Prof. Dr. Leandro Luque
Faculdade de Tecnologia de Mogi das Cruzes

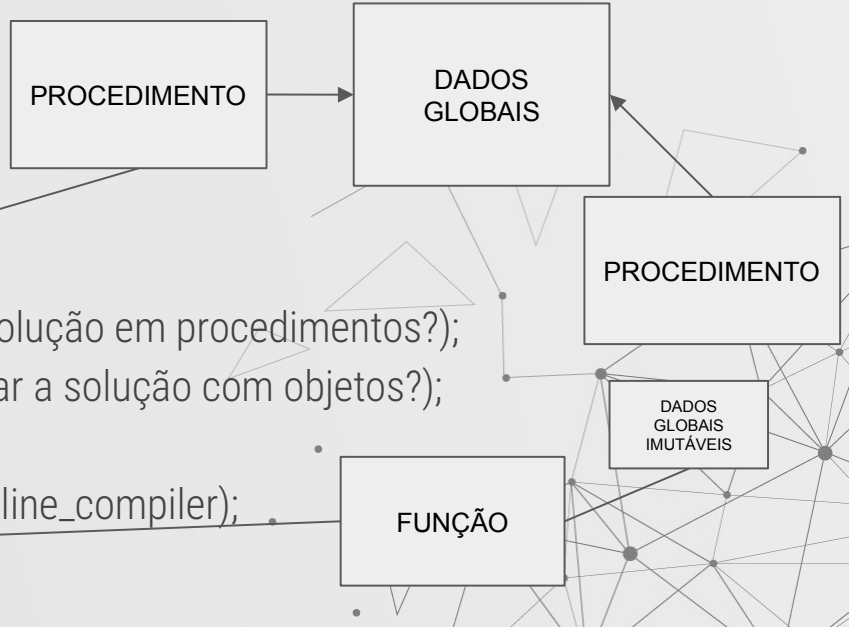
JAVASCRIPT

Programação funcional

PARADIGMAS DE PROGRAMAÇÃO

Modelo que orienta como projetamos e implementamos uma solução computacional para um problema.

- Imperativo:
  - Não-estruturado:
    - GoTo;
    - Sem estruturas de repetição;
  - Estruturado:
    - Procedimental (como organizar a solução em procedimentos?);
    - Orientado a Objetos (como organizar a solução com objetos?);
- Declarativo:
  - Lógico (https://rextester.com/l/prolog_online_compiler);
  - Funcional.



PROGRAMAÇÃO FUNCIONAL

- Mais antiga que a orientação a objetos;
- Baseada num sistema formal chamado Cálculo Lambda, equivalente a máquinas de Turing;

- Principais conceitos relacionados:
  - Declarativa;
  - Funções são cidadãos/objetos de primeira classe;
  - Funções de alta ordem (recebem e/ou retornam outras funções);
  - Valoriza:
    - Imutabilidade;
    - Funções puras;
    - Recursão (em contraponto à iteração);
  - Currying;

DECLARANDO FUNÇÕES

Função nomeada (Hoisted)

function nome(...) {...}

Expressão de função anônima

var funcao = function (...) {...};

Expressão de função nomeada

var funcao = function nome (...) {...};

Constructor Function

new Function('x', 'y', 'return x ** y');

Função flecha

Sem parâmetros:
() => 2 ** 2;
Único parâmetro:
x => x ** 2;
Mais de um parâmetro:
(x, y) => x ** y;

Com corpo:
() => { return 2 ** 2 };

PARÂMETROS E ARGUMENTOS

- Forma de passagem:
  - Objetos passados por referência;
  - Primitivos por valor;
- Valores padrões:
  - parametro = valorPadrao
- Qualquer quantidade de parâmetros:

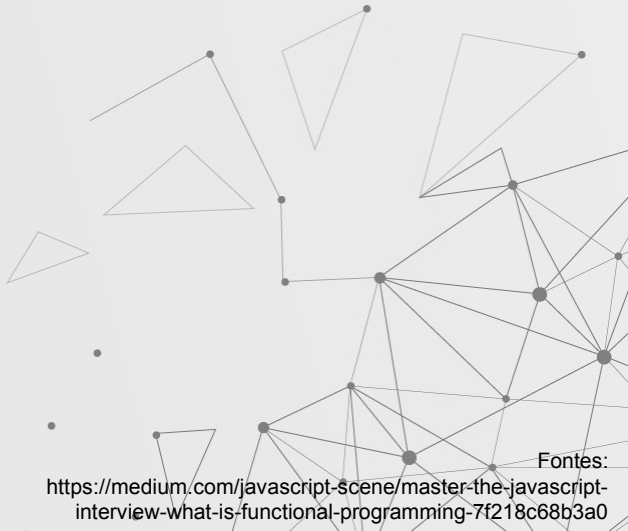
```
function soma() { // ES5
  let resultado = 0;
  for(let i = 0; i < arguments.length; i++) {
       resultado += arguments[i];
  }
       return resultado;
}
```

```
function soma(...restParameters) { // ES6
  let resultado = 0;
  for(let i = 0; i < restParameters.length; i++) {
       resultado += restParameters[i];
  }
       return resultado;
}
```

```
function soma(...restParameters) { // ES6
  return restParameters.reduce((acc, cur) => acc +
cur);
}
```

IMPERATIVO VS DECLARATIVO

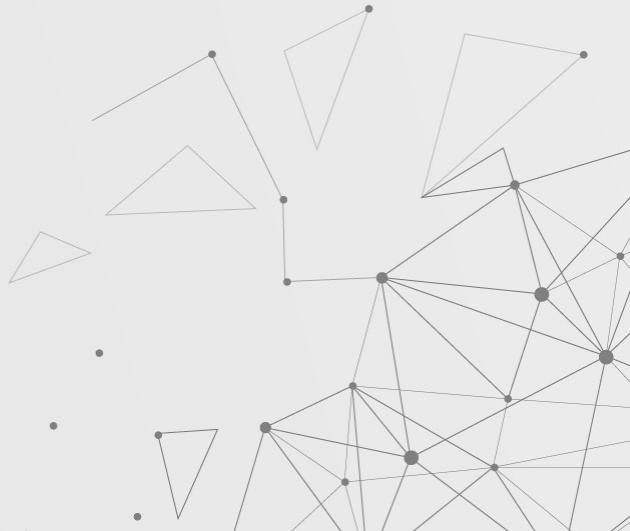
```
// A partir de uma array, gera uma nova array com todos os valores multiplicados por 2.
// IMPERATIVA.
const doubleMap = numbers => {
  const doubled = []; // O ponteiro não é modificado em nenhum lugar do código - por isso OK com
const.
  for (let i = 0; i < numbers.length; i++) {
      doubled.push(numbers[i] * 2);
  }
  return doubled;
};

console.log(doubleMap([2, 3, 4])); // [4, 6, 8]

-x-

// DECLARATIVA.
const doubleMap = numbers => numbers.map(n => n * 2);
console.log(doubleMap([2, 3, 4])); // [4, 6, 8]
```



Fontes:
https://medium.com/javascript-scene/master-the-javascript-interview-what-is-functional-programming-7f218c68b3a0

CIDADÃOS DE PRIMEIRA CLASSE E FUNÇÕES DE ALTA ORDEM CONTRA EXEMPLO DE IMUTABILIDADE

```
let array = ["abc", "def", "ghi"];
array.sort();

array.sort((a, b) => b < a ? -1 : 1);
console.log(array);



let array2 = [100, 20, 30];
array2.sort();

console.log(array2);
???
```

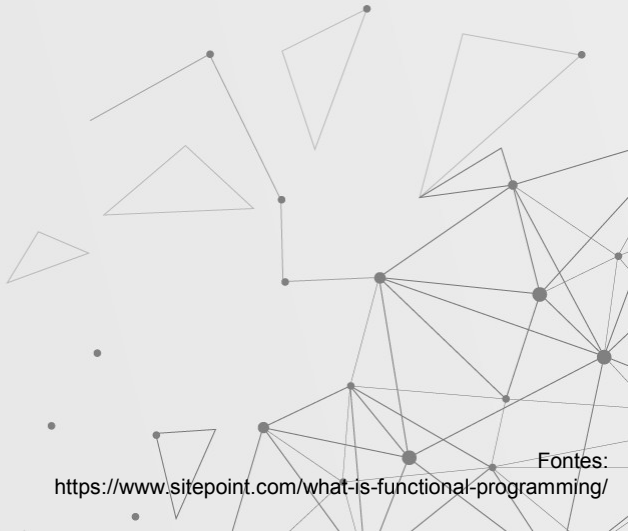


FUNÇÕES DE ALTA ORDEM (ÚTEIS)

```
const myArr = [1, 2, 3, 4, 5];
const evens = myArr.filter(x => x % 2 === 0); // [2, 4]
```

```
const myArr = [1, 2, 3, 4, 5];
const doubled = myArr.map(i => i * 2); // [2, 4, 6, 8, 10]
```

```
const myArr = [1, 2, 3, 4, 5];
const sum = myArr.reduce((i, runningSum) => i + runningSum); // 15
```

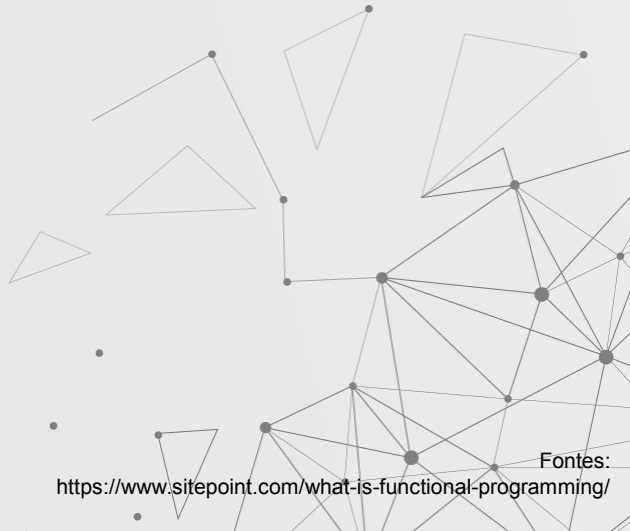


Fontes:
https://www.sitepoint.com/what-is-functional-programming/

IMUTABILIDADE

```
// Mutável
const myArr = [1, 2, 3];
myArr.pop();
// [1, 2]
```

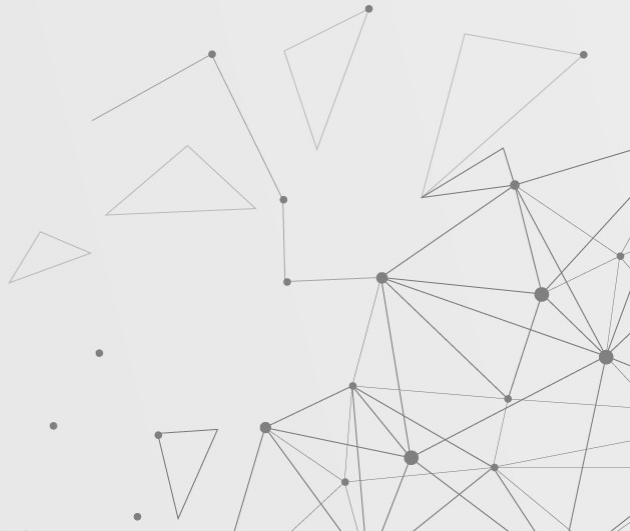
```
let myArr = [1, 2, 3];
// let myNewArr = Array.from(myArr);
// ES6: let myNewArr = [...myArr];
let myNewArr = myArr.slice(0, 2)
// [1, 2]
```



Fontes:
https://www.sitepoint.com/what-is-functional-programming/

IMUTABILIDADE

- Mutáveis:
  - Object;
    - Array;
    - Function.
- Imutáveis:
  - Todos os tipos primitivos são imutáveis;
  - string;
  - number;
  - boolean;
  - null;
  - undefined;
  - symbol.



FUNÇÕES PURAS

- OBS: (Valorização) - Nem toda função precisa/deve ser pura.

- Retornam sempre o mesmo valor quando são chamadas com os mesmos parâmetros;
- Não produzem efeitos colaterais:
  - Requisição HTTP;
  - Modificação de dados;
  - Impressão de algo na tela ou console;
  - Manipulação de DOM;
  - Geração de números aleatórios: Math.random();
  - Utilizar data/hora atual.

```
Impura:
let taxa = 20;
function calcularTaxa(precoProduto) {
    return (precoProduto * (taxa/100)) + precoProduto;
}

Pura:
function taxaPorPreco(precoProduto, taxa) {
    return (precoProduto * (taxa/100)) + precoProduto;
}
```

Fontes:
https://medium.com/@jamesjefferyuk/javascript-what-are-pure-functions-4d4d5392d49c

CURRYING

- Gerador de URLs:

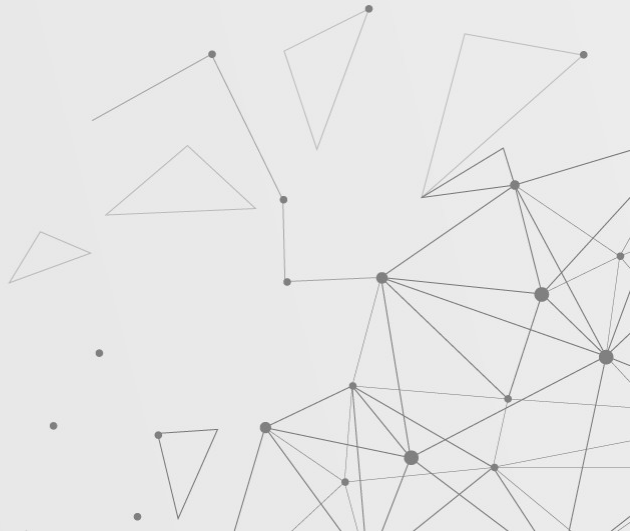
```
function construirURL (protocolo, dominio, caminho) {
  return `${protocolo}://${dominio}/${caminho}`;   // Template literals
}

var url = protocolo => dominio => porta => caminho => ${protocolo}://${dominio}:${porta}/$
{caminho}`;
var https = url(“https”);
var uolWeb = https(“uol”)(“80”);
```

Fontes:
Inspirado por: https://www.digitalocean.com/community/tutorials/javascript-functional-programming-explained-partial-application-and-currying

CLOSURE

- Escopo

```
var contador = function() {
  let contadorPrivado = 0;
  return function() {
        contadorPrivado++;
        return contadorPrivado;
  }
}
```

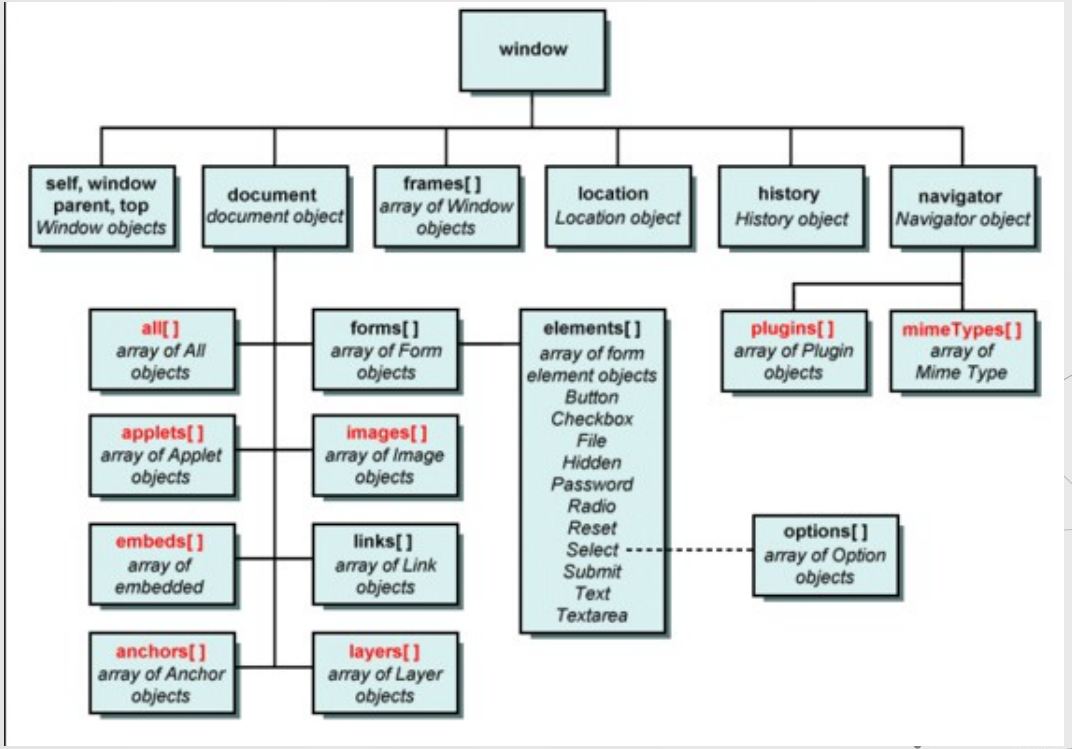


EXERCÍCIOS

1. Implementar sua versão do filter;
2. Implementar sua versão do map;
3. Implementar sua versão do reduce;
4. Implementar uma versão funcional de um botão em HTML/Javascript que, quando clicado a primeira vez, exibe um log no console. A partir da segunda vez, ele exibe outra mensagem no console;
5. Implementar uma função que retorna um novo estado de posicionamento de um jogador em um jogo (considerando seu posicionamento no plano cartesiano) - ver explicação do professor.

```
let personagem = {
    x: 10,
    y: 20
}

moverCima(moverDireita(moverDireita(personagem)));

{ // deve ser um novo objeto, sem alterar o original
    x: 12,
    y: 21
}
```

DOM



Fontes:
https://kalbojsprogramming.wordpress.com/