Funções

FUTURE4

Big Picture e Sumário

O que vimos até agora...

- HTML: estrutura e conteúdo do site

- CSS: organização e estilização do site

- Como podemos tornar isso "vivo"?
  - Usando o **JavaScript**!
  - Para isso, precisamos do conceito de funções

O que são funções?

- Bloco de código projetado para executar uma tarefa específica

- Só executa essa tarefa quando algo a invoca
  - Veremos quais são as opções para esse algo

Por que utilizar funções?

- Reutilização de código: facilita alteração

Por que utilizar funções?

- Reutilização de código: facilita alteração

  Exemplo: Transformar reais em dólar

Por que utilizar funções?

- Abstração de partes do código: facilita leitura

Por que utilizar funções?

- Abstração de partes do código: facilita leitura

  Exemplo: Transformar graus Celsius em Fahrenheit

Em resumo...

Entrada
(Input)

Saída
(Output)

FUTURE4

Pausa 

FUTURE4

Sintaxe de Funções

Funções Nomeadas

- São criadas a partir de uma **declaração**

```
function nomeDaFuncao(parametros){
    // tarefa a ser executada
}
```

Exemplo de função nomeada

```
function dizOi(){
    console.log("Oi!");
}
```

Chamando uma função

- Para que a função execute, você deve invocá-la:

```
nomeDaFuncao();
```

Demo time 1 😈

FUTURE4

Funções com parâmetros

- Certas funções precisam de informações específicas: **parâmetros**
- Esses parâmetros funcionam como variáveis dentro do bloco da função
- Uma função pode ter mais de um parâmetro

Funções com parâmetros

```
function imprimeValorEmDolar(valorEmReais){

    const valorEmDolar = valorEmReais * 4.17;

    console.log(valorEmDolar);
}
```

Chamando uma função com parâmetros

```
imprimeValorEmDolar(100);

// ou

let valorEmReais = 100;

imprimeValorEmDolar(valorEmReais);
```

Obtendo a saída de uma função

- Certas funções **retornam** informações para o código que as chamou

- O return **interrompe** a execução da função

```
function converteParaDolar(valorEmReais){
    const valorEmDolar = valorEmReais * 4.17;
    return valorEmDolar;
}
```

Chamando uma função com saída

```
let quantiaEmReais = 100;

console.log(converteParaDolar(quantiaEmReais));

// ou

let quantiaEmDolar = converteParaDolar(quantiaEmReais);
```

Demo time 2 😈

FUTURE4

Outras Sintaxes

Funções Não Nomeadas

- São criadas a partir de uma **expressão**
- O nome é geralmente omitido

```
let nome = function(parametros){
  // tarefa a ser executada
}
```

Exemplo de função não nomeada

```
let valorEmDolar = function(valorEmReais){
    return valorEmReais * 4.17;
}

// ou
let valorEmDolar = function(valorEmReais){
    console.log(valorEmReais * 4.17);
}
```

Arrow Functions

- Apenas outra forma de fazer uma **expressão**

```
let nome = (parametros) => {
  // tarefa a ser executada
}
```

FUTURE4

Chamando uma função não nomeada

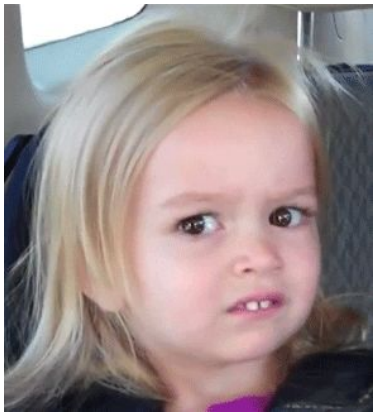
- Da mesma forma que as nomeadas:

```
nome();
```

Demo time 3 😈

FUTURE4

Então qual a diferença entre funções nomeadas e não nomeadas???



FUTURE4

Nomeada x Não nomeada

- Principal diferença: **Hoisting**
    - Nomeada: pode invocar antes de declarar
    - Não nomeada: **NÃO** pode invocar antes de declarar

Demo time 4 😈

FUTURE4

Pausa 

FUTURE4

Mas e o HTML com isso???

Veremos formas de o JS atuar junto com o HTML

FUTURE4

Eventos

- “Coisas” que acontecem com a página HTML

- Podem ser algo que:
  - O browser faz
    Ex: a página acabou de carregar

  - O usuário faz
    Ex: clique de um botão

- Usados para chamar funções JS

Eventos

- Sintaxe

```
<elemento evento="nomeDaFuncao()">
```

- Eventos comuns
  - onchange - um elemento foi alterado
  - onclick - usuário clicou no elemento
  - onmouseover - usuário posicionou mouse em cima do elemento
  - onmouseout - usuário tirou o mouse do elemento
  - onkeydown - usuário pressionou uma tecla
  - onload - browser terminou de carregar a página

Exemplo: Botão que, quando clicado, chama uma função chamada mudaCor

```
<button onclick="mudaCor()">
  Mudar cor!
</button>
```

getElementById( )

- Função que retorna o elemento com o id passado como parâmetro

- Caso não exista um elemento com o id passado, retorna null

- Sintaxe
`document.getElementById(“idDoElemento”)`

Demo time! 

Construindo a função mudaCor

FUTURE4

querySelector( )

- Função que retorna o **primeiro** elemento correspondente a um seletor CSS especificado

- Sintaxe
`document.querySelector(“seletorCSS”)`

Exemplos

```
document.querySelector("p");
// Seleciona o primeiro elemento de tag p

document.querySelector("#demo").innerHTML;
// Seleciona o elemento de id demo

document.querySelector("div > p");
//Seleciona o primeiro elemento p cujo pai é uma div
```

Demo time! 😈

Alterando a função mudaCor

Coding Together

Minha lista de compras

FUTURE4

Pausa 

FUTURE4

Review

- Funções são blocos de código que executam uma tarefa específica quando invocadas

- Sintaxes: nomeadas, não nomeadas e arrow function

- Evento: ações no site que podem servir para invocar uma função

- Seletores de elementos HTML: getElementById( ) e querySelector( )

FUTURE4

Boas práticas

- Assim como variáveis, uma função deve ter um nome que reflita seu propósito

- A função deve fazer apenas uma coisa
  - Se for possível extrair outra função de nome significativo, está fazendo mais de uma coisa

- As funções devem estar em uma ordem lógica, na medida do possível

Mão na Massa!

Obrigada!

FUTURE4