Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parceiros para o Crescimento
PORTUGAL DIGITAL

Innominatum CONSULTORIA

JavaScript

Módulo 2

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Innominatum CONSULTORIA

Lógica e Controlo de Fluxo

Sessão 6

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Objetivo geral

Capacitar os formandos na construção de algoritmos estruturados em JavaScript, dotando-os de competências para implementar mecanismos de tomada de decisão, gerir a iteratividade através de estruturas de repetição e aplicar estratégias de resiliência no código. Os formandos deverão ser capazes de interpretar o comportamento booleano intrínseco aos dados (Truthy & Falsy) e garantir a estabilidade da aplicação através do tratamento estruturado de exceções e erros.

Objetivos específicos

Ao final da sessão, os formandos deverão ser capazes de:

- Aplicar estruturas condicionais (**if / else, switch e operador ternário**);
- Avaliar condições com base em valores **truthy e falsy**;
- Utilizar estruturas de repetição (**for, while, do-while**);
- Controlar ciclos através das instruções **break e continue**;
- Implementar tratamento básico de erros com **try, catch e finally**.

Estruturas Condicionais (if / else)

Estruturas condicionais – definição

São blocos de código que permitem ao programa tomar decisões lógicas, executando diferentes ações com base em se uma condição é **verdadeira (true)** ou **falsa (false)**.

Tipos de condições

- If;
- If/else;
- If/else/if;
- Switch.

Condicional if

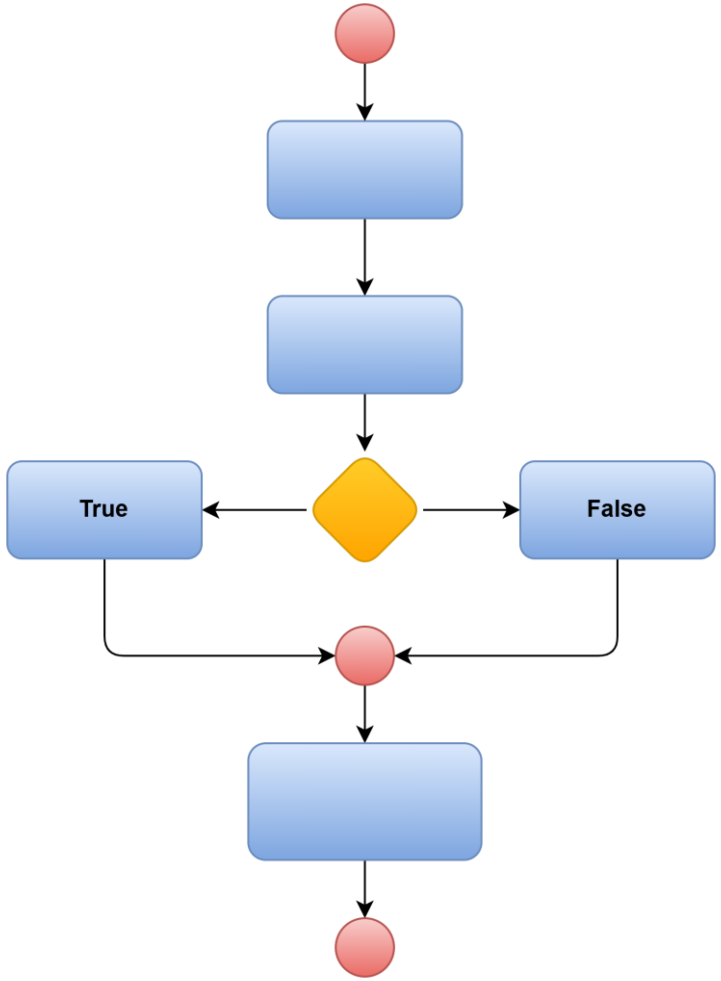
```
if (condicao) {
  // código executado se a condição for true
}
// Sintaxe básica do if.
```

```
let idade = 20;

if (idade < 18) {
    console.log("Menor de idade");
}
```

Condicional if/else

```
if (condicao) {
    true
} else {
    false
}
```



Formas de construção de condições

- Comparações Simples: Validar valores (Ex: estoque < 10; idade <= 18);
- Condições Compostas (Expressões lógicas): Unir várias regras com && ou || (Ex: possui 18 anos && permissão para conduzir);
- Verificação de Estado: Usar a própria existência do dado (Truthy/Falsy) para decidir.

Condições encadeadas

```
let nota = 15;

if (nota >= 18) {
  console.log("Excelente");
} else if (nota >= 10) {
  console.log("Aprovado");
} else {
  console.log("Reprovado");
}
// O JS para no primeiro bloco verdadeiro.
```

Ordem das condições importa

```
if (nota >= 10) {
  console.log("Aprovado");
} else if (nota >= 18) {
  console.log("Excelente");
}
// O segundo bloco nunca será executado.
// Condições devem ir do mais específico para o mais geral.
```

Blocos {} e escopo

- Um bloco de instruções (ou instrução composta) é definido como uma sequência de instruções colocadas dentro de chaves ({ });
- Uma variável possui escopo local quando ela é declarada dentro de um bloco de instruções.
- Quando ela é declarada fora desse bloco, ela possui escopo global.

```
if (true) {
  // O código dentro do if vive num bloco
  // Variáveis declaradas aqui não existem fora
  const x = 10;
}
// Fora do bloco, x não existe
```

Switch e Operador Ternário

Innominatum CONSULTORIA
IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Quando if / else começa a ficar confuso

```
if (dia === 1) {
  console.log("Segunda");
} else if (dia === 2) {
  console.log("Terça");
} else if (dia === 3) {
  console.log("Quarta");
}
```

- Aninhamento de if em grandes quantidades prejudicam a legibilidade do código e logo a manutenção;
- O código funciona, mas começa a ficar longo e repetitivo.

Além do if / else: Ferramentas de Precisão Conteúdo

- Em JS, **não existe apenas uma forma de decisão**, podemos escolher a estrutura que mais se adeque a nossa necessidade, destacamos as seguintes:
  - Switch Case: Ideal para quando temos uma única variável com muitas opções fixas (ex: dias da semana, estados de um pedido, cargos). Evita a "escada" infinita de else if;
  - Operador Ternário: A versão "express" do if/else. Resolve decisões simples em apenas uma linha. Ideal para atribuições rápidas.

Estrutura switch

```
switch (expressao) {
  case valor1:
    // código
    break;
  case valor2:
    // código
    break;
  default:
    // código
}
// O JS compara a expressão com cada case.
```

- Quando todas as ramificações dependem do valor da mesma expressão, o uso do **switch** é bastante recomendado.
- A sua estrutura evita o desperdício de avaliar a mesma expressão repetidamente em várias instruções **if**, tornando o código mais limpo.

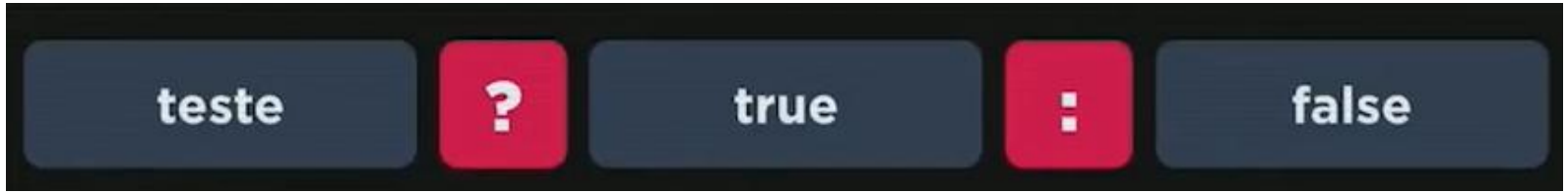
Exemplo simples de switch

```
let dia = 2;

switch (dia) {
  case 1:
    console.log("Segunda");
    break; // impede a execução dos próximos cases
  case 2:
    console.log("Terça");
    break;
  default:
    console.log("Dia inválido");
}
```

Operador ternário

```
condicao ? valorSeVerdadeiro : valorSeFalso
// O operador ternário é uma forma compacta de if / else.
```



Exemplo de ternário

```
let idade = 20;
let mensagem = idade >= 18 ? "Maior de idade" : "Menor de idade";
```

Quando usar ternário

Usamos o ternário quando:

- A condição é simples;
- Para tornar o código mais legível;
- Dentro uma string;
- Como argumento de funções;
- Precisa de um valor imediato.

Evitamos o uso quando:

- Há várias condições;
- O ternário fica aninhado.

Innominatum CONSULTORIA

Arrays – A Nossa Lista de Dados

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Array - Definição

Um array em JavaScript é um **conjunto ordenado de valores**, onde cada valor é chamado de elemento e possui uma posição numérica denominada índice. Seus elementos podem ser de qualquer tipo e diferentes elementos do array podem ter tipos distintos (incluindo objetos ou outros arrays). Eles são dinâmicos e crescem ou diminuem conforme a necessidade, sem ser necessário declarar um tamanho fixo na criação.

Array - Sintaxe

```
let frutas = ["Maçã", "Banana", "Morango"];
```

```
console.log(frutas[0]); // Saída: Maçã
console.log(frutas.length); // Saída: 3 (O total de elementos)
```

Array – Métodos push e pop

```
let fila = ["Cliente 1", "Cliente 2"];

// Chegou um novo cliente
fila.push("Cliente 3");
console.log(fila); // Saída: ["Cliente 1", "Cliente 2", "Cliente 3"]

// O último cliente desistiu da fila
let desistente = fila.pop();
console.log("O " + desistente + " saiu.");
console.log(fila); // Saída: ["Cliente 1", "Cliente 2"]
```

Estruturas de Repetição (Loops)

Por que precisamos de loops?

Muitas tarefas precisam ser repetidas:

- Percorrer listas;
- Repetir cálculos;
- Validar dados.

- Loops evitam repetição manual de código.

O que é um Loop?

Um **loop** (ou laço) em JavaScript é uma instrução de estrutura de controle projetada para executar outras instruções repetidas vezes. Essencialmente, os laços desviam o fluxo de execução para si mesmos a fim de repetir partes específicas do código-fonte.

Tipos de loop

- while e for: São os laços básicos da linguagem. O **while** repete as instruções enquanto uma expressão for verdadeira, enquanto o **for** simplifica laços que seguem padrões comuns, como o uso de uma variável contadora;
- do/while: É semelhante ao **while**, mas a expressão é testada no final, o que garante que o corpo do laço seja executado pelo menos uma vez.
- for/in: Um tipo de laço completamente diferente, utilizado para iterar pelas propriedades de um objeto.

Estrutura for

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

O for junta:

- Inicialização;
- Condição;
- Incremento.

Como funciona o for

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

1. I = 0;
2. Testa i < 5;
3. Executa o bloco;
4. Incrementa i;
5. Repete.

Quando usar o for?

O for é extremamente utilizado quando:

- Sabes quantas vezes vai repetir;
- Trabalhas com contadores;
- Percorre índices.

- Muito utilizado com arrays.

Estrutura while

```
while (i < 5) {
  console.log(i);
  i++;
}
```

Executa enquanto a condição for verdadeira.

Diferente do **for**, que temos um número de repetições conhecidos, o **while** depende de condição dinâmica

Estrutura do...while

```
let i = 10;
while (i > 0) {
    console.log("Contagem regressiva: " + i);
    i--;
}
```

Executa pelo menos uma vez, o seu uso é muito recomendado em menus, validações e sempre que as repetições precisarem acontecer pelo menos uma vez.

Erros comuns com loops

1. Esquecer o incremento;
2. Condição errada;
3. Usar == em vez de === (coerção);
4. Alterar a variável errada;

Loops exigem atenção, um erro pode tornar esse loop infinito, o que irá causar o travamento do programa, portanto, **atenção!**

Síntese

- Estruturas condicionais;
- Truthy & Falsy;
- Estruturas de Repetição;
- Controlo de ciclos;
- Tratamento de erros.

Conclusão

Conclusão

“Um bom design é aquele que é o mais simples possível.”

Dieter Rams

Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parceiros para o Crescimento
PORTUGAL DIGITAL