# JavaScript
# Aula 1

# JS

Também chamada de JS, é a linguagem de criação de scripts para a Web.

Originalmente criada na Netscape por Brendan Eich em 1994.

Java e JavaScript são “coisas” completamente distintas e desconexas.

JavaScript não permite a criação de applets nem de aplicativos.

JavaScript reside dentro de documentos HTML e pode prover diferentes níveis de interatividades não suportados pelo HTML sozinho.

# JS

Diferenças chaves em relação ao Java:

- Java é uma linguagem de programação;
- JavaScript é uma linguagem de script;
- Aplicativos Java são executados pela máquina virtual Java;
- Scripts JavaScript são executados pelos browsers;
- Java é compilado;
- JavaScript é texto puro;
- Cada tecnologia requer um plug-in diferente.

É utilizado por bilhões de páginas para:

- Adicionar funcionalidades;
- Verificar formulários; e
- Comunicar com servidores.

# JS

Com o tempo, muitas funcionalidades foram criadas em forma de Script para os browser e foram “incorporadas” ao JavaScript. JavaScript hoje é um conjunto de funcionalidades e, até mesmo, diferentes padrões.

## Formas de utilização do JS

- No elemento HTML.
- Na tag <script>.
- No arquivo externo.

Assim, dispomos de três opções para poder fazer um trabalho bem direcionado e atender com excelência as demandas do dia a dia.

# JS

## Log e Console

**Log** → Forma de comunicação com o sistema operacional.

**Console** → Local que recebe estas mensagens é chamado de console.

**Exemplo:**

```
console.log("Houve uma falha ao salvar o arquivo");
```

# JS

O uso do **console.log** é uma forma genérica de registrar uma mensagem, mas a maioria das linguagens de programação permite registrar mensagens com tipos específicos, tais como:

- **Log** → mensagem genérica, mais comumente utilizada.
- **Info** → registra um ***log*** do tipo "informação".
- **Debug** → registra um ***log*** do tipo "depuração" de código.
- **Warm** → registra um ***log*** do tipo alerta.
- **Error** → registra um ***log*** do tipo erro.

Com isso, podemos acessar o console de cada sistema operacional para acompanhar as mensagens. No Windows, por exemplo, o console é chamado de **"Event Viewer"** ou Visualizador de Eventos

# JS

## A Tag <script>

Para inserir códigos JavaScript, iremos fazê-lo em uma Tag HTML apropriada para a interpretação dos códios JS:

- <script>...</script>

Na tag **<script>**, podemos também usar uma **função nomeada (function)** e, no elemento, devemos inserir o nome da função que será executada.

```
<script>
        function limpar()        limpar é o nome da função a ser chamada em
                {                outro ponto
                <comando>;   Instrução a ser executada
                }
</script>
```

# JS

Em documentos HTML, a utilização da linguagem JavaScript, se dá sob a forma de funções, as quais são chamadas em determinadas situações ou em resposta a determinados eventos, estas funções podem estar localizadas em qualquer parte do código HTML, a única restrição é que devem começar com a declaração <SCRIPT> e termina com o respectivo </SCRIPT>.

Por convenção costuma-se colocar todas as funções no início do documento **(entre as TAG <HEAD> e </HEAD>)**, isso para garantir que o código JavaScript seja carregado antes que o usuário interaja com a Home Page, ou seja, antes da TAG <BODY>.

# JS

## Classe document

### Propriedades

- title – Define ou Retorna o Título da Página;
- url – Retorna o URL completo da página;

### Métodos

- write() – Escreve texto no documento;
- writeln() – Escreve uma linha de texto no documento;

# JS

## JavaScript com arquivo externo

Da mesma forma como nos arquivos CSS, podemos deixar funções e comandos JavaScript em arquivos externos:

- Estes arquivos devem ter a extensão .JS

Para importar, é preciso colocar o código na TAG <script> ...</script>:

**Exemplo:**

<script src="**meuscript.js**"></script>

meuscript.js → nome do arquivo externo.

# JS

```
<HTML>
  <HEAD>
      <TITLE> Exemplo </TITLE>
          <!--
              ...
              Se houvesse alguma função seria bom declará-la aqui!!!
              ...
           -->
    </HEAD>
    <BODY>
       Esta linha está escrita em HTML
      <SCRIPT>
         <!-- Esconde o código JavaScript dos browsers mais antigos
            document.write("Aqui já é JavaScript");
         // -->
      </SCRIPT>
      Voltamos para o HTML
     </BODY>
</HTML>
```

# JS

É importante ressaltar que todas as linhas devem ser terminadas com **“;” (ponto e virgula)** a menos que a próxima instrução seja um **“else”** e se você precisar escrever mais de uma linha para executar uma condição seja ela em uma estrutura **“for”, “if” ou “while”**, este bloco de instruções deve estar entre **“{ }” (chaves)**.

Inclusive a definição de funções segue este modelo, ou seja, todo o código da função deve estar limitado por **{ (no início) e } (no final)**.

Um browser que não suporta JavaScript, ele não conhece a TAG.

# JS

## Comentários

### Comentários de linha única

Os comentários de uma única linha começam com //.

### Comentários multilinhas

Começam com /*e terminam com */.

# JS

## Método alert()

A finalidade deste método é emitir uma caixa de diálogo do windows passando com uma mensagem e um botão de OK.

Este método é pertencente ao **objeto window** do JavaScript. Observe a sintaxe de seu funcionamento.

**Exemplo:**

```
window.alert("Meu Primeiro Script");
ou
alert("Meu Primeiro Script");
```

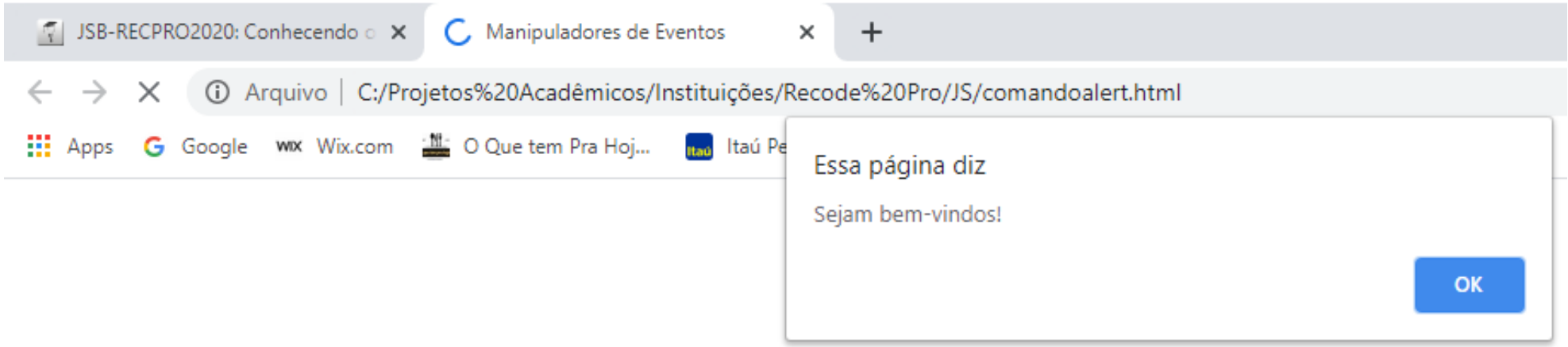
# JS

Comando inicial para dar uma mensagem para o usuário.

**Exemplo:**

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <h1>Comando alert para o usuario</h1>
                <script>
                        alert (“Sejam bem-vindos!”);
                </script>
        </body>
</html>
```

# JS


JSB-RECPRO2020: Conhecendo
Manipuladores de Eventos
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/comandoalert.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Pe
Essa página diz
Sejam bem-vindos!
OK

# Atividade Prática 1

Duração de 20 minutos.

1. Crie um arquivo HTML de nome exercalert.html;
2. Criar um título “Trabalhando com o Alert”;
3. Criar uma tag <h1> com o título “Comando alert para o usuário”; e
4. Dentro da TAG <script> crie três mensagens quaisquer para informações ao usuário.

# Atividade Prática 1

**Gabarito:**

```
<html>
        <head>
                <title>Trabalhando com o Alert</title>
        </head>
        <body>
                <h1>Comando alert para o usuario</h1>
                <script>
                        alert ("Você acaba de ser sorteado");
                        alert ("Agora é preciso entender as regras");
                        alert ("seu prêmio já pode ser resgatado");
                </script>
        </body>
</html>
```

# JS

## Método document.write()

Esta instrução na realidade segue a sintaxe de ponto da linguagem JavaScript, uma das maneiras de seguir a hierarquia dos objetos presentes na linguagem.

Nesta linha de comando temos o **método write()** que é pertencente ao **objeto document** que retrata o documento como um todo.

**Exemplo:**

```
document.write("Texto inserido com instruções JavaScript");
```

# Atividade Prática 2

Duração de 20 minutos.

1. Criar um arquivo HTML de nome docwrite1.html.
2. Criar três tags de h1, h2 e h3; e
3. Em cada uma das tags, colocar uma mensagem com o uso do document.write.

# Atividade Prática 2

## Gabarito:

```
<html>
        <head>
                <title>Metodo document</title>
        </head>
        <body>
                <h1>Uso do document.write no H1</h1>
                <script>
                        document.write("Entendendo o uso deste metodo no H1");
                </script>
                <h2>Uso do document.write no H2</h2>
                <script>
                        document.write("Entendendo o uso deste metodo no H2");
                </script>
                <h3>Uso do document.write no H3</h3>
                <script>
                        document.write("Entendendo o uso deste metodo no H3");
                </script>
        </body>
</html>
```

# Atividade Prática 2

# JS

## Método confirm()

Exibe uma caixa de diálogo e os botões de **OK** e **CANCELAR**. Caso seja pressionado o botão OK, o método retornará o valor booleano **TRUE** e pressionado o botão CANCELAR, é retornado o valor **FALSE**.

Com isto, o usuário poderá determinar uma tomada de decisão dentro de seu script.

**Exemplo:**

```
window.confirm("Tem Certeza??");
ou
confirm("Tem Certeza??");
```

# Atividade Prática 3

Duração de 20 minutos.

Criar um arquivo HTML de nome exercconfirm.html;

1. Criar um título na tag <head> de nome “Confirm”; e
2. Criar na tag <script> o código para capturar a opção que foi clicada pelo usuário.

# Atividade Prática 3

**Gabarito:**

```
<html>
        <head>
                <title>Confirmação</title>
        </head>
        <body>
                <script>
                        document.write(confirm("Deseja Prosseguir para a
                        proxima etapa?"));
                </script>
        </body>
</html>
```

# JS

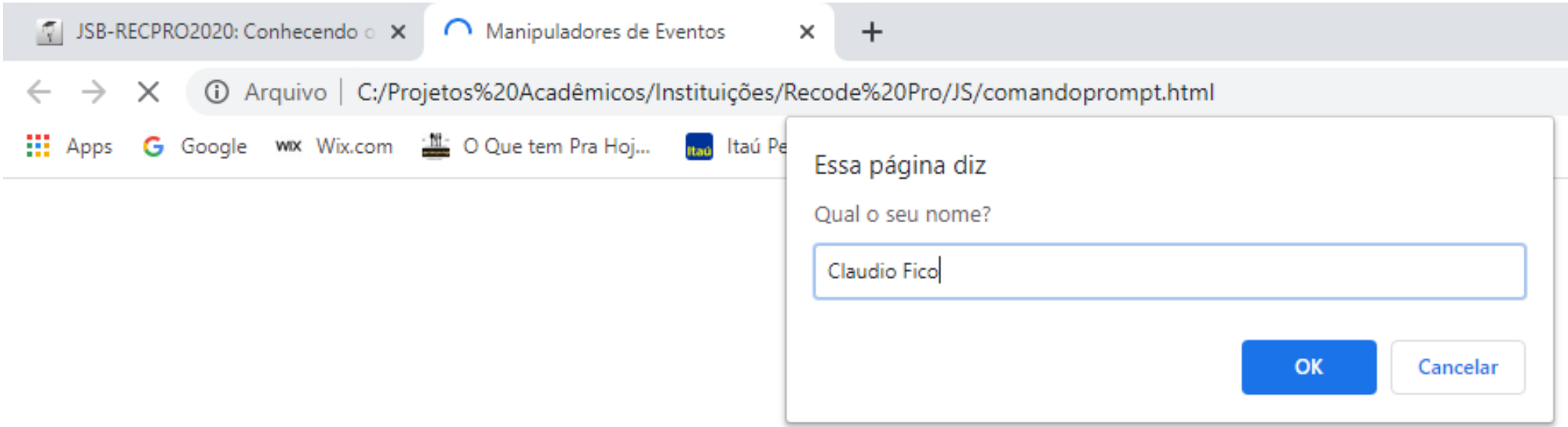
## Prompt (caixa de mensagem para o usuário preencher)

Recuperar uma informação digitada pelo usuário.

## Exemplo:

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <h1>Comando prompt para o usuario</h1>
                <script>
                        prompt (“Qual o seu nome?”);
                </script>
        </body>
</html>
```

# JS


JSB-RECPRO2020: Conhecendo o
Manipuladores de Eventos
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/comandoprompt.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Pe
Essa página diz
Qual o seu nome?
Claudio Fico
OK
Cancelar

# JS

**Exemplo:**

```
<html>
	<head>
		<title>Trabalhando com o Alert</title>
	</head>
	<body>
		<h1>Comando alert para o usuario</h1>
		<script>
			let nome = prompt ( "Qual o seu nome?" );
			document.write (nome + ", seja bem vindo ao site!" );
		</script>
	</body>
</html>
```

# JS

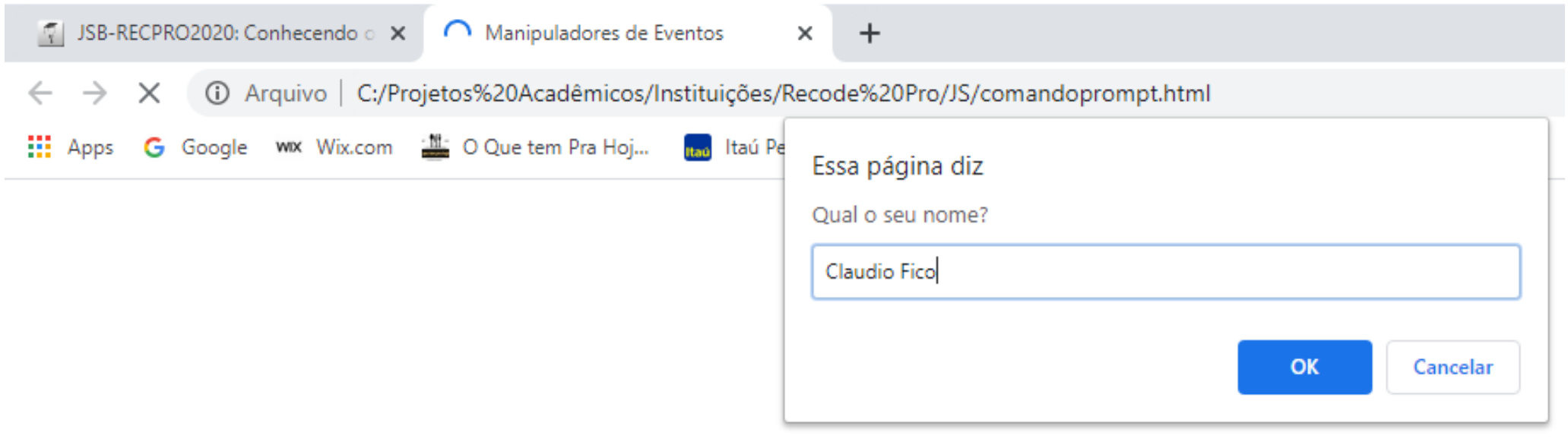
## Var

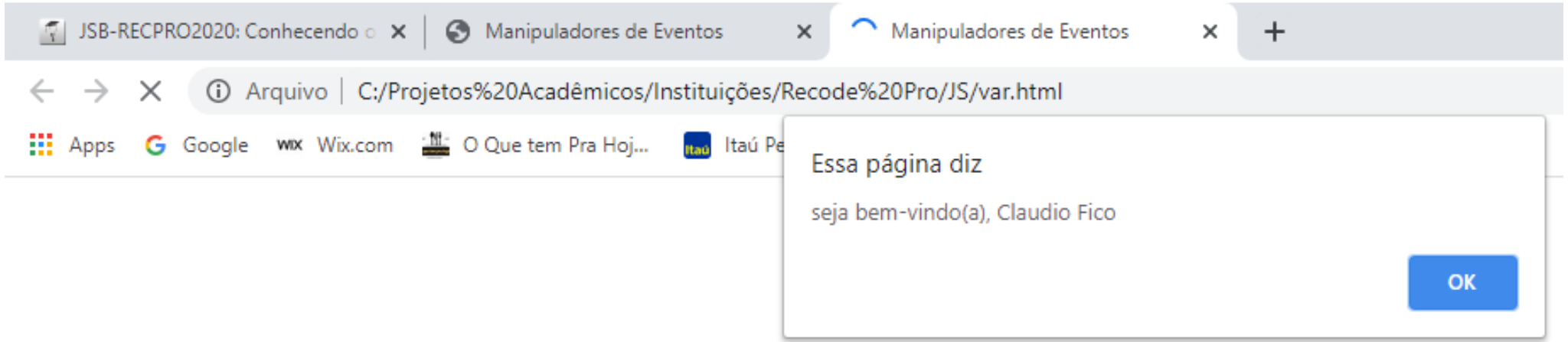
Usado para declaração de uma variável. Mas depois conheceremos o **let**.

## Exemplo:

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <h1>Comando alert para o usuario</h1>
                <script>
                        var nome;
                        nome = prompt ("Qual o seu nome?");
                        alert("seja bem-vindo(a), " + nome);
                </script>
        </body>
</html>
```

# JS


JSB-RECPRO2020: Conhecendo
Manipuladores de Eventos
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/comandoprompt.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Pe
Essa página diz
Qual o seu nome?
Claudio Fico
OK
Cancelar


JSB-RECPRO2020: Conhecendo
Manipuladores de Eventos
Manipuladores de Eventos
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/var.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Pe
Essa página diz
seja bem-vindo(a), Claudio Fico
OK

# Atividade Prática 4

Duração de 20 minutos.

1. Criar um título na tag <head> de nome “Prompt”; e
2. Crirar na tag <script> o código para capturar o bairro digitado pelo usuário e exibir a informação após a confirmação do botão de OK feita pelo usuário.

# Atividade Prática 4

**Gabarito:**

```
<html>
	<head>
		<title>Manipuladores de Eventos</title>
	</head>
	<body>
		<h1>Comando alert para o usuario</h1>
		<script>
			let bairro;
			bairro = prompt ("Qual o seu bairro?");
			alert("O bairro digitado foi, " + bairro);
		</script>
	</body>
</html>
```

# JS

## JavaScript com arquivo externo

Da mesma forma como nos arquivos CSS, podemos deixar funções e comandos JavaScript em arquivos externos:

- Estes arquivos devem ter a extensão .JS

Para importar, é preciso colocar o código na TAG <script> ...</script>:

**Exemplo:**

<script src="**meuscript.js**"></script>

meuscript.js → nome do arquivo externo.

# JS

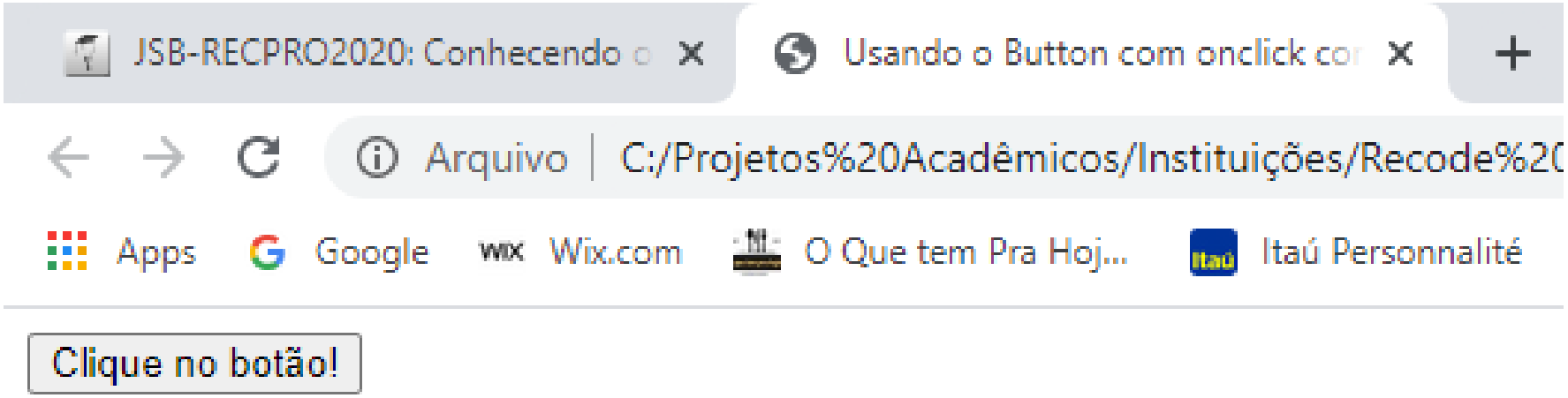
**Exemplo (arquivo externo – func.js):**

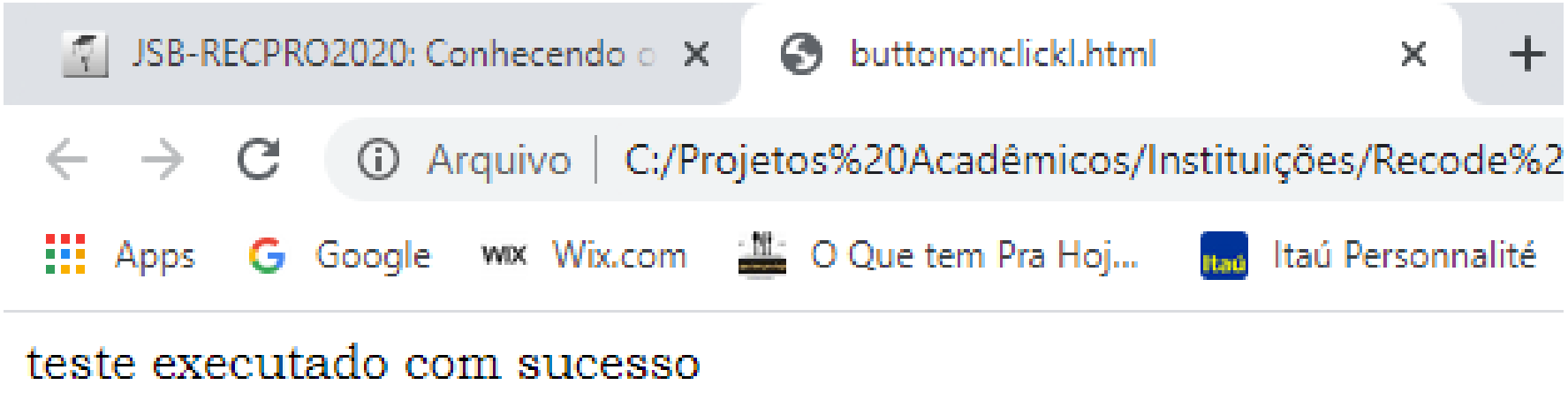
```
function exibe() {
   document.write("teste executado com sucesso");
}
```

**Exemplo (arquivo HTML)**

```
<html>
   <head>
      <title>Usando o Button com onclick com função externa</title>
      <script src="func.js"></script>
   </head>
   <body>
         <button onclick='exibe()'>Clique no botão!</button>
   </body>
</html>
```

# JS


JSB-RECPRO2020: Conhecendo
Usando o Button com onclick co
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%2(
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Personnalité
Clique no botão!


JSB-RECPRO2020: Conhecendo
buttononclickl.html
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%2
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Personnalité
teste executado com sucesso

# JS

## Variáveis

Assim como as propriedades que armazenam dados sobre os objetos, é possível com JavaScript a utilização das variáveis que têm a finalidade de armazenar temporariamente informações como textos, valores, datas, entre outros.

O conteúdo de uma variável pode ser simplesmente atribuído ou vir de um resultado de uma ação dada de uma expressão ou função.

Em JavaScript, variáveis dinâmicas podem ser criadas e inicializadas sem declarações formais.

# JS

## Let e Const

**Let** → Foi introduzida no EcmaScript 6, trazendo uma melhoria estrutural de um problema que existe na **keyword var**, além de um problema de escopo que também foi melhorado.

**Const** → Foi introduzida em 2015 com o lançamento do ES6. Ela possui toda a nova confiabilidade e previsibilidade existente no **let**, porém, as variáveis criadas com o **const** não podem ser modificadas, sendo apenas para leitura. Ou seja, elas definem **valores constantes**!

Este recurso é muito importante no controle do código, impedindo a troca acidental de valores de uma variável.

# JS

## CamelCase

Existe um padrão internacionalmente conhecido de nomenclatura de variáveis, chamado **CamelCase**, onde a variável, quando composta, tem a **primeira letra de cada palavra em maiúscula**.

**Exemplo:**

**var nomeAluno = “Francisco”;**

Outros formatos também são permitidos, mas não são considerados como boas práticas de programação, tais como:

- var Nomealuno = “Antonia”;
- var nomeprofessor = “Chico”;
- var NOMEDADISCIPLINA = “JavaScript”;

# JS

## Inteiros (integer)

Representam números positivos, negativos ou fracionários. Exemplo:

- A = 500
- B = 0
- C = -32

## Ponto flutuante

Este literal também chamado de notação científica é representado da seguinte maneira:

- 2.34

# JS

## Booleanos

Este tipo de literal representa valores lógicos que podem ser:

- TRUE ou 1; e
- FALSE ou 0

## String

Este literal representa qualquer cadeia de caracteres envolvida por aspas ou apóstrofo. Veja abaixo alguns exemplos:

- “Adriano Lima”
- ‘CFP-INFORMÁTICA’
- “”
- “500”

# JS

## Object

Pode armazenar diversos valores estruturados em pares de chave-valor:

```
{
chave: 'valor',
nome: "Matheus",
idade: 21,
 }
```

## Array

Pode armazenar diversos valores, organizados por posição:

```
[1,2,3,"Matheus"]
```

# JS

## Outro tipos

Uma variável ainda pode obter valores:

- Null (vazio);
- Undefined (indefinido); e
- NaN (not a number – não é um número).

# JS

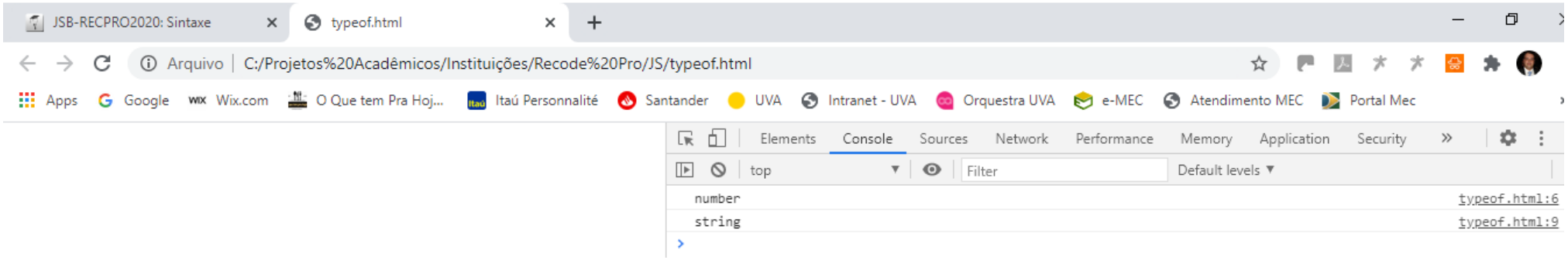
## Typeof

Identificar o tipo da variável.

```
<!DOCTYPE HTML>
<html>
        <head>
                <script>
                        let idade = 19;
                        console.log(typeof (idade));

                        idade = "melhor valor";
                        console.log(typeof (idade));
                </script>
        </head>
</html>
```

# JS


JSB-RECPRO2020: Sintaxe
typeof.html
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/typeof.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Personnalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
number
typeof.html:6
string
typeof.html:9

# JS

## Conversão de valores

Convertendo uma variável para obter o valor desejado:

```
<html>
	<head>
		<title>Manipuladores de Eventos</title>
	</head>
	<body>
		<script>
			let n1 = prompt ("Digite o primeiro numero");
			n1 = parseFloat(n1);
			let n2 = prompt ("Digite o segundo numero");
			n2 = parseFloat(n2);
			alert("Resultado " + (n1 + n2));
		</script>
	</body>
</html>
```

# JS

Em JavaScript, variáveis dinâmicas podem ser criadas e inicializadas sem declarações formais.

Existem dois tipos de abrangência para as variáveis:
**Global** → Declaradas/criadas fora de uma função. As variáveis globais podem ser acessadas em qualquer parte do programa.
**Local** → Declaradas/criadas dentro de uma função. Só podem ser utilizadas dentro da função onde foram criadas e precisa ser definida com a instrução Var.

Com relação à nomenclatura, as variáveis devem começar por uma letra ou pelo caractere sublinhado "_", o restante da definição do nome pode conter qualquer letra ou número.

# JS

## Case sensitive

É importante ressaltar que a variável “Código” é diferente da variável “código”, que por sua vez é diferente de “CODIGO”, sendo assim, muito cuidado quando for definir o nome das variáveis, utilize sempre um mesmo padrão.

Existem três tipos de variáveis:

- Numéricas;
- Booleanas; e
- Strings.

# JS

## Caracteres especiais

Podem ser incluídos dentro de uma string alguns caracteres especiais, a saber:

| Caracteres | Descrição |
|---|---|
| \t | Posiciona o texto a seguir, na próxima tabulação |
| \n | Passa para outra linha |
| \f | Form feed |
| \b | Back space |
| \r | Carrige return |
| \\ | Barra Invertida |
| \” | Aspas |
| \’ | Apóstofre |

# JS

Na prática isso é utilizado para a manipulação de variáveis não inicializadas sem que ocorra um erro no seu programa.

Quando uma variável possui o valor NULL, significa dizer que ela possui um valor desconhecido ou nulo.

A representação literal para NULL é a string 'null' sem os delimitadores.

Quando referenciado por uma função ou comando de tela, será assim que NULL será representado. Observe que NULL é uma palavra reservada.

# JS

## Expressões

Uma expressão é normalmente uma combinação de variáveis, literais, métodos, funções e operadores que retornam um valor qualquer. Usada para atribuir valores em variáveis ou até mesmo para testá-la e atribuir uma ação específica com base do seu resultado. Veja o exemplo da criação de uma variável numérica:

- numero=5
- numero=5*2

# JS

## Operadores

Junto com funções e variáveis, operadores são blocos de construção de expressões. Um operador é semelhante a uma função no sentido de que executa uma operação específica.

| Operadores Relacionais | |
|---|---|
| = | Atribuição |
| != | Diferente (valor) |
| !== | Diferente (valor e tipo) |
| > | Maior |
| < | Menor |
| <= | Menor igual |
| >= | Maior igual |
| == | Igualdade (valor) |
| === | Igualdade (valor e tipo) |
| % | Resto da divisão |

| Operadores Lógicos | |
|---|---|
| && | and (e) |
| \|\| | or (ou) |
| ! | not (Negação) |

# JS

## Operadores Aritméticos

| Aritmético | Operação | Prioridade |
|---|---|---|
| + | Adição | 5 |
| - | Subtração | 5 |
| % | Resto da divisão | 4 |
| * | Multiplicação | 3 |
| / | Divisão | 3 |
| ++ | Incremento | 2 |
| -- | Decremento | 2 |
| + | Manutenção do sinal | 1 |
| - | Inversão do sinal | 1 |

# JS

## Módulo da divisão ou resto ( % )

**Exemplo:**

V01=5

V02=2

V=V01%V02 // resulta em: 1

## Incremento ( ++ )

++Variável

**Exemplo:**

V01 = 5

V02 = ++V01 // Resulta em 6

# JS

## Decremento ( -- )

--Variável

**Exemplo:**

V01 = 5

V02 = --V01 // Resulta em 4

# JS

## Operadores de atribuição

= Atribuir

+= Soma ou concatenação e atribuição: x+=5 // é o mesmo que: x=x+5

-= Subtração e atribuição. x-=5 // é o mesmo que: x=x-5

*= Multiplicação e atribuição. x*=5 // é o mesmo que: x=x*5

/= Divisão e atribuição. x/=5 // é o mesmo que: x=x/5

# JS

## Operadores de strings

Operador de concatenação (***+***)

**Exemplo:**

```
Nome1 = "José"
Nome2 = "Silva"
Nome = Nome1+" da "+Nome2
```

O resultado é: "José da Silva"

# JS

## Estrutura de Decisão

### If else

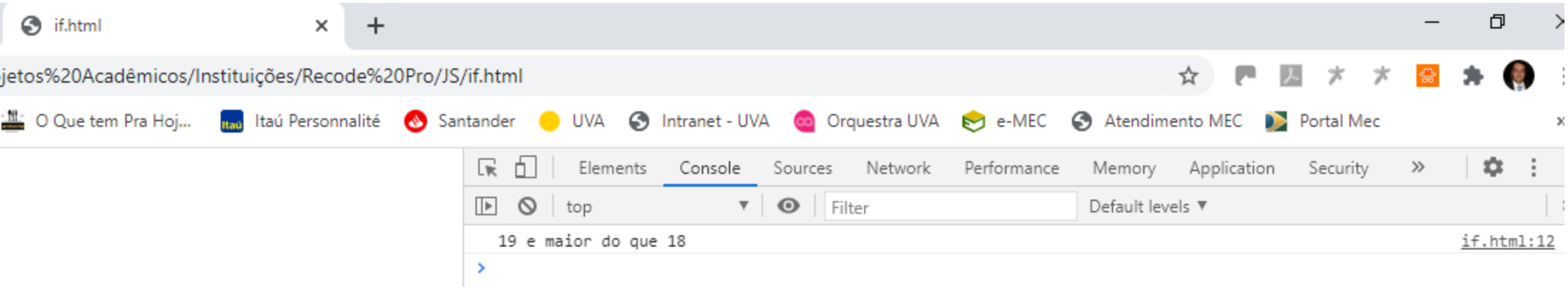
```
if (condição) // caso a expressão verificada retorne true
{
        <comando>;
        <comando>;
}
else // caso a expressão verificada retorne false
{
        <comando>;
        <comando>;
}
```

# JS

## Exemplo:

```
<!DOCTYPE HTML>
<html>
        <head>
                <script>
                        let idade = 19;
                         if (idade < 18)
                        {
                                console.log("19 e menor do que 18");
                         }
                        else
                        {
                                console.log("19 e maior do que 18");
                        }
                </script>
        </head>
</html>
```

# JS


if.html
jetos%20Acadêmicos/Instituições/Recode%20Pro/JS/if.html
O Que tem Pra Hoj...
Itaú Personnalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
19 e maior do que 18
if.html:12

# JS

## Estrutura de decisão

### If else if

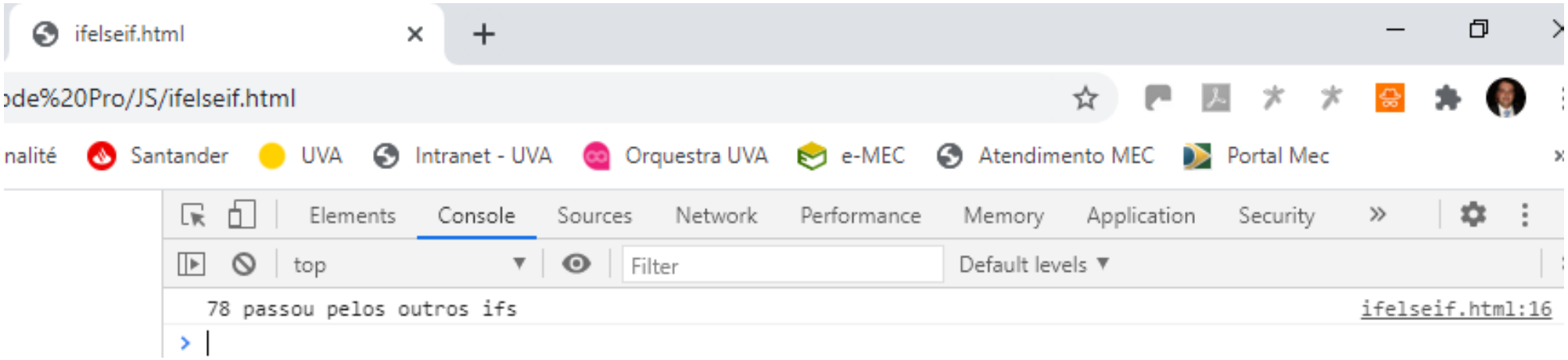
```
if (condição) // expressão verificada retorne true
{
        <comando>;
}
else if (condição) // retorna false, poderá fazer uma nova condição
{
        <comando>;
}
else // retorna false do else if
{
        <comando>;
}
```

# JS

## Exemplo:

```
<!DOCTYPE HTML>
<html>
	<head>
		<script>
			let idade = 78;
			if (idade < 16)
			{
				console.log("78 nao e menor do que 16");
			}
			else if (idade < 59)
			{
				console.log("78 nao e maior do que 59");
			}
			else
			{
				console.log("78 passou pelos outros ifs");
			}
		</script>
	</head>
</html>
```

# JS


ifelseif.html
ode%20Pro/JS/ifelseif.html
nalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
78 passou pelos outros ifs
ifelseif.html:16

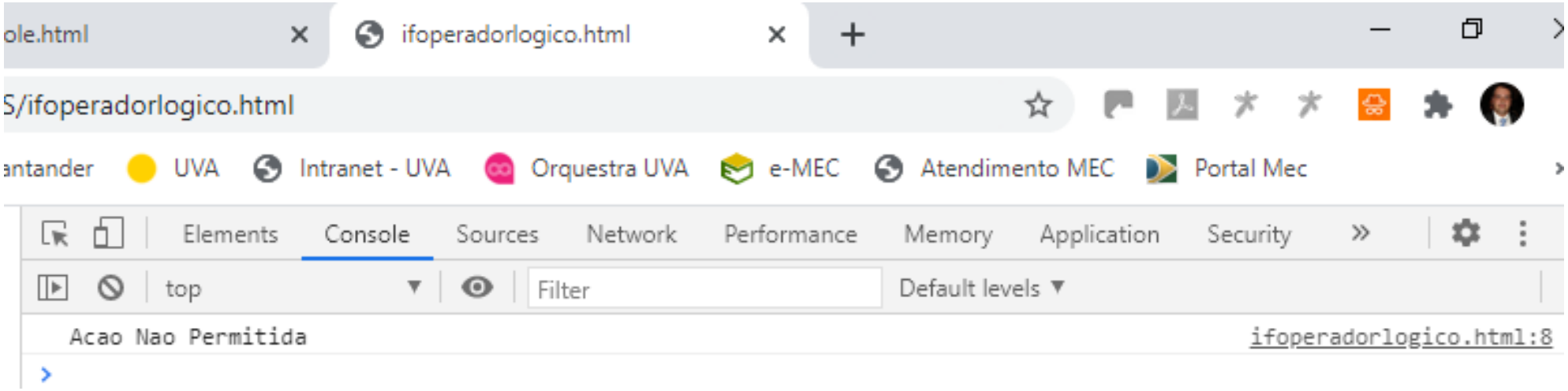
# JS

## Exemplo de if com operadores lógicos

```
<script>
        let idade = 16;
        let sexo = "f";

        if ((idade < 20) && (sexo == "m"))
                console.log("Acao Permitida");
        else
                console.log("Acao Nao Permitida");
</script>
```

# JS


ole.html
ifoperadorlogico.html
S/ifoperadorlogico.html
antander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
Acao Nao Permitida
ifoperadorlogico.html:8

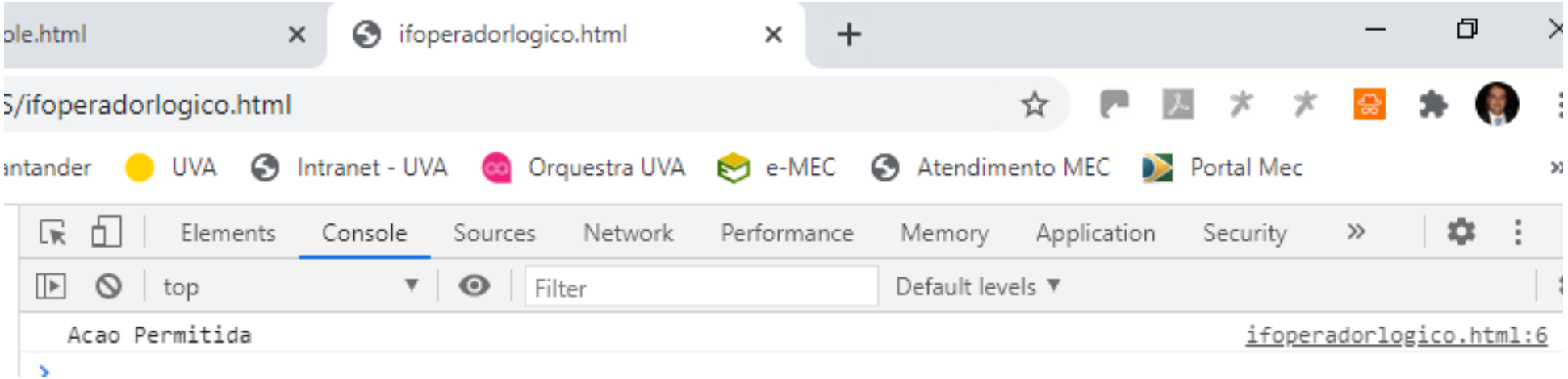
# JS

## Exemplo de if com operadores lógicos

```
<script>
        let idade = 19;
        let sexo = “m";

        if ((idade < 20) && (sexo == "m"))
                console.log("Acao Permitida");
        else
                console.log("Acao Nao Permitida");
</script>
```

# JS


ole.html
ifoperadorlogico.html
5/ifoperadorlogico.html
antander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
Acao Permitida
ifoperadorlogico.html:6

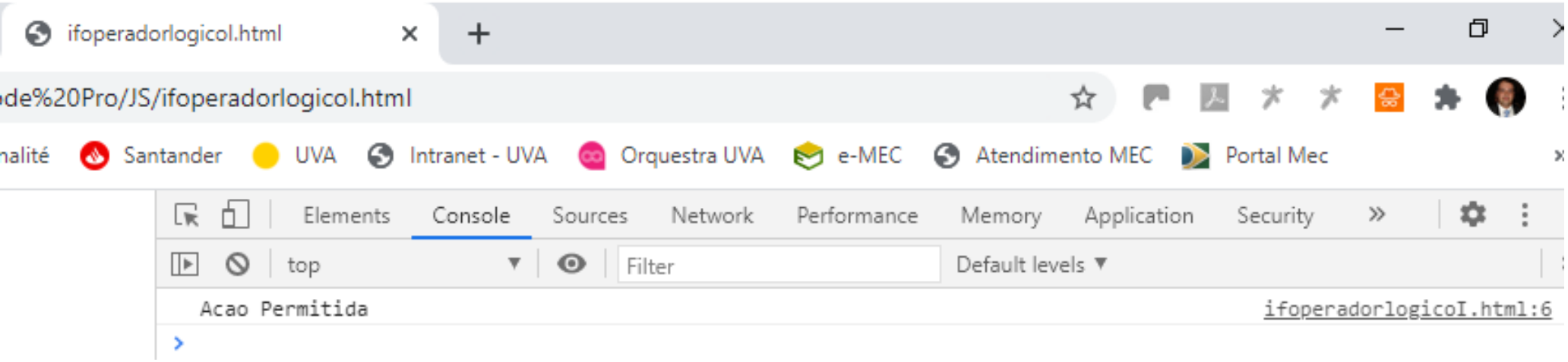
# JS

## Exemplo de if com operadores lógicos

```
<script>
        let idade = 36;
        let sexo = "m";

        if (((idade >= 20) && (idade <= 50)) || (sexo == "m"))
                console.log("Acao Permitida");
        else
                 console.log("Acao Nao Permitida");
</script>
```

# JS


ifoperadorlogicol.html
de%20Pro/JS/ifoperadorlogicol.html
nalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
Acao Permitida
ifoperadorlogicoI.html:6

# JS

## Switch

Esta instrução é bem semelhante com uma estrutura IF, porém é mais eficiente em razão de ser mais simples sua utilização e seu entendimento.

```
switch (variável)
{
        case CONSTANTE:    Valor numérico e string
                comandos; Executa o comando
                break;        Força a parada do código sem ir adiante nos demais cases
        case CONSTANTE2:
                comandos;
                break;
        case default:
                comandos;
                break;
}
```

# JS

**switch()** → significa “desvio”, é a abertura da estrutura e recebe a entrada da variável que será comparada.

**Case** → significa “caso”, é o espaço onde incluímos cada opção de resposta, que será executada “caso” seja igual ao seu valor de entrada. O case define o ponto de entrada, seguindo sua execução linha-a-linha até o fim, ou até ser interrompido (break).

**Break** → significa “interrompimento”, comando necessário ao final de cada bloco de linhas da opção case. O break deve ficar entre o início do case e seu término, criando uma estrutura de blocos e definindo um ponto de saída. Se não informado, todas as demais linhas do switch serão executadas.

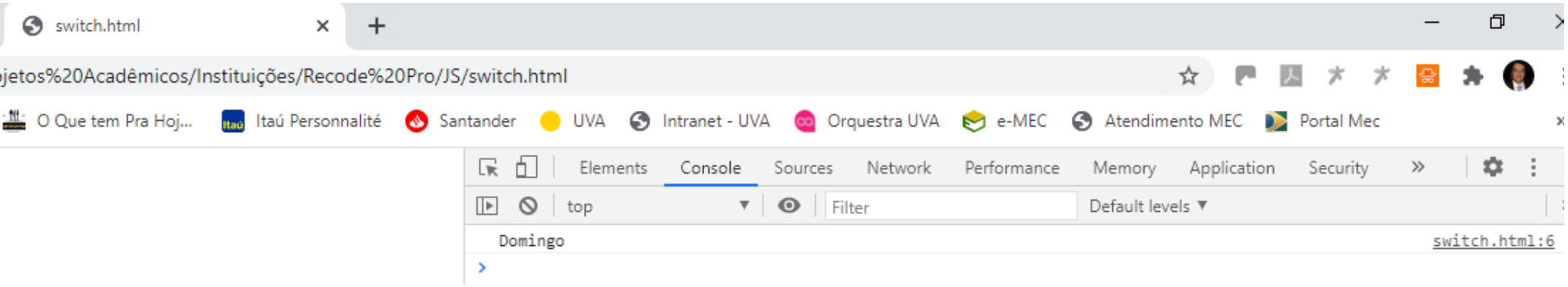
**Default** → significa “padrão”, é o bloco de código que será executado caso nenhuma opção existente seja a correta.

# JS

## Exemplo:

```
<script>
let dia = 0;
switch (dia)
{
   case 0:
      console.log("Domingo");
      break;
   case 1:
      console.log("Segunda");
      break;
   case 2:
      console.log("Terça");
      break;
   case 3:
      console.log("Quarta");
      break;
   case 4:
      console.log("Quinta");
      break;
   case 5:
      console.log("Sexta");
      break;
   case 6:
      console.log("Sabado");
      break;
   default:
      console.log("Dia da Semana Invalido");
}
</script>
```

# JS


switch.html
jetos%20Acadêmicos/Instituições/Recode%20Pro/JS/switch.html
O Que tem Pra Hoj...
Itaú Personnalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
Domingo
switch.html:6

# JS

## Exemplo:

```
<script>
let cor = "b";
switch (cor)
{
   case "a":
      console.log("azul");
      break;
   case "b":
      console.log("branco");
      break;
   case "c":
      console.log("caqui");
      break;
   case "d":
      console.log("dourado");
      break;
   case "e":
      console.log("esmeralda");
      break;
   case "f":
      console.log("firebrik");
      break;
   case "g":
      console.log("gold");
      break;
   default:
      console.log("Cores Invalidas");
}
</script>
```

# JS

## Exemplo:

```
<script>
let trimestre = 5;
switch (trimestre)
{
   case 1:
   case 2:
   case 3:
      console.log("primeiro trimestre");
      break;
   case 4:
   case 5:
   case 6:
      console.log("segundo trimestre");
      break;
   default:
      console.log("Trimestre Invalido");
}
</script>
```