Funções

Programação para Internet I

IFB - Tecnologia em Sistemas para Internet

Marx Gomes van der Linden

Funções

- Funções são trechos de código reutilizável
- Podem **receber argumentos** e **retornar valores**
- Palavra-chave: `function`

Exemplo

Definição

```
function hello(){
    console.log('Olá mundo!');
}
```

Chamada

```
hello();
```

Parâmetros (argumentos)

```
function hello(nome, email){

    console.log(
     `  Olá, ${nome}! Seu e-mail é ${email}`
     );

}

hello('Fulano', 'fulano@gmail.com');
```

Retorno de valores

```
function hello(nome, email){
    return `Olá, ${nome}! Seu e-mail é ${email}`;
}

let valor = hello('Beltrano', 'beltrano@gmail');
console.log('O valor retornado foi ' + valor);
```

Escopo de variáveis

- Escopo: em que parte do programa a variável existe
- Nenhuma variável existe fora da função em que é definida (**escopo local**)
- Variáveis definidas fora de função valem para todo o script (**escopo global**)

Exemplo

```
let x = 10;

teste();

function teste(){
    console.log(x);
}
```

Exemplo

```
let x = 10;

teste();

console.log(x);

function teste(){
    let x = 20;
    console.log(x);
}
```

Diferença entre `var` e `let`

```
let i = 3;
for (let i = 0; i < 6; i++) {
  console.log('Loop: ' + i);
}

console.log('Resultado final: ' + i);
```

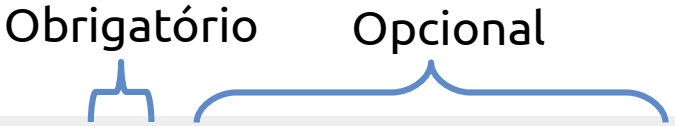
```
var i = 3;
for (var i = 0; i < 6; i++) {
  console.log('Loop: ' + i);
}

console.log('Resultado final: ' + i);
```

Diferenças entre `var` e `let`

- **`var`**
  - » Hoisting (declaração é movida para o início)
  - » Escopo de **função**
- **`let`**
  - » Não faz hoisting
  - » Escopo de **bloco**

Parâmetros com valores-padrão

- Cada parâmetro com valor padrão se torna opcional




```
function teste(a, b = 'batata') {
    console.log("a = " + a);
    console.log("b = " + b);
}
```

- ES2015 (não funciona no Internet Explorer)

Função recursiva

- Uma função em JavaScript pode chamar a si mesma
- Sempre precisa haver uma condição que evita a recursividade!

```
function fatorial(n) {
    if(n == 0) {
        return 1;
    } else {
        return n * fatorial(n - 1);
    }
}
```

Funções internas

- Uma função pode ser definida dentro de outra função

```
externa();
//interna(); ← Erro!

function externa(){
    console.log('Testando...');
    interna();
    console.log('Pronto');

    function interna(){
        console.log('Executando a função interna');
    }
}
```