SENAC

Campus Santo Amaro

.

TADS - Análise Desenvolvimento de Sistemas

*PW - Programação Web*

# Aula #6
# JavaScript:Elementos Básicos

Professor: Veríssimo - carlos.hvpereira@sp.senac.br

11/09/2023

## Sobre este documento

Este documento objetiva deixar registrado o conteúdo abordado em sala de aula pelo professor. Importante destacar que a Nota de Aula serve como guia ao professor, bem como serve aos alunos como um norte, quanto ao conteúdo desenvolvido em sala de aula.
Este documento não tem a pretenção de ser uma única fonte para estudo. Para tal, o aluno deverá assistir às aulas e fazer uso (consulta) à bibliograa recomendada na ementa da disciplina, e à bibliograa complementar, apontada pelo professor.

## Preâmbulo da Aula

Esta aula aborda coneitos fundamentais da linguagem JavaScript. Neste sentendido, trataremos dos seguintes tópicos:

- Entendendo JavaScript
- Manipulção de Variáveis (var, let, const)
- Controle de fluxo

Importante destacar que esta aula possui uma abordagem prática, na qual os elementos conceituai servem de guia para a parte prática da aula.

# Contents


**1 JavaScript- Introdução** **1**

1.1 O que é JavaScript . . . . . 1

1.2 Algumas Caracterísiticas Importantes do **JS** . . 2

1.3 Como podemos testar programas JavaScript? . . 3

1.3.1 Console do Navegador (Browser Console) 3

1.3.2 Editor de Código com Extensões de Execução de Código . . . . . 4

1.3.3 Chamada por HTML . . . . . 4

**2 Utilizando JavaScript** **7**

2.1 Ambiente Web - Font-End . . . . . 7

2.2 Ambiente Servidor - Back-End . . . . . 10

**3 JavaScript - Manipulando Variáveis** **11**

3.1 Tipos de Dados . . . . . 11

3.2 Regras para os Identificadores . . . . . 13

3.3 Bloco de Comando . . . . . 13

3.4 Escopo de Variável . . . . . 14

3.5 Identificador var . . . . . 15

3.6 Identificadores let e const . . . . . 15

**4 Tratamento de Fluxo** **17**

4.0.1 Estruturas Condicionais . . . . . 17

4.0.2 Comando Swich . . . . . 24

4.0.3 Estruturas while e do...while e for . . . . 26

# Chapter 1

# JavaScript- Introdução

## 1.1 O que é JavaScript

O JavaScript **(JS)** é uma linguagem de programação que permite que implementemos itens complexos em páginas Web. **(JS)** é geralmente utilizada em páginas Web, onde possui um papel importante para que uma página HTML possua elementos dinâmicos, pois, o HTML/CSS por si só, não possuem esta caracterísitica (Lembramdo que HTML é uma linguagem de marcação, e não uma linguagem procedural).

O poder da linguagem JavaScript transcende ao ambiente Web pois, com esta poderosa linguagem podemos desenvolver **App** Android/IoS, bem como também desenvolver servidores, **API-Rest** (Back-end).

## 1.2 Algumas Caracterísiticas Importantes do JS

Atenção a pontos importantes a considerar sobre a linguagem **(JS)**:

- É uma linguagem Interpretada (Não é compilada)
- É **Case Sentisitve**
- É uma linguagem **não-tipada**
- Linguagem baseada em protótipos
- Suportando estilos de orientação a objetos, imperativos e declarativos (programação funcional)

## 1.3 Como podemos testar programas JavaScript?

O *JavaScript* pode ser executado tanto no lado do servidor, como no lado do cliente. Podemos seguir várias abordagens, dependendo do tipo de execução (cliente/servidor) que desejamos realizar. Nas subseções que seguem abordaremos a execução pelas seguintes abordagens:

- Navegador
- Editor VS Code
- Código por chamada em HTML

### 1.3.1 Console do Navegador (Browser Console)

Esta é uma maneira rápida e simples de testar pequenos trechos de código *JavaScript*.

1. Abra o console do navegador pressionando **F12** ou clicando com o botão direito do mouse na página e selecionando "Inspecionar" ou "Console".
2. Localize a aba "***console***"
3. Podemos inserir código JavaScript **diretamente** no console e pressionar Enter para ver os resultados.

O resultado podemos ver na figura 1.1 , onde podemos observar qur a mensagem está em destaque.

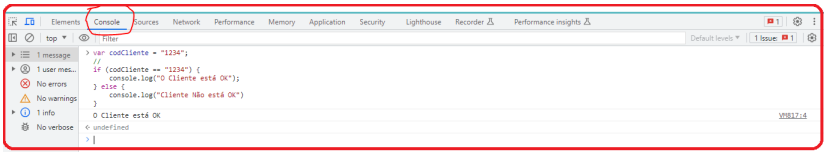


Figure 1.1: Utilizando Browser para executar JS

### 1.3.2 Editor de Código com Extensões de Execução de Código

Muitos editores de código, como **Visual Studio Code**, oferecem extensões que permitem executar código *JavaScript* diretamente. Podemos escrever o código em um arquivo *.js* e usar essas extensões para executá-lo no ambiente de desenvolvimento, conforme podemos observar àa figura 1.2, onde utilizamos o comando *node*: **node teste.js**

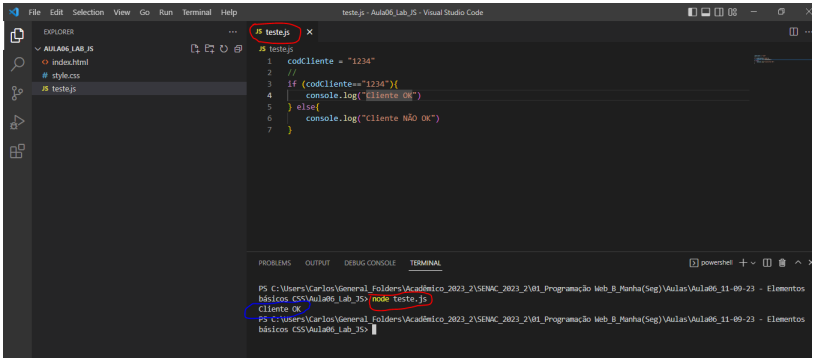


Figure 1.2: Executando JS com *Node*

### 1.3.3 Chamada por HTML

Para chamadas de script JavaScript em códigos HTML, podemos fazê-lo de duas formas, conforme seguem às subseções abaixo.

### Chamada incorporada no código HTML

Podemos criar scripts no código HTML ou fazer chamadas de arquivos .js para executar códigos, a partir de páginas HTML, conforme podemos observar à figura 1.3, onde utilizamos o chamamos o script, a partir de um evento de botão.

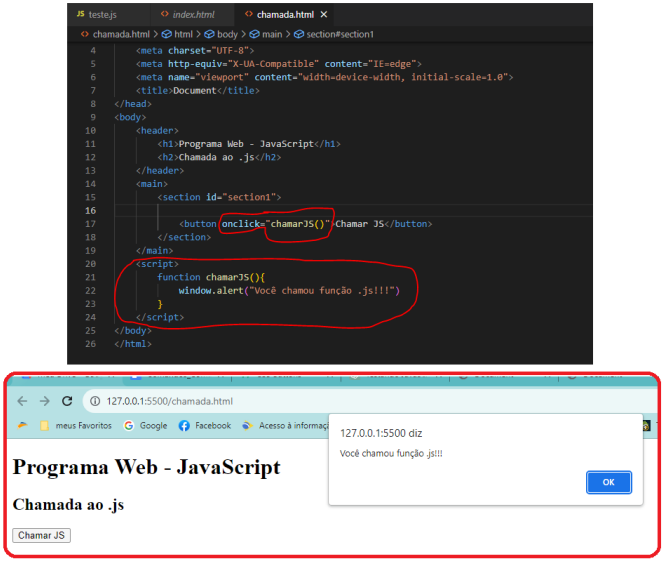


Figure 1.3: Executando JS chamada no HTML

## Chamada à arquivos externos ao código HTML

Podemos executar scripts Javascript, a partir de arquivos externos ao código HTML. Esta é a forma mais recomedada, conforme abordado à aula anterior (que versou sobre HTML). Podemos observar à figura 1.4, onde chamamos uma function,partir de um evento de botão, que se encontra em um arquivo .js.

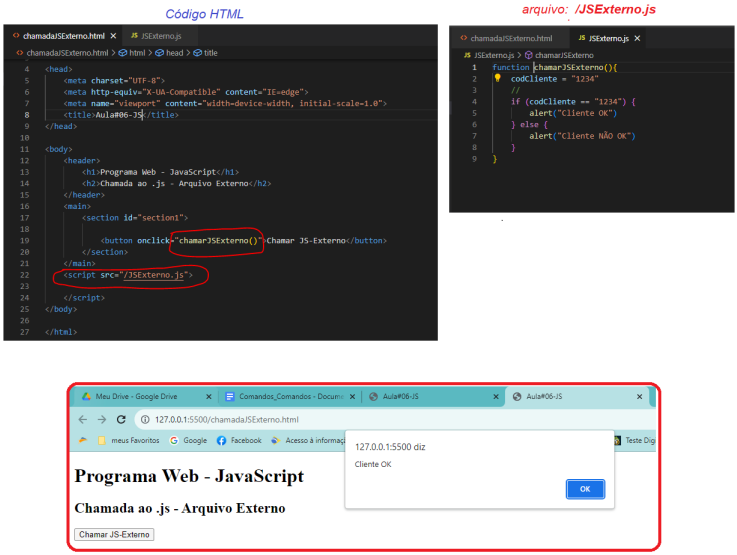


Figure 1.4: Executando JS por chamada à arquivo externo

# Chapter 2

# Utilizando JavaScript

O **JS** é uma linguagem de programação que permite implementar funcionalidades mais complexas em páginas web **(Front-end)**, bem com em ambientes de servidores **(Back-end)**

## 2.1 Ambiente Web - Font-End

O trecho de código 2.1 abaixo, ilustra a utilização da linguagem **JS** em uma página **HTML**. Observe que a página HTML aciona a função **clicou()**, a partir do envento **onclick** - **"Button"**

```
1
2  /*-----------------------------------------------*
3    * SENACAnaslie e Desenvolvimento de Sistemas   *
4    *           Programacao Web                     *
5    *  Prof. Verissimo                  Set/2023   *
6    *-----------------------------------------------*
7  */
8  <!DOCTYPE html>
9  <html lang="en">
10
11 <head>
12     <meta charset="UTF-8">
```

```
13      <meta http-equiv="X-UA-Compatible" content="IE
            =edge">
14      <meta name="viewport" content="width=device-
            width, initial-scale=1.0">
15      <title>PW-JavaScript-Introducao</title>
16      <link rel="stylesheet" href="style.css">
17  </head>
18
19  <body>
20      <header>
21          <h1>Programa Web - JavaScript</h1>
22          <h2>Interacao com Usuario</h2>
23      </header>
24      <main>
25          <section id="section1">
26              <h3>Interacaocom o Botao</h3>
27              <button onclick="clicou()">Clicar Aqui
                    </button>
28          </section>
29      </main>
30      <script>
31          function clicou(){
32              window.alert("Voce CLICOU no Botao!!!"
                    )
33          }
34      </script>
35  </body>
36
37  </html>
```

Listing 2.1: Exemplo inteção com JS

O resultado podemos ver na figura 2.1 , onde podemos observar qur a mensagem está em destaque.

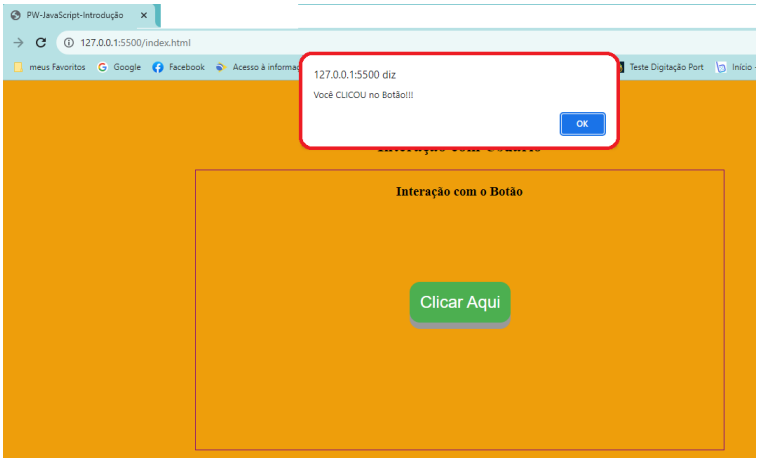


Figure 2.1: Resultado do exemplo JS acionado à página HTML

## 2.2 Ambiente Servidor - Back-End

A figura 2.2 ilustra a utilização da linguagem **JS** em uma Aplicação no Servidor **API**. Neste exemplo a API responde a uma requisição: Devolve em formato de protocolo **JSON**

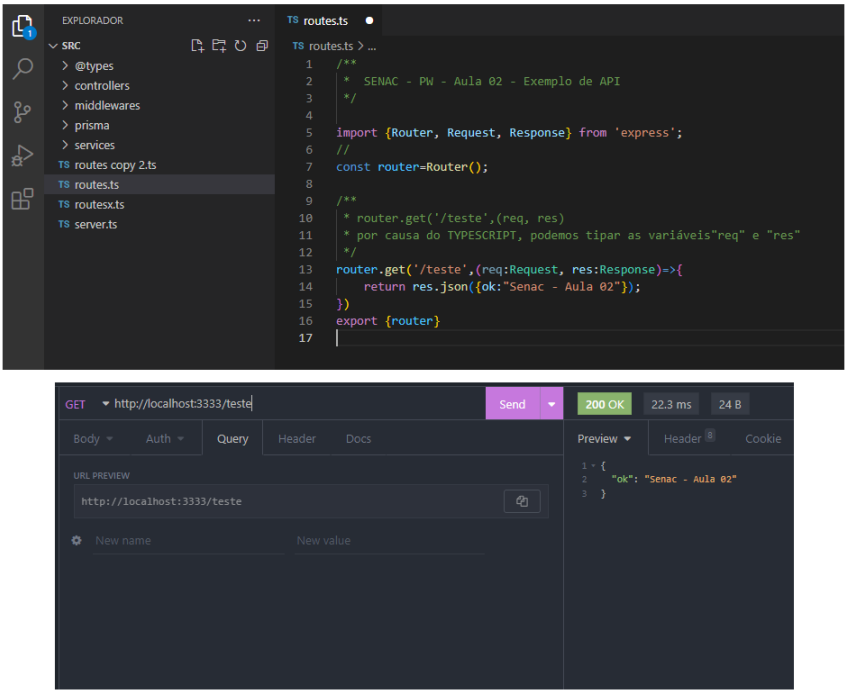


Figure 2.2: Exemplo JS no Servidor - API

# Chapter 3

# JavaScript - Manipulando Variáveis

## 3.1 Tipos de Dados

- Boolean: entidade lógica e pode ter dois valores: verdadeiro(true) ou falso(false).
- Null:tem exatamente um valor: null
- Undefined:Uma variável que não foi atribuída a um valor específico
- Number:O tipo number possui apenas um inteiro que tem duas representações: 0 é representado como -0 ou +0.
- BigInt:é um tipo de dado numérico que representa inteiros no formato de precisão arbritrária.
- String:O tipo String em JavaScript é usado para representar dados textuais. Isto é um conjunto de "elementos" de valores de 16-bits

- Symbol: Um Symbol é um valor primitivo único e imutável e pode ser usado como chave de uma propriedade[1] de Object
- **Object**: refere-se a uma estrutura de dados contendo dados e instruções para se trabalhar com estes dados.

O operador **typeof** retorna uma string indicando o tipo de um operando.
O operador typeof pode ser utilizado das seguintes maneiras:

- **typeof** operando

Exemplo:

- console.log(typeof "3.14");

Resultado: **string**

[1]Uma propriedade Javascript é uma característica de um objeto, frequentemente descrita como atributos associados à uma estrutura de dados.

## 3.2 Regras para os Identificadores

Os nomes de variáveis [2], chamamos de **"Identificadores"**. Temos que atentar para as seguintes regras:

- Podemos começar com letra, $ ou _ (Underline)
- Não pode começar com **números**
- podem conter letras e números
- é possível usar acentos e símbolos
- Não podem conter espaços
- não podem ser palavras reservadas

## 3.3 Bloco de Comando

Em JavaScript, um bloco de comando é um conjunto de instruções delimitado por chaves ( ). O bloco de comando é usado para agrupar uma ou mais instruções e é frequentemente usado em estruturas de controle de fluxo, como if/else, while, for, switch, try/catch, entre outros. O trecho de código **??** abaixo, demonstra blocos de comando em uma estrutura de desição:

[2] Na W3schools usamos camelCase para nomes de identificadores (variáveis e funções).

```
1  /**-----------------------------------------*
2   *     SENAC - TADS - Programacao  Web       *
3   * Exemplo de bloco de comando               *
4   *-----------------------------------------*/
5  if (true) {
6    let x = 10; // a vari vel x    declarada dentro
         do bloco if
7      console.log(x); // a vari vel x pode ser
           acessada dentro do bloco if
8    }
9
10   //cls
11   console.log("Valor de x",x); // a vari vel x
        n o pode ser acessada fora do bloco if,
       resultando em um erro
```

Listing 3.1: Exemplo Bloco de Comando

## 3.4 Escopo de Variável

Escopo é a acessibilidade de objetos, variáveis e funções em diferentes partes do código, isto significa dizer que, escopo é o que determina quais são os dados que podem ser acessados em uma determinada parte do código . As palavras chaves **let** e **const** permitem que trabalhemos não só com o escopo de funções (**var**), mas também com o escopo dos blocos.

Ao utilizamos JS é imprescindível que entendamos como aplicar escopo de variável:[3] de forma correta.

- Escopo global
  - É definida quando declaramos uma variável fora de qualquer função - ela torna acessível a qualquer parte da nossa aplicação ou site, podendo ser lida e alterada

[3]Escopo de variável é o local de nosso código onde uma determinada variável pode ser acessada:global ou local

- Escopo Local
  - É declarada dentro de uma função

Escopos criados por funções são chamados de **function scopes**, enquanto escopos criados por estruturas de controle são chamados de **block scopes**.

## 3.5 Identificador var

O identificador **var** age sobre o **escopo da função**:Ao declaramos uma variável sem o uso da palavra reservada **var** estaremos criamos uma variável **global implicitamente**, e automaticamente ela se torna global independente de onde ela for definida.

Uma variável global é definida quando declaramos uma variável fora de qualquer função, portanto, estão acessíveis em qualquer lugar em nosso código. O escopo de uma variável declarada com ***var*** é seu contexto atual em execução, o qual é a função a qual pertence ou, para variáveis declaradas fora de qualquer função, o escopo é o global.

## 3.6 Identificadores let e const

A grande mudança trazida pelo **ES2015** foram a introdução de **let** e **const** como maneiras de definirmos variáveis.
Essas **keywords** permitem que trabalhemos não só com o escopo de funções, mas também com o escopo dos blocos

- let tem escopo de bloco
- let pode ser atualizado, mas não declarado novamente.
- Declarações com const têm escopo de bloco

- Variáveis declaradas com const mantêm valores constantes
- const não pode ser atualizado nem declarado novamente
- Cada declaração com const deve ser inicializada no momento da declaração.
- Declarações de const somente podem ser acessadas dentro do bloco onde foram declaradas.

# Chapter 4

# Tratamento de Fluxo

### 4.0.1 Estruturas Condicionais

As declarações condicionais, em qualquer linguagem de programação, nos permitem representar tomadas de decisão,a partir da escolha que deve ser feita.

Abordaremos aqui as seguintes estruturas condicionais:

- Estrutura de Decisão simples:*if*
- Estrutura de Decisão *if*...*else*
- Encadeamento de decisões ( *if*...*else* **encadeado** )

#### Estrutura de Decisão *if*

Esta estrutura permite avaliar uma condição e, a partir dela, executar um bloco de código, somente se o resultado for Verdadeiro.

Conforme demonstrado à figura 4.1 podemos verificar que esta estrutura permite a execução de um bloco de código, de

acordo com o resultado da análise da condição (Comando *if*)

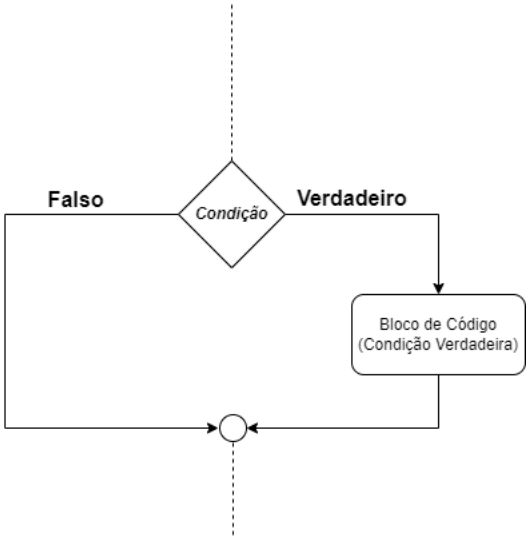


Figure 4.1: Estrutura de decissão if

A **sintaxe** para declarar um comando *if é:*

```
1 /**-----------------------------------------*
2  *       SENAC - TADS - Programacao Web      *
3  *              Sinstaxe comando if          *
4  *-----------------------------------------*/
5  if (condicao){
6   //bloco de codigo: Condicao Verdadeira
7 }
```

Listing 4.1: Sintaxe da estrutura if simples

### Comando *if...else*

*Esta estrutura condicional permite avaliar uma condição e, a partir dela, executar diferentes linhas de código.*

*Conforme demonstrado à figura 4.2 podemos verificar que esta estrutura permite a execução de um bloco de código verdadeiro ou um bloco de código falso, de acordo com o resultado da análise da condição (Comando if)*

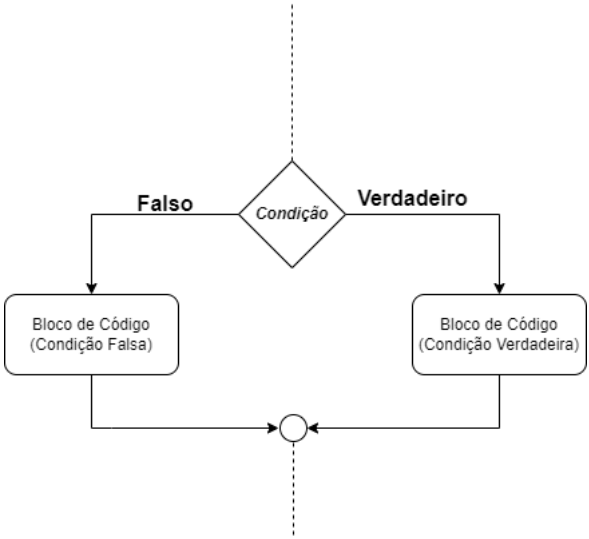


Figure 4.2: Estrutura de decissão if...else

*A **sintaxe** para declarar um comando if...else é:*

```
/**-----------------------------------------*
 *       SENAC - TADS - Programacao Web      *
 *         Sinstaxe comando if...else        *
 *-----------------------------------------*/
 if (condicao){
  //bloco de codigo: Condicao Verdadeira
} else {
  //bloco de codigo: Condicao Falsa
}
```

Listing 4.2: Sintaxe da estrutura if...else

### Comando *if...else* encadeados

*Esta estrutura condicional valiar permite sucessivas condições e, a partir dela, executar diferentes blocos de código.*

*Conforme demonstrado à figura 4.3 podemos verificar que esta estrutura permite a execução de um bloco de código verdadeiro ou um bloco de código falso, de acordo com o resultado da análise da condição (Comando if)*

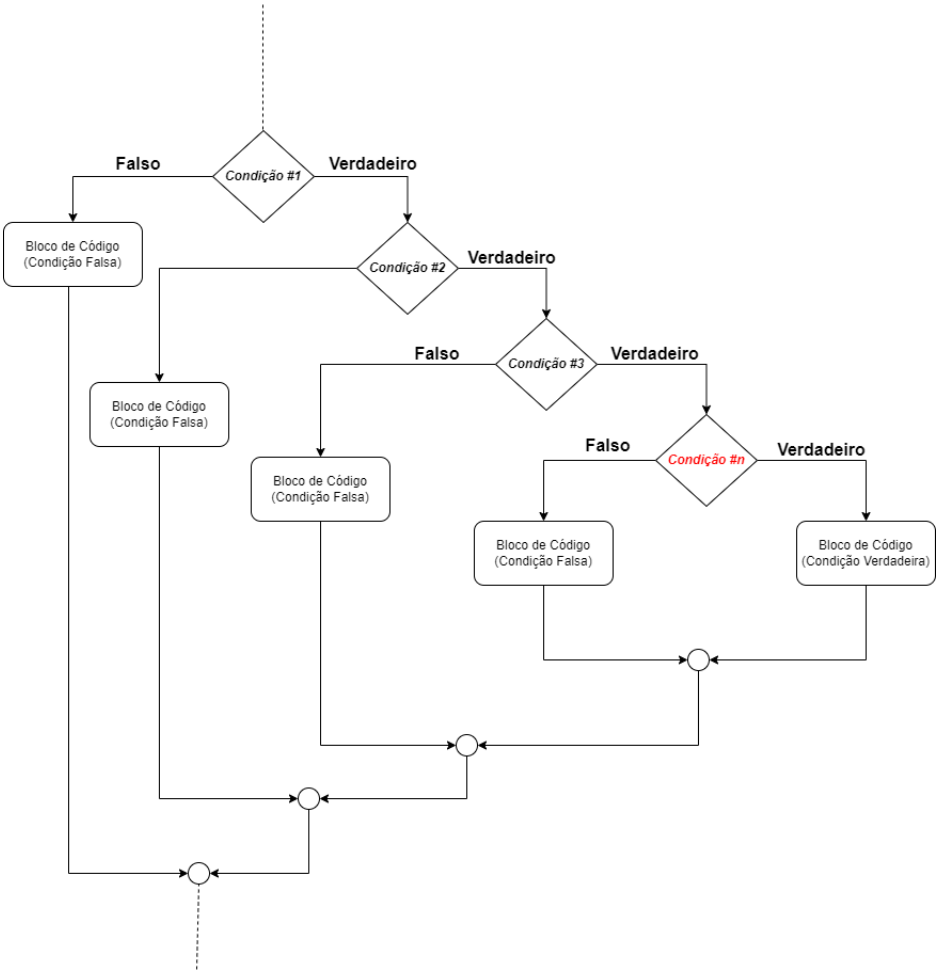


Figure 4.3: Estrutura de if...else encadeados

*A **sintaxe** para declarar if...else encadeados é:*

```
/**-----------------------------------------*
 *       SENAC - TADS - Programacao Web      *
 *    Sinstaxe comando if...else encadeados  *
 *-----------------------------------------*/
if (condicao#1){
    if (condicao#2) {
        if (condicao#3) {
            if (condicao#n) {
                    // bloco#n de codigo: Condicao
                       verdadeira
            } else {
                   //bloco#n de codigo: Condicao
                      Falsa
            }
        } else {
            //bloco#3 de codigo: Condicao Falsa
        }
    } else {
        //bloco#2 de codigo: Condicao Falsa
    }
} else {
    //bloco#1 de codigo: Condicao Falsa
}
```

Listing 4.3: Sintaxe da estrutura if...else encadeado

### Operadores Condicionais

*O quadro 4.1 mostra os operadores de comparação na linguagem javScript.*

Table 4.1: OPeradores de comparação JavaScript

| *Operador* | *Operação* | *Exemplo* |
|---|---|---|
| $>$ | Maior que | $(a > b)$ |
| $<$ | Menor que | $(a < b)$ |
| $>=$ | Maior ou igual a | $(a >= b)$ |
| $<=$ | Menor ou igual a | $(a <= b)$ |
| $==$ | Igual a | $(a == b)$ |
| $!=$ | Diferente a | $(a! = b)$ |
| $===$ | Idêntico a | $(a === b)$ |
| $!==$ | Não Idêntico a | $(a! == b)$ |
| && | E/AND | $(a\&\&b)$ |
| $\|\|$ | OU/OR | $(a\|\|b)$ |

### 4.0.2 Comando Swich

*A instrução switch é usada para executar diferentes ações com base em diferentes condições. Este comando é uma alternativa ao comando* ***if****.*

*A sintax para declaração de switch é:*

```
1  switch(expression) {
2    case x:
3      // code block
4      break;
5    case y:
6      // code block
7      break;
8    default:
9      // code block
10 }
```

Listing 4.4: Sinxtax para declarar Swich

*A listagem 4.5 abaixo, podemos ver as a utilização do switch.*

*Observer os seguintes pontos:*

- ***linha 8*** *inicia-se o comando, utilizando a variável diaObtido*
- *O comando break dever ser colocado dentro de cada case obtido*
- *à linha 30, o comando* ***default*** *serve para identificar que os testes feitos em todos comandos case, não foram atendidos.*

```
/**-----------------------------------------*
 *       SENAC - TADS - Programacao Web      *
 *                 Exemplo de Switch         *
 *-----------------------------------------*/

let diaObtido=new Date().getDay();
diaObtido=5;
switch (diaObtido) {
    case 0:
      diaDaSemana = "Domingo";
      break;
    case 1:
        diaDaSemana = "Segunda-Feira";
      break;
    case 2:
        diaDaSemana = "Terca-Feira";
      break;
    case 3:
        diaDaSemana = "Quarta-Feira";
      break;
    case 4:
        diaDaSemana = "Quinta-Feira";
      break;
    case 5:
        diaDaSemana = "Sexta-Feira";
      break;
    case 6:
        diaDaSemana = "Sabado";
        break;
    default:
    diaDaSemana = "Dia Invalido";
    break;
  }
  console.log("Hoje eh:"+diaDaSemana)
```

Listing 4.5: Exemplo manipulacao de Switch

### 4.0.3 Estruturas while e do...while e for

*Dos paradigmas da lógica de programa, a repetição de um trecho de código, é uma grande ferramenta para os programadores. As estruturas de repetição while; do...while e for* proporcionam implementar este paradigma.

Para quem quer ter sucesso em qualquer linguagem de programação, é imperativo ter pleno domínio, destas estruturas de repeção.

#### Repetição while

A **sintaxe** para declarar *while* é:

```
1 /**----------------------------------------*
2  *      SENAC - TADS - Programacao Web      *
3  *       Sinstaxe da estrutura while      *
4  *----------------------------------------*/
5  while (condicao) {
6     //Bloco de comandos
7  }
```

Listing 4.6: Sintaxe da estrutura while

#### Repetição do...while

A **sintaxe** para declarar *do...while* é:

```
1 /**----------------------------------------*
2  *      SENAC - TADS - Programacao Web      *
3  *     Sinstaxe da estrutura do...while     *
4  *----------------------------------------*/
5  do {
6     //Bloco de comandos
7     }
8     while (condicao)
```

Listing 4.7: Sintaxe da estrutura do...while

### for

A **sintaxe** para declarar *for* é:

```
1 /**-------------------------------------------*
2  *       SENAC - TADS - Programacao Web        *
3  *          Sinstaxe da estrutura for          *
4  *-------------------------------------------*/
5  for (inicializacao; condicao; incremento) {
6     //Bloco de comandos
7     }
```

Listing 4.8: Sintaxe da estrutura for