



Universidade Federal
de Campina Grande

Versionamento de código e Introdução à JavaScript

**José Glauber - UFCG
2025.1**

O que é versionamento de código?

- Processo de gerenciar mudanças no código-fonte ao longo do tempo, garantindo o histórico e rastreabilidade.



rastreabilidade

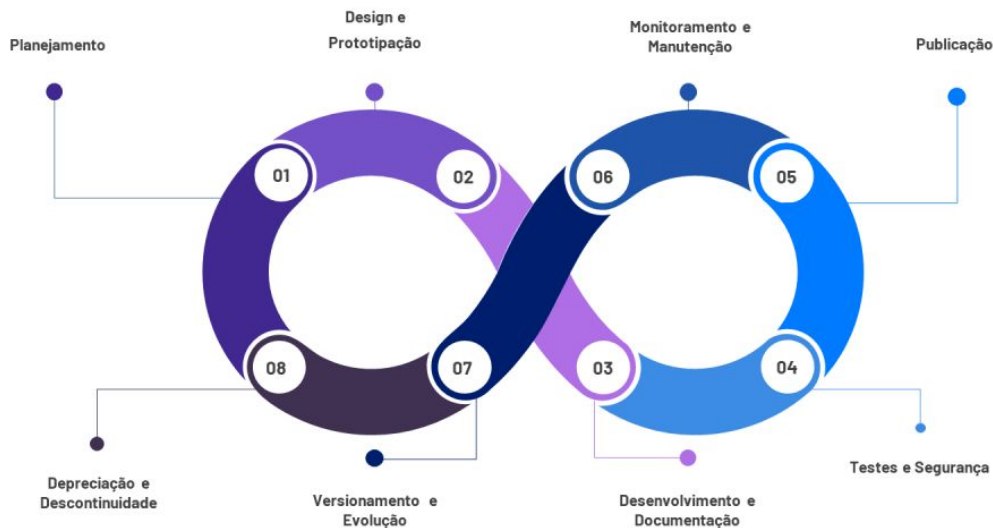
colaboração

controle de versões

EVITAR: profissionais desenvolvendo código que sobrescreva o do outro!

O que é versionamento de código?

API LIFE CYCLE



Ferramentas de versionamento de código



Ferramenta de controle de versão distribuída, rápida, confiável e muito utilizada!

Plataformas baseadas no Git



Ferramentas de versionamento de código



Actions: Automação de pipelines CI/CD. Ideal para executar testes em APIs, deploys e validações de segurança.

Code review: Permite revisão colaborativa através de pull requests, ajudando garantir a qualidade de novos endpoints.

Ferramentas de versionamento de código



Deploy automatizado: Pipelines diretamente conectadas ao Kubernetes ou servidores para deploy de backend.

Ambientes dinâmicos: Permite criar ambientes temporários para testar alterações na API antes do merge.

Ferramentas de versionamento de código

Integração com o Atlassian: Ideal para quem já usa Jira e Confluence.



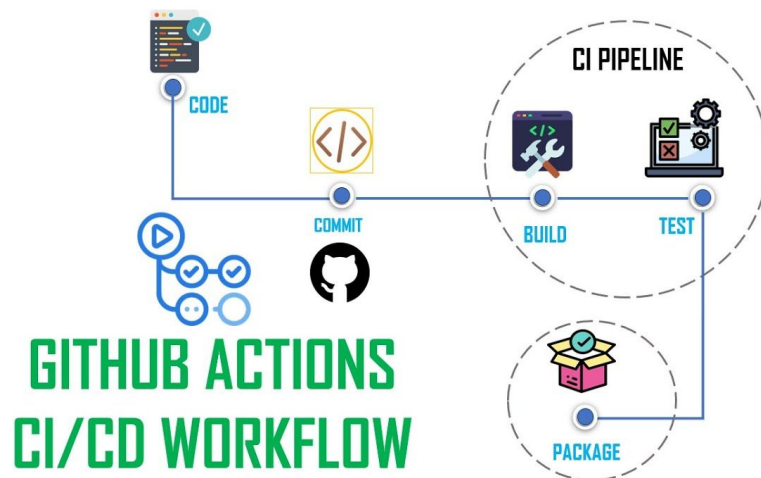
Branch permission: Controle rígido sobre quem pode fazer alterações na branch principal.

Ferramenta	Principal uso no backend	Diferencial
Git	Controle distribuído; manipulação local de repositórios	Performance e flexibilidade em grandes repositórios.
Github	Open source e projetos colaborativos	GitHub Actions e comunidade ampla para projetos backend.
GitLab	Pipelines avançados; deploy automatizado	CI/CD nativo e integração com Kubernetes
Bitbucket	Atrelado ao Atlassian	Integração com Jira para rastreabilidade de tarefas

Impacto do uso de ferramentas

- Colaboração eficiente
- Automação CI/CD: o que é muito importante para testes, build e deploy.

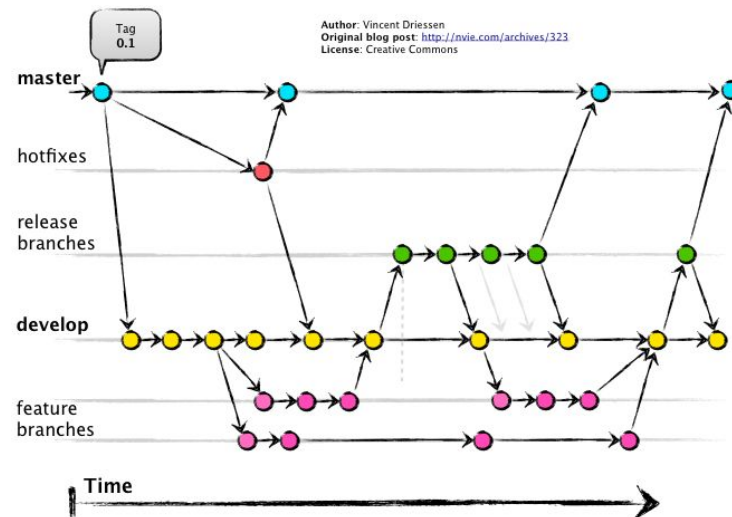
cada push para a branch principal pode disparar testes automatizados, verificar quebras de endpoints e fazer o deploy para o ambiente de staging.



Impacto do uso de ferramentas

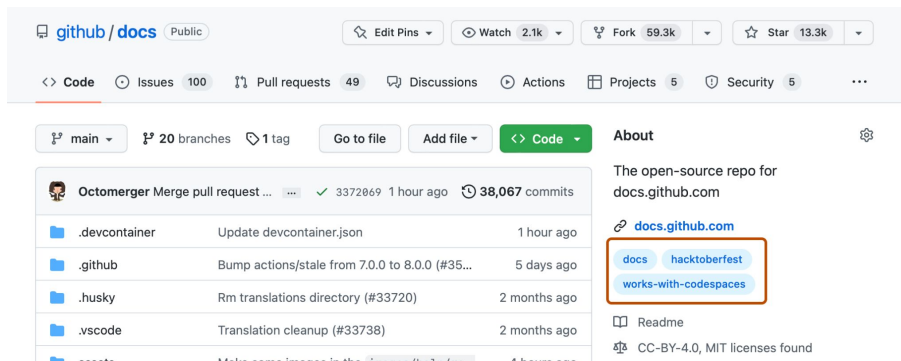
Gerenciamento de Hotfixes e emergências: importante usar branches e tags, sem afetar as funcionalidades que estão em desenvolvimento.

Bug crítico em produção é detectado, daí uma branch pode ser aberta diretamente de uma versão estável, corrigindo e ser mandado novamente sem atrapalhar outras mudanças em desenvolvimento.

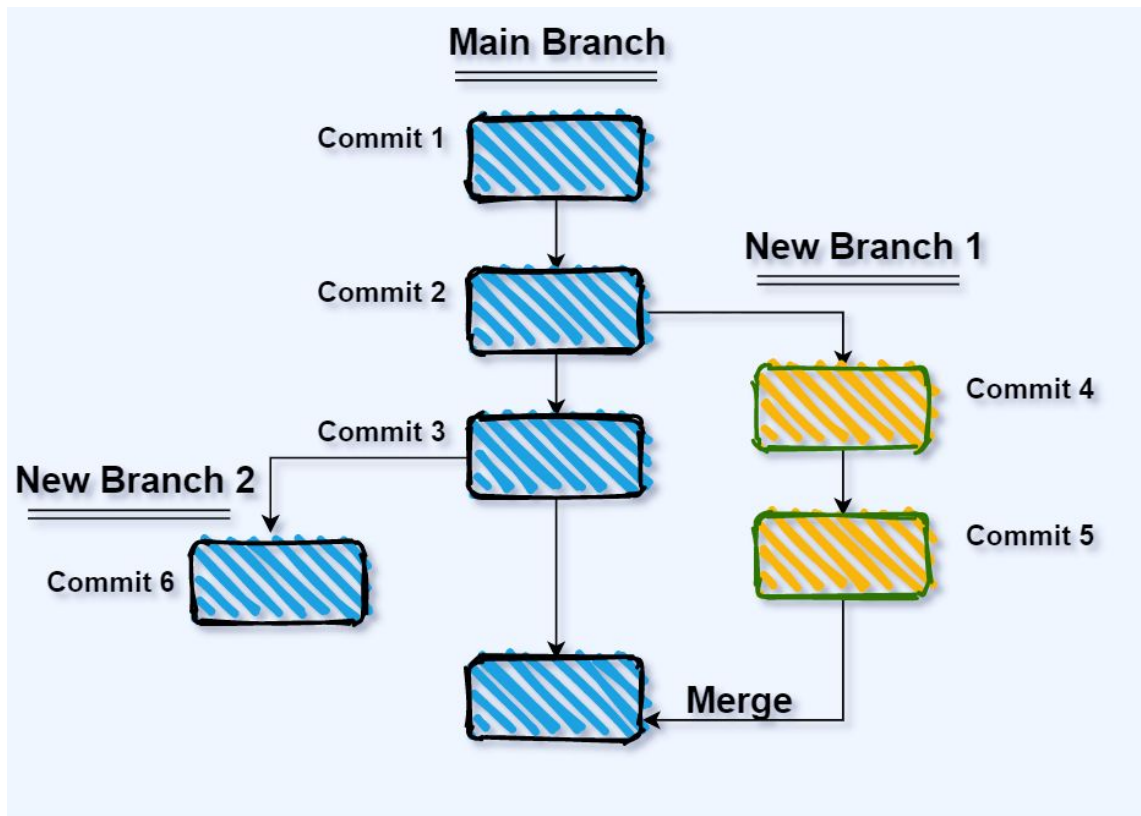


Conceitos principais do Git

Repositórios



Conceitos principais do Git

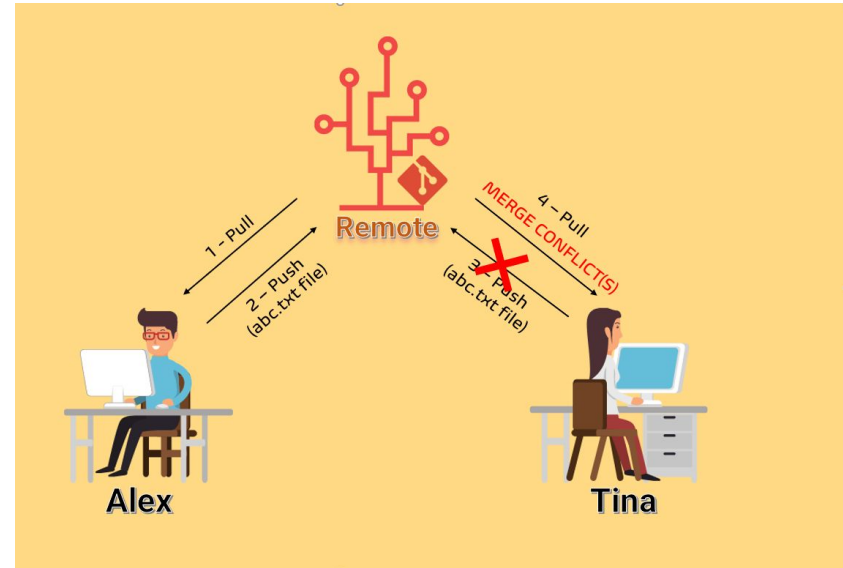
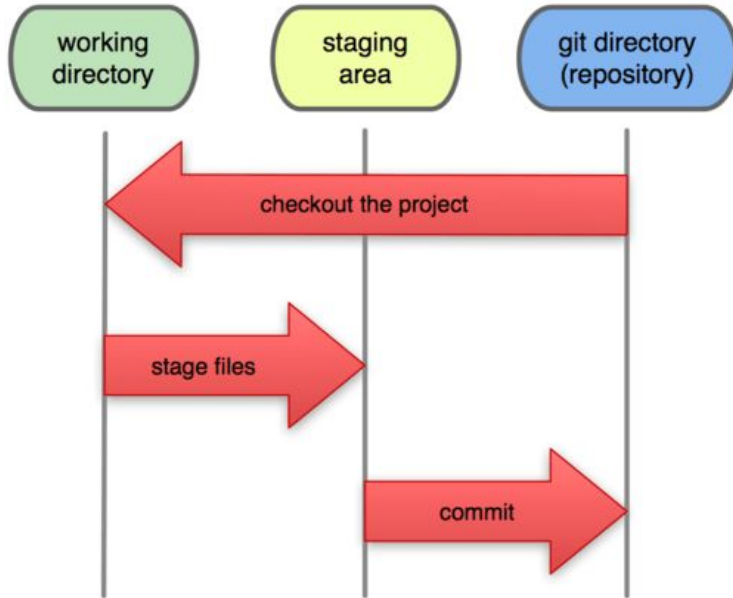


Commits,
branches e
merges

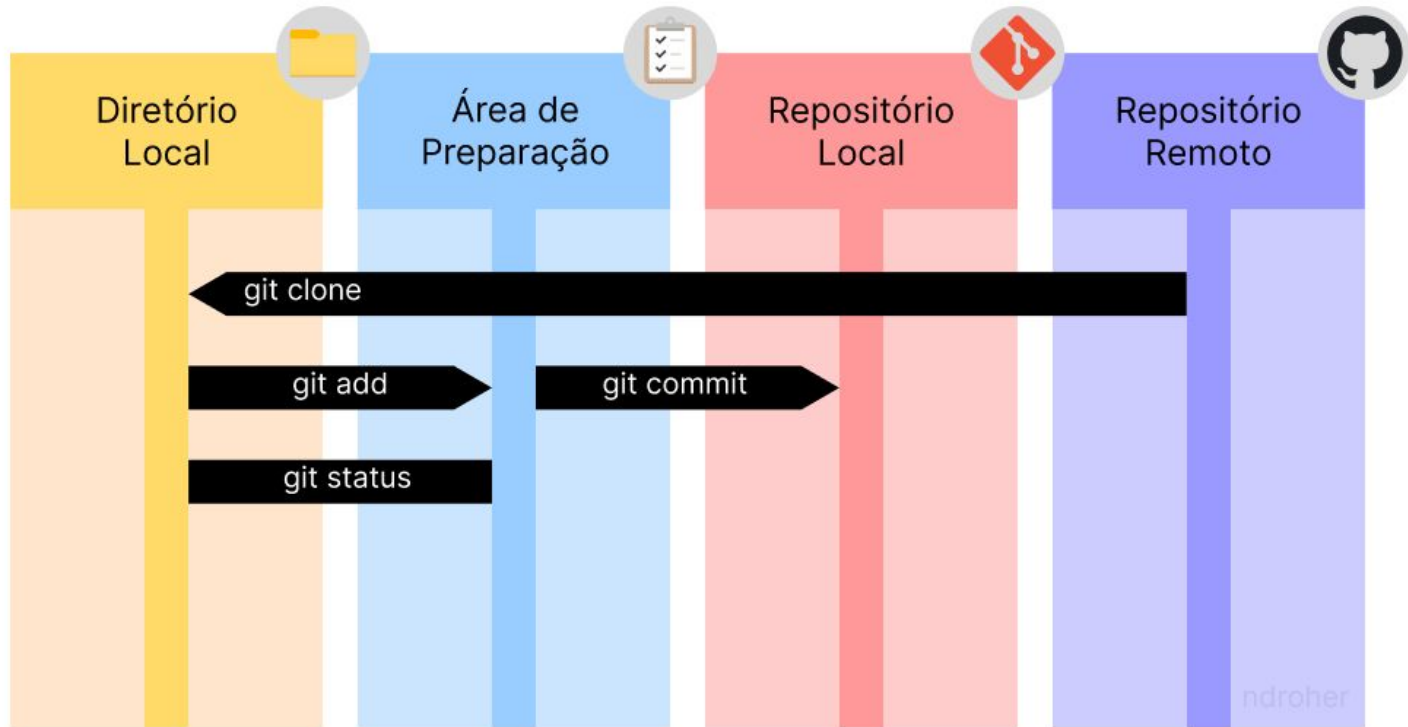
Conceitos principais do Git

staging area e conflitos de merge

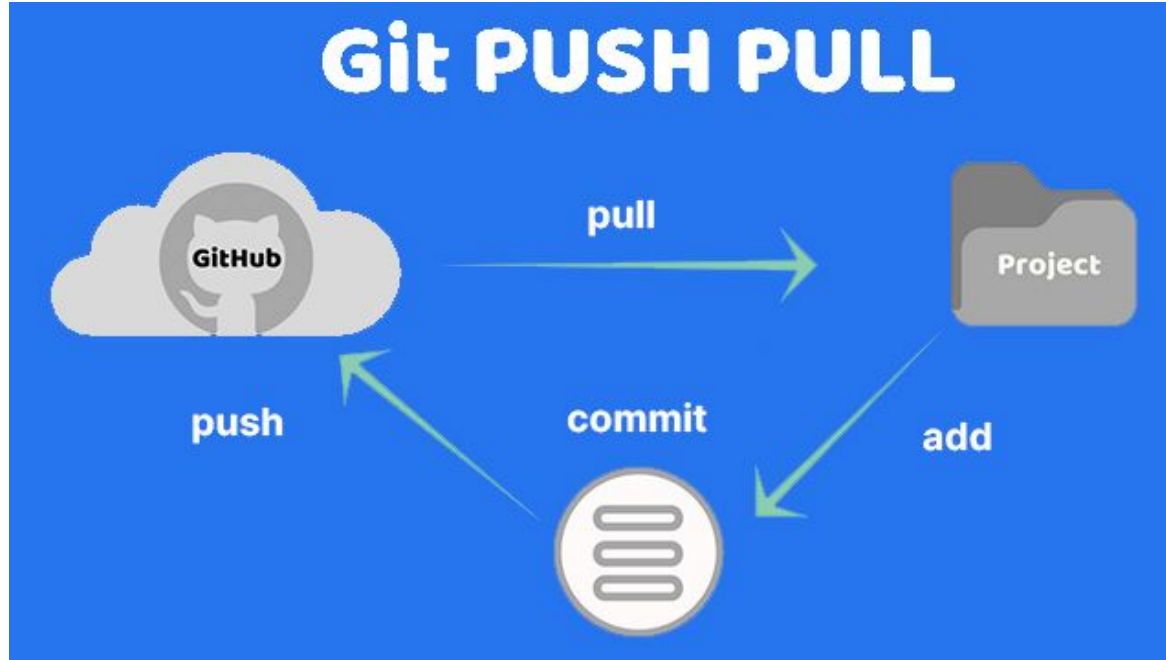
Local Operations



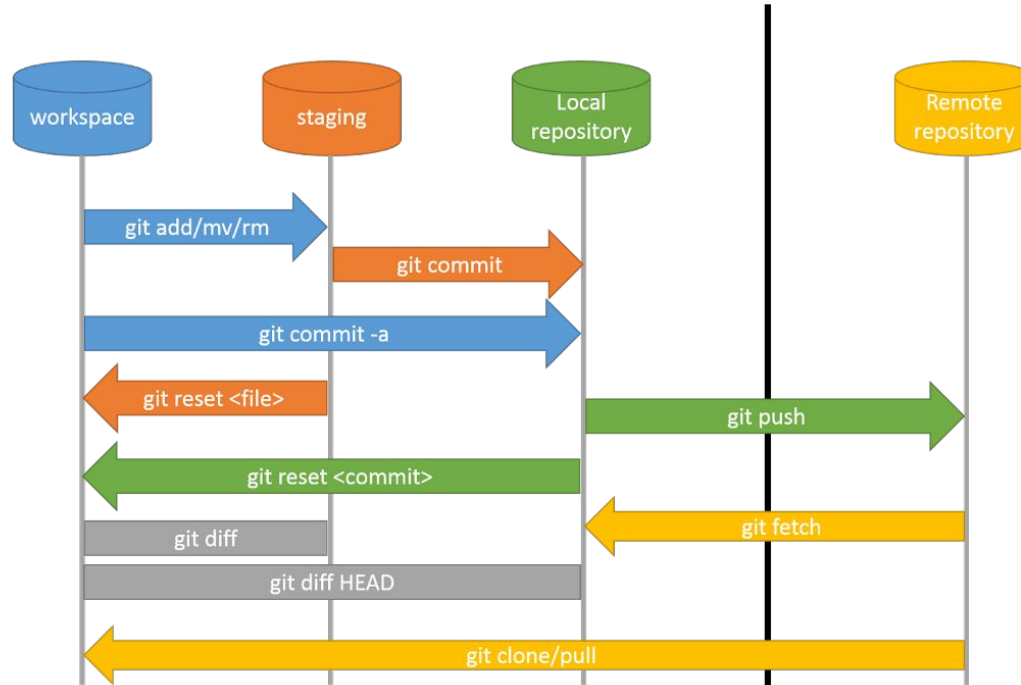
Fluxo básico de trabalho no Git



Fluxo básico de trabalho no Git



Fluxo básico de trabalho no Git



Introdução à JavaScript

O que é Javascript?

Linguagem de programação de alto nível, interpretada, leve e baseada em eventos, usada principalmente para adicionar interatividade em páginas web.



responde à ação dos usuários

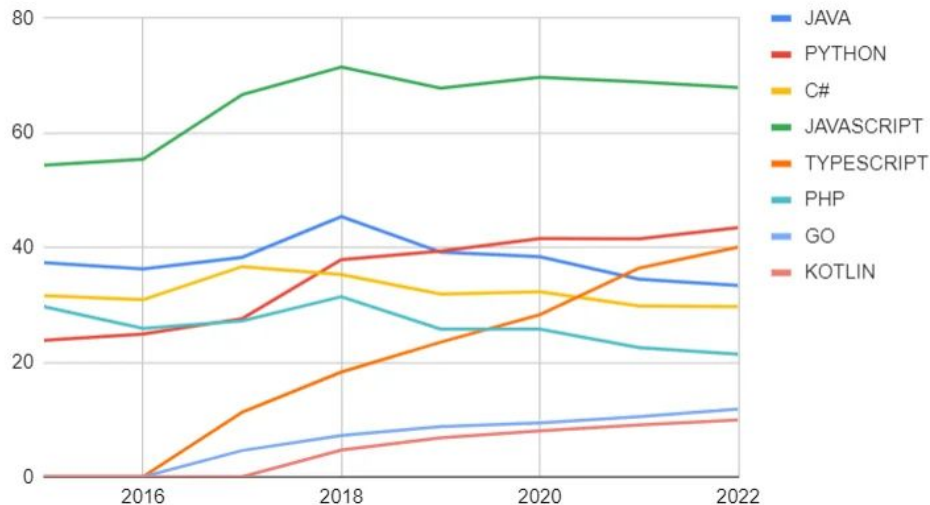
multiparadigma: funcional, OO e procedural

interpretada no navegador

Atualmente é a linguagem mais conhecida para frontend.

O que é Javascript?

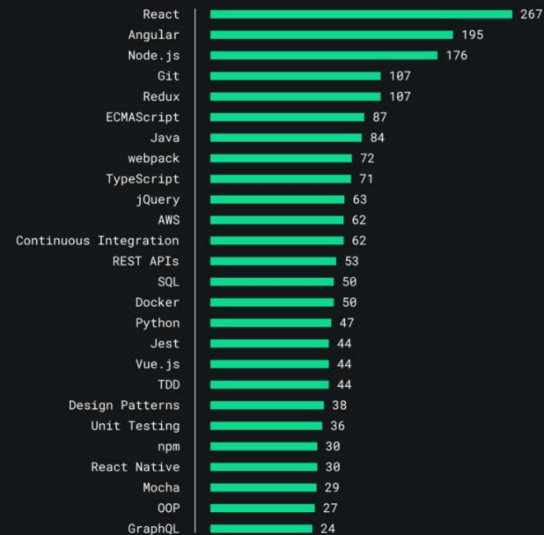
JAVA, PYTHON, C#, JAVASCRIPT, TYPESCRIPT...



[Crescimento e queda de algumas linguagens no Stack Overflow Survey \(2015-2022\) : r/brdev](#)

The skills JavaScript developers need today

(based on 300 job listings from tech companies in April 2019)



[O que o programador JavaScript deve saber - MundoJS](#)

História

- Criada em 1995 por Brendon Eich.

Problema principal: Para tornar os sites mais dinâmicos, os desenvolvedores precisavam de uma linguagem que rodasse diretamente no navegador, manipulando elementos da página sem depender dos servidores.

Na época existiam duas principais:

Java

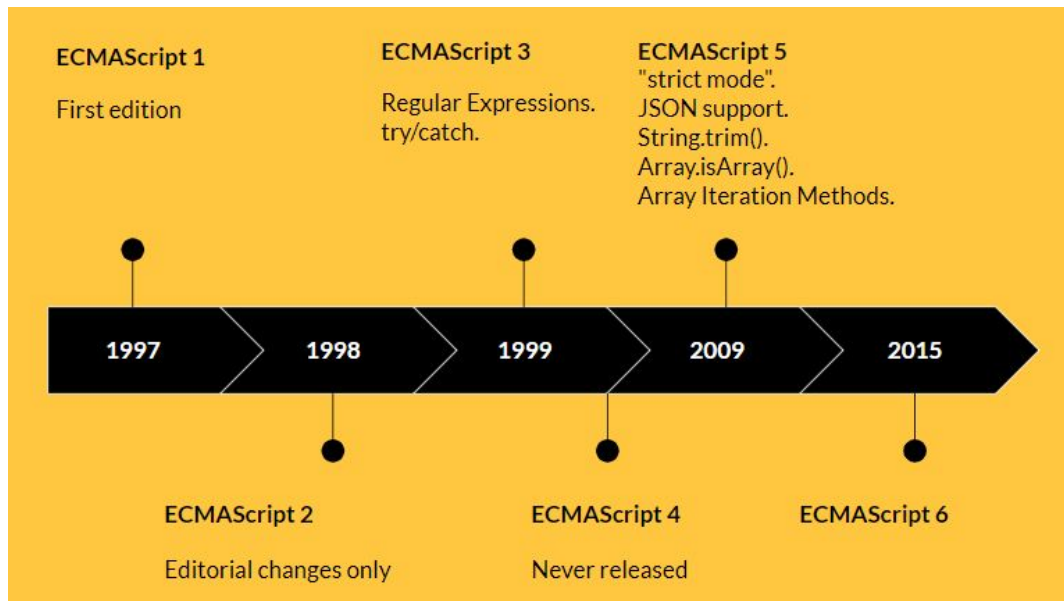
Muito complexa para tarefas pequenas e lentas no contexto de web.

VBScript

Exclusivo para navegadores da Microsoft e incompatível com outros navegadores.

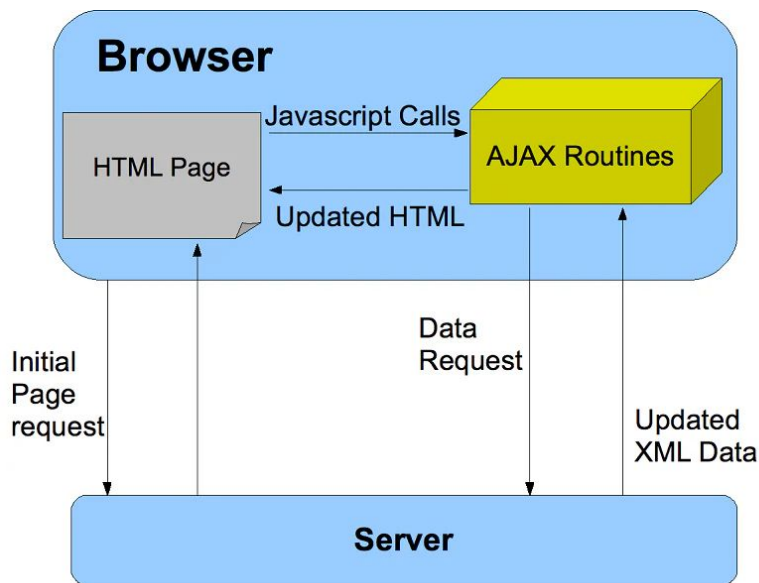
História

Em 1997, JavaScript foi padronizado pelo ECMA Internacional com ECMAScript (ES) tornando-se especificação oficial. Isso foi necessário para que vários navegadores pudessem implementar a linguagem de forma consistente.



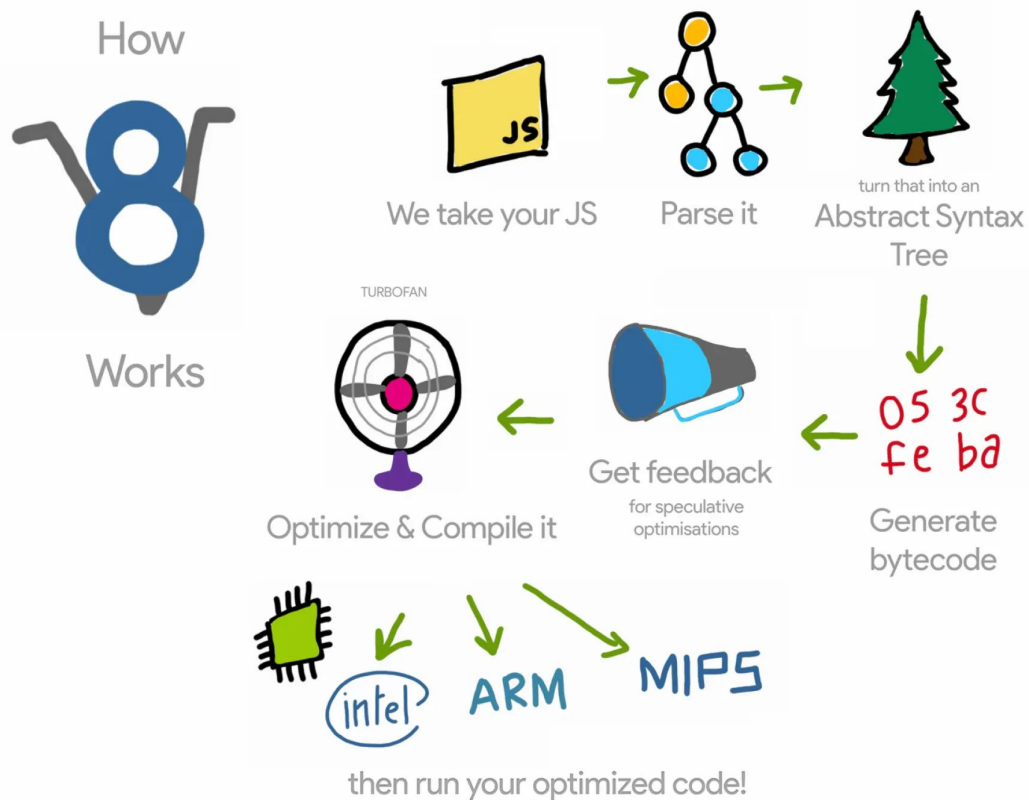
História

2010 - 2015 surgiu o **AJAX (Asynchronous JavaScript and XML)**: permitiu a criação de aplicações web mais rápidas e dinâmicas, como Google Maps e Gmail. Com o AJAX era possível atualizar partes da página sem precisar atualizá-la inteira.



História

Surgiu o
Chrome V8,
que executa
código
JavaScript
dentro ou fora
do navegador.



História

Com todos esses avanços, Javascript impactou diretamente no desenvolvimento web.

Dominância do Frontend

Expandiu para backend, mobile e Machine Learning.

Principais elementos JavaScript

Sintaxe bem próxima a linguagem Java e C.

Variáveis

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

Principais elementos JavaScript

Tipos de Dado Primitivos

String

"hello world"

'abacaxi'

`5 patinhos`

Number

20

3.1415

-18

-9.04

Infinity

NaN

Boolean

true

false

Object
















{}

Outros

null

undefined

Principais elementos JavaScript

 Operador	 Nome	 Ex.	 Resultado
 <code>==</code>	Igual	<code>a == b</code>	Verdadeiro se <code>a</code> for igual a <code>b</code>
 <code>!=</code>	Diferente	<code>a != b</code>	Verdadeiro se <code>a</code> não for igual a <code>b</code>
 <code>===</code>	Idêntico	<code>a === b</code>	Verdadeiro se <code>a</code> for idêntico a <code>b</code>
 <code>!==</code>	Não Idêntico	<code>a !== b</code>	Verdadeiro se <code>a</code> não for igual a <code>b</code> ou eles não tiverem o mesmo tipo
 <code><</code>	Menor que	<code>a < b</code>	Verdadeiro se <code>a</code> for menor que <code>b</code>
 <code>></code>	Maior que	<code>a > b</code>	Verdadeiro se <code>a</code> for maior que <code>b</code>
 <code><=</code>	Menor ou igual	<code>a <= b</code>	Verdadeiro se <code>a</code> for menor ou igual a <code>b</code>
 <code>>=</code>	Maior ou igual	<code>a >= b</code>	Verdadeiro se <code>a</code> for maior ou igual a <code>b</code>
 <code>&&</code>	E	<code>a && b</code>	Verdadeiro se existir <code>a</code> e <code>b</code>
 <code> </code>	Ou	<code>a b</code>	Verdadeiro se existir <code>a</code> ou existir <code>b</code>
 <code>!</code>	Negação	<code>!a</code>	Negação, se o valor for verdadeiro vai transformar em falso e vice versa

Principais elementos JavaScript

REGULAR VS ARROW FUNCTION

```
function regular() {  
}
```

```
const regular = () => {  
}
```

Funções regulares permitem um `this` dinâmico, já `arrow functions` fixa o `this` no escopo em que foi criado.

Importante para preservar o contexto em callbacks e funções aninhadas.

Principais elementos JavaScript



Array Methods

<code>findLastIndex()</code>	<code>findLast()</code>	<code>reduce()</code>	<code>fill()</code>
<code>lastIndexOf()</code>	<code>includes()</code>	<code>filter()</code>	<code>sort()</code>
<code>reduceRight()</code>	<code>unshift()</code>	<code>concat()</code>	<code>find()</code>
<code>copyWithin()</code>	<code>indexOf()</code>	<code>delete()</code>	<code>from()</code>
<code>toReversed()</code>	<code>isArray()</code>	<code>splice()</code>	<code>flat()</code>
<code>toSpliced()</code>	<code>reverse()</code>	<code>every()</code>	<code>join()</code>
<code>findIndex()</code>	<code>valueOf()</code>	<code>slice()</code>	<code>push()</code>
<code>toSorted()</code>	<code>entries()</code>	<code>shift()</code>	<code>pop()</code>
<code>toString()</code>	<code>forEach()</code>	<code>some()</code>	<code>map()</code>

@abidullah786

Principais elementos JavaScript

- Promises:
 - um objeto que representa a eventual conclusão de uma operação assíncrona e seu valor resultante;
 - pendente, cumprida e rejeitada
 - then(onFulfilled, onRejected), catch(onRejected), finally(onFinally)

```
1 const myPromise = new Promise((resolve, reject) => {  
2   let sucesso = true;  
3  
4   if (sucesso) {  
5     resolve("Successfull!");  
6   } else {  
7     reject("Error");  
8   }  
9 });
```

```
11 myPromise  
12   .then((message) => {  
13     console.log(message);  
14   })  
15   .catch((error) => {  
16     console.error(error);  
17   })  
18   .finally(() => {  
19     console.log('Successfull!');  
20   });
```

Principais elementos JavaScript

- Classes
 - sintaxe para criação de objetos e herança baseada em protótipos
- Modules
 - permitiu a exportação e importação de código entre arquivos;

```
1 class Person {
2   constructor(name, age) {
3     this.name = name;
4     this.age = age;
5   }
6   ..
7   message() {
8     return `My name is ${this.name} and i have ${this.age} old.`;
9   }
10 }
11
12 const p1 = new Person("Alice", 25);
13 console.log(p1.message());
```

```
1 export const name = "Alice";
2
3 export function message() {
4   console.log("Hello world!");
5 }
6
7 import { name, message } from './modulo.js';
8 message();
```

Como utilizar Javascript?

```
<!DOCTYPE html>
<html>
  <head>
    <title>Teste de botão</title>
    <meta charset = "utf-8"/>
    <script>
      quantidadeClique = 0;
      function ContarClique() {
        alert("Você me clicou!");
        quantidadeClique++;
      }
    </script>
  </head>
```

Incorporado no HTML

```
6 function registerThemeAssets()
7 {
8   $baseUrl = get_stylesheet_directory_uri();
9
10  # CSS
11  $nomeStyle = 'meu-style';
12  $urlStyle = $baseUrl . '/main.css';
13  $dependenciasStyle = [];
14  $versaoStyle = null;
15  $media = 'all';
16  wp_enqueue_style($nomeStyle, $urlStyle, $dependenciasStyle, $versaoStyle, $media);
17
18  # JS
19  $nomeScript = 'meu-script';
20  $urlArquivoScript = $baseUrl . '/main.js';
21  $dependenciasScript = ['jquery'];
22  $versaoScript = null;
23  $carregarNoFooter = true;
24  wp_enqueue_script($nomeScript, $urlArquivoScript, $dependenciasScript, $versaoScript, $carregarNoFooter);
25
26  # Parametros extras ao arquivo JS
27  $nomeObjeto = 'meuScriptParams';
28  $objeto = [
29    'parametro0' => 132,
30    'parametro1' => is_user_logged_in()
31    // ...
32  ];
33  wp_localize_script($nomeScript, $nomeObjeto, $objeto);
34 }
35 add_action('wp_enqueue_scripts', 'registerThemeAssets', 99);
```

Arquivo .js

Referências

- [Git Documentation](#)
- [Git - Basic Branching and Merging](#)
- Pro Git - Scott Chacon e Ben Straub
- [Understanding GitHub Actions - GitHub Docs](#)
- [Git](#)
- <https://www.atlassian.com/git/tutorials>
- MDN Web Docs - JavaScript
- Eloquent JavaScript (Marijn Haverbeke)
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- <https://tc39.es/ecma262/>