

Condicionais, funções e loop



Condicionais

No JavaScript as instruções são executadas em ordem (linha 1, depois a linha 2, depois a linha 3, etc). Mas podemos fazer com que o nosso programa execute algumas linhas apenas **se** uma condição for verdadeira.

Para isso temos o **if**, **else if** e **else**

```
const aluno = "Gabriel";
const idade = 20;

if (idade < 20) {
  console.log(aluno + " Cordeiro");
} else if (idade < 42) {
  console.log(aluno + " dos Santos");
} else {
  console.log(aluno + " Araujo");
}

console.log("top")
```

O que isso vai logar?

Condicionais

O código dentro do if será executado sempre que a condição dentro de () for **true** e vai ignorar os "elses" daquele bloco.

Se for **false** ele irá para o próximo else. Caso nenhum if seja atendido o ultimo else será executado.



```
const aluno = "Gabriel";  
const idade = 60;  
  
if (true) {  
  console.log(aluno, " Cordeiro");  
} else if (idade < 42) {  
  console.log(aluno, " dos Santos");  
} else {  
  console.log(aluno, " Araujo");  
}
```

isso vai logar "Gabriel Cordeiro" porque ele vai entrar no primeiro if e nem vai testar ou outros else-

Operadores lógicos e comparações

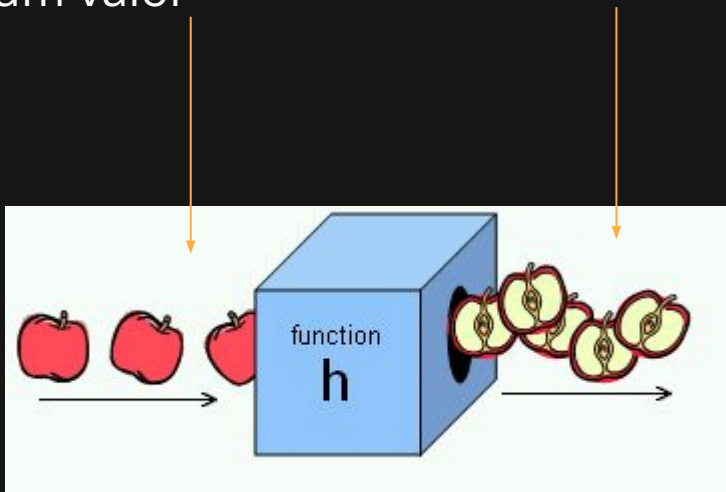
Os principais são

	ou (uma condição ou a outra serem verdadeiras)
&&	e (uma condição e a outra serem verdadeiras)
>	maior que
<	menor que
===	igual em valor e tipo
!==	diferente em valor ou tipo

Funções

Funções são pedaços de código que você pode reutilizar

Elas podem receber variáveis (conhecidas como parâmetros ou argumentos) e retornar algum valor



Funções



```
function iniciarAula (nomeAluno) {  
  return "Olá " + nomeAluno + ", a aula começou!"  
}  
  
const iniciarAula = function (nomeAluno) {  
  return "Olá " + nomeAluno + ", a aula começou!"  
}  
  
const iniciarAula = (nomeAluno) => {  
  return "Olá " + nomeAluno + ", a aula começou!"  
}
```

Pode-se criar funções dessas formas

declaração de função

function nomeDaFuncao() {}

expressão de função

const nomeDaFuncao = function () {}


const nomeDaFuncao = () => {}

funções

anônimas

arrow function

Funções



```
function iniciarAula (nomeAluno) {  
    return "Olá " + nomeAluno + ", a aula começou!"  
}
```

function é a palavra reservada que indica a declaração de uma função.

iniciarAula é o nome que demos a nossa função.

nomeAluno é o parâmetro que essa função recebe quando ela é invocada.

return indica o valor que será recebido por quem invocar essa função.

Como executo uma função?

- Lembra do console.log? Aquilo é uma função* e você a executa sempre que faz console.log()
- Tudo que coloca dentro das () são argumentos que você está passando para a função
- O que o código ao lado vai logar?



```
const mensagem = iniciarAula("Alan");  
console.log(mensagem)
```

*na verdade é um método, uma função ligada a um objeto

Nomenclatura

Javascript é case-sensitive, ou seja, tem diferença entre **variavel** e **Variavel**

Convenção

Em JS nomeia-se funções e variáveis em camelCase

nomes com mais de uma palavra tem a primeira letra da primeira palavra minúscula e das outras palavras, maiúscula (exemplo: minhaFuncao)



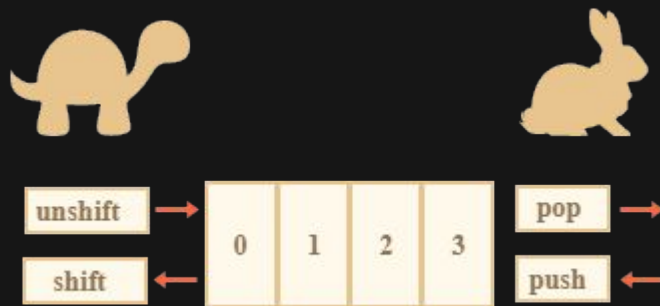
Arrays

Array é um tipo de dado que guarda um conjunto de informações que tem uma **sequência**, uma **ordem**. Os itens de um array podem ser de qualquer tipo e é possível acessar um item pelo seu **index**.



```
const meuArray = [1, "oi", false, [2, 3]];
```

```
meuArray[0] // 1  
meuArray[1] // "oi"  
meuArray[2] // false  
meuArray[3] // [2, 3]
```



Laços de repetição - for

As vezes queremos repetir um código várias vezes, uma vez para cada item de um array por exemplo. Para isso temos laços de repetição e um deles é o **for**



```
for (let i = 0; i < 9; i++)  
{ console.log(i);  
  // more statements  
}
```

Laços de repetição - for of



```
const alunos = ["Aida", "Fernanda", "Camilla", "Rafaela"]
```

```
for (const aluno of alunos) {  
  console.log("Boa noite, " + aluno)  
}
```

```
// Boa noite, Aida  
// Boa noite, Fernanda  
// Boa noite, Camilla  
// Boa noite, Rafaela
```

Hands On

- Criar a função (**inverter**) que recebe uma string e retorna um inverso dela “salve” => “evlas”
- Criar a função (**contarNumeroVogais**) que recebe uma string e retorna o número de vogais que ela tem (maiúsculas e minúsculas) “salve” => 2
- **Extra:** Criar a função (**fazerRelatorio**) que recebe uma string e retorna um objeto com a própria string, o inverso da string e quantas vogais ela tem “salve” => { palavra: “salve”, palavraInverso: “evlas”, numeroVogais: 2 }
- **Extra2:** Se receber algo diferente de string da um console.log(“oh carai”) e retorna undefined

