

AceLeraDev React Online

/

Módulo 3: Javascript I, Fundamentos da Linguagem



Quem sou eu?

/


Henrique Jensen

Front end developer na Daitan Group

Cientista da Computação pela Universidade de Lavras(UFLA)

Co-organizador do Campinas FrontEnd

Mentor React no AceleraDev React



Entendendo o curso

/

O curso é dividido em 10 módulos.

Cada módulo é complementar ao outro e cada um possui um desafio a ser entregue.

Os desafios são a base para estimulá-los a aprender.

Nosso objetivo é guiá-los a desenvolverem aplicações web utilizando React.

Ao final do curso teremos um Demo Day, para que vocês mostrem o que aprenderam durante o curso.

Ementa geral do programa

Módulo 1: Browser Engine, Motores e Debugging

Módulo 2: HTML5 e CSS3, Atomic Design e CSS Modular

Módulo 3: Javascript I, Fundamentos da Linguagem

Módulo 4: Javascript II, Paradigmas e Testes

Módulo 5: React I, Fundamentos e React OO

Módulo 6: React II, Fundamentos e React OO

Módulo 7: React Hooks, React Funcional

Módulo 8: Redux, Gerenciamento de Estado

Módulo 9: Redux, Build e Deploy

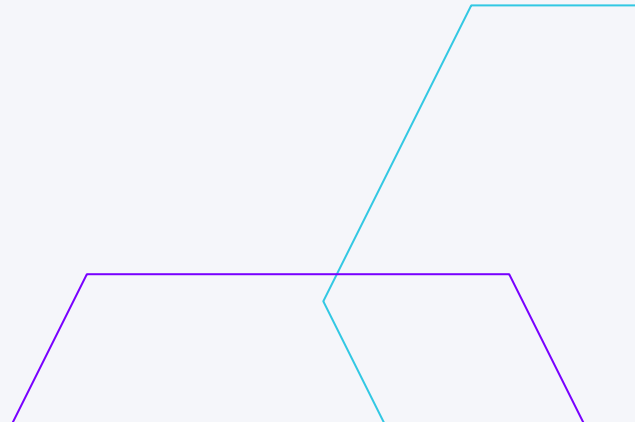
Módulo 10: Demo Day



Objetivos da Semana

/

Iniciar o entendimento de JavaScript e da plataforma
NodeJS





Requisitos

/

VS Code

Browser

Internet

NodeJS - versão 12





Tópicos desta aula:

/

Tópico 1: Evolução do Javascript

Tópico 2: A Engine Javascript

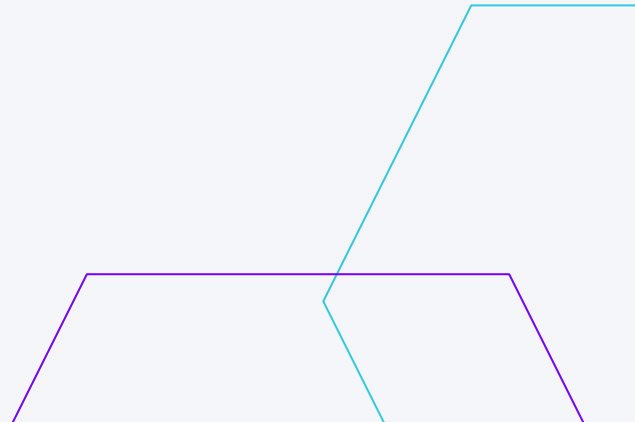
Tópico 3: Declaração de variáveis

Tópico 4: Tipos de dados primitivos

Tópico 5: Valores e Inferência

Tópico 6: Tipos de dados não-primitivos

Tópico 7: Funções



The background is a dark navy blue. It features abstract geometric line art in teal and purple. In the top-left corner, there are several overlapping lines forming a series of connected shapes. In the bottom-right corner, there are more lines forming a similar but more complex geometric pattern.

Evolução do Javascript

Evolução do Javascript

/

É uma linguagem de programação

Criada por Brendan Eich em 1995 para a Netscape

Foi inspirado por Java, Scheme e Self

Ela tem sintaxe de chaves, tipagem dinâmica, orientação a objetos baseada em protótipos e funções de primeira classe

É um dos core da web ao lado do HTML e do CSS

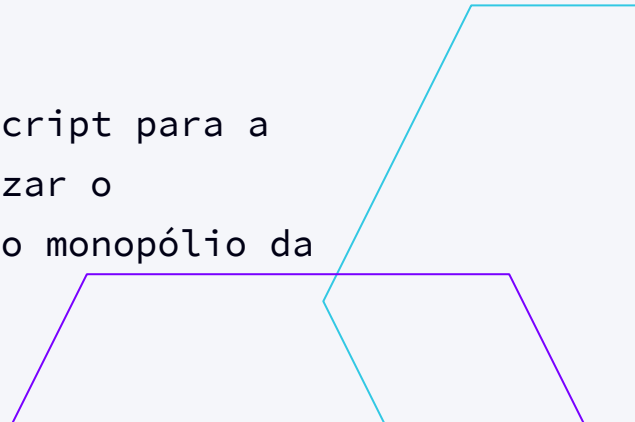


/

Seu primeiro nome foi Mocha, depois LiveScript, porém antes do lançamento oficial mudou para JavaScript para aproveitar a popularidade do Java na época.

Em 1995 a Microsoft lança o Internet Explorer (IE) com o JScript como linguagem de script, dando início a guerra dos browsers.

Em Novembro de 1996 a Netscape envia o JavaScript para a ECMA International, com o intuito de padronizar o JavaScript para todos os navegadores e tirar o monopólio da Microsoft





/


Junho de 1997, especificação do ECMAScript 1

Junho de 1998, especificação do ECMAScript 2

Dezembro de 1999, especificação do ECMAScript 3

No início de 2000 a popularidade do IE era de 95%, tornando o JScript a linguagem do frontend.

Sem a colaboração da Microsoft a especificação do ECMAScript 4 nunca saiu





/

Em 2004 a Mozilla (sucessora da Netscape), lança o Firefox, que foi bem recebido pelo mercado.

Em 2005 eles entram na ECMA Internacional e participam do trabalho para a próxima versão do ECMAScript

Em 2005, Jesse James Garret lança um artigo com o termo Ajax e descreve as tecnologias das quais o JavaScript era o sustentação, para criar web applications.

Algumas bibliotecas são lançadas, como jQuery. Prototype.

Em 2008 a Google lança o Chrome com a engine V8






/

Em Julho de 2008, na conferência de Oslo, há um acordo para a nova versão do ECMAScript, que foi lançada em 2009 como ECMAScript 5.

Em 2009 foi lançado a plataforma Node.js, permitindo que o JavaScript se tornasse uma linguagem do lado do servidor também.

Em 2016 a versão ECMAScript 6 é lançada, com uma coleção extensiva de adições e refinamentos.

De 2016 até 2019, uma nova versão do padrão ECMAScript foi lançado por ano, tornando a linguagem madura.



The background is a dark navy blue. It features abstract geometric line art in teal and purple. In the top-left corner, there are several overlapping lines forming a series of connected shapes, including a prominent teal 'V' shape and a purple line extending downwards. In the bottom-right corner, there are more teal and purple lines forming a series of connected shapes, including a teal 'V' shape and a purple line extending upwards.

A Engine Javascript

A Engine Javascript

/

É um programa de computador que executa um código JavaScript.

Primeiros eram somente interpretadores, atualmente todos usam o just-in-time(JIT) para melhorar a performance

Primeiro foi escrito por Brendan Eich para o Netscape




/

A engine do Netscape evoluiu para o SpiderMonkey do Firefox

O primeiro navegador moderno com o just-in-time foi o V8 do Chrome.

A apple desenvolveu o Nitro(JavaScriptCore) para o navegador Safari

Em 2017 os navegadores adicionaram suporte ao WebAssembly. A JavaScript engine executa o código WebAssembly no mesmo lugar que o código JavaScript



The background is a solid dark blue. In the top-left corner, there are several overlapping geometric shapes formed by thin lines in teal and purple. These shapes resemble stylized polygons or abstract architectural elements. In the bottom-right corner, there are more similar geometric line patterns, also in teal and purple, mirroring the design in the top-left.

Declaração de variáveis

Declaração de variáveis

/

Antes da versão 6, o ECMAScript 2015, para se declarar uma variável no JavaScript utilizamos a palavra chave **var**

```
`var num = 45`
```

Um variável declarada com var ela pode ser utilizada no escopo de uma função, ou seja, no abre e fecha chaves de uma função



/

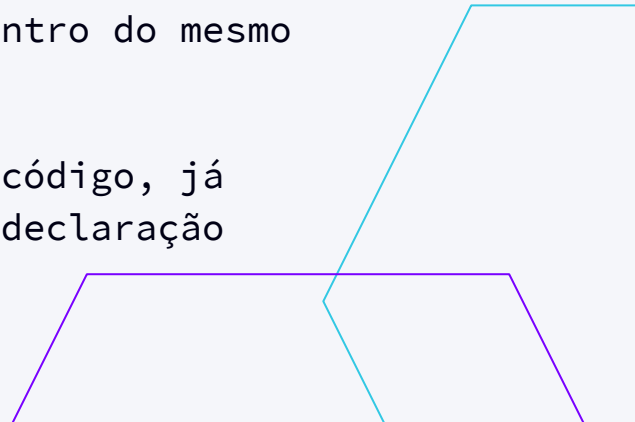
A ECMAScript 2015 introduziu duas novas palavras chaves para declarar: `let` e `const`

```
`let num = 45;`
```

```
`const text = 'Hello'`
```

`let` e `const` são do escopo de bloco. Uma variável declarada com `let` e `const` não pode ser re-declarada dentro do mesmo escopo.

`let` pode receber outro valor no decorrer do código, já `const` não pode ser retribuída depois da sua declaração



The background is a solid dark blue. It features decorative geometric lines in teal and purple. In the top-left corner, there are several overlapping lines forming a complex shape. In the bottom-right corner, there are more lines forming a similar but less complex shape.

Tipos de dados primitivos

Tipos de dados primitivos

/

Os tipos de dados primitivos do JavaScript são:

- Strings
- Numbers
- BigInts
- Booleans
- Null
- Undefined

Strings

/

É a representação de textos, caracteres. Declarada utilizando aspas simples, aspas duplas ou crase.

Propriedades:

String.length => propriedade que retorna o tamanho da string

Numbers

/

É o tipo para números

Principais propriedades:

Number.NaN => valor especial que representa que não é um número

Number.MAX_SAFE_INTEGER => valor do maior inteiro no JavaScript
($2^{53} - 1$)

Number.MIN_SAFE_INTEGER => valor do menor inteiro ($-(2^{53} - 1)$)

BigInts

/

Tipo de dado que representa inteiros com precisão arbitrária, com ele podemos armazenar e operar inteiros gigantes que vão além do limite de Numbers, $2^{53} - 1$

Boolean

/

Tipo de dado para true e false.

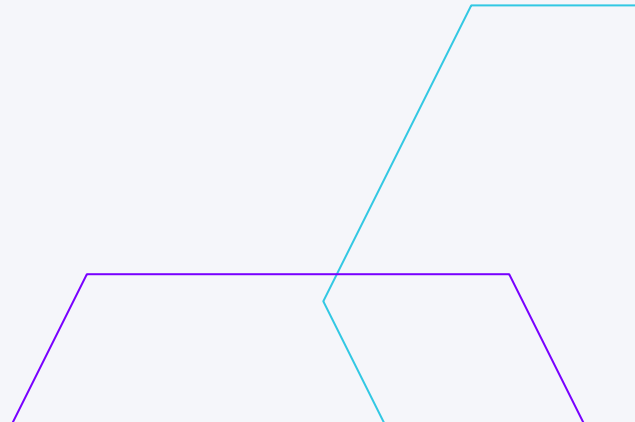
Para operações booleanas possuímos outros valores considerados como false no JavaScript: 0, ""(string vazia), null, undefined, NaN(not a number)



Null

/

É o tipo que representa nenhum valor no JavaScript.
Diferente do undefined ele deve ser atribuído a variável.

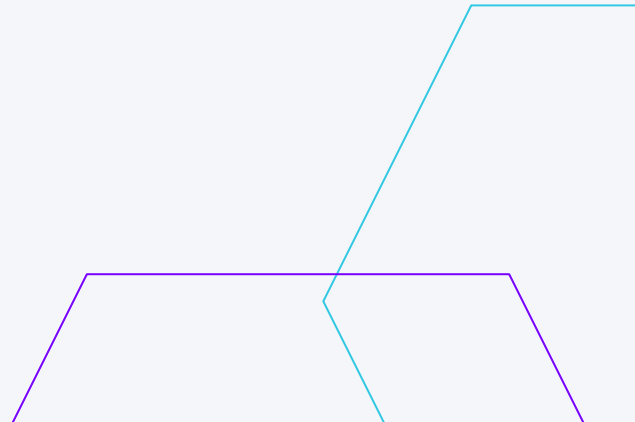




Undefined

/

É um valor primitivo que é atribuído automaticamente as variáveis que foram declaradas porém não receberam nenhum valor.



The background is a solid dark blue. It features abstract geometric line art in teal and purple. In the top-left corner, there are several connected line segments forming a jagged, angular shape. In the bottom-right corner, there is another set of connected line segments, also forming a jagged, angular shape. The lines are thin and create a modern, minimalist aesthetic.

Valores e Inferência

Valores e Inferência

/

Em ciência da computação, o tipo de dado é uma combinação de valores e operações que uma variável pode executar, varia de acordo com a linguagem e o sistema operacional

Indicam ao compilador ou interpretador como o dado deve ser armazenado

/

Tipos de dados comuns são: Inteiro, Ponto Flutuante, Character, String, Boolean.

Tipos de dados são utilizados dentro de sistemas de tipos. Existem diferentes sistemas de tipos porém os mais comuns são o de tipagem estática e o de tipagem dinâmica.

Tipagem estática é utilizado nas linguagens como C, C++, Java, nele o programador deve definir qual o tipo de dado que determinada variável irá armazenar

``int num = 45;`` => num será do tipo inteiro

``num = "quarenta"`` => vai dar erro de compilação pois a variável num só pode receber tipos de dados inteiro




/

Tipagem dinâmica a responsabilidade de dizer o tipo de dado fica com o interpretador, é utilizado em linguagens como Python, JavaScript, Ruby.

``num = 45` => num será do tipo inteiro`

``num = "quarenta"` => num será do tipo string`



The image features a dark blue background with white text. In the top-left corner, there are several overlapping geometric shapes (hexagons and lines) in light blue and purple. Similarly, in the bottom-right corner, there are more overlapping geometric shapes in the same colors. The text is centered and reads "Tipos de dados não-primitivos".

Tipos de dados não-primitivos

Tipos de dados não-primitivos

/

A classe **Object** representa um dos tipos de dados do JavaScript. É utilizado para armazenar várias coleções de chave e valor e entidades mais complexas

Objetos podem ser declarados utilizando o constructor `Object()` ou a sintaxe literal (utilizando o abre e fecha chaves)

/

Todos os objetos no JavaScript são instâncias de Object, um objeto herda propriedades do Object.prototype.

,

```
let o = new Object()
let Person = {
  name: 'React'
}
console.log(Person.name)
let User = {
  email: 'user@email',
  login: () => return 'OK'
}
console.log(User.login())
```

,

/

A classe Array do Javascript é um objeto que é usado na construção de arrays

Arrays são objetos tipo lista, não possuem tamanho e nem tipos fixos, podendo mudar durante a execução do programa

,

```
let fruits = ['maça', 'banana']  
console.log(fruits.length)
```

```
let array = [123, 'carro', { name: 'react' }]  
console.log(array)  
console.log(array[2])
```

,

The background is a solid dark blue. It features abstract geometric lines in teal and purple. In the top-left corner, there are several overlapping lines forming a series of connected segments. In the bottom-right corner, there are more lines forming a similar geometric pattern, including a small hexagonal shape.

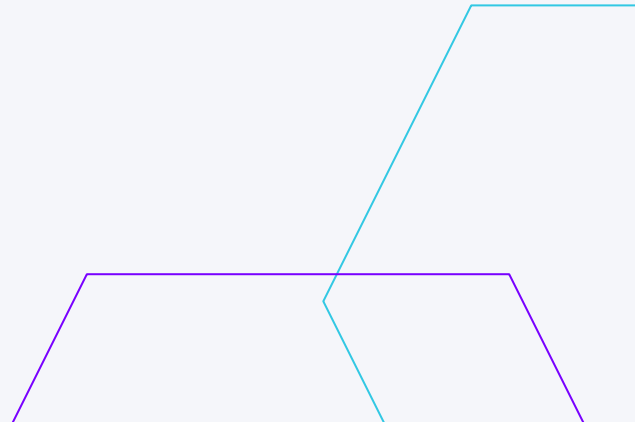
Funções



Funções

/

No JavaScript as funções são tratados como membros de primeira classe, ou seja, podem ser passadas como parâmetro para outras funções.

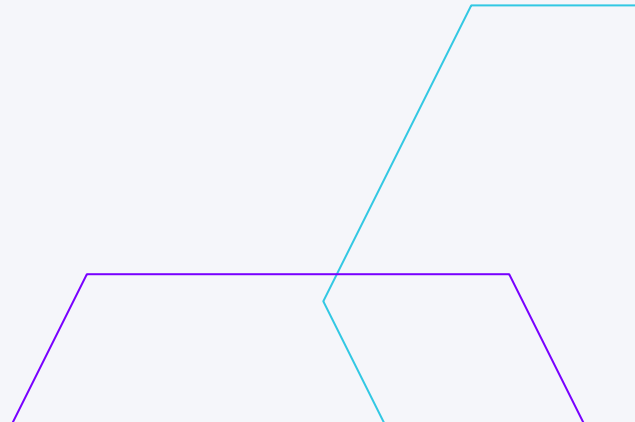




/

Podemos declarar uma função utilizando a palavra chave **function** mais o nome da função

```
`function minhaFuncao() {  
    return 'Hello'  
}  
minhaFuncao()  
,
```



/

Ou podemos declarar uma função anônima utilizando a sintaxe da arrow function

,

```
const minhaFuncao = () => { return 'Hello' }  
minhaFuncao()
```

,



/

Podemos passar parâmetros para as nossas funções, e eles podem ser de qualquer tipo de dados

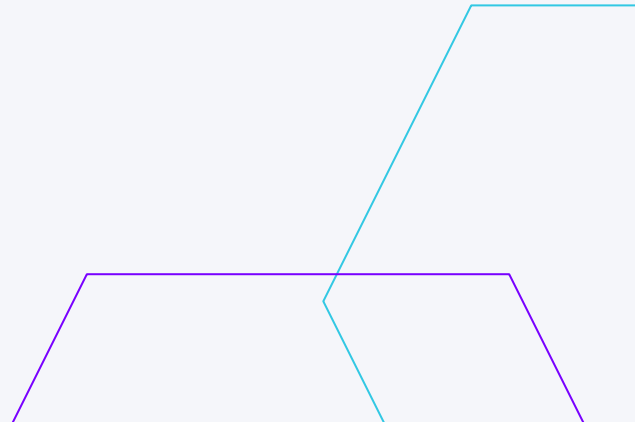
```
`function calc(num1, num2) {  
    return num1 + num2  
}
```

```
calc(23,12)
```

,

```
`function meuNome(name) {  
    return "Ola " + name  
}
```

```
meuNome('React')
```



,

Podemos passar funções como parâmetro para as nossas funções. Chamamos essas funções de callbacks, ou seja, funções que serão executadas após uma computação.

```
`function print(num) {  
    console.log(num)  
}
```

```
function calc(func) {  
    const num1 = 23  
    const num2 = 12  
    func(num1 + num2)  
}
```

```
calc(print)`
```

Hora do código

design: Calculadora Terminal

código: [github](#)

Referências

/

<https://en.wikipedia.org/wiki/JavaScript>

[https://en.wikipedia.org/wiki/JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine)

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global Objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects)



Próximos tópicos:

/

Tópico 1: Testes

Tópico 2: Estruturas de controle e repetição

Tópico 3: Closures

Tópico 4: O valor This

Tópico 5: Promises

Tópico 6: JavaScript Orientado a Objetos

Tópico 7: JavaScript Funcional

