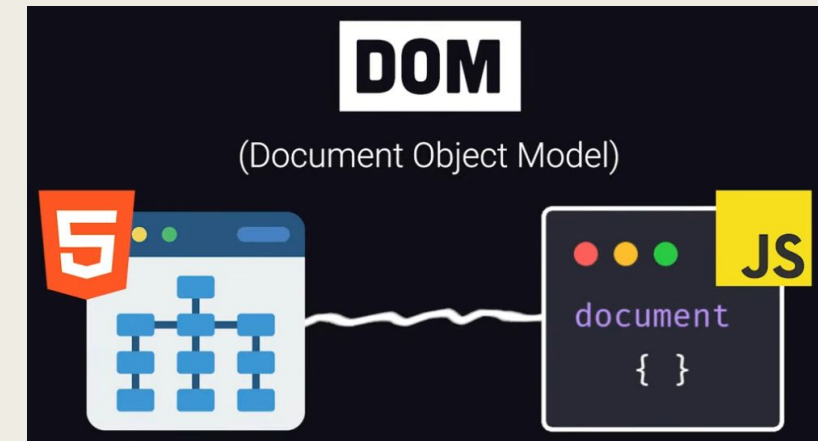
- Vamos fazer um hello world com JS!!!
- Aperte 
- Escreva `console.log('Hello World');`
- Aperte:



```
> console.log('Hello World')  
Hello World  
← undefined
```

Document Object Model (DOM)

- É uma interface que representa a estrutura do HTML;
- É através da manipulação deste objeto que o JavaScript pode interagir com a página;
- É possível:
 - selecionar elementos específicos;
 - alterar elementos;
 - adicionar eventos(ex: ao clicar em um botão uma notificação é exibida);
 - adicionar elementos à página;
- Exemplo de manipulação de DOM:
 - acesse <https://www.google.com/>
 - abra o console com 
 - digite: **document.body.style.background='red'**



Mão no código

F12

Tipos de Variáveis

Primitivos

booleanos
strings
números

Compostos

arrays []
objetos { }

	const	let	var
escopo global	✗	✗	○
escopo de função	○	○	○
escopo de bloco	○	○	✗
pode ser reatribuída	✗	○	○

ARRAYS

```
const arr = [1, 2, 3];
```

```
console.log(arr[0])  
// → 1
```

OBJETOS

```
const obj = {  
  nome: "Nickolas",  
  idade: 23,  
};
```

```
console.log(obj.nome)  
// → 'Nickolas'
```

Boas práticas:

- não utilizar o **var** pois ela pode ser reatribuída em **qualquer parte do código**;
- utilizar o máximo de **const** que puder;
- usar **let** em casos que a variável precise mudar de valor e há uma certeza de que o usuário não conseguirá reatribuí-la;

Sintaxe

- se assemelha a outras linguagens, incluindo estruturas de controle como:

- loops

```
for (let index = 0; index < bound; index++) {  
  ...  
}
```

```
while (0) {  
  ...  
}
```

- condicionais

```
> if () {  
  ...  
} else if () {  
  ...  
} else {  
  ...  
}
```

- Dicas: usar **for-in** em **objetos** e **for of** em **arrays**

```
const nums = [1, 2, 3];  
for (num of nums) {  
  console.log(num);  
}
```

```
for (key in obj) {  
  console.log(key, obj[key]);  
}
```

Sintaxe

- JS também tem em comum com outras linguagens as funções, que podem ser declaradas conforme o exemplo:

```
> function somar(a,b) {  
    return a+b;  
}  
↵ undefined  
> somar(7,98)  
↵ 105
```

- Há também os métodos, que são funções pertencentes a uma determinada classe/tipo.
 - string tem um método de colocar todos os caracteres em caixa alta:

```
> 'javascript'.toUpperCase()  
↵ 'JAVASCRIPT'
```

Sintaxe

- **Arrow Functions:**

- outro método de declarar funções:

```
> let somar = (a,b) =>{  
    return a+b;  
}  
↵ undefined  
  
> somar(1,6)  
↵ 7
```

- podem ter retorno implícito se for declarada em uma única linha:

```
> let somar = (a,b)=> a+b  
↵ undefined  
  
> somar(9,88)  
↵ 97
```

Frameworks



Frameworks

- Para criar sites bonitos e responsivos, por exemplo, é necessário uma grande cooperação entre HTML, CSS e JS;
- Ao desenvolver uma solução para um projeto, não por que reinventar a roda, é possível reutilizar estruturas já criadas antes para o novo projeto;
- E isso deu a luz aos **Frameworks**:
 - são bases de códigos que são uma fundação para projetos(tanto para **front-end** quanto para **back-end**)



Express

NEXT.js

PHASER

Vue.js

VUETIFY

NUXTJS

ANGULAR

SVELTE

nest

Gatsby

Continue estudando...

- DUCKETT, Jon. JAVASCRIPT & JQUERY Interactive Front-End Web Development. John Wiley & Sons, Inc., Indianapolis, Indiana ISBN: 978-1-118-53164-8.
- Referência JS W3 <https://www.w3schools.com/js/>
- Referência JS MDN <https://developer.mozilla.org/en-US/docs/Web/JavaScript>