Recode Pro 2020

JavaScript

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

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

Orientação a objetos

Diferente da Linguagem HTML, a linguagem JavaScript corresponde a programação orientada a objetos, isto significa que todos os elementos de uma página da Web são tratados como objetos.

Estes objetos são agrupados de acordo com seu tipo ou finalidade.

Dentro da linguagem JavaScript, são criados automaticamente objetos que permitem que o usuário possa criar novos objetos de acordo com sua conveniência.

propriedades e valores próprios que são ajustados pelo conteúdo da própria página.

Todos eles seguem uma hierarquia que reflete toda a estrutura de uma página HMTL.

A linguagem JavaScript pode ser utilizada para a criação de scripts tanto do lado cliente como do lado servidor. Seguindo a hierarquia de objetos da linguagem JavaScript, são criados os seguintes objetos ao ser carregada uma página:

JS

- **window** → O objecto mais acima na hierarquia, contém propriedades que se aplicam a toda a janela. Há também um objecto desta classe para todas as "sub-janelas" de um documento com frames location: Contém as propriedades da URL actual.

- **history** → Contém as propriedades das URLs visitadas anteriormente.

- **document** → Contém as propriedades do documento contido na janela, tais como o seu conteúdo, título, cores, etc.

JS

Objetos

Os objetos também são variáveis. Mas os objetos podem conter muitos valores.

Você define (e cria) um objeto JavaScript com um literal de objeto.

Criação do objeto

var pessoa = {primName:“Marcos", ultName:“Antunes", idade:50};

JS

Propriedade dos Objetos

Os pares **nome: valores** em objetos JavaScript são chamados de **propriedades**

Propriedade	Valor da propriedade
primeiro nome	John
último nome	Corça
era	50
cor dos olhos	azul

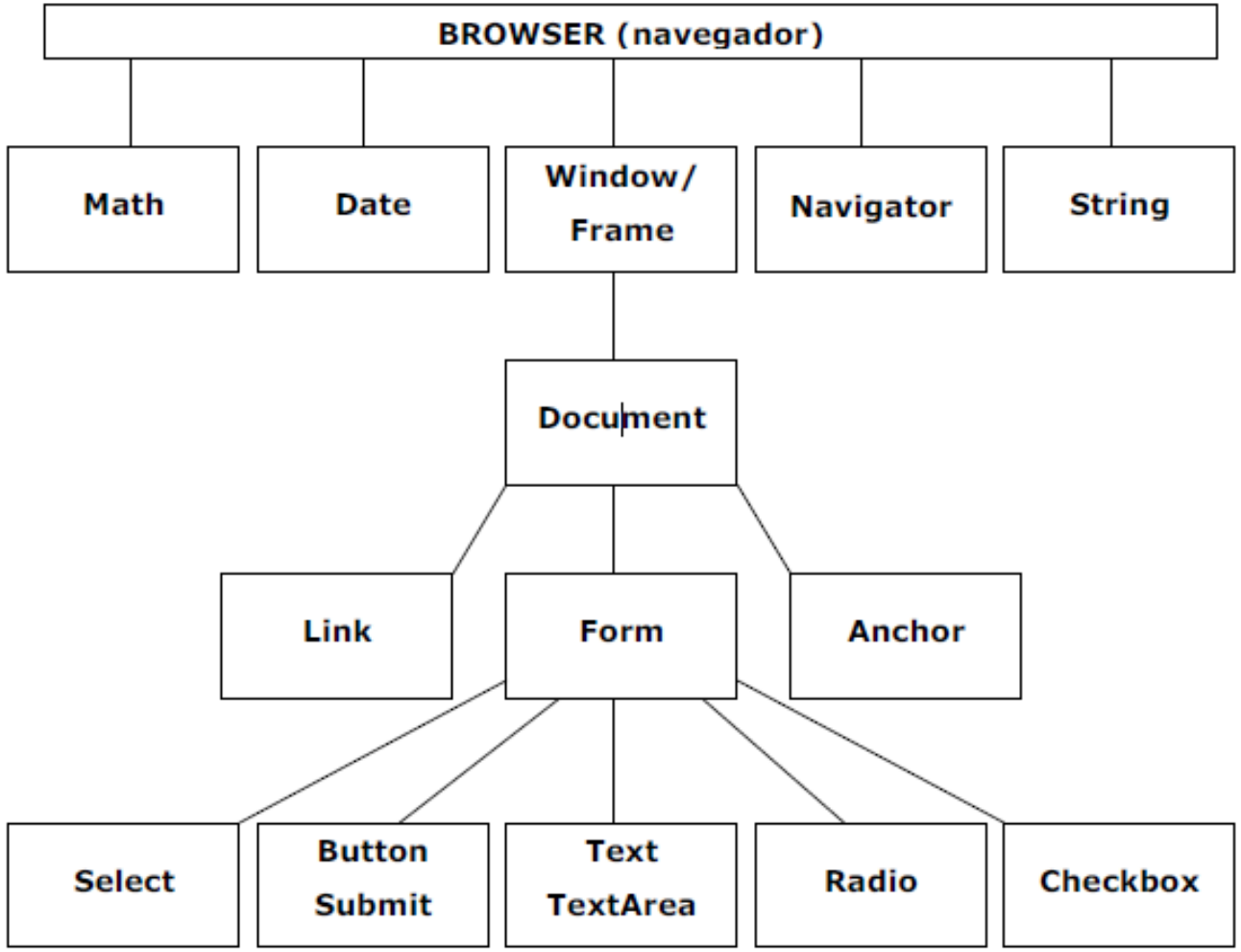
Acessando a propriedade do objeto

objectName.propertyName ou
objectName["propertyName"]

JS

Hierarquia dos objetos do JavaScript

A linguagem JavaScript manipula vários tipos de objetos através do uso de suas propriedades e métodos. Estes objetos são representados por uma hierarquia, fazendo com que alguns objetos se tornem propriedades de outros.



JS

Propriedade de objetos

Cada objeto existente na manipulação do JavaScript possui propriedades (características).

Exemplo: sabemos que um documento HTML possuem título e corpo, estas características do documento podemos chamar de propriedades que existem neste documento.

Estas propriedades existem de dois tipos, algumas são os objetos propriamente ditos enquanto outras não.

Exemplo: o objeto form (formulário) que é uma propriedade do objeto document (documento), conforme mostrado no organograma apresentado anteriormente.

JS

Já a propriedade de título da página (title), é pertencente ao objeto document não havendo nenhuma propriedade sobre ela. Concluindo, podemos dizer que a propriedade form do objeto document é um objeto-filho e o objeto document é o objeto-pai.

Em geral, as propriedades podem conter valores (string, números, boleanos, array entre outros tipos). A utilização de propriedades se dá acompanhada de seu objeto sendo separados por um ponto apenas.

Veja abaixo a sintaxe de utilização de propriedades:
nomeobjeto.propriedade

JS

Métodos dos objetos

Além das propriedades, os objetos podem conter métodos que são funções pré-definidas pela linguagem JavaScript que irão executar determinada operação.

Exemplo: dentro de um documento o usuário poderá utilizar o método de escrever neste documento para exibir um texto qualquer.

Os métodos estarão sempre associados à algum objeto presente no documento e cada método faz parte de um objeto específico.

JS

Não use métodos em objetos que não o utilizam, isto causa erro na execução do script. Na maior parte os métodos são usados para alterar o valor de uma propriedade ou executar uma tarefa específica. **Sintaxe:**

nomeobjeto.metodo(argumento)

nomeobjeto → faz referência ao objeto a ser utilizado e o qual sofrerá uma ação do método.

método → é o nome de identificação do método usado e entre parênteses.

argumento → é a expressão ou valor opcional que será usada para alterar sobre o objeto.

JS

Eventos

Em linguagens orientadas a objetos é comum a manipulação de eventos que é qualquer reação ou ação que executará determinado procedimento, normalmente ocorre por ato executado pelo usuário:

- Clicar em um botão;
- Selecionar algum objeto; e
- Pressionar alguma tecla.

Eventos são quaisquer ações iniciadas por parte do usuário.
Sua utilização se dá como atributos da linguagem HTML, ou seja dentro dos próprios Tag's HTML.

JS

Sintaxe:

<TAG nomeevento="Instruções JavaScript">

Onde é apresentado **TAG** é uma instrução da linguagem HTML.
Onde é evento é o nome do evento gerado da linguagem JavaScript.

Onde **“Instruções JavaScript”** serão as instruções JavaScript à serem executadas.

Elas estarão sempre entre aspas.

JS

Caso haja mais de um comando JavaScript a ser executado para o mesmo evento estes deverão estar separados por ponto e vírgula (;), conforme mostrado no exemplo a seguir:

Exemplo:

<TAG nomeevento="JavaScript1; JavaScript2; JavaScript3">

JS

Manipulador de eventos que podem ser usados

EVENTO	MANIPULADOR	DESCRIÇÃO
blur	onBlur	Ocorre quando o usuário retira o foco de um objeto de formulário.
change	onChange	Ocorre quando o usuário muda o valor de um objeto de formulário.
click	onClick	Ocorre quando o usuário clica sobre o objeto.
focus	onFocus	Ocorre quando o usuário focaliza o objeto.
load	onLoad	Ocorre quando o usuário carrega a página.
unload	onUnload	Ocorre quando o usuário abandona a página.
mouseOver	onMouseOver	Ocorre quando o ponteiro do mouse passa sobre um link ou âncora. **Válidos apenas para hiperlinks.**
select	onSelect	Ocorre quando o usuário seleciona um elemento de um formulário.

JS

EVENTO	MANIPULADOR	DESCRIÇÃO
submit	onSubmit	Ocorre quando o usuário envia um formulário.
mouseDown	onMouseDown	Ocorre quando o botão do mouse é pressionado.
mouseMove	onMouseMove	Ocorre quando o ponteiro do mouse se movimenta sobre o objeto.
mouseOut	onMouseOut	Ocorre quando o ponteiro do mouse afasta de um objeto. **Válidos apenas para hiperlinks.**
mouseUp	onMouseUp	Ocorre quando o botão do mouse é solto.
keyDown	onKeyDown	Ocorre quando uma tecla é segurada.
keyPress	onKeyPress	Ocorre quando uma tecla é pressionada.
keyUp	onKeyUp	Ocorre quando uma tecla é solta.

JS

Exemplo do javascript na TAG <button>:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Usando o Button com onclick</title>
   </head>
   <body>
         <button    onclick='document.write("Ação    realizada    com
         sucesso")'>Clique no botão!</button>
   </body>
</html>
```

JS

A Tag <script>

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

- <script>...</script>

Uma das formas de trabalho na tag **<script>**, é o uso de 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

Observação:

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

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

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:

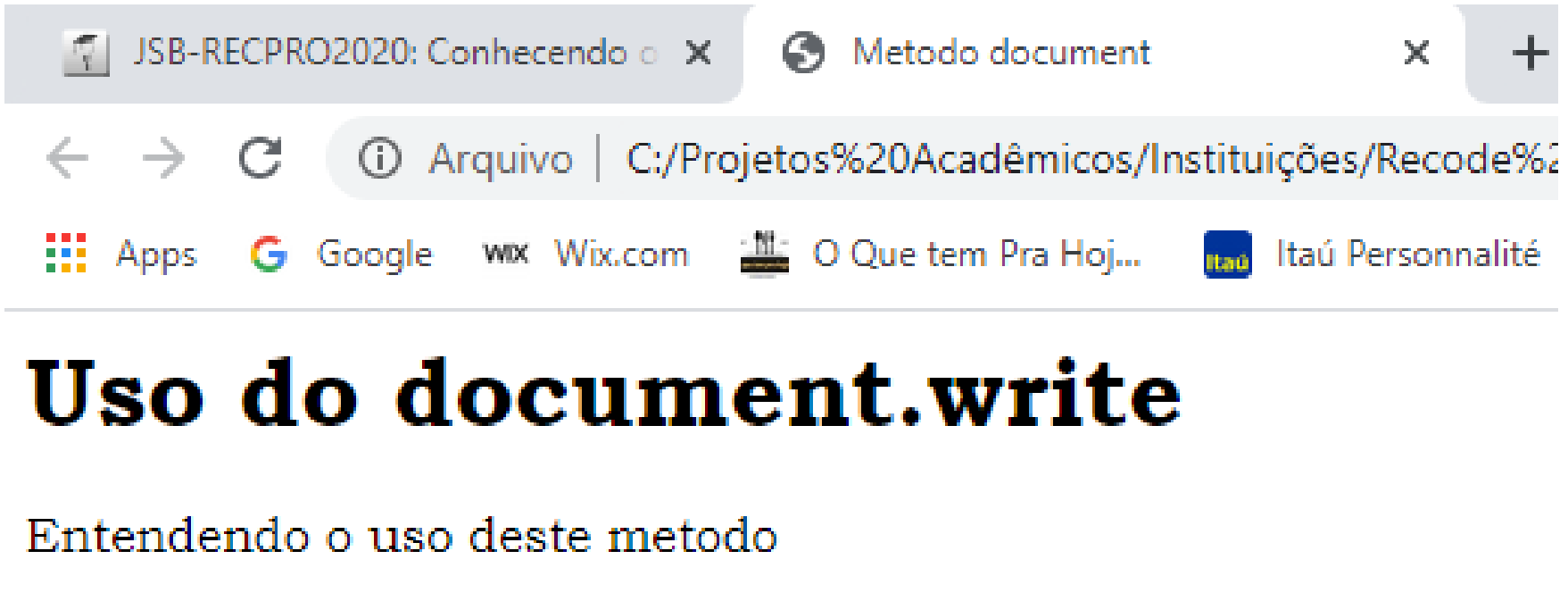
```
document.write("Texto inserido com instruções JavaScript");
```

JS

Exemplo:

```
<html>
        <head>
                <title>Metodo document</title>
        </head>
        <body>
                <h1>Uso do document.write</h1>
                <script>
                        document.write("Entendendo o uso deste
                        metodo");
                </script>
        </body>
</html>
```

JS


JSB-RECPRO2020: Conhecendo
Metodo document
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%2
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Personnalité
Uso do document.write
Entendendo o uso deste metodo

JS

Exemplo do javascript no evento onclick na TAG <script>:

```
<html>
	<body>
		<script>
			function data()
			{
				document.write(Date());
			}
		</script>
		<button onclick="data()">Clique Aqui!</button>
	</body>
</html>
```

JS

Exercício 1:

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 tag script e uma mensagem com o uso do document.write.

JS

Código do exercício 1:

```
<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>
```

JS


Metodo document
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/docwrit.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Personnalité
Santander
UVA
Intranet - UVA
Orquestra UVA
e-MEC
Atendimento MEC
Portal Mec
Uso do document.write no H1
Entendendo o uso deste metodo no H1
Uso do document.write no H2
Entendendo o uso deste metodo no H2
Uso do document.write no H3
Entendendo o uso deste metodo no H3
POR
08:07

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

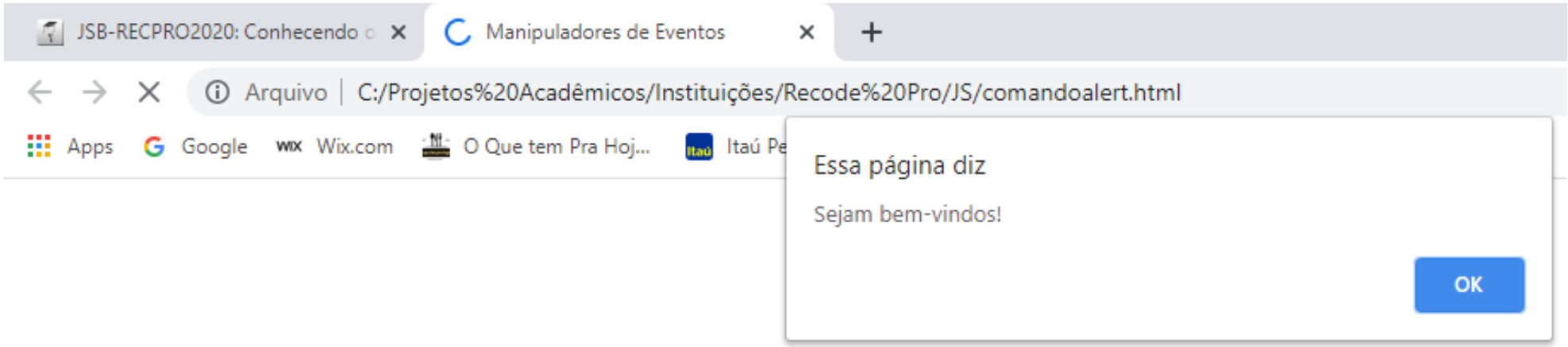
Alert (caixa de mensagem para o usuário)

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

JS

Exercício 2:

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.

JS

Código do exercício 2:

```
<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 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??");
```

JS

Confirm

Como o próprio nome sugere, serve para **efetuarmos confirmações.** A caixa de mensagem **confirm** serve exatamente para esses casos, pois ela exibe uma mensagem com 2 botões, que, dependendo do navegador, podem ser:

- “sim” e “não;
- “ok” e “cancelar”;
- “ok” e “cancel”;
- “yes” e “no”;

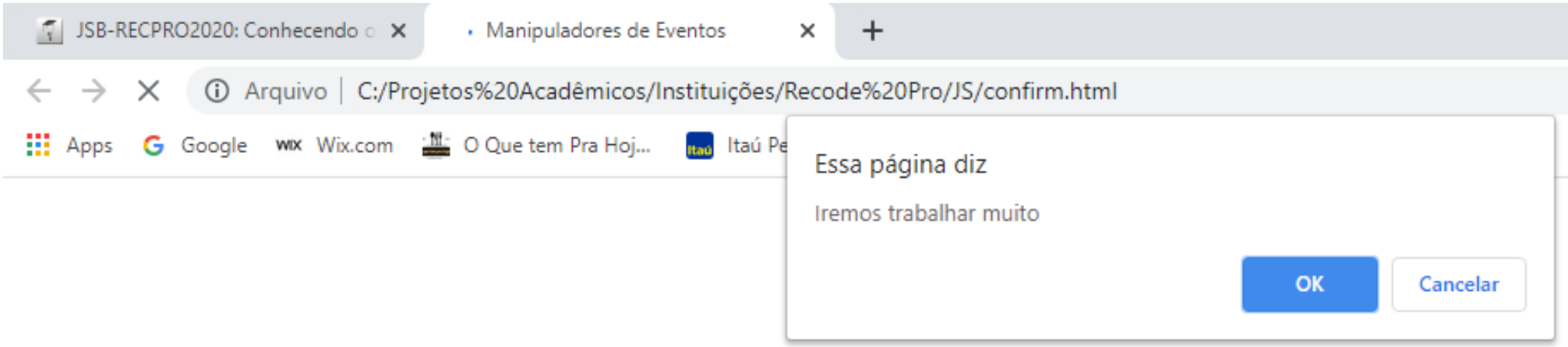
Por mais que os nomes dos botões possam variar, podemos perceber que estamos sempre falando dos nossos velhos conhecidos **TRUE** e **FALSE.**

JS

Exemplo:

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <h1>Metodo Confirm para o usuario</h1>
                <script>
                        confirm ("Iremos trabalhar muito");
                </script>
        </body>
</html>
```

JS

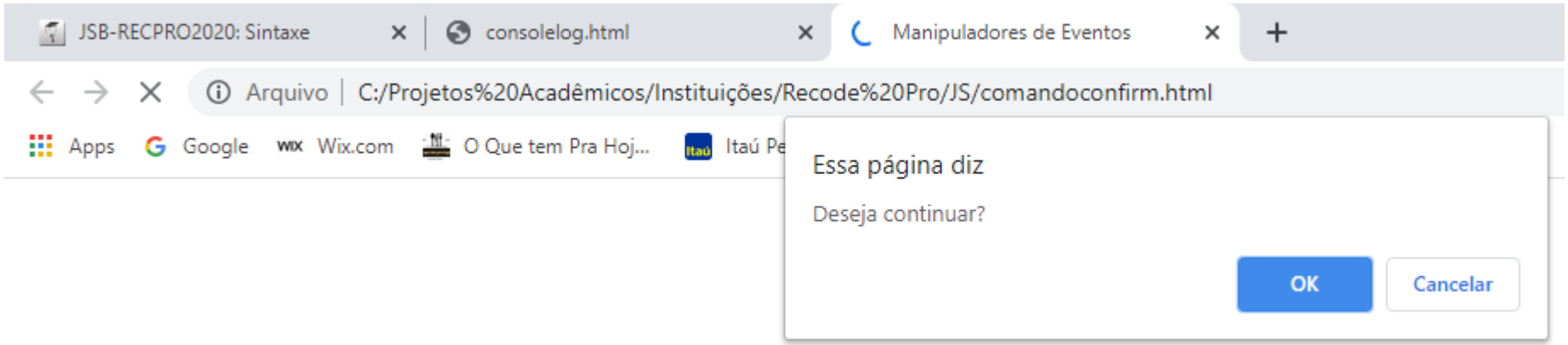

JSB-RECPRO2020: Conhecendo
Manipuladores de Eventos
Arquivo | C:/Projetos%20Acadêmicos/Instituições/Recode%20Pro/JS/confirm.html
Apps
Google
Wix.com
O Que tem Pra Hoj...
Itaú Pe
Essa página diz
Iremos trabalhar muito
OK
Cancelar

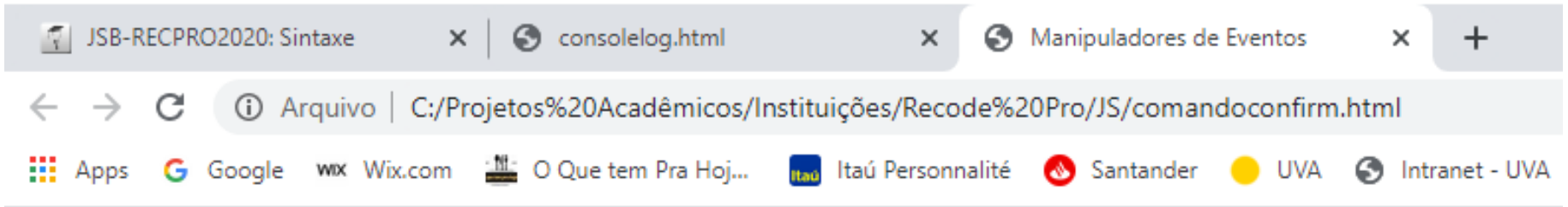
JS

Exemplo:

```
<html>
        <head>
                <title>Confirmação</title>
        </head>
        <body>
                <h1>Comando confirmação para o usuario</h1>
                <script>
                        let opcao = confirm("Deseja continuar?");
                         document.write("Sua escolha foi: " + opcao);
                </script>
        </body>
</html>
```

JS





Comando confirmação para o usuario

Sua escolha foi: true

JS

Exercício 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.

JS

Código do exercício 3:

```
<html>
	<head>
		<title>Confirmação</title>
	</head>
	<body>
		<script>
			let opcao = confirm("Deseja Prosseguir para a proxima
			etapa?");
			document.write("Sua escolha foi: " + opcao);
		</script>
	</body>
</html>
```

JS

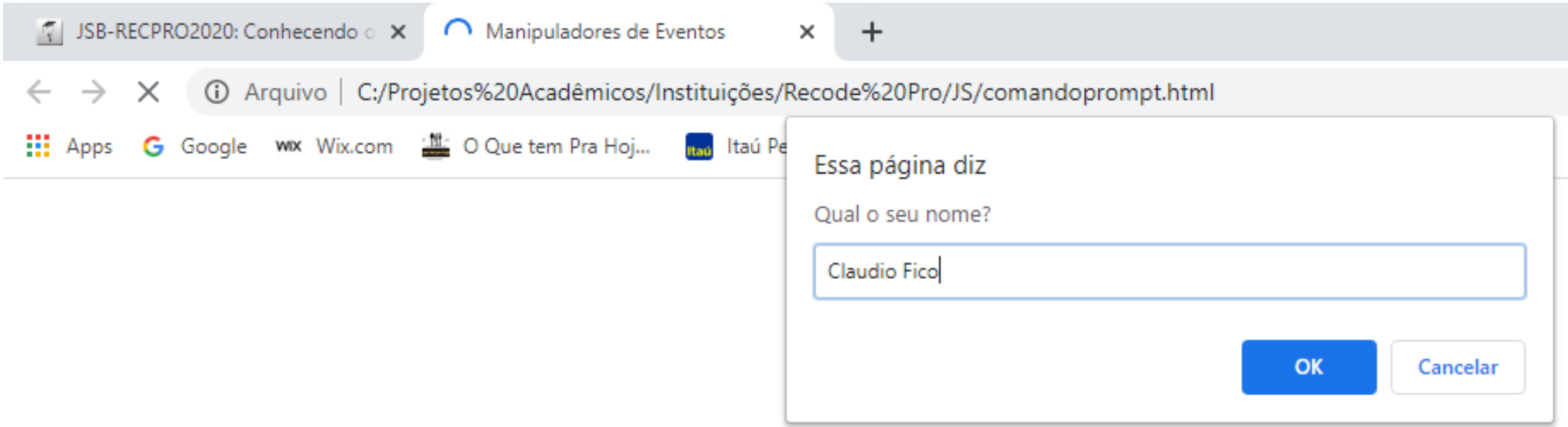
Prompt (caixa de mensagem para o usuário preencher)

Recuperar uma informação do usuário.

Exemplo:

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <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

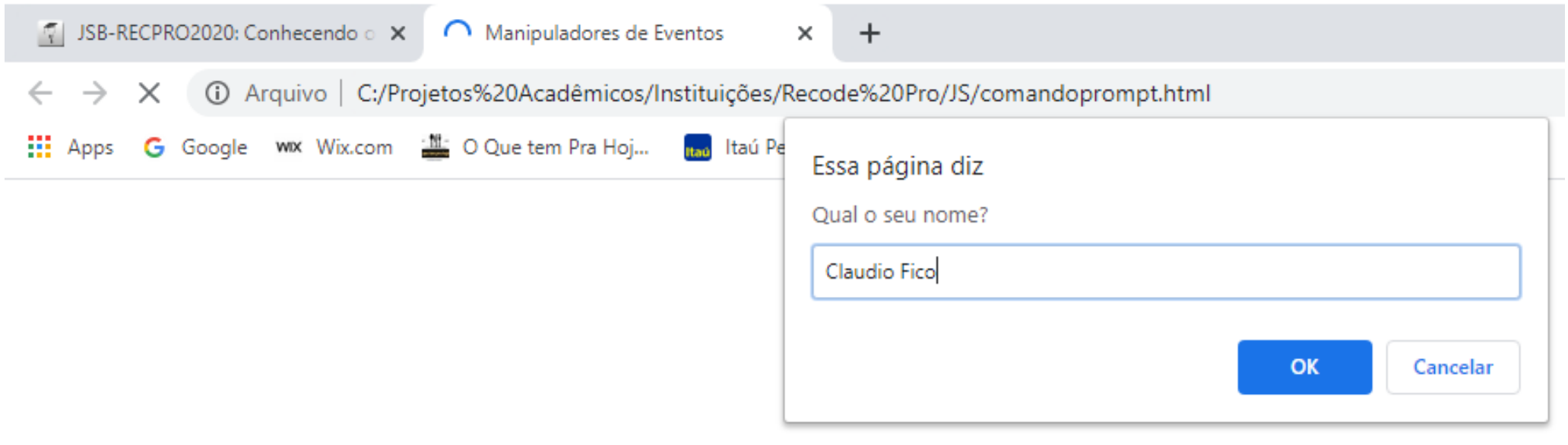
Var

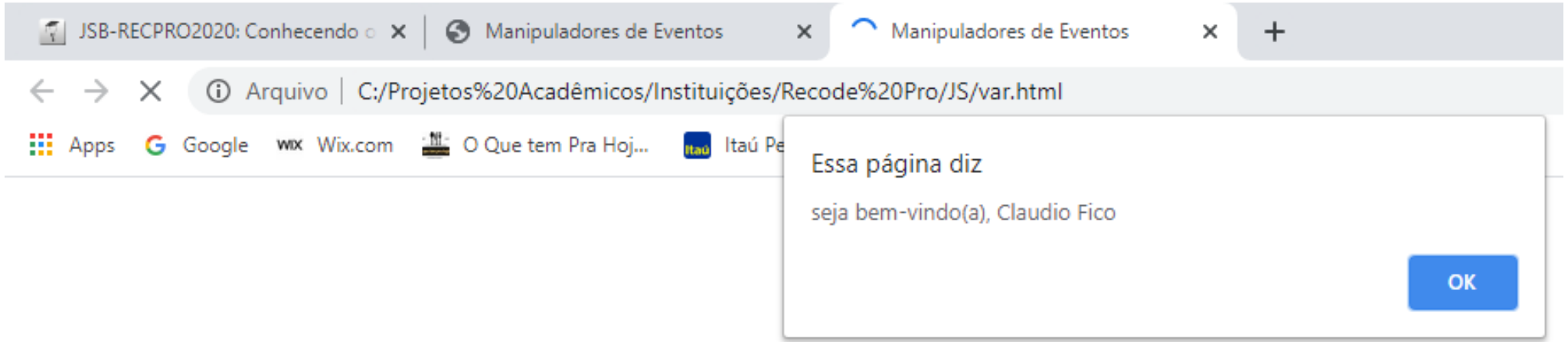
Usado para declaração de uma variável. Mas depois conheceremos o **let**.

Exemplo:

```
<html>
        <head>
                <title>Manipuladores de Eventos</title>
        </head>
        <body>
                <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

JS

Exercício 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.

JS

Código do exercício 4:

```
<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

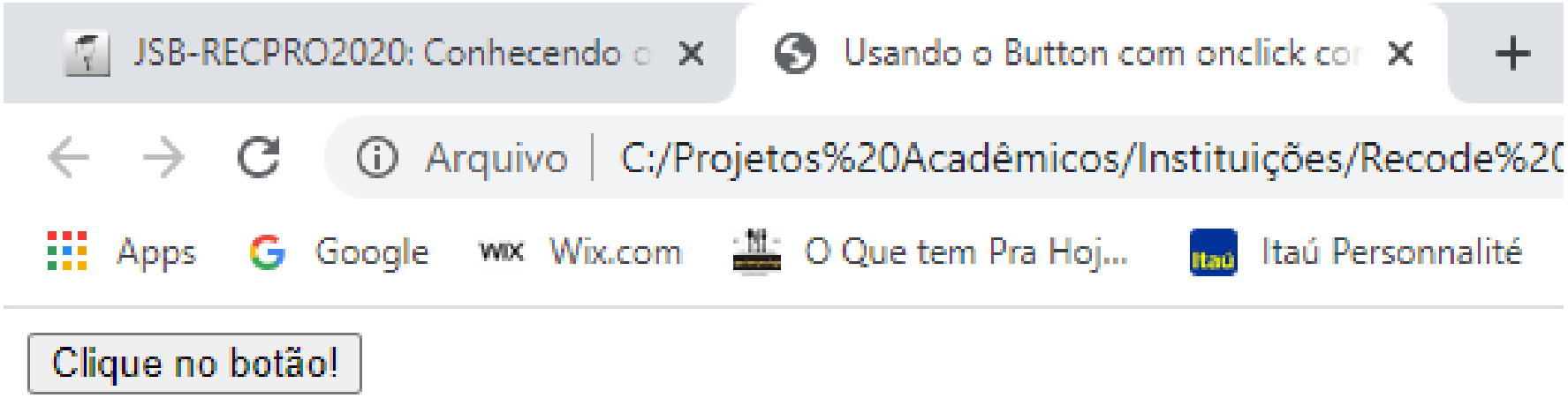
Exemplo (arquivo externo – func.js):

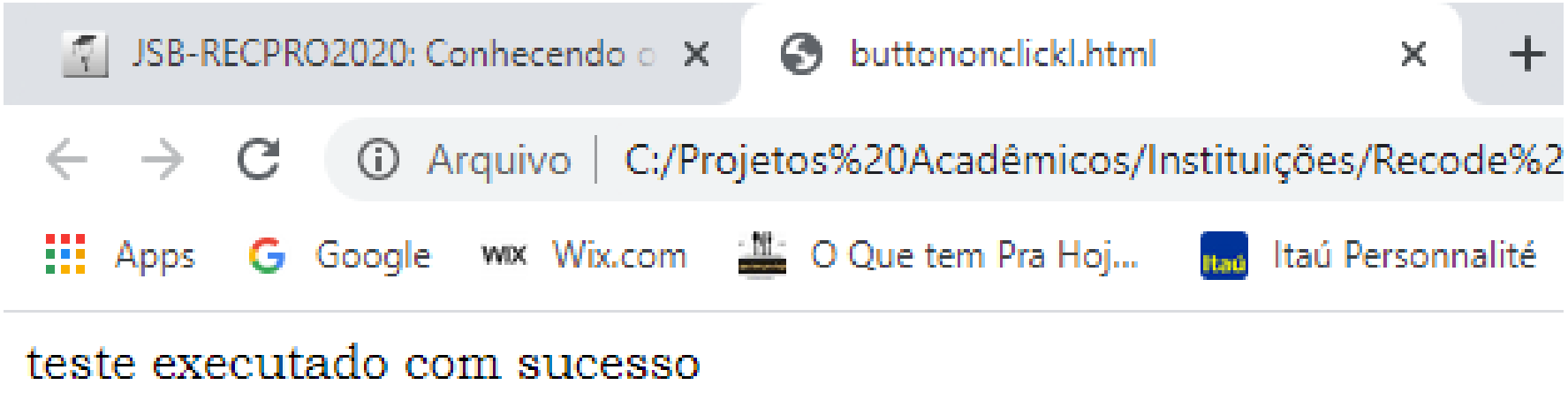
```
function exibe() {
   document.write("teste executado com sucesso");
}
```

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

JavaScript

Aula 2

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

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

Tipos de dados

Na programação, os tipos de dados são um conceito importante.
Para operar em variáveis, é importante saber algo sobre o tipo.
Sem tipos de dados, um computador não pode resolver isso com segurança

Não declare strings, números e booleanos como objetos!
Quando uma variável JavaScript é declarada com a palavra-chave "**new**", a variável é criada como um objeto:

```
let x = new String();       // Declares x as a String object
let y = new Number();    // Declares y as a Number object
let z = new Boolean();    // Declares z as a Boolean object
```

JS

Exemplo:

```
<html>
 <body>
  <h2>Tipos de Dados</h2>
  <p>A importancia</p>
  <script>
        let num = 10 + 15 + "otimo";
        document.write(num);
   </script>
 </body>
</html>
```

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.

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:

let nomeAluno = “Francisco”;

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

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

JS

Números

Os números JavaScript são sempre pontos flutuantes de 64 bits.

Inteiros (integer)

Representam números positivos, negativos ou fracionários. Exemplo:

- A = 500
- C = -32

Ponto flutuante

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

- X = 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

Comprimento da string

Para encontrar o comprimento de uma string, use a propriedade length.

```
<!DOCTYPE html>
<html>
        <body>
                <h2>String</h2>
                <p>Propriedade length</p>
                <script>
                        let txt = “MARTELO DE OURO";
                        let tam = txt.length;
                        document.write(tam);
                </script>
        </body>
</html>
```

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

Data

Por padrão, o JavaScript usará o fuso horário do navegador e exibirá uma data como uma string de texto completo:
Sáb, 03 de outubro de 2020 07:06:36 GMT-0300 (Horário Padrão de Brasília)

Nota: JavaScript conta meses de 0 a 11.
Janeiro é 0. Dezembro é 11.

ISO 8601 é o padrão internacional para representação de datas e horas. A sintaxe ISO 8601 (AAAA-MM-DD) também é o formato de data JavaScript.

JS

Objeto Data

Os objetos de data são criados pelo construtor **new Date()**

Observação:

Para que os métodos possam ser utilizados é necessária a utilização do new no objeto Date(). Caso contrário, o método não será reconhecido.

JS

Exemplo do uso da Data:

```
<!DOCTYPE html>
<html>
	<body>
	  <h2>JavaScript new Date()</h2>
	  <p>Formatação da data</p>
	  <script>
		let dat = new Date();
		document.write(dat);
	  </script>
	</body>
</html>
```

JS

Exemplo do uso da Data completa com valor fixo:

```
<!DOCTYPE html>
<html>
	<body>
	  <h2>JavaScript new Date()</h2>
	  <p>Formatação da data</p>
	  <script>
		let dat = new Date(2020, 10, 21, 10, 33, 30);
		document.write(dat);
	  </script>
	</body>
</html>
```

JS

Exemplo do uso da Data só com ano e mês:

```
<!DOCTYPE html>
<html>
	<body>
	  <h2>JavaScript new Date()</h2>
	  <p>Formatação da data</p>
	  <script>
		let dat = new Date(2020, 08);
		document.write(dat);
	  </script>
	</body>
</html>
```

JS

Métodos de obtenção de data

Método	Descrição
getFullYear()	Retorna o ano
getMonth()	Retorna o mês de (0-11)
getDate()	Retorna o dia (1-31)
getHours()	Retorna a hora (0-23)
getMinutes()	Retorna o minuto (0-59)
getSeconds()	Retorna o segundo (0-59)
getMilliseconds()	Retorna o milessegundo (0-999)
getTime()	Retorna a hora
getDay()	Retorna o dia da semana (0-6)
Date.now()	Retorna a hora. ECMAScript 5.

JS

Uso de métodos no Date()

```
<!DOCTYPE html>
<html>
	<body>
		<h2>Metodo getFullYear</h2>
		<p>Capturando o ano da data</p>
		<script>
				let dataobj = new Date();
				let ano = dataobj.getFullYear();
				document.write(ano);
		</script>
	</body>
</html>
```

JS

Outro tipos

Uma variável ainda pode obter valores:

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

JS

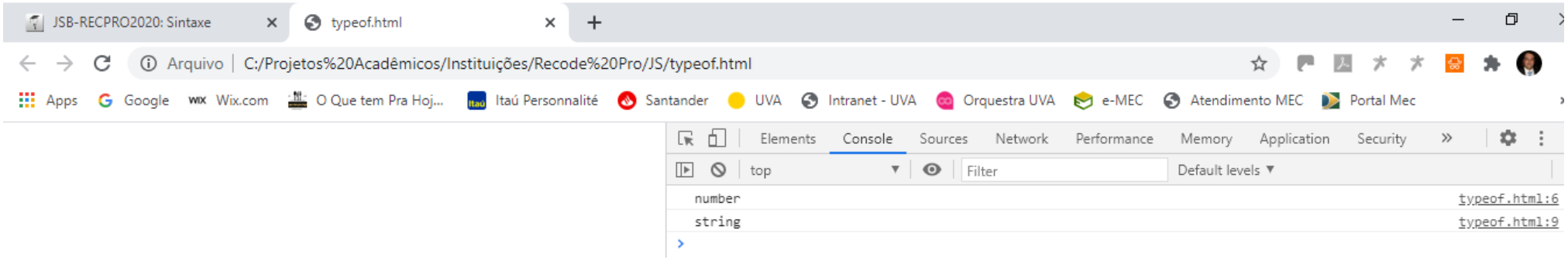
Typeof

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>
                <h1>Conversão de Valores</h1>
                <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

Exercício 5:

Criar um arquivo HTML de nome exerc5.html;

1. Criar um prompt para fazer entrada de um valor;
2. Criar outro prompt para fazer entrada de um segundo valor;
3. Exibir o resultado através do Alert.....

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“
```

Boas Práticas!

```
nome = "José"
sobrenome = "Silva"
nome = nome+" da "+sobrenome
```

JS

Estrutura de decisão

If else

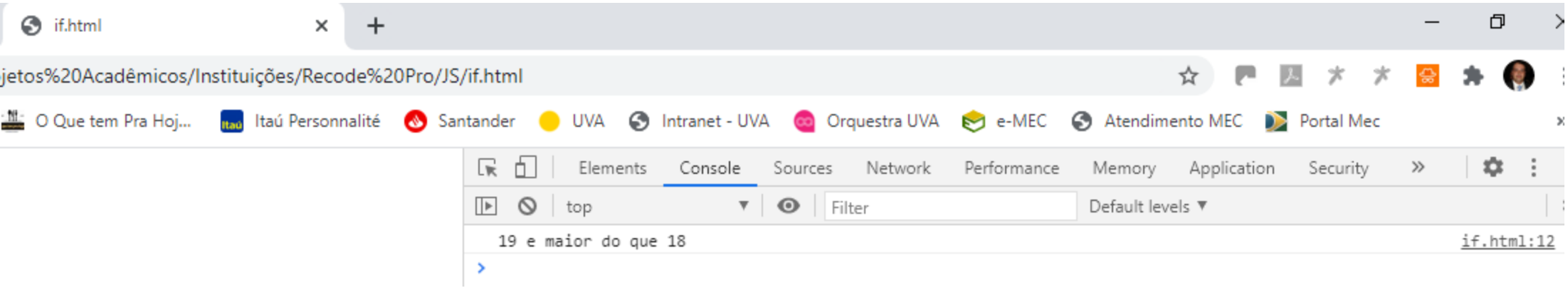
```
if (condição) // caso a expressão verificada retorne true
{
        <comando>;
        <comando>;
}
else // caso a expressão verificada retorne false
{
        <comando>;
        <comando>;
}
```

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

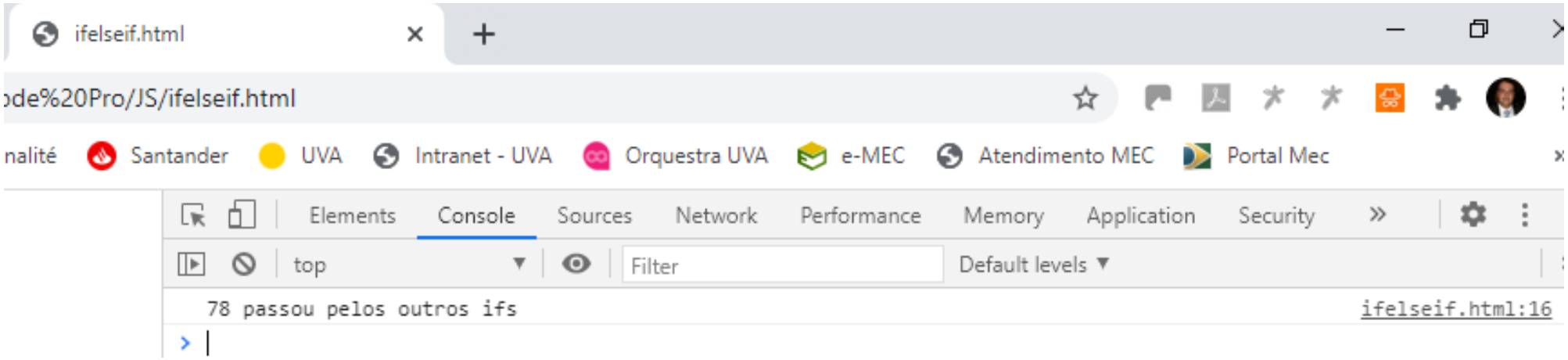
```
if (condição) // expressão verificada retorne true
{
        <comando>;
}
else if (condição) // retorna false, poderá fazer uma nova condição
{
        <comando>;
}
else // retorna false do else if
{
        <comando>;
}
```

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

Exercicio 6:

Criar arquivo exerc6.html

1. Criar um primeiro prompt para entrada do valor numérico;
2. Criar um segundo prompt para entrada de outro valor numérico;
3. Testar qual o maior valor;
4. Exibir uma mensagem com o maior valor.

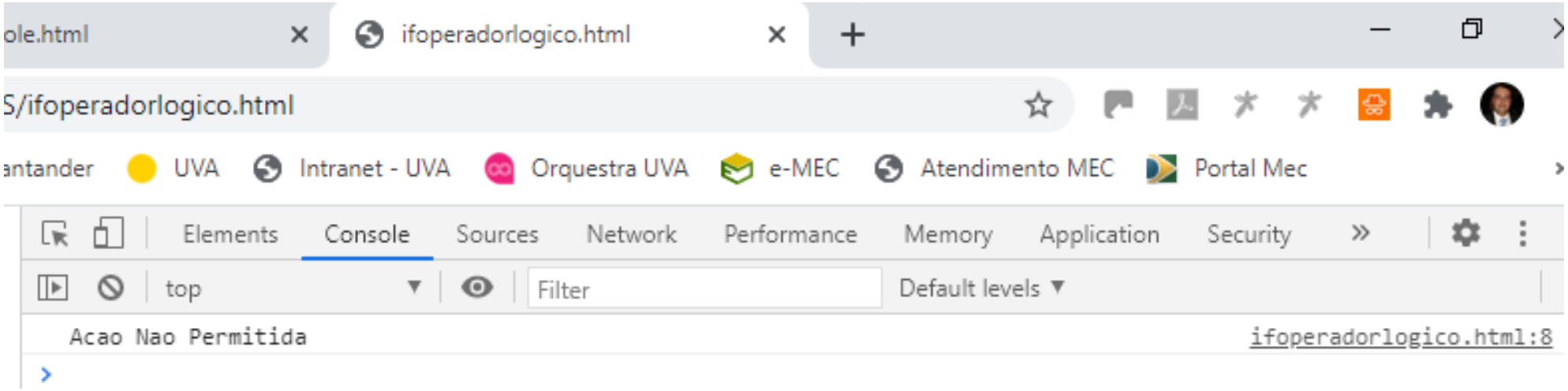
JS

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

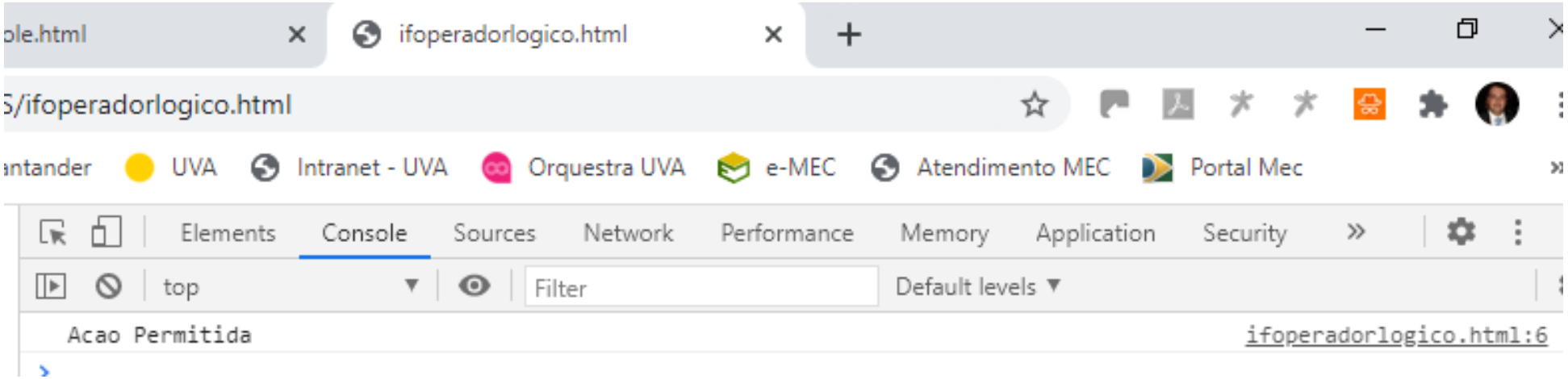
JS

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

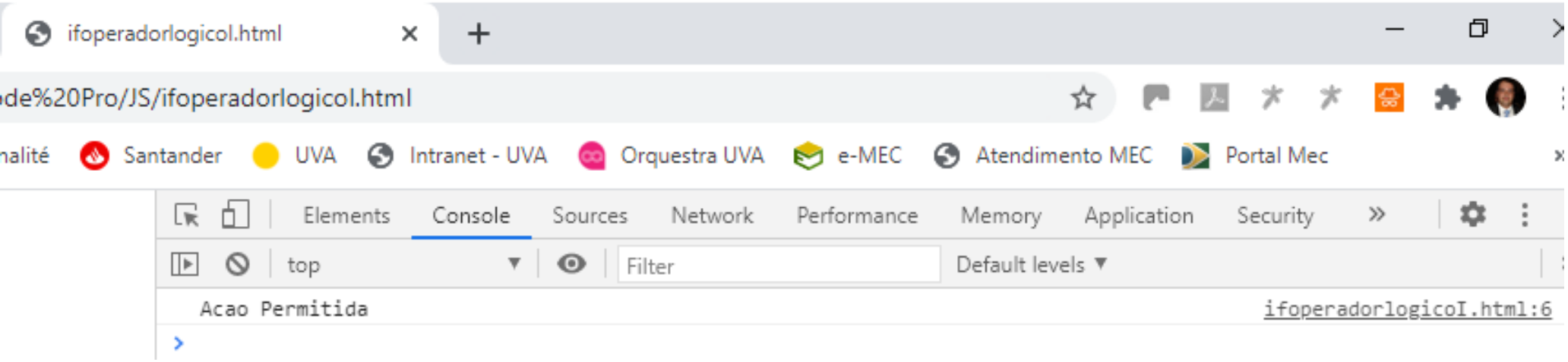
JS

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

Exercício 7:

Criar um arquivo HTML para cada atividade.

Primeira atividade:

Fazer um programa para que o usuário digite um número e o programa verifique se os números estão entre o intervalo de valores de 10 e 20. Exibir mensagem “numero esta no intervalo desejado”, caso contrario, “numero invalido”.

Segunda atividade:

Fazer um programa para que o usuário digite um número e o programa verifique se o número é múltiplo de 5. Exibir mensagem “numero múltiplo de 5”, caso contrario, numero não e múltiplo”.

JS

Exercício 8:

Criar um arquivo HTML para cada atividade.

Terceira atividade:

Fazer um programa para que o usuário digite um número e o programa verifique se o número é múltiplo de 2 ou de 5 ou de 10. Exibir mensagem. “numero e multiplo”, caso contrário “numero invalido.

Quarta atividade:

Fazer um programa para que o usuário digite três notas, o programa deverá calcular a média aritmética e exibir o status.
Media >= 7, aprovado
Media <3, reprovado
Senão, prova final

JS

Exercício 9:

Criar um arquivo HTML para cada atividade.

Quinta atividade:

Fazer um programa para que o usuário digite um salário e o programa deverá calcular seu reajuste. Exibir mensagem com o valor do desconto.
Salario <= 600 e sexo = “M”, aplicar desconto de 20%
Salario <= 1200 e sexo = “F”, aplicar desconto de 20%
Salario <= 2000 e sexo “M”, aplicar desconto de 25%
Senão, desconto de 30% - ambos os sexos

Fórmula: desconto = salario * 0.2 ou 0.25 ou 0.30;

JavaScript

Aula 3

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.

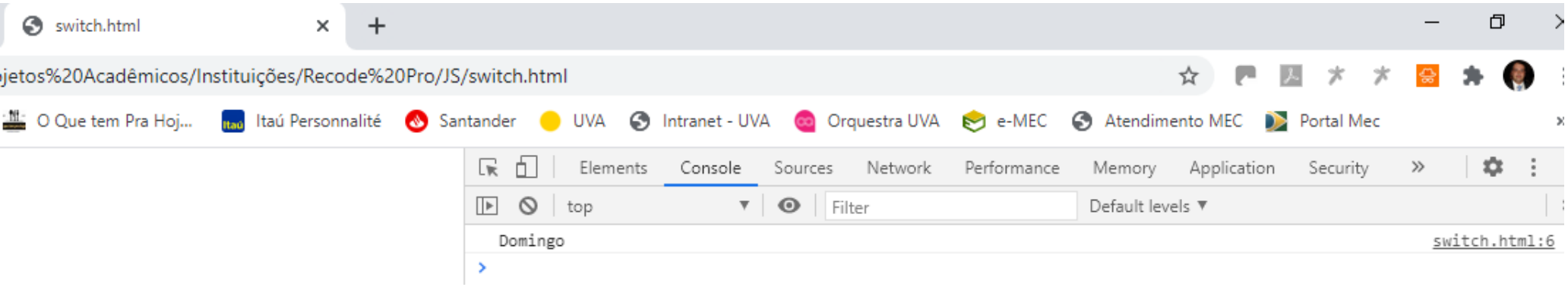
Default → significa “padrão”, é o bloco de código que será executado caso nenhuma opção existente seja a correta.

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>
```

JS

Comandos condicionais e repetição

Instrução While

A instrução while realiza uma ação enquanto determinada condição for satisfeita.

Sua sintaxe básica é:

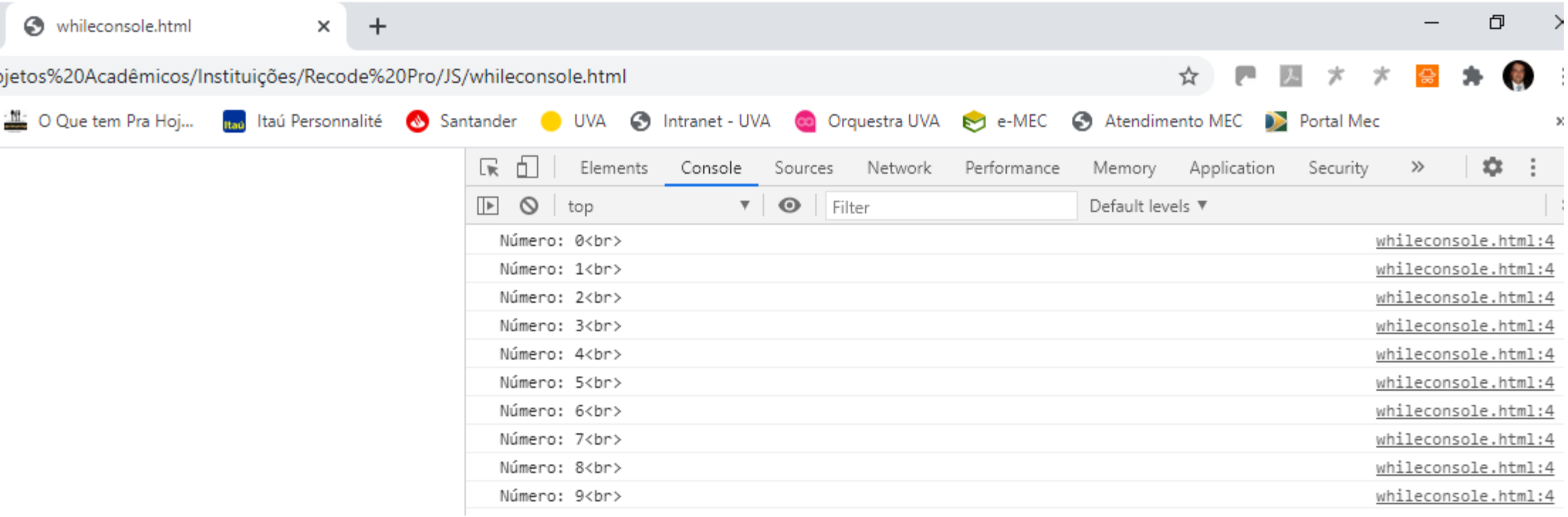
```
while (expressão) {
        <comando>;
        <comando>;
        }
```

JS

Veja no exemplo seguinte a utilização do laço while que é repetido por total de 10 vezes:

```
<script>
        num = 0;
        while (num < 10) {
                console.log("Número: "+num+"<br>");
                num++;        /* faz o incremento para a variável
                              num*/
                }
</script>
```

JS


whileconsole.html
ojetos%20Acadêmicos/Instituições/Recode%20Pro/JS/whileconsole.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
Número: 0
 whileconsole.html:4
Número: 1
 whileconsole.html:4
Número: 2
 whileconsole.html:4
Número: 3
 whileconsole.html:4
Número: 4
 whileconsole.html:4
Número: 5
 whileconsole.html:4
Número: 6
 whileconsole.html:4
Número: 7
 whileconsole.html:4
Número: 8
 whileconsole.html:4
Número: 9
 whileconsole.html:4

JS

Instrução for

A instrução for realiza uma ação até que determinada condição seja satisfeita.

Sua sintaxe básica é:

```
for (início; condição; incremento) {
        <comando>;
        }
```

O início determina o valor inicial do laço for. Normalmente é 0 ou 1, porém poderá ser especificado qualquer outro valor. O valor especificado é atribuído em uma variável, por exemplo i=0, count=1.

JS

A condição determina a expressão que irá controlar o número de repetições do laço. Enquanto esta expressão for verdadeira, o laço continuará, caso o laço seja falso, o laço terminará. Por exemplo: i<20. Enquanto o valor de i for menor que 20, a condição é verdadeira.
O incremento determina como o laço irá contar, de 1 em 1, 2 em 2, 5 em 5, 10 em 10, enfim.

Exemplo: i++. Será aumentado o valor da variável i a cada repetição. Diferente de todas as outras linguagens, em JavaScript, a instrução for, utiliza ponto e vírgula para separar os argumentos ao invés de vírgula.

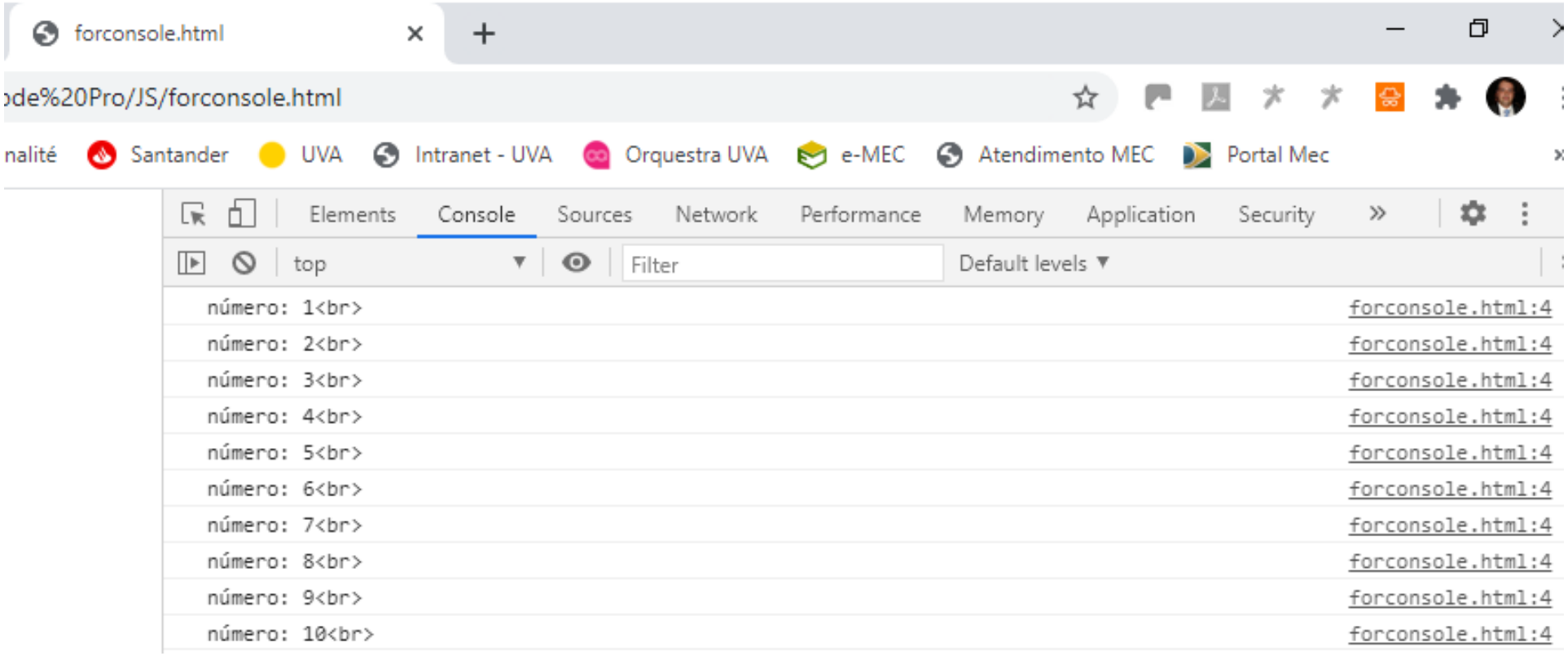
JS

Vejamos um exemplo prático de utilização do laço for que conta valores de 1 até 10, acrescentando um valor de cada vez:

Exemplo:

```
<script>
        let num = 0;
        for (i=1 ; i<=10 ; i++){
                console.log("número: "+ i +"<br>");
                }
</script>
```

JS


forconsole.html
ode%20Pro/JS/forconsole.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
número: 1
 forconsole.html:4
número: 2
 forconsole.html:4
número: 3
 forconsole.html:4
número: 4
 forconsole.html:4
número: 5
 forconsole.html:4
número: 6
 forconsole.html:4
número: 7
 forconsole.html:4
número: 8
 forconsole.html:4
número: 9
 forconsole.html:4
número: 10
 forconsole.html:4

JS

Arrays

são usados para armazenar vários valores em uma única variável.

Exemplo:

let carros = ["Mercedes", "Volvo", "BMW"]

Carros	Mercedes	Volvo	BMW
Índices	0	1	2

O que é um Array?

Um Array é uma variável especial, que pode conter mais de um valor por vez.

JS

Se você tem uma lista de itens (uma lista de nomes de carros, por exemplo), armazenar os carros em variáveis individuais pode ser assim:

- let carro1 = “Mercedes";
- let carro2 = "Volvo";
- let carro3 = "BMW";

No entanto, e se você quiser percorrer os carros e encontrar um específico?
E se você não tivesse 3 carros, mas 300?
A solução é um Array!

JS

Criando um Array

Sintaxe:

```
let array_name = [item1, item2, ...]
```

Exemplo:

```
let carros = ["Mercedes", "Volvo", "BMW" ]
```

Espaços e quebras de linha não são importantes. Uma declaração pode abranger várias linhas:

```
let carros = ["Mercedes",
              "Volvo",
              "BMW" ];
```

JS

Acesse os elementos de um Array

Você se refere a um elemento do Array referente ao número de índice. Esta declaração acessa o valor do primeiro elemento em carros:

```
let name = carros[0];
```

Esta declaração modifica o primeiro elemento em carros:

```
carros[0] = "Mercedes";
```

Exemplo:

var carros = [“Mercedes", "Volvo", "BMW"];

carros[0] → [0] é o primeiro elemento em um Array. [1] é o segundo e assim por diante. Os índices do Array sempre irão começar com o valor 0.

JS

Acessando um Array completo

Com o JavaScript, é possível acessar a matriz completa consultando o nome do Array:

```
let carros = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = carros;
```

Arrays são objetos

- Arrays são um tipo especial de objetos.
- O tipo de operador em JavaScript retorna "objeto" para arrays.
- Mas, os arrays de JavaScript são melhor descritos como arrays.

JS

Arrays usam números para acessar seus "elementos“

Neste exemplo, a pessoa[0] retorna Carlos

```
var pessoa = [ "Carlos", "Silva", 46 ];
```

Exemplo:

```
<script>
    var pessoa = ["Carlos", "Silva", 46];
    console.log(pessoa[1]);
    console.log(pessoa[2]);
</script>
```

JS

Iterando sobre os itens de um array

A melhor maneira de percorrer um Array é usando um loop "for":

```
<script>
var frutas, text, fLen, i;
frutas = ["Banana", "Laranja", "Manga", "Manga"];
fLen = frutas.length;
text = "<ul>";  //lista não ordenada
for (i = 0; i < fLen; i++)
        {
                console.log(text += "<li>" + frutas[i] + "</li>");
        }
</script>
```

JS

Método push()

Para adicionar elementos em um Array com JavaScript, podemos utilizar o método push(), que significa “empurrar”. Desta forma, o próximo dado será sempre inserido ao final e não precisamos nos preocupar com a posição do índice.

Exemplo:

```
<script>
        let frutas = [];
        frutas.push(“carambola”);
        frutas.push(“maça”);
        frutas.push(“amora”);
        console.log(frutas);
</script>
```

Exibição do Array

frutas	carambola	maça	Amora
índices	0	1	2

JS

Método pop(), splice() e shift()

Podemos também remover um elemento do Array. Para isso, o primeiro método de remoção que usamos é o pop(). Ele remove simplesmente e apenas o último elemento de um array.

Exemplo:

```
<script>
        let frutas = [];
        frutas.push(“carambola”);
        frutas.push(“maça”);
        frutas.push(“amora”);
        frutas.pop();
        console.log(frutas);
</script>
```

Exibição do Array

frutas	carambola	Maça
índices	0	1

JS

Existem outros métodos de remoção como o splice() e o shift(). O **splice()** remove um elemento específico do Array e recebe **2 parâmetros**: o primeiro é a posição inicial, o segundo é a quantidade de elementos. Portanto, se quisermos excluir a pessoa "Ana", faríamos a seguinte declaração:

Exemplo:

```
<script>
        let frutas = [];
        frutas.push("carambola");
        frutas.push("maça");
        frutas.splice(1,1);
        console.log(frutas);
</script>
```

frutas	carambola	maça
índices	0	1

Exibição do Array

carambola

JS

O método **shift()**, que, ao contrário do pop(), remove o primeiro elemento de um Array.

Exemplo:

```
<script>
        let frutas = [];
        frutas.push(“carambola”);
        frutas.push(“maça”);
        frutas.shift();
        console.log(frutas);
</script>
```

frutas	carambola	maça
índices	0	1

Exibição do Array

maça

JS

Proriedade e Métodos do Array

A força real dos Arrays de JavaScript são as propriedades e métodos do Array incorporada:

- let x = carros.length; → A propriedade length retorna o número de elementos.

- elementos let y = carros.sort(); → O método sort() classifica os arrays.

JS

A Propriedade length para o array

A propriedade length de um Array retorna o comprimento de um Array (o número de elementos do Array).

let frutas = ["Banana", "Laranja", "Maça", "Manga"];

frutas.length; → o length de frutas é 4 (quatro dados no array).

Exemplo:

```
<script>
   let frutas = ["Banana", "Laranja", "Maça", "Manga"];
   console.log(frutas.length);
</script>
```

A propriedade length é sempre mais do que o índice do Array mais alto, pois o índice inicia com 0 e o length inicia com 1.

JS

A Propriedade sort() para o array

A propriedade sort() de um Array faz a ordenação das informações em ordem descendente.

let frutas = ["Banana", "Laranja", "Maça", "Manga"];

Exemplo:

```
<script>
    let frutas = ["Banana", "Laranja", "Maça", "Manga"];
    frutas.sort();
    console.log(frutas);
</script>
```

frutas	amora	banana	carambola	maça
índices	0	1	2	3

JavaScript

Aula 4

JS

Funções

Uma função JavaScript é definida com a palavra chave **function**, seguida por um **nome** , seguido por parênteses **()** .

Os nomes das funções podem conter letras, dígitos, sublinhados e cifrões (mesmas regras das variáveis).

Os parênteses podem incluir nomes de parâmetros separados por vírgulas:

(parâmetro1, parâmetro2, ...)

O código a ser executado, pela função, é colocado entre chaves: **{}**

JS

Invocação/chamando uma Função

O código dentro da função será executado quando "algo" **invocar** (chamar) a função:

- Quando ocorre um evento (quando um usuário clica em um botão);
- Quando é invocado (chamado) a partir do código JavaScript; e
- Automaticamente (auto-invocado).

JS

Retorno de Função

Quando o JavaScript atinge uma instrução **return**, a função para de ser executada.

Se a função foi chamada a partir de uma instrução, o JavaScript "retornará" para executar o código após a instrução de chamada.

As funções geralmente calculam um **valor de retorno** . O valor de retorno é "retornado" de volta ao "chamador"

JS

Exemplo de uma função simples:

```
<html>
  <body>
        <script>
        function exibe()
        {
        document.write("Teste bem sucedido");
        }
        </script>
        <button onclick='exibe()'>Clique para calcular!</button>
  </body>
</html>
```

JS

Exemplo com o uso de return:

```
<html>
   <body>
         <script>
                  function mult(num1, num2)
                  {
                  return num1 * num2;
                  }
                  let val = mult(3, 5);
                  document.write(val);
         </script>
   </body>
</html>
```

JS

Exercício:

1. Criar um arquivo .html;
2. Criar uma função na tag <head> para somar dois números;
3. Exibir o resultado do somatório com o comando alert;
4. Criar na tag <body> a entrada de dados de dois valores utilizando o prompt;
5. Criar um título com a tag <h2>;
6. Criar um botação para chamar a função;

JS

Codigo do exercício:

```
<html>
        <head>
                <title>Funcoes</title>
                <script>
                        function calcsoma(num1,num2)
                        {
                                let tot;
                                 tot = num1+num2;
                                alert("A soma foi, " + tot);
                        }
                 </script>
        </head>
        <body>
                <form>
                        <label for="var1">Digite o primeiro numero</label><br>
                         <input type="text" name="var1"> <br><br>
                         <label for="var2">Digite o segundo numero</label><br>
                         <input type="text" name="var2">
                        <h1 style = "font-family: fantasy; color: cornflowerblue">Agora faremos o informe de
                        valores</h1>
                        <Button onclick = "calcsoma(parseInt(var1.value), parseInt(var2.value))">Clique aqui para
                        calcular</Button>
                </form>
        </body>
</html>
```

JS

Exercício:

1. Criar um arquivo .html;
2. Criar uma função na tag <head> para somar dois números;
3. Criar uma função na tag <head> para subtrair dois números;
4. Criar uma função na tag <head> para multiplicar dois números;
5. Exibir o resultado do somatório com o comando alert. Um em cada função;
6. Criar na tag <body> a entrada de dados de dois valores utilizando o prompt;
7. Criar um título com a tag <h2>;
8. Criar um botão para chamar a função soma;
9. Criar um botão para chamar a função subtrair;
10. Criar um botão para chamar a função multiplicar;

JS

Codigo do exercício:

```
<html>
<head>
    <title>Funcoes</title>
    <script>
      function calcsoma(val1, val2)
      {
        let tot;
        tot = val1 + val2;
        alert("o valor total da soma e, " + tot);
      }
      function calcsub(val1, val2)
      {
        let tot;
        tot = val1 - val2;
        alert("o valor total da subtracao e, " + tot);
      }
      function calcmult(val1, val2)
      {
        let tot;
        tot = val1 * val2;
        alert("o valor total da multiplicacao e, " + tot);
      }
    </script>
</head>
<body>
      <form>
        <label for="var1">Digite o primeiro numero</label><br>
        <input type="text" name="var1"> <br><br>
        <label for="var2">Digite o segundo numero</label><br>
        <input type="text" name="var2">
        <h2 style="color: darkcyan">Agora faremos o informe dos resultados</h2>
        <Button onclick = "calcsoma(parseInt(var1.value), parseInt(var2.value))">Clique aqui para somar</Button>
        <Button onclick = "calcsub(parseInt(var1.value), parseInt(var2.value))">Clique aqui para subtrair</Button>
        <Button onclick = "calcmult(parseInt(var1.value), parseInt(var2.value))">Clique aqui para multiplicar</Button>
      </form>
</body>
</html>
```

JS

Document Object Model - DOM

Quando uma página é carregada, o navegador cria um **Document Object Model** da página.

O navegador interpreta cada palavra, letra ou símbolo de um HTML e exibe o resultado na tela:

- Esse resultado é uma página visível para o usuário do navegador

Resultado da interpretação do HTML é armazenado em uma estrutura de objetos:

- Document Object Model (DOM)

JS

Document Object Model – DOM

Cada elemento, atributo e texto HTML no DOM torna-se um objeto

- Objetos podem ser acessados de modo independente pelos scripts.

É um padrão definido pelo W3C para acesso a documentos

DOM do HTML define:

- Objetos e propriedades de todos os elementos em um documento HTML/XHTML;
- Métodos para manipular (obter/modificar/adicionar/apagar) cada elemento.

JS

Document Object Model – DOM

Todo documento XHTML é uma hierarquia de elementos:

- Todos são subordinados à *tag<html>;*
- <html> possui um cabeçalho e um corpo (body); e
- Corpo possui outras tags ligadas a ele.

Essa hierarquia, em DOM, é representada como uma estrutura de dados de árvore.

DOM define objetos e propriedades para cada elemento:

- Cada objeto de elemento é ligado ao seu objeto “pai” na árvore DOM

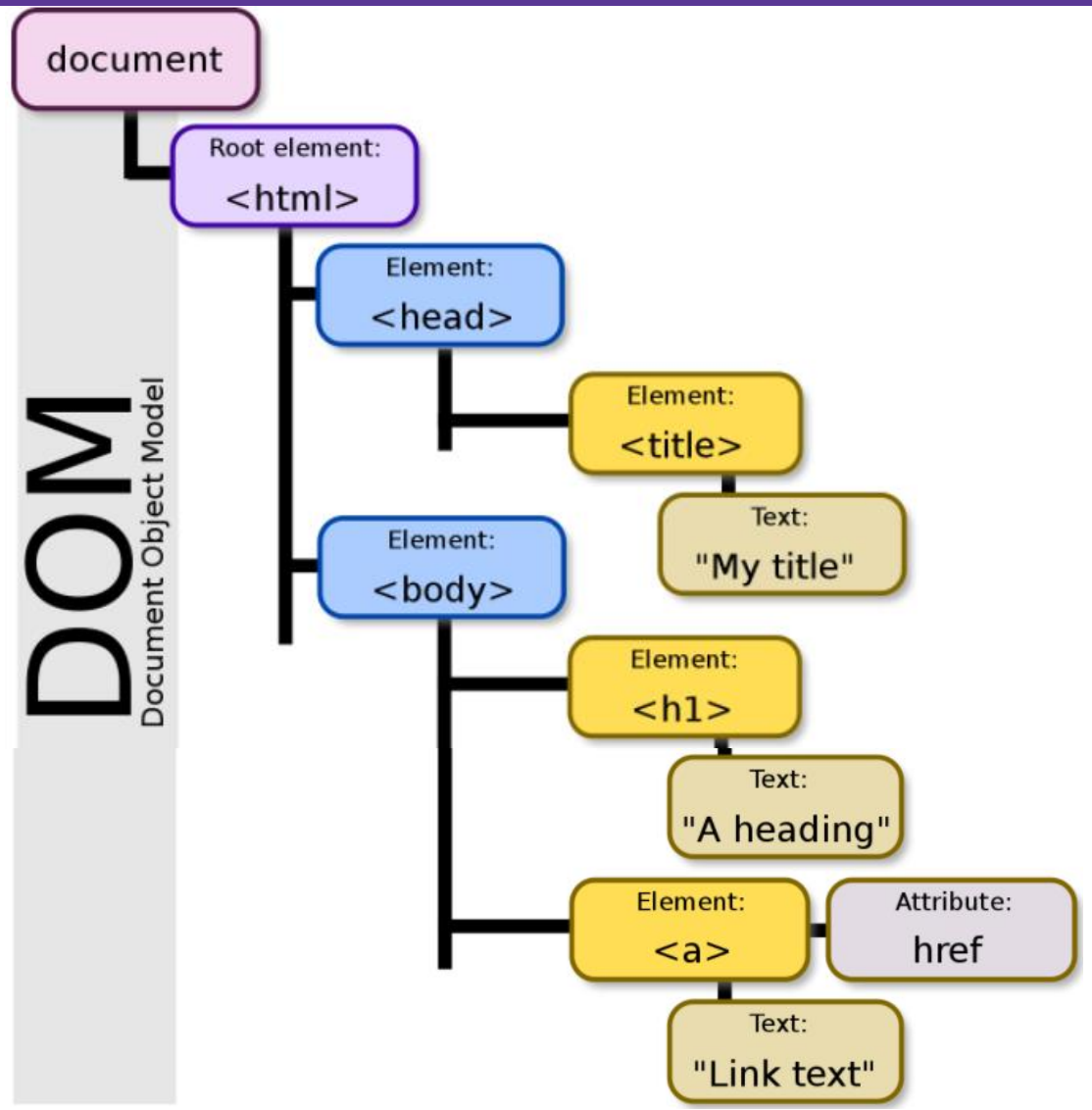
JS

Document Object Model – DOM

Com o modelo de objeto, JavaScript tem todo o poder necessário para criar HTML dinamicamente:

- JavaScript pode alterar todos os elementos HTML na página;
- JavaScript pode alterar todos os atributos dos elementos HTML na página;
- JavaScript pode alterar todos os estilos CSS;
- JavaScript pode remover um elemento HMTL e seus atributos;
- JavaScript pode adicionar um elemento HMTL e seus atributos;
- JavaScript pode reagir a todos os eventos que ocorrerem em uma página; e
- JavaScript pode criar novos eventos na page.

JS


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribute:
href
Text:
"Link text"

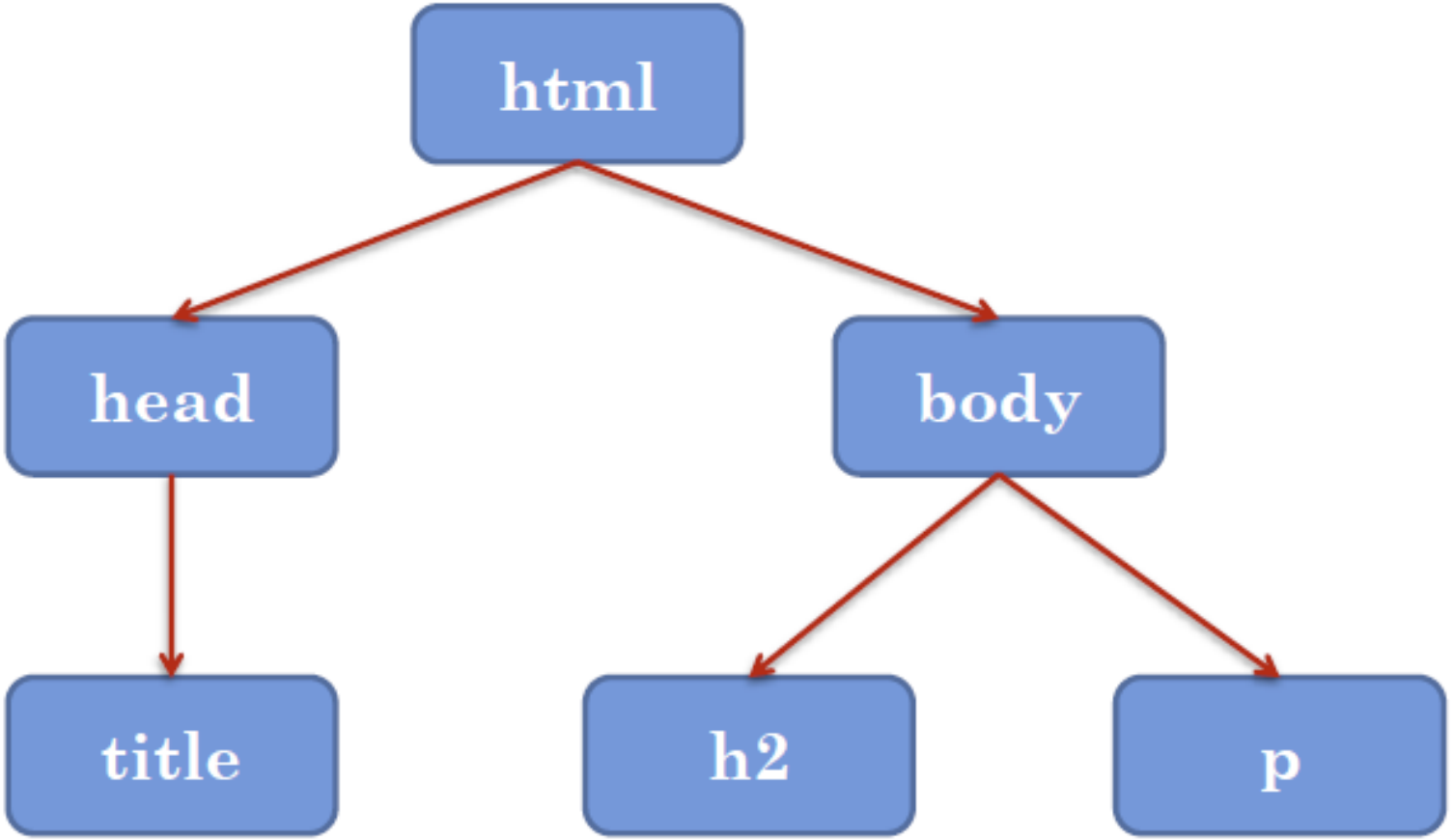
JS

EXEMPLO:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Testando DOM</title>
        </head>
        <body>
                <h2>DocumentObjectModel(DOM)</h2>
                <p>DOM permite o acesso e a manipulação de
                documentos HTML por meio de funções
                acessíveis ao JavaScript.
                </p>
        </body>
</html>
```

JS

Representação do exemplo anterior na árvore DOM



JS

NÓS da árvore DOM

Cada elemento do HTML é representado como um “nó” na árvore DOM.

Nó de documento (document):

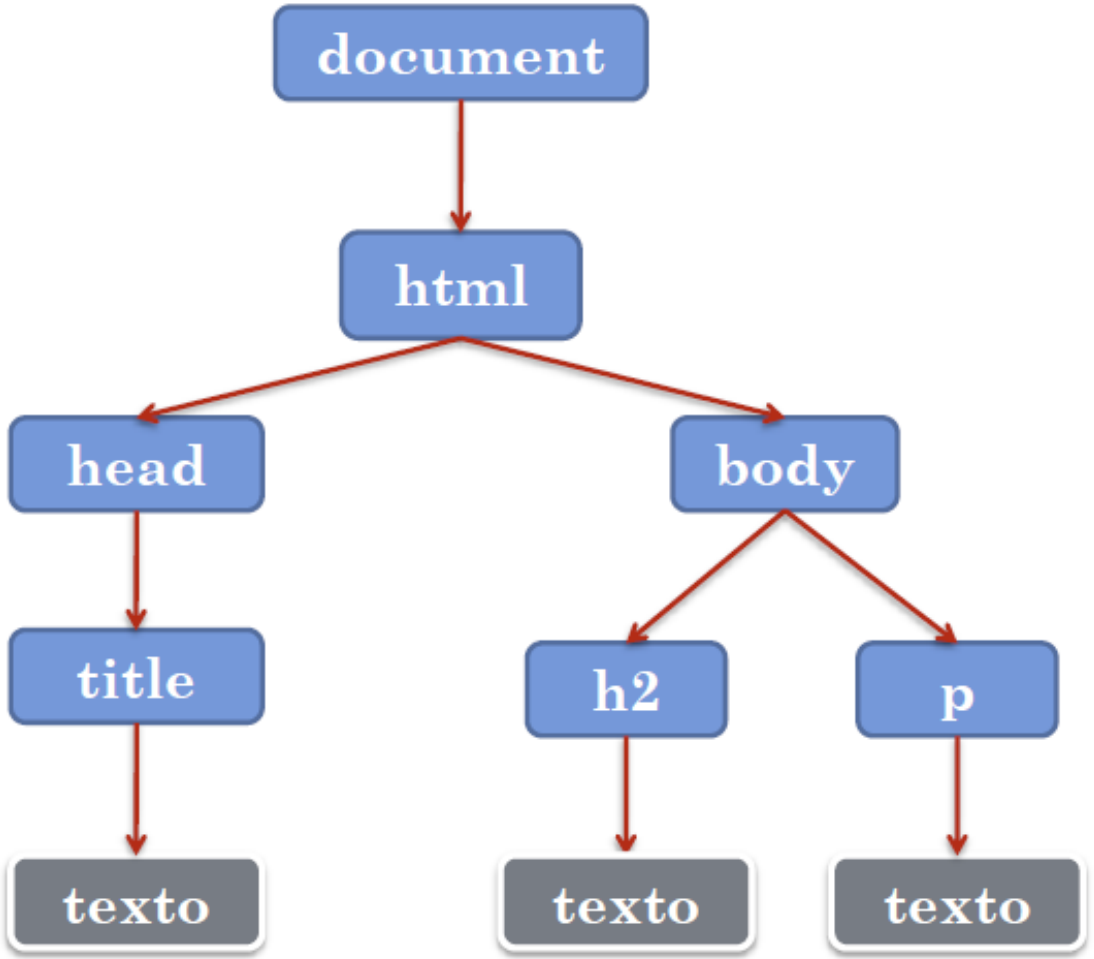
- É a raiz de toda a representação DOM; e
- Ponto de entrada para obter os outros nós.

Tipos de nós:

- Elemento *<a>*
- Atributo *href, id, class, etc.*
- Texto “*http://www.abc.com*”
- Comentário *<!--texto ->*

JS

Exemplo anterior com informações complementares



JS

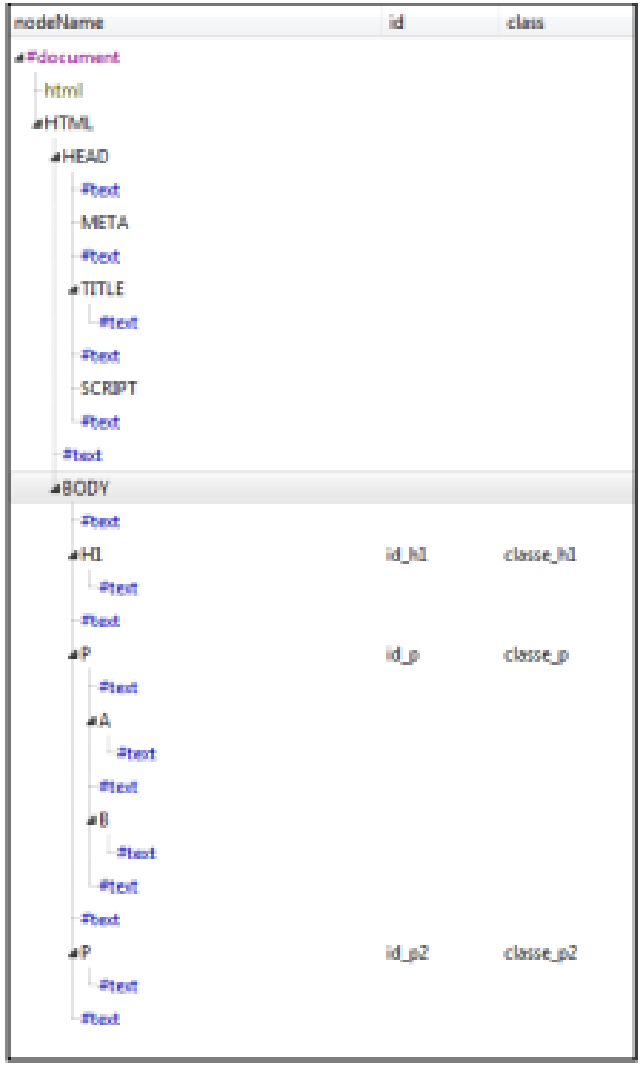
Relacionamento entre os nós: pais, filhos e irmãos

Em uma árvore:

- Nó topo é chamado de **principal** (root);
- Todo nó possui um **pai** (exceto o root);
- Um nó pode ter qualquer número de **filhos**;
- **Irmãos** (siblings) são nós com o mesmo pai; e
- Um nó sem filhos é chamado de **folha.**

JS


#document
Pai de todos
HTML
HEAD
BODY
Filhos de BODY e Irmãos
META
SCRIPT
TITLE
H1
P
P
#text
#text
#text
#text
Filhos de HEAD e Irmãos
A
B
#text
#text
Filhos de P e Irmãos


nodeName
id
class
#document
html
HTML
HEAD
#text
META
#text
TITLE
#text
#text
SCRIPT
#text
#text
BODY
#text
H1
id_h1
classe_h1
#text
#text
P
id_p
classe_p
#text
A
#text
#text
B
#text
#text
#text
P
id_p2
classe_p2
#text
#text

JS

Existe um outro objeto que é pai de todos estes elementos (não representado na estrutura), que representa a página do navegador: o objeto **window**.

Com o uso do objeto **window**, temos acesso à vários recursos do navegador, como, por exemplo:

- Abrir uma nova janela no navegador (window.open);
- Fechar uma janela (window.close);
- Definir que site será acessado (window.location);
- Definir o tamanho e posicionamento das janelas (propriedades width, height, top e left de window.open); e
- Exibir ou ocultar itens da janela, como barra de endereços, barra de status, barras de rolagens, entre outros.

JS

O objeto **document**, como podemos imaginar, representa o documento HTML em si e toda sua estrutura. Com ele, por meio de seus métodos e atributos, podemos:

- acessar um elemento específico, pelo seu id, nome, tag ou classe (apenas por id retorna um único elemento, os demais podem retornam um array de elementos).
    - document.getElementById();
    - document.getElementsByName()
    - document.getElementsByTagName()
    - document.getElementsByClassName()
- Acessar o título do documento (document.title);
- acessar a URL do documento (document.URL); e
- Criar um elemento HTML pelo JavaScript (document.createElement), entre outros.

JS

O objeto **element**, por sua vez, representa os elementos html, como **<body>, , <p>** etc. Com esse objeto, podemos acessar o elemento e:

- Modificar sua classe CSS **(elemento.className)**;
- Modificar seu conteúdo **(elemento.innerHTML e document.outerHTML)**;
- Saber quem são seus elementos filhos **(document.children)**; e
- Ter acesso a seus atributos **(document.attributes)**, entre outros.

JS

Graças ao DOM, podemos ter uma padronização de comportamentos nos diferentes navegadores. **Seus métodos e atributos deram liberdade de programação para tornar as páginas ainda mais dinâmicas, usando o JavaScript**. Logo, outros frameworks criaram seus métodos para facilitar o uso do DOM, automatizando alguns processos e criando uma linguagem para usar o DOM de uma forma mais simples. Um dos primeiros frameworks a fazer isso foi o **JQuery**, em uso até hoje.

Veja na sequência uma comparação na forma de modificação de um conteúdo com o **DOM** puro, em seguida com o **JQuery**, que usa “atalhos” para executar os comandos DOM por trás:

JS

Document Object Model – DOM

Enquanto objeto, possui **métodos (funções) e propriedades (atributos)**:

Funções já conhecidas do objeto document:

- getElementById;
- write; e
- addEventListener.

Propriedades já conhecidas do objeto document:

- innerHTML;
- Value; e
- Style.

JS

Encontrando elementos

Propriedade	**Descrição**
document.getElementById(*id*)	Encontra um elemento pelo id
document.getElementsByTagName(*name*)	Encontra um elemento pelo nome da tag
document.getElementsByClassName(*name*)	Encontra um elemento pelo nome da classe

Método	**Descrição**
element.setAttribute*(attribute, value)*	Altera o valor do atributo de um element HTML

JS

innerHTML

A propriedade innerHTML define ou retorna o conteúdo HTML (HTML interno) de um elemento.

A maneira mais fácil de obter o conteúdo de um elemento é usando a propriedade innerHTML.

A propriedade innerHTML é útil para obter ou substituir o conteúdo de elementos HTML.

Exemplo com evento onclick:

```
<html>
        <body>
                <button onclick="innerHTML=Date()">Veja a hora
                </button>
        </body>
</html>
```

JS

Exemplo:

O valor de um nó de atributo é uma referência ao próprio atributo:

- Como um apontador para o atributo.

O texto dentro de um atributo pode ser acessado com a propriedade **innerHTML**

Exemplo:

<title>Testando DOM</title>

- Elemento nó é <title>
- Texto “Testando DOM” é “filho” do elemento <title>

JS

O método getElementById

A maneira mais comum de acessar um elemento HTML é usar o **id** do elemento.

No exemplo acima, o método **getElementById** usado **id="demo"** para encontrar o elemento.

id="demo“ → Nome de referência para que haja o elo de execução no momento do uso do getElementById e do referencial no HTML.

JS

getElementById

```
<html>
        <body>
                <p id="p1">Ola pessoal!</p>
                <script>
                document.getElementById("p1").innerHTML = "Nova
                insercao";
                </script>
        </body>
</html>
```

JS

Exercício:

1. Criar um novo arquivo .html;
2. Criar um parágrafo na tag <body> com a frase “ola pessoal!”;
3. Criar um botão na tag <body> com o dizer “Clique aqui para trocar o texto”;
4. Criar uma função (usar tag <script>) de nome trocacor();
5. O botão ao ser acionado deverá executar a função;

Função:

1. Colocar o parágrafo na cor vermelha;
2. Colocar a nova informação com a frase “Novo texto inserido”;

JS

Código do exercício:

```
<html>
        <body>
                <p style = "color: aqua" id="p1">Ola pessoal!</p>
                <Button onclick = "trocacor()"> Clique aqui para trocar o texto </Button>
                <script>
                        function trocacor()
                        {
                                document.getElementById("p1").style.color = "red";
                                document.getElementById("p1").innerHTML    =    "texto
                                novo inserido com sucesso";
                        }
                </script>
        </body>
</html>
```

JS

getElementByClasse

O método **getElementsByClass()** retorna uma coleção de todos os elementos no documento com o nome de classe especificado, como um objeto HTML Collection.

O objeto **Collection** representa uma coleção de nós.
Os nós podem ser acessados por números de índice. O índice começa em 0.

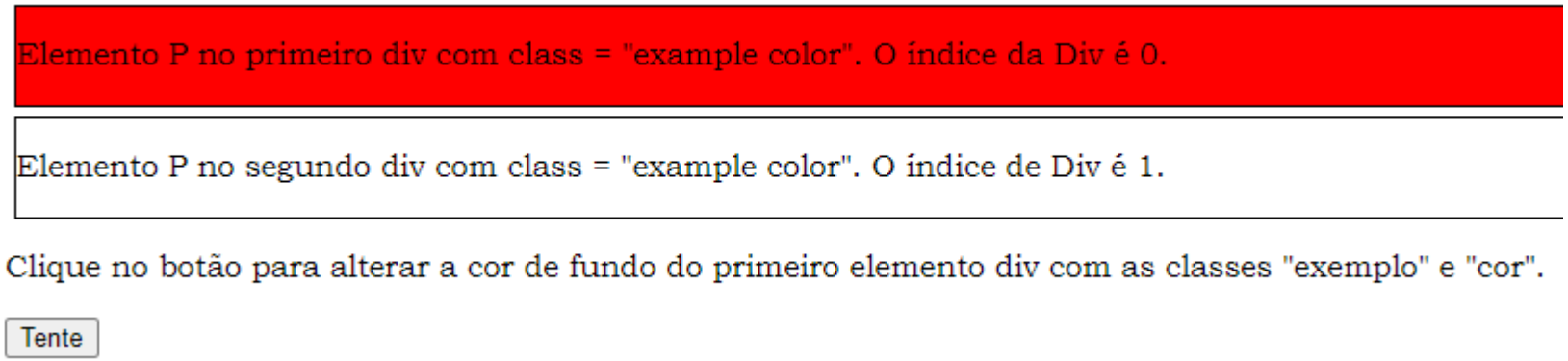
JS

Exemplo:

```
<!DOCTYPE html>
<html>
<body>
   <div class="primeiro teste">Primeiro elemento da classe="primeiro teste".</div>
   <div class="primeiro teste">Segundo elemento da classe="segundo teste".</div>
   <p>Clique no botão para o teste do primeiro elemento da classe="primeiro teste“ (índice
0)</p>
   <button onclick="myFunction()">Clique aqui!</button>
   <script>
      function myFunction() {
      var x = document.getElementsByClassName("primeiro teste");
      x[0].innerHTML = "Alterou a primeira frase";
      }
   </script>
</body>
</html>
```

JS

Exercício:



1. Criar um arquivo HTML com o nome que quiser;
2. Criar uma tag <style> na tag <head> para configurar as bordas das tags ;
    1. border: 1px solid black;
    2. margin: 5px;
3. Criar duas tags com a referência das classes;
4. Criar uma função para alterar a cor da primeira tag ;
5. Criar uma função para trocar o backgroundcolor da primeira tag ;
6. Criar um botão para executar a função;

JS

Código do exercício:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        border: 1px solid black;
        margin: 5px;
        }
    </style>
  </head>
  <body>
    <div class="example color">
    <p>P Elemento P no primeiro div com class = "example color". O índice da Div é 0.</p>
    </div>
    <div class="example color">
    <p>Elemento P no secundario div com class = "example color". O índice da Div é 1.</p>
    </div>
    <p>Clique no botão para alterar a cor de fundo do primeiro elemento div com as classes "exemplo" e "cor".</p>
    <button onclick="myFunction()">Tente</button>
    <script>
      function myFunction() {
      var x = document.getElementsByClassName("example color");
      x[0].style.backgroundColor = "red";
      }
    </script>
  </body>
</html>
```

JS

Document.write()

Nunca use **document.write()** depois que o documento for carregado. Isso substituirá o documento.

Exemplo:

Document.write(“Texto aprovado com sucesso”);

JS

Eventos

Representam toda a força motriz que permite toda a dinâmica da página.

Os eventos fazem parte do DOM e podem ser entendidos como “gatilhos” que são disparados sempre que algo acontecer, como um clique do mouse sobre um elemento, a movimentação do mouse, alguma tecla pressionada no teclado ou uma ação de abrir ou fechar uma página.

O DOM fica sempre observando tudo o que está acontecendo na página, no documento e em seus conteúdos. Assim que houver alguma interação do usuário, o DOM irá disparar o gatilho.

JS

Os eventos mais comuns são:

- **onclick**: disparado quando o usuário clica sobre o elemento.
- **onchange**: disparado quando um elemento HTML foi alterado.
- **onmouseover**: disparado quando o usuário passa o mouse “por cima” do elemento.
- **onmouseout**: disparado quando o usuário movimenta o mouse “para fora” do elemento.
- **onkeydown**: disparado quando o usuário pressiona alguma tecla do teclado.
- **onload**: disparado quando o navegador termina de carregar a página.

JS

Exemplo do evento onclick:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Usando o evento onclick</title>
        </head>
        <body>
                <h1 onclick="innerHTML='Isso acontece quando usamos o evento
                onclick!'">Clique nesse link para testar o evento onclick!</h1>
        </body>
</html>
```

JS

Exemplo do evento onmousedown e onmouseup:

```
<!DOCTYPE html>
<html>
		<head>
			<title>Eventos onmousedown e onmouseup</title>
		</head>
<body>
		<div onmousedown="mDown(this)" onmouseup="mUp(this)"
		Style="background-color:#D94A38;width:90px;height:20px;padding:40px;">Clique aqui</div>

		<script>
			function mDown(obj)
				{
				obj.style.backgroundColor="#1ec5e5";
				obj.innerHTML="Solte o clique"
				}

			function mUp(obj)
				{
				obj.style.backgroundColor="#D94A38";
				obj.innerHTML="Obrigado"
				}
		</script>

</body>
</html>
```

JS

getElementById ()

O método retorna o elemento que possui o atributo **ID** com o valor especificado. Usado para manipular ou obter informações de um elemento em seu documento.

Exemplo:

```
<html>
    <body>
        <h1 id="demo">Passe o mouse sobre mim</h1>
        <script>
            document.getElementById("demo").onmouseover = function() {mouseOver()};
            document.getElementById("demo").onmouseout = function() {mouseOut()};

            function mouseOver() {
            document.getElementById("demo").style.color = "red";
            }
            function mouseOut() {
            document.getElementById("demo").style.color = "black";
            }
        </script>
    </body>
</html>
```

JS

Jquery

O **JQUERY** é um biblioteca leve, rápida e cheia de recursos para Javascript.

Ele facilita a manipulação de eventos, animações, elementos HTML e utilização de Ajax. Basicamente, ele mudou e facilitou a escrita de códigos em Javascript.

Foi lançado oficialmente em 2006 e possui código aberto. A biblioteca também oferece a possibilidade de criação de plugins sobre ela. Através do jQuery é possível desenvolver aplicações web de alta complexilidade.

Site oficial: https://jquery.com

JS

Jquery

O **JQUERY** é um biblioteca leve, rápida e cheia de recursos para Javascript.

Ele facilita a manipulação de eventos, animações, elementos HTML e utilização de Ajax. Basicamente, ele mudou e facilitou a escrita de códigos em Javascript.

Foi lançado oficialmente em 2006 e possui código aberto. A biblioteca também oferece a possibilidade de criação de plugins sobre ela. Através do jQuery é possível desenvolver aplicações web de alta complexilidade.

Site oficial: https://jquery.com

JS

A biblioteca jQuery é um único arquivo JavaScript, e você faz referência a ele com a tag <script> (observe que a tag <script> deve estar dentro da tag <head>):

```
<head>
        <script src="jquery-3.5.1.min.js"></script>
</head>
```

Dica: coloque o arquivo baixado no mesmo diretório das páginas onde deseja usá-lo.

jQuery CDN

Se você não quiser fazer o download e hospedar o jQuery por conta própria, poderá incluí-lo de um CDN (Content Delivery Network).

O Google é um exemplo de alguém que hospeda jQuery:

Google CDN:

```
<head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

JS

Sintaxe:

É feita sob medida para **selecionar** elementos HTML e executar alguma **ação** no(s) elemento(s).

A sintaxe básica é: $(seletor). ação()

- Um sinal **$** para definir / acessar jQuery;
- Um **(seletor)** para "consultar (ou localizar)" elementos HTML; e
- Uma **ação jQuery()** a ser realizada no(s) elemento(s).

Exemplos:

$(this).hide() → oculta o elemento atual.
$("p").hide() → oculta todos os elementos <p>.
$(".teste").hide() → oculta todos os elementos com class = "teste".
$("#teste").hide() → oculta o elemento com id = "teste".

JS

O evento de documento pronto

Os métodos jQuery devem ficar dentro de um evento de documento pronto:

```
$(document).ready(function()
{
        // método jQuery ...
});
```

Isso evita que qualquer código jQuery seja executado antes que o documento termine de carregar (esteja pronto). É uma boa prática aguardar que o documento esteja totalmente carregado e pronto antes de trabalhar com ele. Isso também permite que você tenha o código JavaScript antes do corpo do documento, na tag <head>.

JS

O elemento Selector

O seletor de elemento jQuery seleciona elementos com base no nome do elemento.

Você pode selecionar todos os elementos <p> em uma página como esta:

```
$("p")
```

Exemplo:

Quando um usuário clica em um botão, todos os elementos <p> ficam ocultos:

```
$(document).ready(function()
{
        $("button").click(function()
        {
                $("p").hide();
        });
});
```

JS

```
<!DOCTYPE html>
<html>
	<head>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
		<script>
			$(document).ready(function(){
				$("button").click(function(){
					$("p").hide();
				 });
			});
		</script>
	</head>
	<body>
		<h2>Titulo da pagina</h2>
		<p>Primeiro paragrafo</p>
		<p>Segundo paragrafo</p>

		<button>Clique aqui para esconder os paragrafos</button>
	</body>
</html>
```

JS

O seletor #id

O seletor jQuery usa o **atributo id** de uma tag HTML para encontrar o elemento específico.

Um **id** deve ser único em uma página, então você deve usar o seletor **#id** quando quiser encontrar um elemento único e único.

Para encontrar um elemento com um **id** específico, escreva um caractere **hash**, seguido pelo **id** do elemento HTML:

```
$("#teste")          a palavra teste é o id, então temos #teste
```

Exemplo:

Quando um usuário clica em um botão, o elemento com **id = "teste"** fica oculto:

```
$(document).ready(function()
        {
        $("button").click(function()
        {
                $("#teste").hide();
        });
});
```

JS

```
<!DOCTYPE html>
<html>
	<head>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
		<script>
			$(document).ready(function(){
				$("button").click(function(){
					$("#teste").hide();
				});
			});
		</script>
	</head>
	<body>
		<h2>Titulo da pagina</h2>
		<p>Isto e um paragrafo</p>
		<p id="teste">Este e outro paragrafo.</p>

		<button>Clique aqui!</button>
	</body>
</html>
```

JS

O seletor .class

O **.classseletor** jQuery encontra elementos com uma classe específica.

Para encontrar elementos com uma classe específica, escreva um caractere de ponto, seguido do nome da classe:

```
$(".teste")
```

Exemplo:

Quando um usuário clica em um botão, os elementos com **class = "teste"** serão ocultados:

```
$(document).ready(function(){
        $("button").click(function(){
                $(".teste").hide();
        });
});
```

JS

```
<!DOCTYPE html>
<html>
	<head>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
		<script>
			$(document).ready(function(){
				$("button").click(function(){
					$(".teste").hide();
				});
			});
		</script>
	</head>
	<body>
		<h2 class="teste">Titulo do paragrafo</h2>
		<p class="teste">Primeiro paragrafo</p>
		<p>Segundo paragrafo</p>

		<button>Clique aqui!</button>
	</body>
</html>
```

JS

Sintaxe	Descrição
$("*")	Seleciona todos os elementos
$(this)	Seleciona o elemento html atual
$("p.intro")	Seleciona todos os elementos <p> com a classe="intro"
$("p:first")	Seleciona o primeiro elemento <p>
$("ul li:first")	Seleciona o primeiro elemento do primeiro
$("ul li:first-child")	Seleciona o primeiro elemento de todos os
$("[href]")	Seleciona todos os elemnetos com o atributo href
$("a[target='_blank']")	Seleciona todos os elementos <a> com o alvo de valor igual a "_blank"
$("a[target!='_blank']")	Seleciona todos os elementos <a> com o alvo de valor diferente de "_blank"
$(":button")	Seleciona todos os elementos <button> e elementos <input> elementos do tipo type="button"
$("tr:even")	Seleciona todos os elementos <tr>
$("tr:odd")	Seleciona todos os elementos de linha ímpar <tr>

JS

Funções em um arquivo separado

Se o seu site contém muitas páginas, e você deseja que suas funções jQuery sejam fáceis de manter, você pode colocar suas funções jQuery em um arquivo **.js separado.**

Quando demonstramos o jQuery neste tutorial, as funções são adicionadas diretamente na **<head>** seção. No entanto, às vezes é preferível colocá-los em um arquivo separado, como este (use o atributo src para se referir ao arquivo .js):

Exemplo

```
<head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery
        .min.js"></script>
        <script src="my_jquery_functions.js"></script>
</head>
```

JS

Métodos de evento jQuery

jQuery é feito sob medida para responder a eventos em uma página HTML.

O que são eventos?

Todas as diferentes ações dos visitantes às quais uma página da web pode responder são chamadas de eventos.
Um evento representa o momento preciso em que algo acontece.

Exemplos:

- mover o mouse sobre um elemento;
- selecionando um botão de rádio; e
- clicando em um elemento.

JS

Aqui estão alguns eventos DOM comuns:

Eventos do mouse	Eventos do teclado	Eventos do formulário	Eventos do Document/Window
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

JS

Sintaxe jQuery para métodos de evento

Em jQuery, a maioria dos eventos DOM tem um método jQuery equivalente.

Para atribuir um evento de clique a todos os parágrafos de uma página, você pode fazer o seguinte:

$("p").click();

A próxima etapa é definir o que deve acontecer quando o evento for disparado. Você deve passar uma função para o evento:

```
$("p").click(function(){
        // ação executada aqui!
});
```

JS

Sintaxe jQuery para métodos de evento

Em jQuery, a maioria dos eventos DOM tem um método jQuery equivalente.

Para atribuir um evento de clique a todos os parágrafos de uma página, você pode fazer o seguinte:

$("p").click();

A próxima etapa é definir o que deve acontecer quando o evento for disparado. Você deve passar uma função para o evento:

```
$("p").click(function(){
        // ação executada aqui!
});
```

JS

Métodos de evento jQuery comumente usados

$ (document) .ready ()

O método **$(document).ready()** → nos permite executar uma função quando o documento está totalmente carregado.

clique()

O método **click()** → anexa uma função de manipulador de eventos a um elemento HTML.

A função é executada quando o usuário clica no elemento HTML.

JS

```
<!DOCTYPE html>
<html>
	<head>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
		<script>
			$(document).ready(function(){
				$("p").click(function(){
					$(this).hide();
				});
			});
		</script>
		</head>
		<body>
			<p>Se clicar em mim eu desapareco</p>
			<p>Clique em mim</p>
			<p>Clique em mim também, por favor</p>
		</body>
</html>
```

JS

Definir conteúdo - text (), html () e val ()

Métodos para **definir o conteúdo** :

- text() → Define ou retorna o conteúdo de texto dos elementos selecionados;
- html() → Define ou retorna o conteúdo dos elementos selecionados (inclui marcação HTML); e
- val() → Define ou retorna o valor dos campos do formulário;

Exemplo:

```
$("#btn1").click(function(){
        $("#test1").text("Maravilha de teste!");
});

$("#btn2").click(function(){
        $("#test2").html("<b>Seja bem-vindo!</b>");
});

$("#btn3").click(function(){
        $("#test3").val("vamos jogar");
});
```

JS

```
<!DOCTYPE html>
<html>
	<head>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
		<script>
			$(document).ready(function(){
				$("#btn1").click(function(){
					alert("Texto: " + $("#teste").text());
				});
				$("#btn2").click(function(){
					alert("codigo HTML: " + $("#teste").html());
				});
			});
		</script>
	</head>
	<body>
		<p id="teste">Isto esta <b>em negrito</b> em algum texto do paragrafo</p>
		<button id="btn1">Exibir o texto</button>
		<button id="btn2">exibir o html</button>
	</body>
</html>
```