EFI FUCHS

JAVASCRIPT

LINGUAGEM E APLICAÇÕES

CONTEXTUALIZANDO

- Introdução ao JavaScript:
  - O que é JavaScript
  - História e evolução
  - O papel do JavaScript no desenvolvimento web
- Sintaxe Básica:
  - Variáveis e tipos de dados
  - Operadores
  - Estruturas de controle (if, else, while, for)
- Funções:
  - Declaração e chamada de funções
  - Parâmetros e retorno
  - Escopo de variáveis
- Arrays e Loops:
  - Arrays e métodos úteis
  - Loops (for, forEach, map)
  - Trabalhando com objetos
- Assíncrono em JavaScript:
  - Callbacks
  - Promises
  - Async/await
-

CARACTERÍSTICAS

Alto nível

- Interpretada.
- Estruturada
- Tipagem Dinâmica.
- Tipagem fraca.
- Multiparadigma

Para a web?

- Roda nativamente em navegadores. (lol)

Hey can you compute this for me?

Anything for you babe <3

Hey honey can you ask Chad if he can compute this for me. I'm in no hurry or anything please

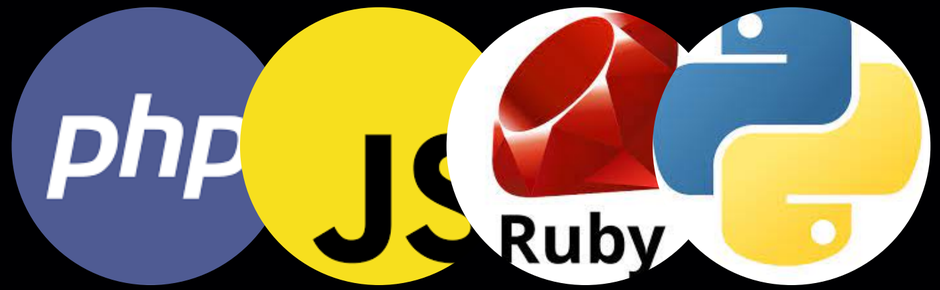
INTERPRETADA

FORTES

- Portável.
- Fácil *Debugging*.
- Direta execução sem compilação.
- Rápido desenvolvimento.

FRACOS

- Precisa de um interpretador.
- Lento.
- *Source code* sempre público (e isso é ruim?).



ESTRUTURADA

TL;DR

- Sequência, Seleção e Repetição.
- Blocos de Código Delimitados.
- !!! **Ausência de Gotos!!! .**
- Legibilidade e Manutenção.
- Controle de Erros

TIPAGEM

TIPAGEM

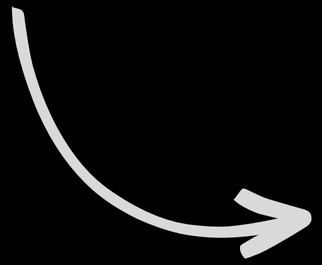
FORTEMENTE TIPADAS:

- Possuem tipos! Propriamente ditas ;D
- Geralmente estaticamente tipadas.

TIPAGEM

FORTEMENTE TIPADAS:

- Possuem tipos! Propriamente ditas ;D
- Geralmente estaticamente tipadas.

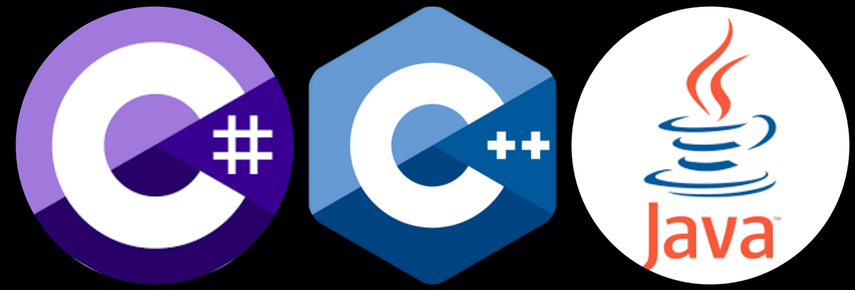


```
int num1 = 10;
String num2 = "5";
int num3 = num1 * num2;
// erro
```

TIPAGEM

FORTEMENTE TIPADAS:

- Possuem tipos! Propriamente ditas ;D
- Geralmente estaticamente tipadas.

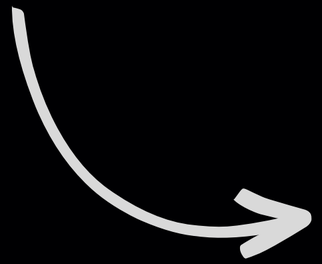


```
int num1 = 10;
String num2 = "5";
int num3 = num1 * num2;
// erro
```

TIPAGEM

FORTEMENTE TIPADAS:

- Possuem tipos! Propriamente ditas ;D
- Geralmente estaticamente tipadas.

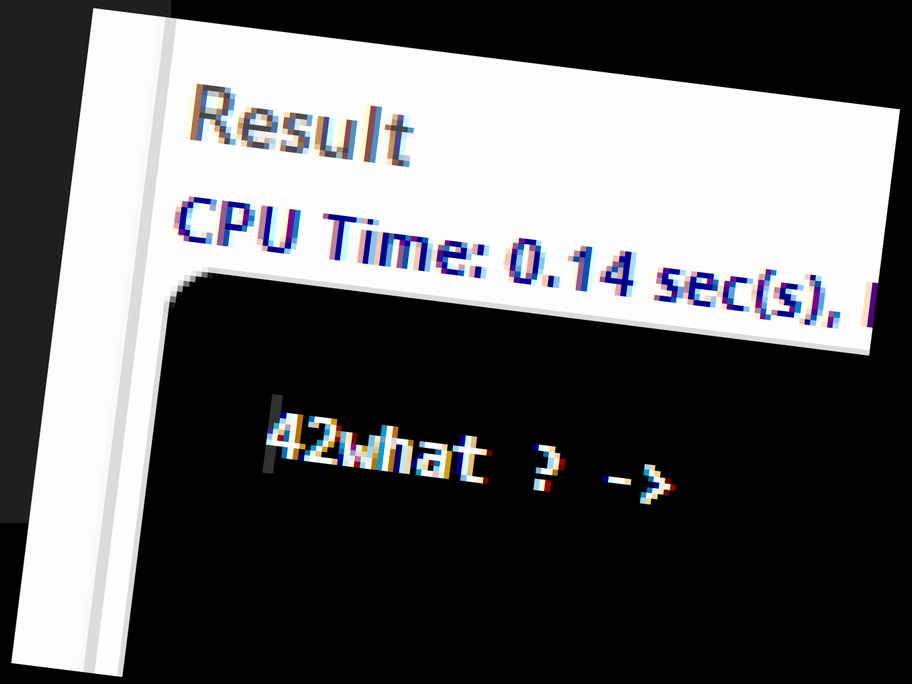


```
int num1 = 10;
String num2 = "5";
int num3 = num1 * num2;
// erro
```


```
class HelloWorld {
    public static void main(String[] args) {
        int value = 42;
        String aaa = "what ? -> ";

        String result = value + aaa;
        System.out.println(result);
    }
}
```

TIPAGEM

```java
class HelloWorld {
    public static void main(String[] args) {
        int value = 42;
        String aaa = "what ? -> ";

        String result = value + aaa;
        System.out.println(result);
    }
}
```



TIPAGEM

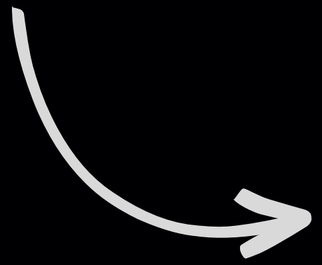
FORTEMENTE TIPADAS E DINÂMICAS

- Não exigem declarações de tipos de dados.
- Podem escolher que tipo utilizar dinamicamente para cada variável

TIPAGEM

FORTEMENTE TIPADAS E DINÂMICAS

- Não exigem declarações de tipos de dados.
- Podem escolher que tipo utilizar dinamicamente para cada variável



```
var1 = 120
var2 = "10"
var3 = var1 * var2
# Erro
```

TIPAGEM

FORTEMENTE TIPADAS E DINÂMICAS

- Não exigem declarações de tipos de dados.
- Podem escolher que tipo utilizar dinamicamente para cada variável

```
var1 = 120
var2 = "10"
var3 = var1 * var2
# Erro
```

```
var1 = 120
var2 = "10"
var2 = 10
var3 = var1 * var2
# resultado: 1200
```

TIPAGEM

FORTEMENTE TIPADAS E DINÂMICAS

- Não exigem declarações de tipos de dados.
- Podem escolher que tipo utilizar dinamicamente para cada variável

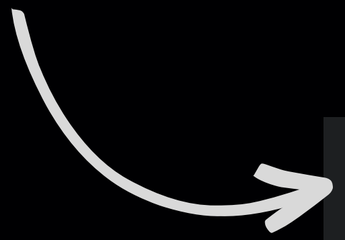

```
var1 = 120
var2 = "10"
var3 = var1 * var2
# Erro
```

```
var1 = 120
var2 = "10"
var2 = 10
var3 = var1 * var2
# resultado: 1200
```

TIPAGEM

FRACAMENTE TIPADAS

- O tipo atribuído não é muito importante...
- Conversões implícitas.
- Operações de diversos tipos.



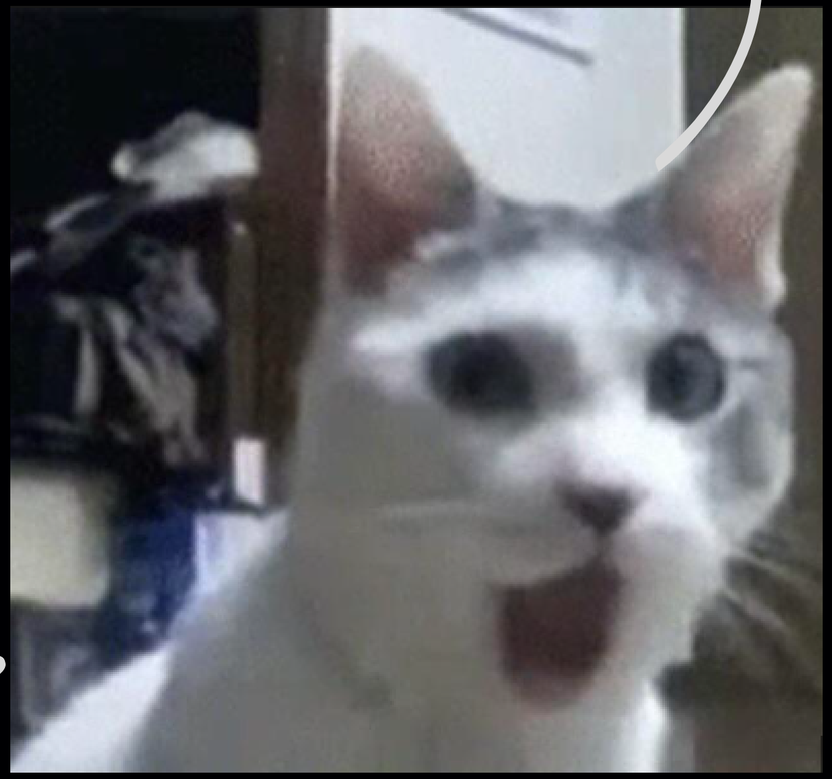
```
var1 = 120
var2 = "10"
var3 = var1 * var2
// resultado: 1200
```

TIPAGEM

FRACAMENTE TIPADAS

- O tipo atribuído não é muito importante...
- Conversões implícitas.
- Operações de diversos tipos.

```
var1 = 120
var2 = "10"
var3 = var1 * var2
// resultado: 1200
```



JAVASCRIPT É ESTRANHO

TIPAGEM

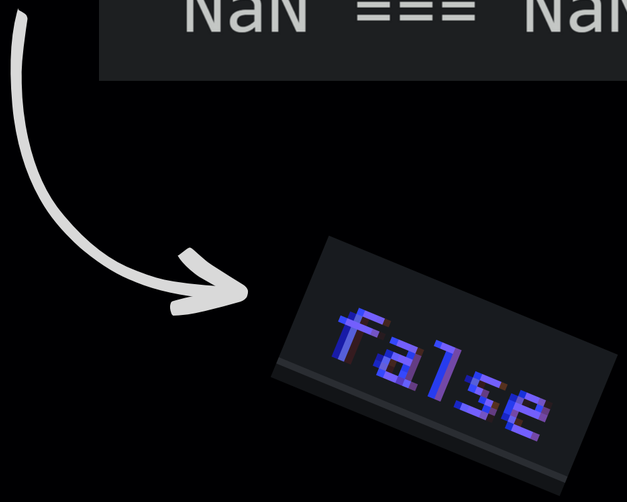
JAVASCRIPT É ESTRANHO

```
NaN === NaN; // ??
```

TIPAGEM

JAVASCRIPT É ESTRANHO

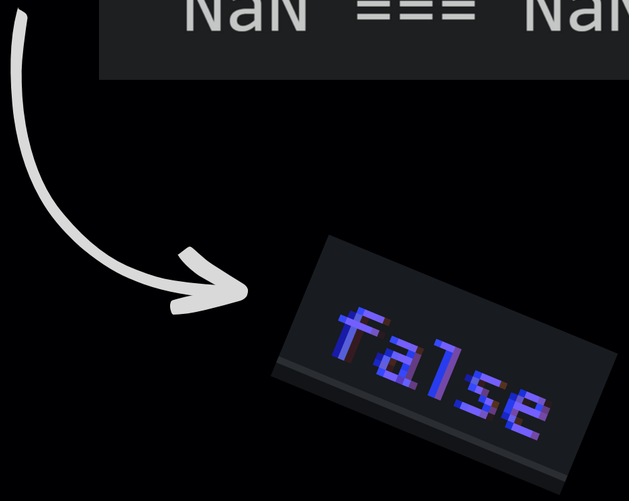
```
NaN === NaN; // ??
```



TIPAGEM

JAVASCRIPT É ESTRANHO

```
NaN === NaN; // ??
```




```
(true + true) * (true + true) - true;
```

TIPAGEM

JAVASCRIPT É ESTRANHO

```
NaN === NaN; // ??
```


```
(true + true) * (true + true) - true;
```

TIPAGEM

JAVASCRIPT É ESTRANHO

```
var1 = (0 == "0");
console.log(var1); // true
```

TIPAGEM

JAVASCRIPT É ESTRANHO

```
var1 = (0 == "0");
console.log(var1); // true
```

```
var1 = (0 == []);
console.log(var1); // true
```

TIPAGEM

JAVASCRIPT É ESTRANHO

```
var1 = (0 == "0");
console.log(var1); // true
```

```
var1 = (0 == []);
console.log(var1); // true
```

```
var1 = ("0" == []);
console.log(var1);
```

TIPAGEM

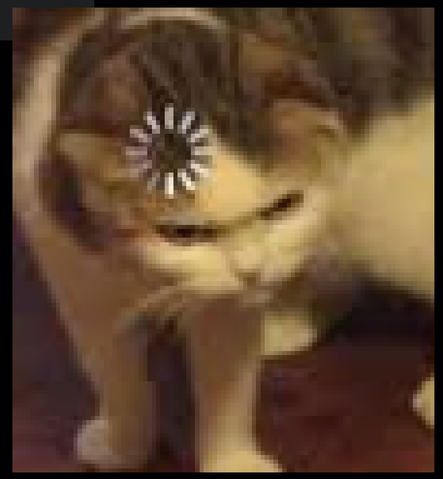
JAVASCRIPT É ESTRANHO

```
var1 = (0 == "0");
console.log(var1); // true
```

```
var1 = (0 == []);
console.log(var1); // true
```

```
var1 = ("0" == []);
console.log(var1);
```

```
false
```



BREVE HISTÓRIA

CRIAÇÃO

ORIGEM

- Criado por Brendan Eich em 1995 na Netscape
- Mocha => LiveScript => JavaScript

MOTIVAÇÕES

- Interatividade na web
- Fácil uso de linguagem

INFLUENCIAS

- Sintaxe semelhante a C e JAVA
- Funcionalidades inspiradas no SCHEME

ECMAScript

- Especificação do JavaScript à ECMA International em 1996

MICROSOFT...

MICROSOFT...

1996:

- VBScript
- JScript

MICROSOFT...

COMUNIDADE

2005:

- JQuery
- Dojo
- Toolkit
- MooTools

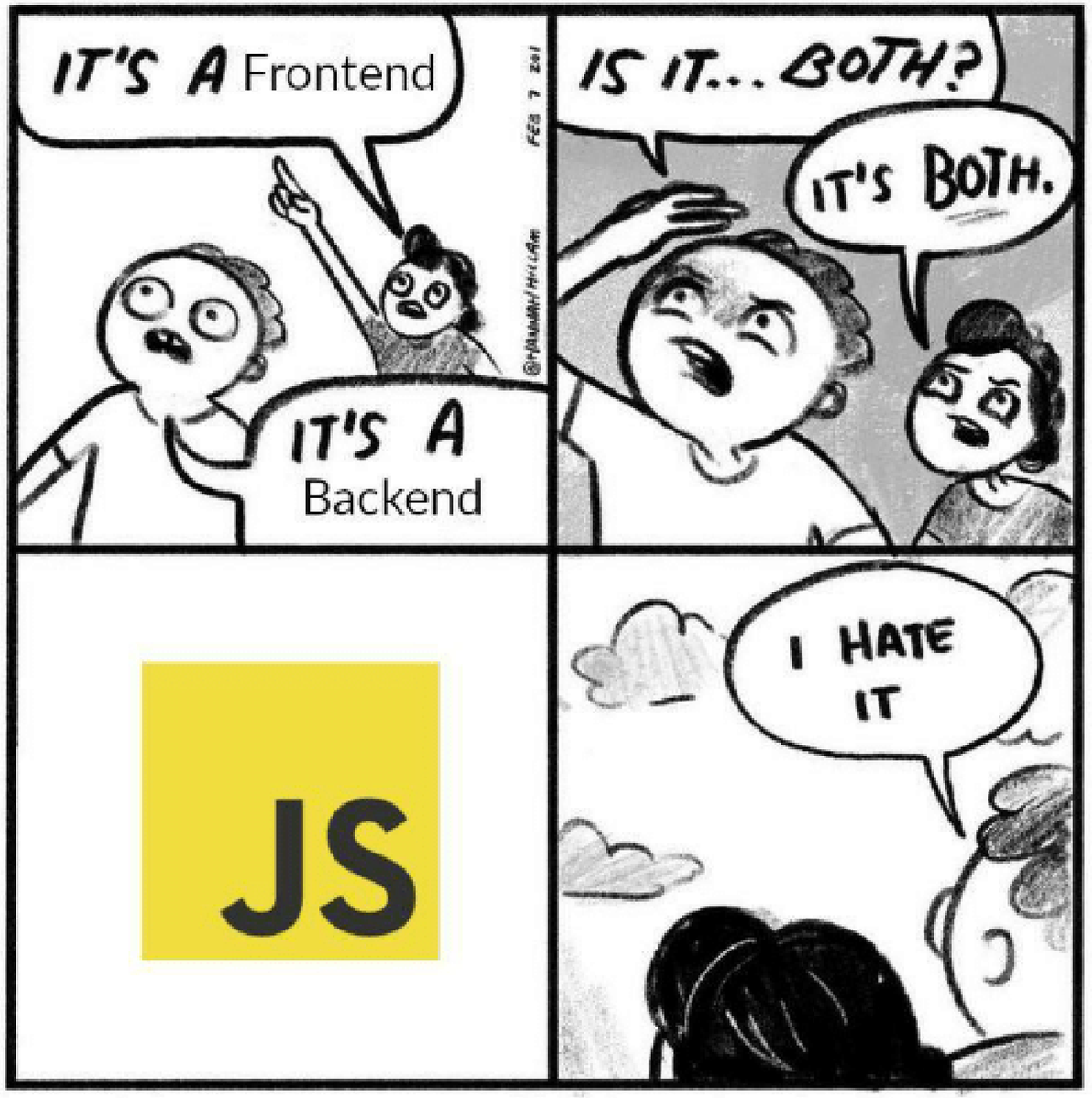
COMUNIDADE

2005:

- JQuery
- Dojo
- Toolkit
- MooTools

NA WEB

PARA A WEB



It's a Meteor !


HTML
JS


HTML
CSS
JS
Content
Structual
Style
Presentational
Behavioral

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

MANIPULAÇÃO

- DOM
- Elementos
- Dinâmicidade
- Sem reload
- Animações
- efeitos

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

MANIPULAÇÃO

- DOM
- Elementos
- Dinâmicidade
- Sem reload
- Animações
- efeitos

VALIDAÇÃO

- Formulários
- Regras simples
- Formatação
- Pré validação

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

MANIPULAÇÃO

- DOM
- Elementos
- Dinâmicidade
- Sem reload
- Animações
- efeitos

VALIDAÇÃO

- Formulários
- Regras simples
- Formatação
- Pré validação

REQ. ASSÍNCRONAS

- Ajax

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

MANIPULAÇÃO

- DOM
- Elementos
- Dinâmicidade
- Sem reload
- Animações
- efeitos

VALIDAÇÃO

- Formulários
- Regras simples
- Formatação
- Pré validação

REQ. ASSÍNCRONAS

- Ajax

API's

- Serviços
- Dados
- Experiências de usuário

INTERATIVIDADE

- Botões
- Eventos
- Entradas diversas

MANIPULAÇÃO

- DOM
- Elementos
- Dinâmicidade
- Sem reload
- Animações
- efeitos

VALIDAÇÃO

- Formulários
- Regras simples
- Formatação
- Pré validação

REQ. ASSÍNCRONAS

- Ajax

API's

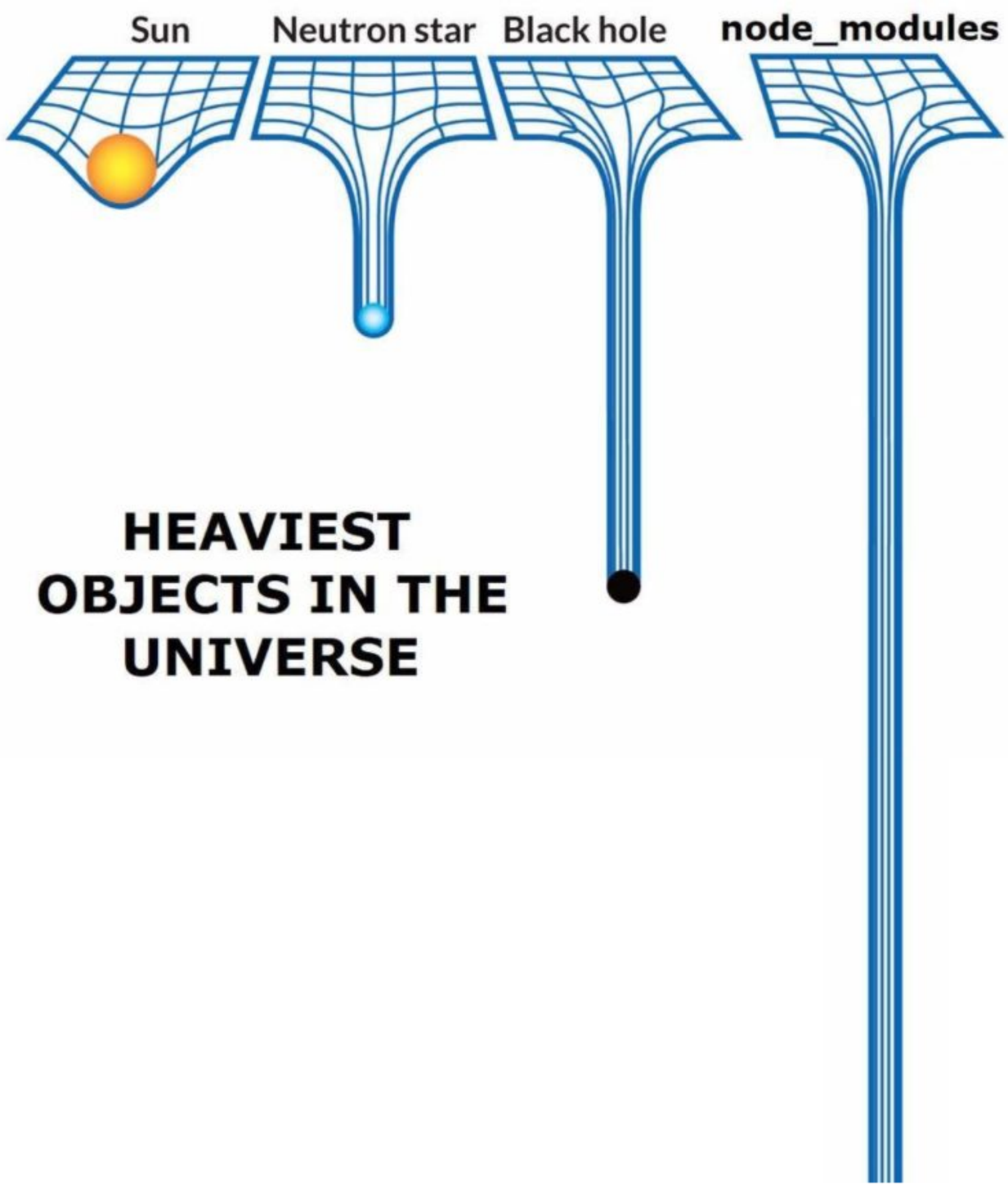
- Serviços
- Dados
- Experiências de usuário

FRAMEWORKS

- Componentização
- Abstração
- Angular
- React
- Vue.*js*

APPS / BACKEND NO JS?

AINDA ESTAMOS FALANDO DE UMA LINGUAGEM DE SCRIPT CERTO?



NODE.JS

CARACTERÍSTICAS

- CrossPlatform
- OpenSource
- Server environment
- Sem navegador
- Arquitetura de eventos


npm

```
var maxTummyPats = 2;
var hiss = false;

if maxTummyPats > 2{
    hiss
}
```

SINTAXE & FUNC

```
// Este comentário ocupa uma única linha

/* Já este comentário
é mais longo e utiliza
várias linhas */
```

```
let name = "Alice";
const age = 30;
var score = 85;

let isStudent = true;
let fruits = ["apple", "banana", "orange"];

console.log(name, age, score, isStudent, fruits);
```

```
let nome = "João";
const idade = 25;
var saldo = 100.50;
```

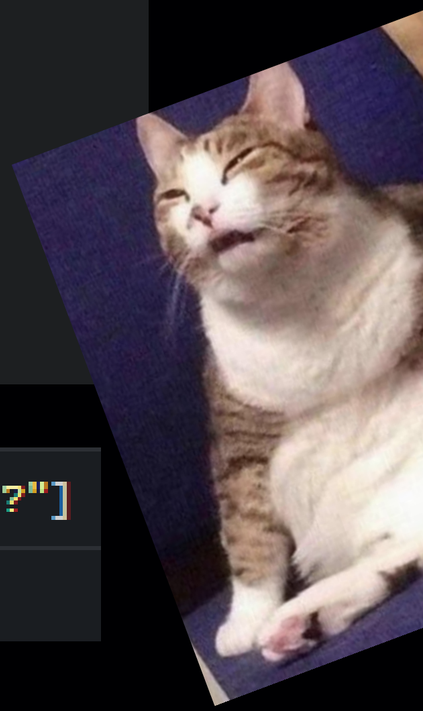
```
if (idade >= 18) {
    console.log("Você é maior de idade.");
} else {
    console.log("Você é menor de idade.");
}

for (let i = 0; i < 5; i++) {
    console.log(i);
}

while (saldo > 0) {
    console.log("Saldo disponível: " + saldo);
    saldo -= 10;
}
```

```
const frutas = ["maçã", "banana", "laranja"];
console.log(frutas[1]); // Saída: "banana"

frutas.push("morango");
console.log(frutas.length); // Saída: 4
```

```
const frutas = ["maçã", "banana", "laranja"];
console.log(frutas[1]); // Saída: "banana"

frutas[5] = "Segmentation fault... ?";
console.log(frutas); // Saída: ?
console.log(frutas.length);
```

```js
const frutas = ["maçã", "banana", "laranja"];
console.log(frutas[1]); // Saída: "banana"

frutas[5] = "Segmentation fault... ?";
console.log(frutas); // Saída: ?
console.log(frutas.length);
```

```
["maçã", "banana", "laranja", undefined, undefined, "Segmentation fault... ?"]
6
```



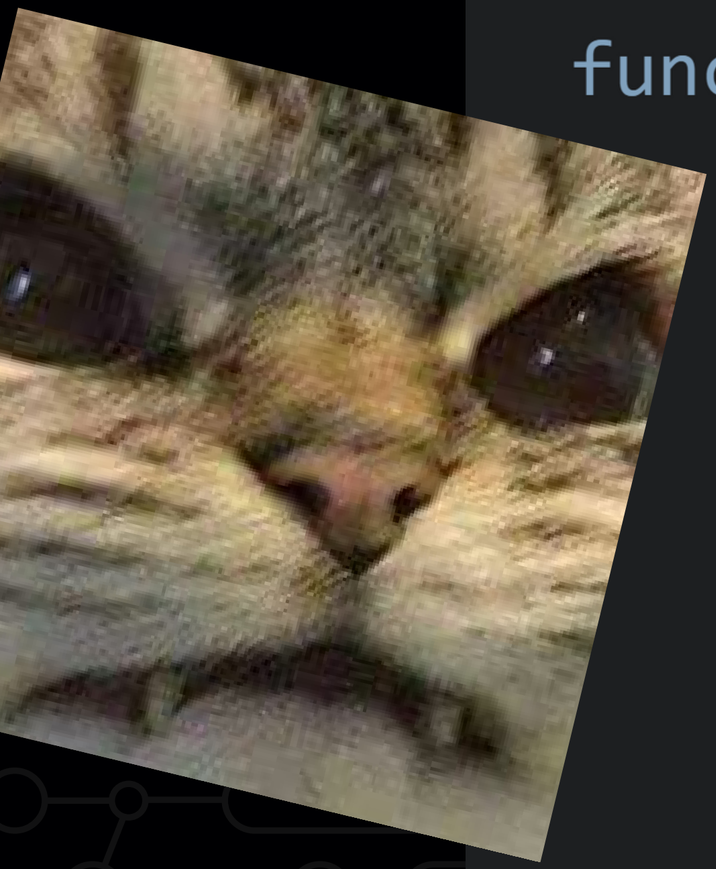
```
function nomeDaFuncao( /*parâmetros*/ ) {
 /* código que será executado */

 return/*Valor retornado*/;
}
```

```
function saudacao(nome) {
    return "Olá, " + nome + "!";
}

const mensagem = saudacao("Maria");
console.log(mensagem);
```

```
const pessoa = {
    nome: "Jiusepe Dal Filho",
    idade: Math.floor((Math.random() * (25 - 20) + 20)),
    profissao: "root"
};

console.log(pessoa.idade); // Saída: "20 inclusivo até 25 exclusivo"
```

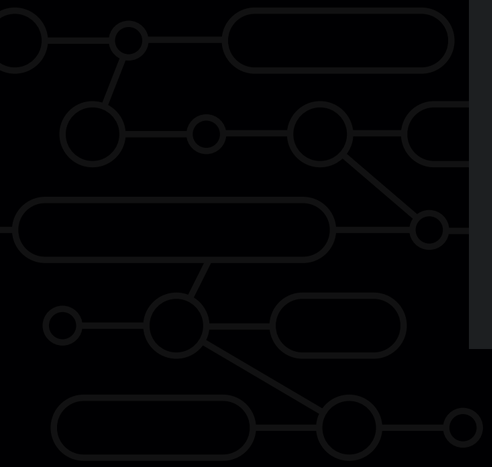
```
//Construtor
//Construtor
function Exemplo() {
    this.propriedade = 'Isso é uma propriedade.',
        this.metodo = function() {
            return 'Isso é um metódo';
        }
}

var objeto = new Exemplo(); //Instância do construtor "Exemplo"

//Alerta os respectivos textos na tela
alert(objeto.propriedade),
    alert(objeto.metodo());
```

```
function greet() {
    let a = 'hello';

    if(a == 'hello'){
        let b = 'world';
        console.log(a + ' ' + b);
    }

    console.log(a + ' ' + b);
}
greet();
```



```
function greet() {
    let a = 'hello';

    if(a == 'hello'){
        let b = 'world';
        console.log(a + ' ' + b);
    }

    console.log(a + '
}
greet()
```

```
Uncaught ReferenceError: b is not defined
```



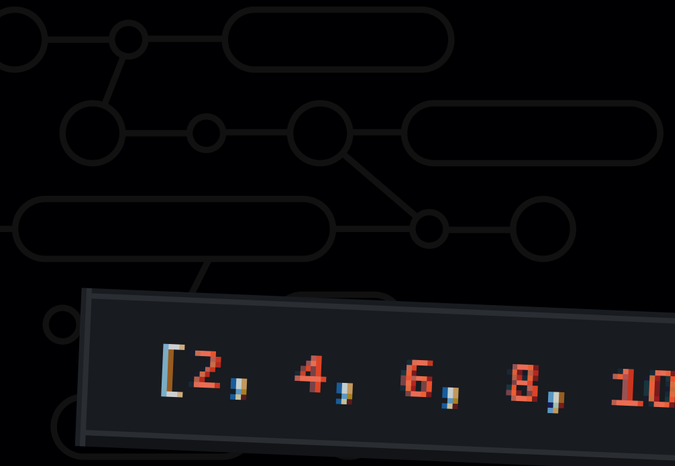
```
function greet() {
    let a = 'hello';

    if(a == 'hello'){
        var b = 'world';
        console.log(a + ' ' + b);
    }

    console.log(a + ' ' + b);
}
greet();
```

```
function greet() {
    let a = 'hello';

    if(a == 'hello'){
        var b = 'world';
        console.log(a + ' ' + b);
    }

    console.log(a + ' ' + b);
}
greet();
```

```
"hello world"
"hello world"
```

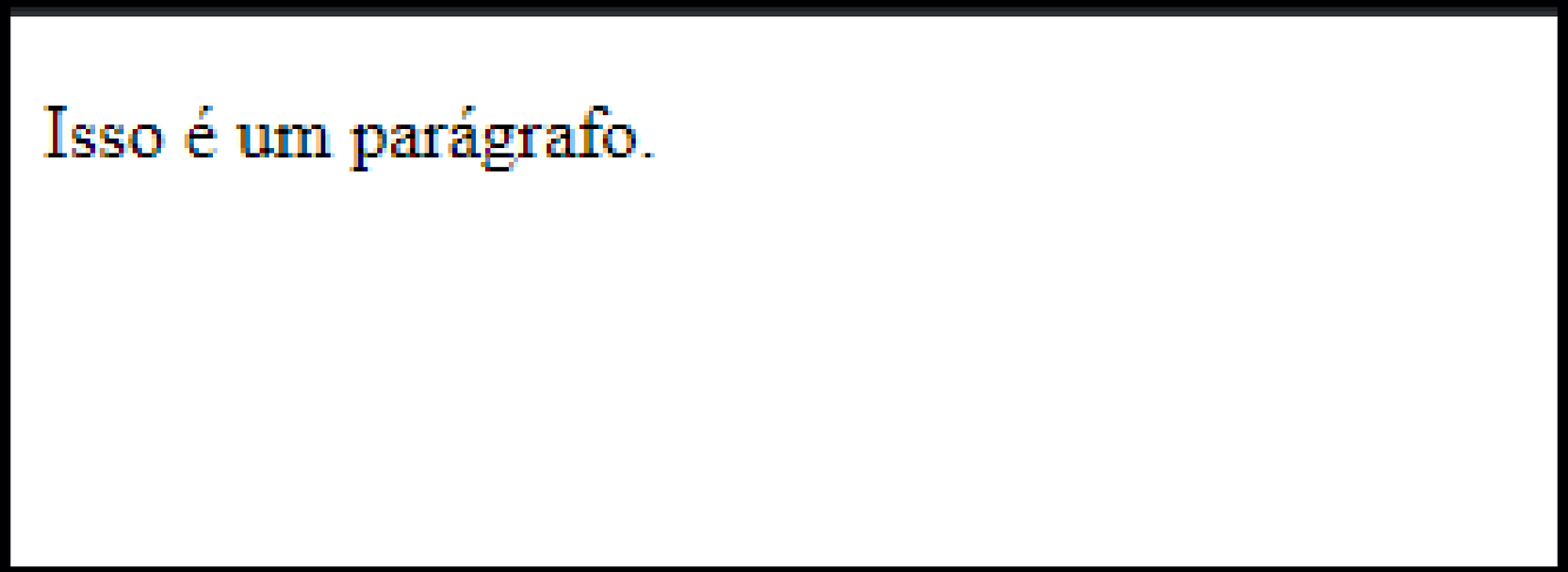
```
const numeros = [1, 2, 3, 4, 5];

numeros.forEach(function(numero) {
    console.log(numero * 2);
});
```

```
const numeros = [1, 2, 3, 4, 5];

const duplicados = numeros.map(function(numero) {
    return numero * 2;
});

console.log(duplicados);
```



```
[2, 4, 6, 8, 10]
```

```
let meuParagrafo = document.createElement("p");
meuParagrafo.textContent = "Isso é um parágrafo.";
document.body.appendChild(meuParagrafo);
```



```
let botao = document.querySelector("button");
botao.addEventListener("click", function() {
    console.log("Botão clicado!");
});
```

```html
<button>
Oii
</button>
```

JavaScript + No-Library (pure JS) ▼ Tidy

```javascript
let botao = document.querySelector("button");
botao.addEventListener("click", function() {
    console.log("Botão clicado!");
});
```

Oii

Console (beta) 6 0 0 0

"Botão clicado!"

"Botão clicado!"

"Botão clicado!"

```
console.log("Início");

setTimeout(function() {
    console.log("Após 2 segundos");
}, 2000);

console.log("Fim");
```

```
"Início"

"Fim"

"Após 2 segundos"
```

```
function aguardeTempo(ms) {
    return new Promise(function(resolve) {
        setTimeout(resolve, ms);
    });
}

console.log("Início");

aguardeTempo(2000).then(function() {
    console.log("Após 2 segundos");
});

console.log("Fim");
```

```
function aguardeTempo(ms) {
    return new Promise(function(resolve) {
        setTimeout(resolve, ms);
    });
}

async function main() {
    console.log("Início");
    await aguardeTempo(2000);
    console.log("Após 2 segundos");
    console.log("Fim");
}

main();
```

```javascript
// Criação de uma Promise
const minhaPromise = new Promise(function(resolve, reject) {
    // Simula uma operação assíncrona
    setTimeout(function() {
        const sucesso = true; // Pode ser true ou false
        if (sucesso) {
            resolve("Operação bem-sucedida!");
        } else {
            reject("Algo deu errado!");
        }
    }, 2000); // Simula um atraso de 2 segundos
});

// Uso da Promise
minhaPromise
    .then(function(resultado) {
        console.log(resultado); // Resolvida: "Operação bem-sucedida!"
    })
    .catch(function(erro) {
        console.error(erro); // Rejeitada: "Algo deu errado!"
    });
```


PRATICA ;D


SQUIRTLE♂ Lv18
PAR
PIKACHU♂ Lv18
HP
03/ 03
EXP
Foe SQUIRTLE is paralyzed!
It may be unable to move!

```
// Endpoint
const apiUrl = 'https://pokeapi.co/api/v2/pokemon/eevee';

// GET request
fetch(apiUrl)
  .then(response => {
    // Check if the response status is OK (200)
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    // Parse the response JSON
    return response.json();
  })
  .then(data => {
    // Handle the data (array of posts)
    console.log(data);
  })
  .catch(error => {
    // Handle any errors
    console.error('Fetch error:', error);
  });
```

```
  const response = await fetch(url, {
    method: "POST", // *GET, POST, PUT, DELETE, etc.
    mode: "cors", // no-cors, *cors, same-origin
    cache: "no-cache", // *default, no-cache, reload, force-cache, only-if-cached
    credentials: "same-origin", // include, *same-origin, omit
    headers: {
      "Content-Type": "application/json",
      // 'Content-Type': 'application/x-www-form-urlencoded',
    },
    redirect: "follow", // manual, *follow, error
    referrerPolicy: "no-referrer", // no-referrer, *no-referrer-when-downgrade,
origin, origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-
cross-origin, unsafe-url
    body: JSON.stringify(data), // body data type must match "Content-Type" header
  });
  return response.json(); // parses JSON response into native JavaScript objects
}
```

store Access to Petstore orders

GET	/store/inventory	Returns pet inventories by status
POST	/store/order	Place an order for a pet
GET	/store/order/{orderId}	Find purchase order by ID
DELETE	/store/order/{orderId}	Delete purchase order by ID

AJUDA:

- PokeAPI API endpoint:https://pokeapi.co/api/v2/
- Documentação: https://pokeapi.co/docs/v2
- Javascript doc: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

Node.js - Wikipedia (27 may 2009). https://en.wikipedia.org/wiki/Node.js.
Denysdovhan (15 dec 2022). GitHub - denysdovhan/wtfjs: 🤪 A list of funny and tricky JavaScript examples. https://github.com/denysdovhan/wtfjs.
ECMAScript – Wikipédia, a enciclopédia livre ([S.d.]). https://pt.wikipedia.org/wiki/ECMAScript.
Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js ([S.d.]). https://ieeexplore.ieee.org/abstract/document/7023652.
Linguagem tipada – Wikipédia, a enciclopédia livre ([S.d.]). https://pt.wikipedia.org/wiki/Linguagem_tipada#Linguagens_fracamente_tipadas.
A Brief History of JavaScript ([S.d.]). https://link.springer.com/chapter/10.1007/978-1-4302-0062-8_1.
JavaScript – Wikipédia, a enciclopédia livre (1 nov 2010). https://pt.wikipedia.org/wiki/JavaScript.
Ryckegem, K. V. (21 dec 2021). 5 Strange JavaScript Quirks Demystified. https://javascript.plainenglish.io/top-5-strange-javascript-quirks-demystified-61edd3e1f3a9.