ESTRUTURAS DE REPETIÇÃO, FUNÇÕES & EVENTOS

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

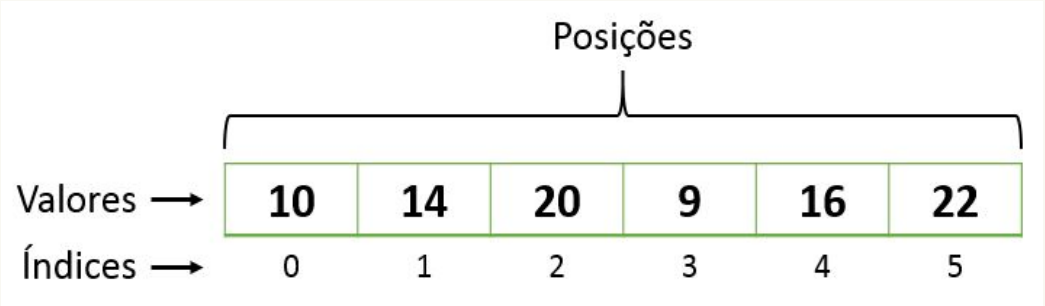
AGENDA | M1S03-A2

- **Revisão** (Estruturas de Controle de Fluxo)
  - if, else, else if, switch case, ternário
  - Array (vetor)
- **Estruturas de repetição**
  - Repetição (while / do while / for)
  - Variações do for (for in / for of / forEach)
- **Funções**
- **Eventos**

REVISÃO | Estruturas de Controle de Fluxo

- **Estruturas de controle de Fluxo**:
  - if
  - else
  - else if
  - switch
  - ternário

- **Array** (vetor)

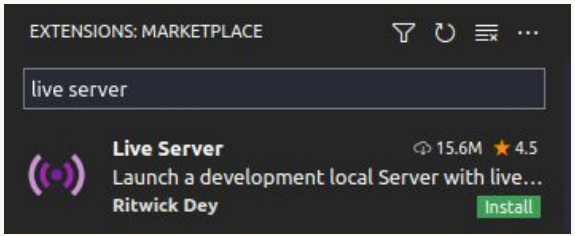


PARA A MÃO NA MASSA

- Instalar **VS Code**
  (ou outro editor que se sentir mais confortável)
  https://code.visualstudio.com

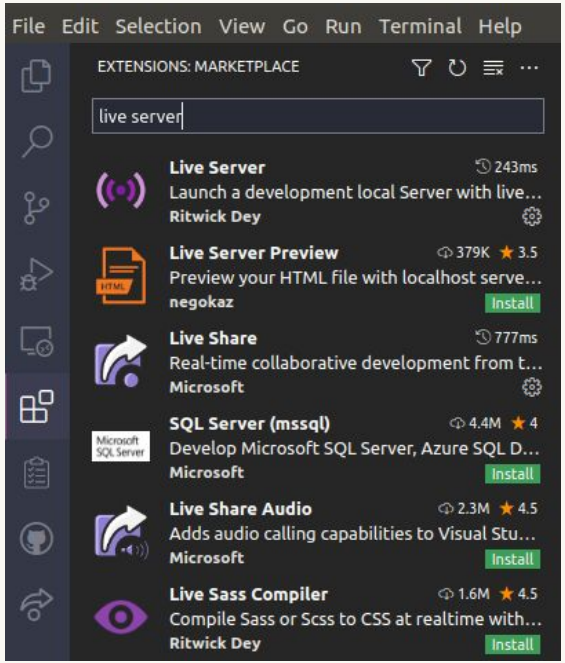
- Sugestão: Instalar extensão **Live Server** no **VS Code**

- Criar um arquivo **index.html** no seu editor

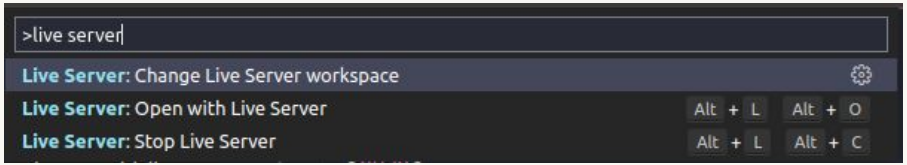


Code Sandbox | https://codesandbox.io
PlayCode | https://playcode.io/new
CodePen | https://codepen.io/pen
JSFiddle | https://jsfiddle.net

PARA A MÃO NA MASSA



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

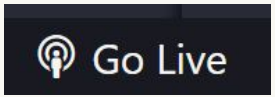


Start

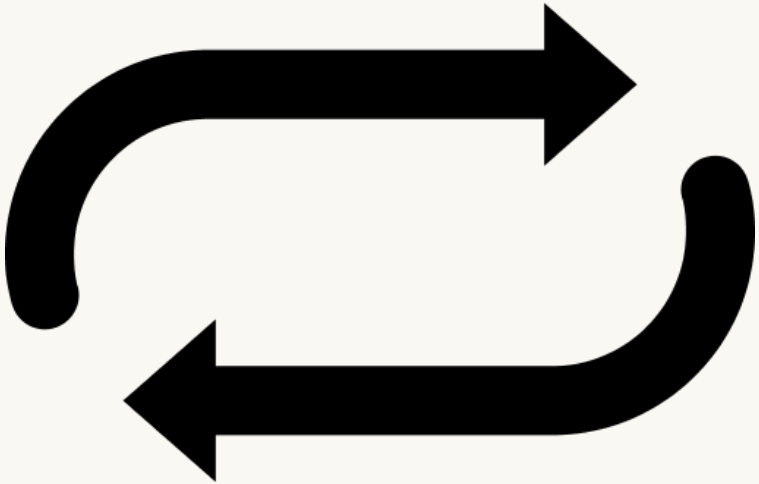
- **Alt** + **L**
- **Alt** + **O**

Stop

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



ESTRUTURAS DE REPETIÇÃO



ESTRUTURAS DE REPETIÇÃO

Às vezes nos deparamos com situações em que precisamos realizar uma **mesma ação várias vezes**.

Exemplo: Exibir na tela todos os elementos de uma lista/array.

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol'];

console.log(listaDeCarros[0]);
console.log(listaDeCarros[1]);
console.log(listaDeCarros[2]);
console.log(listaDeCarros[3]);
```

ESTRUTURAS DE REPETIÇÃO

- Mas não é sempre que sabemos exatamente quantas vezes queremos repetir essa ação.
- Muitas vezes, a quantidade de repetições que precisamos fazer é dinâmica.
- Nessas ocasiões, podemos utilizar as estruturas de repetição que as linguagens de programação nos disponibilizam.

ESTRUTURAS DE REPETIÇÃO

- **while** ("enquanto")
- **for** ("para")

ESTRUTURAS DE REPETIÇÃO | while

Exemplo de **while**:

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol'];
var cont = 0;

while (cont < 4) { // ENQUANTO a condição entre parênteses for VERDADEIRA
  alert(listaDeCarros[cont]); // EXECUTA o bloco de instruções entre chaves
  cont++;
}
```

E se a lista sofrer alterações, e novos carros forem adicionados?

ESTRUTURAS DE REPETIÇÃO | while

Melhorando nosso **while**:

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol', 'hb20', 'corsa'];
var cont = 0;

while (cont < listaDeCarros.length) { // ENQUANTO a condição for VERDADEIRA
  alert(listaDeCarros[cont]); // EXECUTA o bloco de instruções entre chaves
  cont++;
}
```

Agora, nossa repetição é dinâmica, variando de acordo com o tamanho da lista.

ESTRUTURAS DE REPETIÇÃO | do while

While invertido:

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol', 'hb20', 'corsa'];
var cont = 0;

do { // EXECUTA o bloco de instruções entre chaves
  console.log(listaDeCarros[cont]);
  cont++;
} while (cont < listaDeCarros.length) // ENQUANTO a condição for VERDADEIRA
```

Primeiro executa o bloco de código,
depois testa a condição de repetição.

ESTRUTURAS DE REPETIÇÃO | for

For é uma estrutura de repetição que já contém uma variável de controle (o “**cont**” que utilizamos nos exemplos anteriores).

Exemplo:

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol', 'hb20', 'corsa'];
// (início ; condição_de_repetição ; incremento)
for (var i = 0; i < listaDeCarros.length; i++) { // EXECUTA o bloco de instruções
  console.log(listaDeCarros[cont]);
  // ao FINAL de CADA repetição executa o código em INCREMENTO e TESTA a CONDIÇÃO
}
```

Facilita a escrita quando precisarmos de uma variável de controle para definir o final da nossa repetição.

ESTRUTURAS DE REPETIÇÃO | for in

For In é um método de iteração em objetos, vetores e strings.

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol'];

for (var index in listaDeCarros) { // index = índice do ítem do vetor
  console.log(listaDeCarros[index]);
}
```

```
var listaDeCarros = { marca: 'ford', modelo: 'escort', ano: '1996' };

for (var key in listaDeCarros) { // key = chave do ítem do objeto
  console.log(`${chave} - ${listaDeCarros[key]}`);
}
```

ESTRUTURAS DE REPETIÇÃO | break continue

- **break** "quebra" o loop, "sai fora"
- **continue** apenas "quebra" a iteração atual, continuando o loop

```
var listaDeCarros = ['uno', 'fusca', 'escort', 'gol'];

for (var carro of listaDeCarros) {
  if (carro === 'uno') continue;
  if (carro === 'escort') break;
  console.log(carro);
}
```

ESTRUTURAS DE REPETIÇÃO | forEach

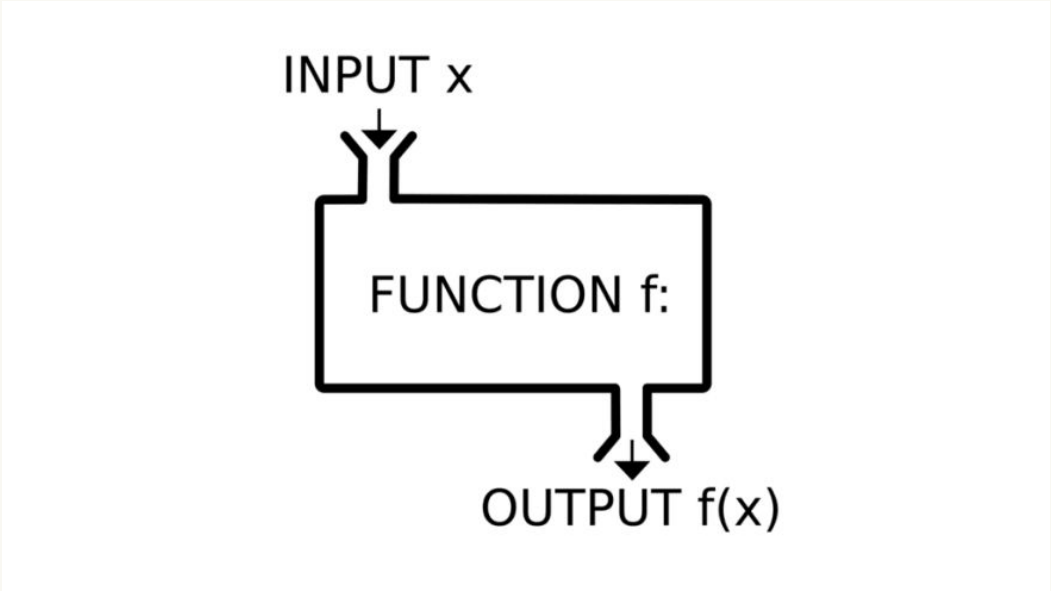
ForEach é um método de iteração em arrays (vetores).

```
const listaDeCarros = ['uno', 'fusca', 'escort', 'gol', 'hb20', 'corsa'];

listaDeCarros.forEach(
  function (nomeDoCarro, indice) {
    console.log(`${indice} - ${nomeDoCarro}`);
  }
)
```

Facilita a escrita do código quando não precisamos manipular a variável de controle.

FUNÇÕES



FUNÇÕES

- Tarefas, rotinas, ações que são executadas apenas quando invocadas (chamadas);
- Podem receber parâmetros, separados por vírgulas;
- E retornar um resultado, se necessário

```
function nomeDaFuncao(parametros) {
  /*
    Sequência de instruções
    a serem executadas
    quando a função for chamada
    Exemplo:
  */
  var valorResultado = parametros * 2;

  return valorResultado;
}
```

FUNÇÕES

```
// Exemplo 1
function somaDoisValores(a, b) {
  var resultadoDaSoma = a + b;
  return resultadoDaSoma;
}

var retorno = somaDoisValores(3, 5);

console.log(retorno);
// 8
```

```
// Exemplo 2
function somaDoisValores(a, b) {
  return a + b;
}

console.log(
  somaDoisValores(3, 5)
);
// 8
```

FUNÇÕES

- Uma função também pode ser armazenada em uma variável;
- A função é um tipo de dados function;

```
typeof contaAteN
// 'function'
```

```
// Exemplo
var contaAteN = function (n) {
  for (var i = 0; i <= n; i++) {
    console.log(n)
  }
}

contaAteN(6);
// 0 1 2 3 4 5 6
```

FUNÇÕES

- Uma função também pode ser armazenada em uma variável;
- A função é um tipo de dados function;

```
typeof contaAteN
// 'function'
```

```
// Exemplo
var contaAteN = function (n) {
  for (var i = 0; i <= n; i++) {
    console.log(n)
  }
}

contaAteN(6);
// 0 1 2 3 4 5 6
```

EVENTOS

EVENTOS

- Ações ou ocorrências que acontecem no sistema que estamos desenvolvendo;
- Podem servir como gatilhos para executar algum script;
- Podemos configurar escutadores de evento (Event Listeners)

```
<button onclick="mostraOi()">
  Me clique
</button>

<script>
  function mostraOi() {
    alert('Oi!')
  }
</script>
```

EVENTOS

Eventos em forma de atributo

- `onclick=""`
- `onchange=""`
- `onmouseover=""`
- `onmouseout=""`
- `onkeydown=""`
- `onkeyup=""`
- `onkeypress=""`
- ...

Eventos em forma de string

- `'click'`
- `'change'`
- `'mouseover'`
- `'mouseout'`
- `'keydown'`
- `'keyup'`
- `'keypress'`
- ...

Event reference | https://developer.mozilla.org/pt-BR/docs/Web/Events

EVENTOS | Event Listeners

Podemos definir listeners por atributos HTML (não use esse):

```
// função para exemplo
function mostraOlaDev() {
  alert('Olá, dev!');
}
```

```
// função para exemplo
function mostraEvento(evento) {
  console.log(evento);
}
```

```
<p onclick="mostraOlaDev()">Clique aqui</p>

<input type="text" onkeydown="mostraEvento(event)" />

// precisamos colocar o 'on' na frente do nome do evento
```

EVENTOS | Event Listeners

Podemos definir listeners por atributos do elemento do DOM:

```
<!-- html exemplo -->
<p id="paragrafo">Clique aqui</p>
<input id="entrada" type="text" />
```

```
var pElement = document.getElementById('paragrafo');
var inputElement = document.getElementById('entrada');

pElement.onclick = mostraOlaDev;
inputElement.onkeydown = mostraEvento;
// precisamos colocar o 'on' na frente do nome do evento
```

EVENTOS | Event Listeners

Podemos definir a função diretamente na igualdade:

```
pElement.onclick = function () {
  mostraOlaDev();
};


inputElement.onkeydown = function (event) {
  mostraEvento(event);
};
```

```
// mostraOlaDev
pElement.onclick = function () {
  alert('Olá, dev!');
};

// mostraEvento*
inputElement.onkeydown = function (e) {
  console.log(e);
};
```

*O navegador fornece à função o objeto do evento por parâmetro

EVENTOS | Event Listeners

Podemos definir listeners pelo método addEventListener:

```
<!-- html exemplo -->
<p id="paragrafo">Clique aqui</p>
<input id="entrada" type="text" />
```

```
var pElement = document.getElementById('paragrafo');
var inputElement = document.getElementById('entrada');

pElement.addEventListener('click', mostraOlaDev);
inputElement.addEventListener('keydown', mostraEvento);
// neste caso não precisamos colocar o 'on' na frente
```

EVENTOS | Event Listeners

Podemos definir uma função diretamente no parâmetro:

```
pElement.addEventListener(
  'click',
  function () {
     mostraOlaDev();
  }
);

inputElement.addEventListener(
  'keydown',
  function (ev) {
    mostraEvento(ev);
  }
);
```

```
pElement.addEventListener(
  'click',
  function () {
    alert('Olá, dev!');
  }
);

inputElement.addEventListener(
  'keydown',
  function (event) {
    console.log(event);
  }
);
```

EVENTOS | Event Listeners

Podemos também remover os eventListeners:

```
// adicionando
pElement.addEventListener('click', mostraOlaDev);
inputElement.addEventListener('keydown', mostraEvento);

// removendo
pElement.removeEventListener('click', mostraOlaDev);
inputElement.removeEventListener('keydown', mostraEvento);
```

EVENTOS | Prevent Default

Podemos evitar o comportamento padrão do navegador:

```
// chamando o método preventDefault do evento
function exemplo(evento) {
  evento.preventDefault()
}
```

MATERIAL COMPLEMENTAR

JavaScript Loops | https://youtu.be/s9wW2PpJsmQ

Repetições (Parte 1) - Curso JavaScript #13 | https://youtu.be/5rZqYPKIwkY

Repetições (Parte 2) - Curso JavaScript #14 | https://youtu.be/eX-lkN_Zahc

for in / for of - Beau teaches JavaScript | https://youtu.be/a3KHBqH7njs

8 Formas de usar Looping em Arrays no JavaScript | https://youtu.be/NfHVPEzo5Ik

JavaScript Functions | https://youtu.be/N8ap4k_1QEQ

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

Funções - Curso JavaScript #16 | https://youtu.be/mc3TKp2XzhI

Eventos DOM - Curso JavaScript #10 | https://youtu.be/wWnBB-mZIvY

MATERIAL COMPLEMENTAR

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

Laços e iterações | https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Loops_and_iteration

Estruturas condicionais e de repetição em JS | treinaweb.com.br/blog/estruturas-condicionais-e-estruturas-de-repeticao-em-javascript

JavaScript While Loop | https://www.w3schools.com/js/js_loop_while.asp

JavaScript For Loop | https://www.w3schools.com/js/js_loop_for.asp

JavaScript For In | https://www.w3schools.com/js/js_loop_forin.asp

JavaScript For Of | https://www.w3schools.com/js/js_loop_forof.asp

JavaScript Break and Continue | https://www.w3schools.com/js/js_break.asp

Break | https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/break

Continue | https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/continue

JavaScript Functions | https://www.w3schools.com/js/js_functions.asp

JavaScript Events | https://www.w3schools.com/js/js_events.asp

Introdução a Eventos | https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/Events

Event reference | https://developer.mozilla.org/pt-BR/docs/Web/Events

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>