# Tema 07 – Estruturas Condicionais

## Introdução

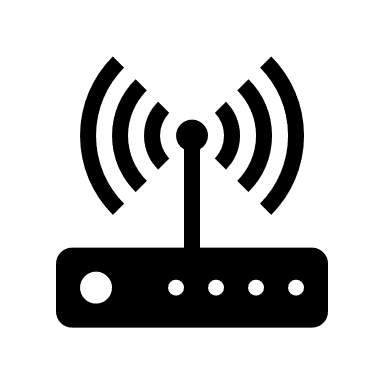
Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá como implementar as estruturas condicionais.

## Criação do Projeto inicial

Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 07\_JS\_Estruturas.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>Estruturas Condicionais</title>      </head>      <body>          <script src="./js/main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas iremos completar a implementação do código JavaScript.
3. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**).

## Desvios condicionais

Os desvios condicionais decidem o fluxo de execução de programa. Esses desvios são construídos com estruturas condicionais simples (if), composta (if-else) e switch. Uma das tarefas fundamentais de qualquer programa é decidir o que deve ser executado a seguir. Os comandos de decisão permitem determinar qual é a ação a ser tomada com base no resultado de uma expressão condicional.

Seja em estruturas condicionais ou laços de repetições, sempre será necessário testar uma condição que irá resultar em um valor booleano (true ou false). Por isso, quase sempre, fazemos usar de operadores de comparação e/ou lógicos:

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores de comparação | == | Igual |
| != | Diferente |
| < | Menor que |
| <= | Menor ou igual |
| > | Maior que |
| >= | Maior ou igual |
| === | Triplo igual |

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores lógicos | && | Lógica “and” ou “e”, que retorna verdadeiro se todos os operandos forem verdadeiros. |
| || | Lógica “or” ou “ou”, que retorna verdadeiro se pelo menos um operando for verdadeiro. |
| ! | Lógica “not” ou “não”, que inverte o valor lógico se é verdadeiro, retorna falso e se é falso retorna verdadeiro. |

### Estrutura condicional simples (if)

A estrutura condicional simples **if** é usada para verificar se dada condição é atendida:

* Se for, um conjunto de instruções deverá ser executado;
* Se não, o fluxo de execução do algoritmo seguirá após o fim do bloco de decisão;

A picture containing night sky

Description automatically generated

A sintaxe da estrutura **if** é:

|  |
| --- |
| if (condição) {  instrucao1;  instrucao2;  ...  instrucaoN;  }  proximaInstrucao; |

Se a condição entre parênteses for verdadeira, o conjunto de instruções entre chaves (dentro do bloco da estrutura condicional **if**) será executado, caso contrário esse conjunto é saltado e o programa irá executar a próxima instrução fora das chaves da estrutura **if**.

Exemplo: Um jogador somente irá passar para a fase 2 se atingir 1000 pontos.

A screenshot of a computer

Description automatically generated with medium confidence

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 1 | // Estrutura condicional simples  // Igual duplo  const a = 10;  if (a == 10) {      console.log('a é 10');  }  const b = '10';  if (b == 10) {      console.log('b é 10');  } |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |

1. Antes de explicar o que foi programado vamos ver o resultado mostrado no console.

Graphical user interface, text, application, email

Description automatically generated

Sempre que testamos se um valor é igual ao outro utilizando o igual duplo, a linguagem JavaScript não verifica o tipo de dados do valor comparado. Por isso, ao comparar a variável **a**, que contém o valor numérico 10 (linha 4), com o valor numérico 10 resulta **verdadeiro**, então o comando para imprimir a mensagem no console (linha 5) é executado:



O mesmo resultado verdadeiro acontece, ao comparar a variável **b**, que contém a string 10, com o valor numérico 10 (linhas 9), então o comando para imprimir a mensagem no console (linha 10) é executado:



Portanto, cuidado, se você quiser diferenciar o tipo de dados deve usar o triplo igual.

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 13 | // Triplo igual - verifica o tipo  console.clear();  const c = 10;  if (c === 10) console.log('c é 10');  const d = '10';  if (d === 10) {      console.log('d é 10');  }  console.log('Estou fora do if que compara string'); |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

Você deve notar a diferença nesse último exemplo, ao comparar uma string com um valor numérico (linha 19), o resultado da comparação é falso, então o comando da linha 20 não é executado e o programa salva para a instrução da linha 22:



**Observe também que quando o bloco do comando if tem apenas uma única instrução, as chaves podem ser ocultadas (linha 16). Essa regra vai valer para todas as estruturas condicionais e laços de repetições ensinados nesse tema.**

### Estrutura condicional composta (if-else)

**A estrutura if-else** prevê dois conjuntos de instruções para serem executadas de acordo com a avaliação da condição:

* Um conjunto de instruções que será executado quando a condição resultar em **Verdadeiro**;
* Um conjunto de instruções quando a condição resultar em **Falso**;

A screenshot of a computer

Description automatically generated with medium confidence

**A sintaxe do if-else é:**

|  |
| --- |
| if (condição) {  instrucao1;  instrucao2;  }  else {  instrucao3;  instrucao4;  }  proximaInstrucao; |

Se a condição entre parênteses for **verdadeira**, o conjunto de instruções (V) será executado, caso a condição for **falsa**, o conjunto de instruções (V) será executado.

**Exemplo: verificar se um número é par ou ímpar.**

A screenshot of a computer

Description automatically generated with medium confidence

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 24 | // Estrutura condicional composta  console.clear();  const e = '10';  if (e === 10) console.log('e é o numeral 10');  else {      console.log(`Resultado da condição: ${e === 10}`);      console.log('e não é o numeral 10');  } |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

Observe que o conteúdo da variável **e** é a string ‘10’, portanto a condição resulta em um valor falso. Por isso o bloco de comandos do **else** será executado (linhas 29 e 30).



1. Podemos também ter múltiplas comparações dentro dos parênteses. Nesse caso, além dos operadores de comparação, devemos usar os operadores lógicos.
2. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 33 | // Multiplas condições  console.clear();  const f = 4;  const g = 11;  if (f > 5 || g > 10) {      console.log(`Condição é ${f > 5 || g > 10}`);      console.log('f é maior do que 5 ou g é maior do 10');  } else console.log('f não é maior do que 5 e g não é maior do 10');  if (f > 5 && g > 10) console.log('f é maior do que 5 e g é maior do 10');  else {      console.log(`Condição é ${f > 5 && g > 10}`);      console.log('f não é maior do que 5 ou g não é maior do 10');  } |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |
| 46 |
| 47 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

Observe a múltipla comparação na linha 38 resulta em **true** (Lembre-se da tabela verdade da lógica OR), então o bloco de comando do **if** é executado:

Graphical user interface, application

Description automatically generated

Também, é possível notar que a comparação na linha 43 resulta em **false** (Lembre-se da tabela verdade da lógica AND), então o bloco de comando do **else** é executado:

Graphical user interface, text, application

Description automatically generated

### Desvios condicionais encadeados

**Você pode encadear vários desvios condicionais quando for necessário** verificar diversas condições. E cada condição depende do resultado da condição anterior. Basicamente, if dentro de if (denominado ifs-elses aninhados):

A screenshot of a computer

Description automatically generated with medium confidence

#### Vamos Praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 49 | // Desvios encadeados  console.clear();  const num = 'Ola';  if (num % 2 == 0 && !isNaN(num)) console.log('Número par!');  else if (num % 2 != 0 && !isNaN(num)) console.log('Número ímpar');  else console.log('Não é um número!'); |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

Nesse exemplo, a condição na linha 53 faz duas comparações:

* Primeiro, compara se o número é par (resto da divisão por 2 for zero, então o número é par), como 10 é par resulta em verdadeiro.
* Segunda, utiliza o método isNaN (is Not a Number, não é um número) para verificar se é um não é um número. Esse método retornar **true** se não for um número e **false** se for um número. Como 10 é um número, então o resultado de isNaN(num) é **false**, mas ao usar o operador de negação exclamação ( ! ), o valor lógico é invertido tornando a expressão **true**.

Como as duas comparações são verdadeiras de acordo com a lógico AND (&&), o resultado final é **true**. Portanto, o bloco de comandos do **if** é executado, imprimindo no console:



Se alteramos o valor da variável num para 11, o número não é mais par, então a primeira comparação retornará **false** e a segunda (num % 2 != 0 && !isNaN(num)) será **true**. Portanto, o bloco de comandos do **else if** é executado, imprimindo no console:



Se alteramos o valor da variável num para ‘Ola’, o conteúdo não é mais um número, então a primeira e a segunda comparações retornarão **false**. Portanto, o bloco de comandos do **else** é executado, imprimindo no console:



### Operador ternário

O operador ternário é uma instrução equivalente a estrutura condicional composta **if…else** e a sua vantagem é torna o código bem mais enxuto. A sua sintaxe é:

**condição ? expr\_1 : expr\_2**

Onde:

* Condição é a condição que será testada.
* expr\_1 é o que fazer quando a condição for verdadeira.
* expr\_2 é o que fazer quando a condição for falsa.

#### Vamos praticar

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 57 | // Operador ternário - exemplo 1  console.clear();  let resultado = 3 > 4 ? "Sim" : "Não";  console.log(resultado) |
| 58 |
| 59 |
| 60 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

No exemplo, a condição é comparar se 3 > 4. Como o resultado é **false**, o valor ‘Não’ é armazenado na variável resultado. E ao imprimirmos essa variável no console (linha 60), podemos ver que o conteúdo é **‘Não’**.

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 57 | // Operador ternário - exemplo 2  console.clear();  let result = Math.PI < 4 ? console.log('Sim') : console.log('Não'); |
| 58 |
| 59 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

No exemplo, usamos o objeto **Math** do JavaScript, que possui diversos valores matemáticas, por exemplo:

|  |
| --- |
| Math.PI; // retorna 3.141592653589793  Math.E // retorna o número de Euler  Math.SQRT2 // retorna a raiz quadrada de 2  Math.SQRT1\_2 // retorna a raiz quadrada de 1/2  Math.LN2 // retorna o logaritmo natural de 2  Math.LN10 // retorna o logaritmo natural de 10 |

Então, Math.PI retorna 3,14… que é um número menor do que 4. Portanto, a condição é verdadeira, executando, assim, a instrução console.log(‘Sim’) para imprimir a mensagem no console.

### Switch

O switch-case trabalha com situações mutuamente exclusivas. A sua sintaxe é:

|  |
| --- |
| switch (expressão) {  case valor1:  //Instruções executadas quando o resultado da expressão for igual á valor1  [break;]  case valor2:  //Instruções executadas quando o resultado da expressão for igual á valor2  [break;]  ...  case valueN:  //Instruções executadas quando o resultado da expressão for igual á valorN  [break;]  default:  //Instruções executadas quando o valor da expressão é diferente de todos os cases  [break;]  } |

Se um caso do switch for verdadeiro, as instruções dele serão executadas, as demais não serão. A expressão dentro dos parênteses será comparada com o valor de cada case, quando ”casar”, executa a lista de comandos e finaliza o switch com o break. Se nenhum caso for verdadeiro, as instruções do caso **default** serão executadas.

#### Vamos praticar

1. Continuando a implementação do projeto, insira o seguinte código no arquivo **main.js**:

|  |  |
| --- | --- |
| 66 | // Switch  const h = 11;  const cor = h > 10 ? 'vermelha' : 'azul';  console.clear();  switch (cor) {      case 'vermelha':          console.log('A cor é vermelha');          break;      case 'azul':          console.log('A cor é azul');          break;      default:          console.log('A cor não é vermelha ou azul');          break;  } |
| 67 |
| 68 |
| 69 |
| 70 |
| 71 |
| 72 |
| 73 |
| 74 |
| 75 |
| 76 |
| 77 |
| 78 |
| 79 |
| 80 |
| 81 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

No exemplo, h vale 11 (linha 67) e a condição h > 10 é verdadeira. Portanto, a string ‘vermelha’ é armazenada na variável cor (linha 68).

Em seguida, na linha 71, a variável cor é comparada no **switch** e “casa” com o primeiro **case** do switch, isso faz com que a instrução da linha 73 seja executada e resultado na impressão do console:



Se alteramos o valor de há para 5, o resultado será azul, pois a condição na linha 68 é falsa e o o segundo case do switch é o que “casa” com azul agora:



## Conclusão

Procure sempre fazer mais do que é passado para você em sala de aula. Por exemplo, você pode alterar as condições, utilizar outras comparações múltiplas para ver o resultado que será gerado.

Seguem alguns links para você estudar e aprender mais:

Operador Ternário:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Conditional_Operator>

<https://www.w3schools.com/js/js_comparisons.asp>

if-else:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/if...else>

Switch-case:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/switch>

Objeto Math:

<https://www.w3schools.com/js/js_math.asp>