ES6+
Variáveis Compostas

DEVinHouse
Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA | M1S03-A1

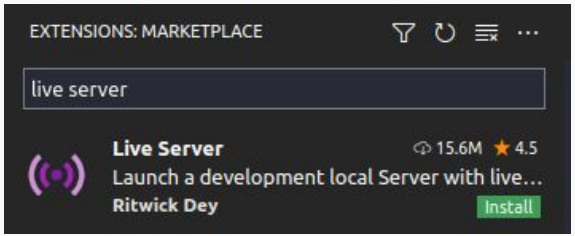
- **Revisão**
  - Variáveis
  - Tipos de dados
  - Operadores Aritméticos
  - Operadores Lógicos
  - Operadores Relacionais
  - Controle de fluxo Condicional
  - Controle de fluxo de Repetição
- **ES6 (ECMA Script 2015)**
  - ECMA
  - ES6 & ES6+
  - Variáveis Compostas

REVISÃO

- **Variáveis**: Espaço de memória nomeado (começa com **letras**, **$** ou _)
- **Tipos de dados**: **string**, **number**, **boolean**, **object**, **function**, **undefined**
  array (object), **null** (object), **NaN** (number)
- **Operadores**:
  - Aritméticos: `+` `-` `*` `/` `%` `**`
  - Atribuição: `=` `+=` `-=` `*=` `/=` `%=` `++` `--`
  - Relacionais: `<` `>` `<=` `>=` `==` `!=` `===` `!==`
  - Lógicos: `!` `||` `&&`
  - Ternário: `a ? b : c`
- **Controle de fluxo condicional**: `if`, `else if`, `else`, `switch case`
- **Controle de fluxo de repetição**: `while`, `do while`, `for`

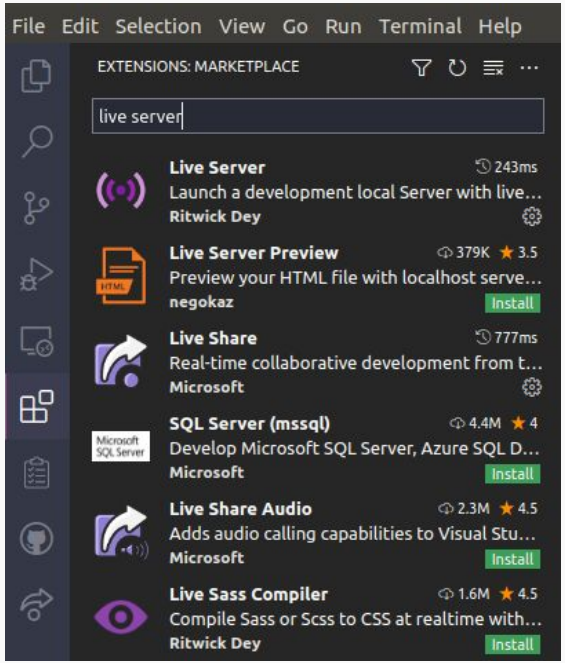
PARA A MÃO NA MASSA

- Instalar **VS Code**
  (ou outro editor que se sentir mais confortável)
  https://code.visualstudio.com
- Sugestão: Instalar extensão **Live Server** no **VS Code**
- Criar um arquivo **index.html** no seu editor

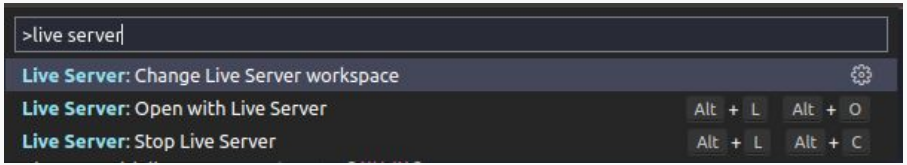


Code Sandbox | https://codesandbox.io
PlayCode | https://playcode.io/new
CodePen | https://codepen.io/pen
JSFiddle | https://jsfiddle.net

PARA A MÃO NA MASSA



- **Ctrl** + **Shift** + **P**
  Live Server: Open with Live Server

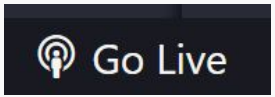


Start

- **Alt** + **L**
- **Alt** + **O**

Stop

- **Alt** + **L**
- **Alt** + **C**



ECMA

ECMA

- **E**uropean **C**omputer **M**anufacturers **A**ssociation (**ECMA**)
- Assumiu o nome de apenas **ECMA** para ser considerada “global”
- Organização sem fins lucrativos que padroniza informação e sistemas de comunicação (um deles, o **ECMAScript**)

JavaScript × ECMAScript

- Na década de 90, o **JavaScript** foi absorvido pela maioria dos navegadores, mas não tinha um padrão (cada navegador fazia sua implementação);
- Com o objetivo de padronizar (para melhorar a vida de nós, desenvolvedores), a **ECMA** recebe a responsabilidade de fazer uma padronização da linguagem.
- Chamamos essa padronização (aceita pela maioria dos navegadores), de **ECMAScript**.

ES6

ES6 | ECMA Script 2015

- **ES6**: **6ª** edição do **ECMAScript** (também conhecida como **ES2015**);
- Mudanças significativas:
  - Declaração de **classes** com sintaxe mais amigável para desenvolvedores de linguagens com orientação a objeto baseada em classes
  - ES6 **Modules**: **`import`**, **`export`**
  - **`for`** ... **`of`** loops
  - **Generators**
  - **Arrow functions**
  - **`let`**, **`const`**
  - etc

ES6+ | ECMA Script 2015~HOJE

- A partir do **ES6**, as mudanças foram mais suaves, seguindo direções estabelecidas nesta revisão da linguagem.

- Convencionou-se usar o sinal “**+**” para se referir às mudanças implementadas no ES da sexta edição em diante.

ES6+ | ECMA Script 2015~HOJE

- Variáveis: **`let`**, **`const`**
- Arrow Functions: **`() => {}`**
- Promises (promessas)
- Parâmetros padrão:
  `function (a = 10) {}`
- Rest: **`function (...args) {}`**
- Operadores **`**`** e **`**=`**
- **`String`** e **`Array.includes()`**
- **`Object.entries()`** e **`.values()`**
- Funções **`async await`**
- Destruct:
  `let { a, b, ...rest } = obj`
- Spread:
  `let obj = { a: 'b', ...o }`
- Módulos **`import export`**
- Optional chaining **`?.`**
- Nullish Coalescing Operator **`??`**
- **`...`**

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

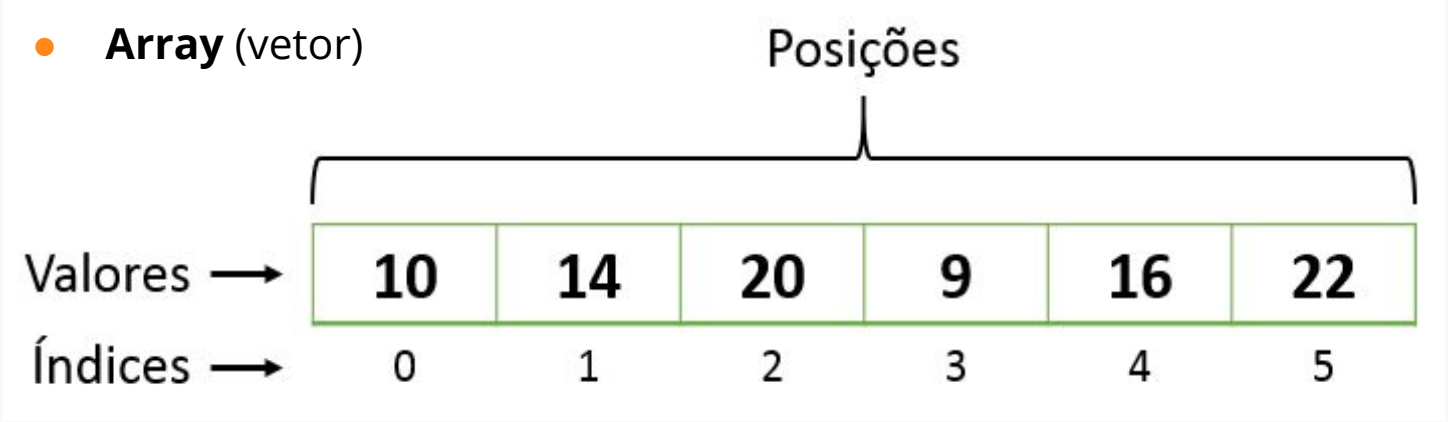
Início: 20:20
Retorno: 20:40

VETORES

VETORES | Arrays

- **Vetores** (*arrays*): são sequências de dados (itens), cada um destes itens da sequência está acessível por um índice numérico de **`0`** a **`N`** (`0`, `1`, `2` ... `N`).
- **Índice** (*index*): Chamamos os índices de um *array* de *indexes*. Quando inserimos um item em um *array*, este pode ser acessado através de seu *index* referente à sua posição ocupada no vetor.
- **O acesso** se dá através de um número inteiro após o nome do *array*, inserido entre colchetes começando em zero (`0`).
  Ex.1: `lista[0]; // 1º item`
  Ex.2: `cursos[42]; // 43º item`

- **Array** (vetor)



VETORES | Exemplos

```
// inicia um vetor com 3 itens
var vetor = [26, 33, 42];

// acessa o segundo item do vetor (33)
vetor[1];    // 33

// acessa o primeiro item do vetor (26)
vetor[0];    // 26
```

Exemplo de vetor

VETORES | Exemplos

```
// inicia um vetor vazio
var vetor = [];

// insere item no vetor
vetor.push(42);

// altera 1º item no vetor
vetor[0] = 33;

// acessa o 1º item do vetor
vetor[0];  // 33
```

Exemplo de uso de vetor (array)

```
// inicia um vetor com vetores
var vetor = [
  [26, 32, 42],
  [55, 99, 11]
]

// acessa chave e índice
vetor[0][2]    // 42
vetor[1][0] // 55
```

Exemplo de uso de vetor com vetores em suas chaves

OBJETOS

OBJETOS

- **Objetos**: são parecidos com vetores, porém, ao invés de estarmos limitados aos índices numéricos de **`0`** a **`N`** (`0`, 1, 2 ... `N`), podemos dar nomes aos índices. Inicializamos um objeto com "**`{}`**" ao invés de "**`[]`**".
- **Chaves** (*Keys*): Chamamos os índices de um objeto de chaves. Quando inserimos um item em um objeto criamos um par chave-valor (*key-value*).
- **O acesso** se dá através de um "**`.`**" (ponto) após o nome do objeto, seguido do nome da chave.
  Ex.1: **`pessoa.nome;`**
  Ex.2: **`pessoa['nome'];`**

OBJETOS | Exemplos

```
// inicia um vetor com 3 itens
var vetor = [26, 33, 42];
// acessa o segundo item do vetor (33)
vetor[1];    // 33

// inicia um objeto com 3 chaves
var objeto = { a: 26, b: 33, c: 42 };

// acessa a segunda chave do objeto
objeto.b;     // 33
objeto['b'];  // 33
```

Exemplo de vetor e objeto

OBJETOS | Exemplos

```
// inicia um vetor vazio
var vetor = [];

// insere item no vetor
vetor.push(42);

// altera 1º item no vetor
vetor[0] = 33;

// acessa o 1º item do vetor
vetor[0];  // 33
```

Exemplo de uso de vetor (array)

```
// inicia um objeto vazio
var objeto = {};

// insere/altera uma chave no objeto
objeto.num = 33;
objeto['num']  = 42;

// acessa a chave "num" do objeto
objeto.num;     // 42
objeto['num'];  // 42
```

Exemplo de uso de objeto (object)

OBJETOS | Exemplos

```
// inicia um vetor de objetos
var vetor = [
  { a: 26, b: 32, c: 42 },
  { a: 55, b: 99, c: 11 }
]

// acessa índice e chave
vetor[1].c     // 11
vetor[0]['a']  // 26
```

Exemplo de uso de vetor de objetos

```
// inicia um objeto com vetores
var objeto = {
  a: [26, 32, 42],
  b: [55, 99, 11]
}

// acessa chave e índice
objeto.a[2]    // 42
objeto['b'][0] // 55
```

Exemplo de uso de objeto com vetores em suas chaves

MATERIAL COMPLEMENTAR

ES6 Tutorial: Learn Modern JavaScript in 1 Hour | https://youtu.be/NCwa_xi0Uuc

Variáveis Compostas - Curso JavaScript #15 | https://youtu.be/XdkW62tkAgU

Curso de Javascript - #23 Objetos | https://youtu.be/kqE3sp_7peQ

Modern JavaScript Tutorial #5 - Objects | https://youtu.be/X0ipw1k7ygU

MATERIAL COMPLEMENTAR

Javascript ES6 | https://www.w3schools.com/js/js_es6.asp

ECMA Script | https://en.wikipedia.org/wiki/ECMAScript

Arrays | https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/Arrays

Array - Javascript | https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array

O básico sobre objetos | https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/Basics

Objeto - JavaScript | developer.mozilla.org/docs/Web/JavaScript/Reference/Global_Objects/Object

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>