Programação para internet

Professor : Romulo Beninca e-mail: romulo.beninca@ifsc.edu.br

Horário de Atendimento : Quarta-Feira 14 as 16h

Plano de atividades da Semana

Objetivo:

Explicação do que é JavaScript e sua importância no desenvolvimento web, uma breve história do JS

Apresentação das ferramentas necessárias para programar em JavaScript (editor de texto, navegador)

1 Breve histórico do JS

Parte 1: Introdução motivação para criação

No em meados dos anos 90 a web começava a ganhar popularidade e muitos desenvolvedores e empresas observavam a falta de iteratividade da web que era construída apenas com HTML, no máximo com CSS. O HTML que era enviado ao navegador cliente era estático, com uma ou outra imagem, como imagens abaixo evidenciam.

Figure 1: Site oficial da IBM em 1996

fonte:http://web.archive.org/web/19961220214620/http://www.ibm.com/

O site da IBM evidencia que documentos HTML eram estáticos com imagens simples e pouca ou nenhuma

iteratividade, podemos fazer essa viagem no tempo graças a sites como *webarchive* que armazena páginas da web a mais de duas décadas e nos permite visualizar como eram as páginas:

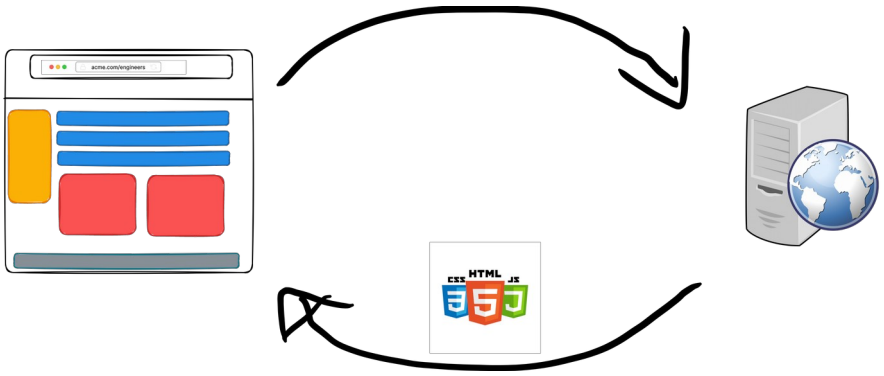
Nesse contexto muitos pesquisadores, principalmente em grandes corporações iniciaram trabalhos para adicionar mais iteratividade a web. Não havia como os usuários interagirem com essas páginas, o que tornava a experiência de navegação limitada e pouco atraente. Foi nesse contexto que surgiu o JavaScript. Criado por *Brendan Eich* em 1995.

Figure 2: Brendan Eich, criador do Java script

Embora os detalhes da história sejam extensos, para os nossos propósitos, podemos sintetizar que a empresa NetScape, para qual Brendan trabalhava o contratou em 1995, inicialmente para desenvolvimento de uma linguagem Script que pudesse rodar sobre os navegadores e intitularam o projeto de Mocha. Em pouco mais de dez dias após iniciar o projeto Bredan entregou a primeira versão do Mocha Script, depois Live Script. Mas somente em 1997 finalmente foi padronizado por solicitação conjunta da NetScape e Sun Microsystem junto a European Computer Manufacturers Association como ECMA-262.

O padrão ECMA-262 define a sintaxe, estrutura, tipos de dados e comportamento da linguagem JavaScript, bem como as regras para sua implementação em diferentes plataformas. Ele é usado por desenvolvedores, fabricantes de navegadores e outras empresas para garantir que o código JavaScript funcione de forma consistente em diferentes ambientes.

A padronização do JavaScript pelo ECMA-262 é importante porque permite que a linguagem seja utilizada de forma mais ampla e interoperável, tornando mais fácil para os desenvolvedores criarem aplicativos web sofisticados e avançados. A ideia consiste em que quando o código HTML e CSS é baixado na requisição inicial também são baixados Código em Javascript que vão ser interpretados pelo navegador do cliente.



Parte 2: Incorporação do JS no documento HTML

O código JS pode ser adicionado ao documento HTML de três diferentes formas ao HTML, inline, incorporado e externo.

- **Inline**

```
<body>
    <div class="main"></div>
    <h5>Meet the</h5>
    <h1 id="titulo">Engineers</h1>
    <ul>
        <li>A. Lovelace</li>
        <li>G. Hopper</li>
        <li>M. Hamilton</li>
    </ul>
    <button type="button" onclick="alert('Olá')">Click</button>
</body>
```

No exemplo é declarado no código HTML que ao detectar o evento "onClick" deve-se executar o código JS entre aspas, no caso exibe uma janela de aletar

Obs: Execute em um navegado externo ao vscode.

- **Incorporado na tag <script>**

```
<body>
    <div class="main"></div>
    <h5>Meet the</h5>
    <h1 id="titulo">Engineers</h1>
    <ul>
        <li>A. Lovelace</li>
        <li>G. Hopper</li>
        <li>M. Hamilton</li>
    </ul>
    <button type="button" onclick="aletaOlaMundo()">Click</button>
</body>
<script >
   function aletaOlaMundo(){
       alert('Olá Mundo')
   }
</script>
```

A tag script nas ultimas linhas contém o código JS com a declaração de uma função chamada de aletaOlaMundo(), que é chamada na ocorrência do evento onclick.

Você pode ser levado a imaginar que para associar um evento a uma função JS sempre será utilizado js inline, mas podemos fazer isso adicionando *Listeners* (Ouvintes) para os eventos do navegador, mas neste momento demonstrarei com JS inline para simplificar.

- **Externo JS ao documento HTML.**

```html
<body>
    <div class="main"></div>
    <h5>Meet the</h5>
    <h1 id="titulo">Engineers</h1>
    <ul>
        <li>A. Lovelace</li>
        <li>G. Hopper</li>
        <li>M. Hamilton</li>
    </ul>
    <button type="button" onclick="aletaOlaMundo()">Click</button>
    <script src="1.js"> </script>
</body>
```

```
alertaOlaMundo
function alertaOlaMundo() {
    alert('Olá mundo!');
}
```

Para incluir esse arquivo JavaScript em seu documento HTML, você deve utilizar a tag <script> com o atributo src apontando para o arquivo 1.js

Neste exemplo incluído no documento HTML, e a função

É observar que, ao utilizar JS externo, o navegador fará uma requisição adicional para baixar o arquivo JavaScript. Portanto, é uma boa prática colocar <script> logo depois do fechamento da tag </body>, para que a renderização do conteúdo da página não seja bloqueada enquanto o arquivo JavaScript é carregado. Mais adiante abordaremos a sintaxe do JS.

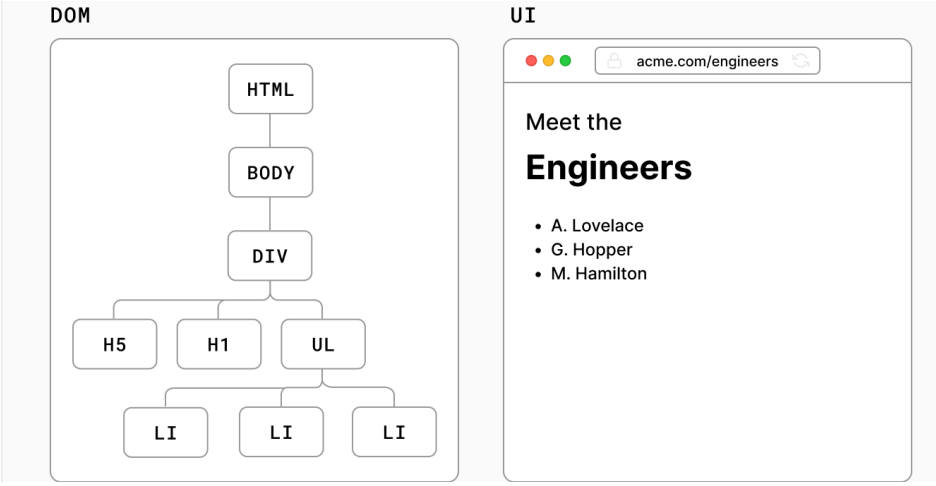
Sobre o nome , JavaScript foi definido com o objetivo de capitalizar o sucesso da linguagem Java, que estava em alta na época, mas cabe destacar que JavaScript não é Java, enquanto Java é uma linguagem compilada, JavaScipt (ECMA-262) é interpretado, a coincidência não passa de estratégia de marketing acertada para o momento.

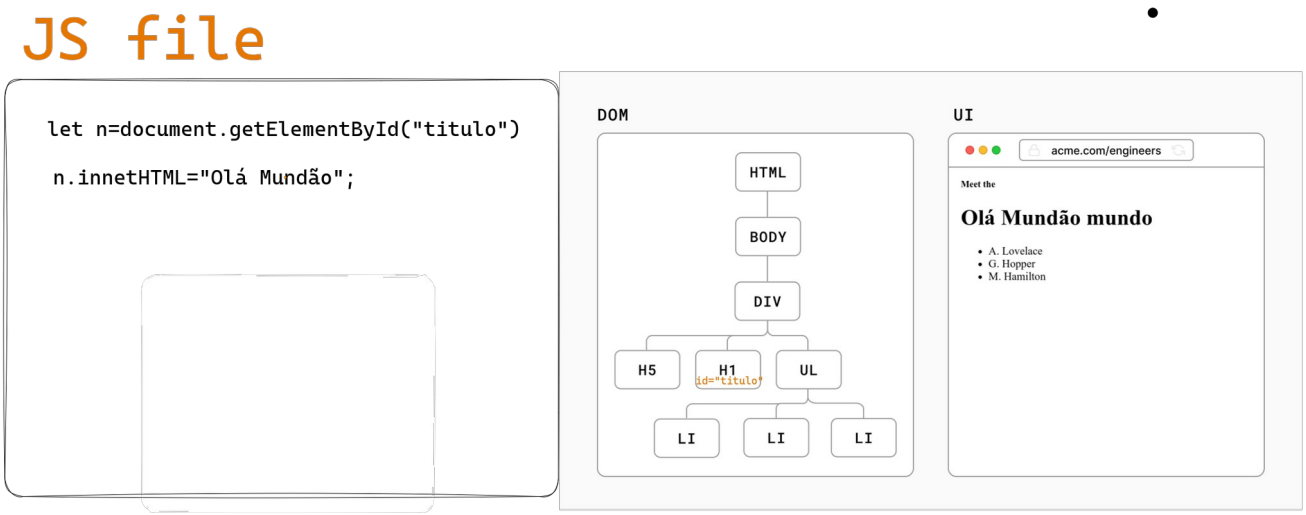
Parte 3: JS não é java

O JavaScript, assim como o Java, é uma linguagem de programação orientada a objetos. No entanto, ela possui algumas diferenças significativas em relação ao Java. Uma das principais diferenças é que o JavaScript é interpretado, enquanto o Java é compilado.

Outra característica marcante do JavaScript é sua capacidade de interagir com o HTML e o CSS. Isso significa que é possível manipular elementos da página, como botões e formulários, a partir de código JavaScript. Para que isso aconteça no ambiente de execução do JS no navegador tempos uma abstração do HTML que esta na tela do navegador e CSS por meio do que denominamos de Arvore DOM.

No código JS podemos temos uma variável especial denominada document que nos da acesso a diversas funções que permitem alterar a estrutura e valores do HTML e CSS, como exemplo abaixo.





Ao longo dos anos, o JavaScript se tornou uma das linguagens de programação mais populares do mundo. Isso se deve, em grande parte, à sua capacidade de criar conteúdo dinâmico e interativo na web, que é executado do lado do cliente. Atualmente, ele é utilizado em uma infinidade de aplicações, desde jogos até aplicações empresariais.

O interpretador JS é executado sobre o navegador, atualmente existem diversos interpretadores alguns deles:

- Google V8: o interpretador de JavaScript de alto desempenho usado pelo Google Chrome e Node.js.
- SpiderMonkey: o interpretador de JavaScript de código aberto usado no navegador Firefox.
- JavaScriptCore: o interpretador de JavaScript de código aberto usado no navegador Safari da Apple e em vários aplicativos da Apple.
- Rhino: um interpretador de JavaScript de código aberto escrito em Java, mantido pela Fundação Apache.
- Nashorn: um interpretador de JavaScript de código aberto escrito em Java, incluído no JDK 8 e posterior.
- Chakra: o interpretador de JavaScript usado no navegador Microsoft Edge.
- Node.js: um ambiente de tempo de execução JavaScript de código aberto que inclui o interpretador V8.

Para acessar um terminal do interpretador basta abrir a ferramenta de desenvolvedor do navegador e digitar o código JS, por se tratar de uma linguagem interpretada ele irá executar diretamente e exibir o resultado.

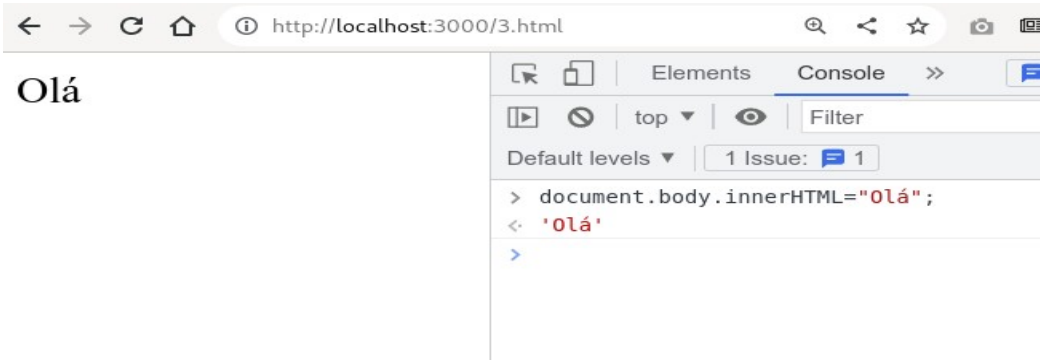


Figure 3: Acesso ao interpretador JS no navegador, aba console da ferramenta de desenvolvimento

Na aba console do navegador você pode incluir comandos como listado abaixo e serão executados diretamente sobre a página.

- Exibir uma mensagem de alerta

```
> alert("Olá, mundo!");
```

- Imprimir uma mensagem no Console:

```
> console.log("Olá, mundo!");
  Olá, mundo!
```

- Alterar o texto de um elemento HTML:

```
> document.querySelector("h1").textContent = "Novo Título";
```

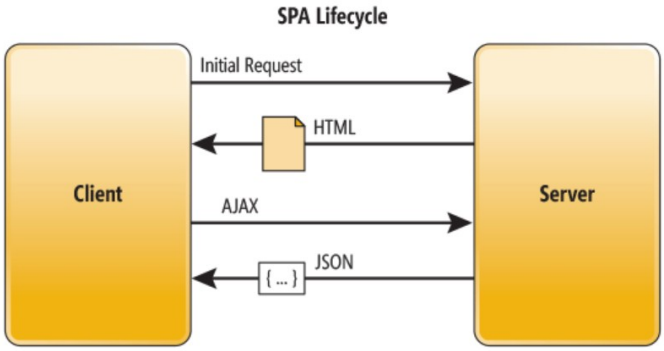
- Calcular uma operação matemática:

```
> var x = 5;
  var y = 10;
  console.log(x + y);
  15
< undefined
```

Antes do Java script, cada iteração do usuário com navegador necessitava de um envio de requisição ao servidor, que processava e recarregava toda a página, ou encaminhava o usuário a outra página, reexecutando dezenas de requisições para download dos html, css, imagens e tudo mais que fosse necessário. Esse modelo é o que denominamos de MPA (Multi Page Application).

Parte 4 – MPS e SPA

Com o surgimento de uma linguagem de programação de uso geral sobre o navegador, foi possível desenvolver aplicações que rodam em parte no cliente, de forma que somete quando ha necessidade de algum dado requisita apenas uma pequena porção de dados ao servidor (AJAX).



Antes das SPAs, as aplicações web eram predominantemente *Multi Page Applications* (MPAs), em que cada interação do usuário resultava em uma solicitação ao servidor e no carregamento de uma nova página. Isso

levava a tempos de carregamento mais longos e a uma experiência de usuário menos fluida, pois as páginas eram constantemente recarregadas. As SPAs (*Single Page Aplication*) começaram a ganhar popularidade por volta de 2002, com a introdução do conceito de AJAX (*Asynchronous JavaScript and XML*). Em 2004, a Google lançou o Gmail, que foi um dos primeiros exemplos bem-sucedidos de uma SPA. O Gmail mostrou como uma aplicação web poderia ter uma interface rica e interativa, semelhante a um aplicativo de *desktop*, melhorando a experiência do usuário. A partir dai muitas outras empresas criaram bibliotecas e recursos como Angular, React e Vue.js, Jquery bibliotecas que utilizam Ajax e possibilitam criação de aplicações rica ou RIA Rich User Interface (RUI) .

Além do relatado até aqui JavaScript também evoluiu com suporte a ser executado no *backend* em 2009, quando Ryan Dhal, e sua equipe modificou o interpretador V8 do crome para executar no servidor e criou o nodeJS. Assim pode-se utilizar código JS no servidor para responder ao cliente.

Em suma, o JavaScript é uma linguagem de programação orientada a objetos que permite acrescentar interatividade às páginas web. Ele foi criado em 1995 por Brendan Eich e, ao longo dos anos, se tornou uma das linguagens mais populares do mundo, graças à sua capacidade de criar conteúdo dinâmico e interativo na web.

Parte 5: Evolução do JS

A linguagem JavaScript evoluiu significativamente desde sua criação em 1995 por Brendan Eich. Exploraremos marcos importantes no desenvolvimento do JavaScript e discutiremos como a linguagem evoluiu ao longo dos anos, fornecendo exemplos para ilustrar as mudanças.

1995: Introdução do JavaScript

Brendan Eich, trabalhando na Netscape, criou o JavaScript (originalmente chamado de Mocha e mais tarde LiveScript) como uma linguagem de script para ser executada em navegadores da web. O objetivo era permitir interações dinâmicas e ricas em páginas da Web, bem como dar suporte à validação de formulários e outras interações básicas do lado do cliente.

- **Exemplo: Validação de formulário simples em JavaScript nativo:**

2.html > html > body > script

```html
<!DOCTYPE html>
<html >
<head>
  <title>Validação de Formulário Simples</title>
</head>
<body>
  <h1>Validação de Formulário Simples</h1>
  <form name="meuFormulario" onsubmit="return validarFormulario()" >
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome">
    <br><br>
    <input type="submit" value="Enviar">
  </form>
  <script src="2.js"></script>
</body>
</html>
```

2.js > validarFormulario

```javascript
function validarFormulario() {
    var nome = document.forms["meuFormulario"]["nome"].value;
    if (nome == "") {
      alert("Por favor, preencha o campo nome.");
      return false;
    }
  }
```

Neste exemplo, temos um arquivo HTML com um formulário, eu js externo.

O formulário tem um único campo de entrada de texto chamado "nome". A função **validarFormulario**() do JS é chamada quando o formulário é enviado, usando o atributo **onsubmit** no elemento <**form**>.

Se o campo "nome" estiver vazio, a função exibirá um alerta solicitando ao usuário

que preencha o campo e impedirá que o formulário seja enviado, retornando false. Se o campo "nome" estiver preenchido, o formulário será enviado normalmente.

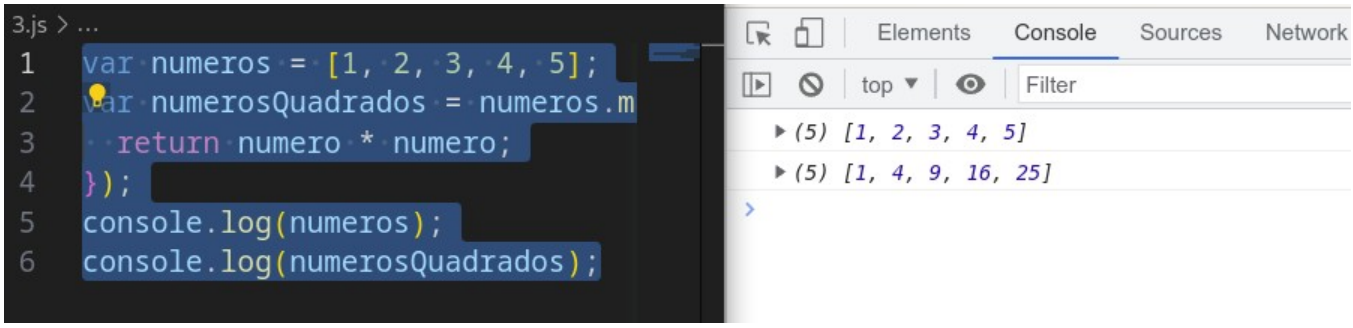
1997: ECMAScript 1 (ES1):

Padronizado pela primeira vez em 1997 pela ECMA International, dando origem ao termo ECMAScript. A primeira versão da especificação ECMAScript, conhecida como ES1, estabeleceu a base da linguagem JavaScript que conhecemos hoje.

1999: ECMAScript 3 (ES3):

O ES3 introduziu várias melhorias na linguagem, como suporte a expressões regulares, manipulação de erros com try/catch e mais funções para manipulação de strings e arrays.

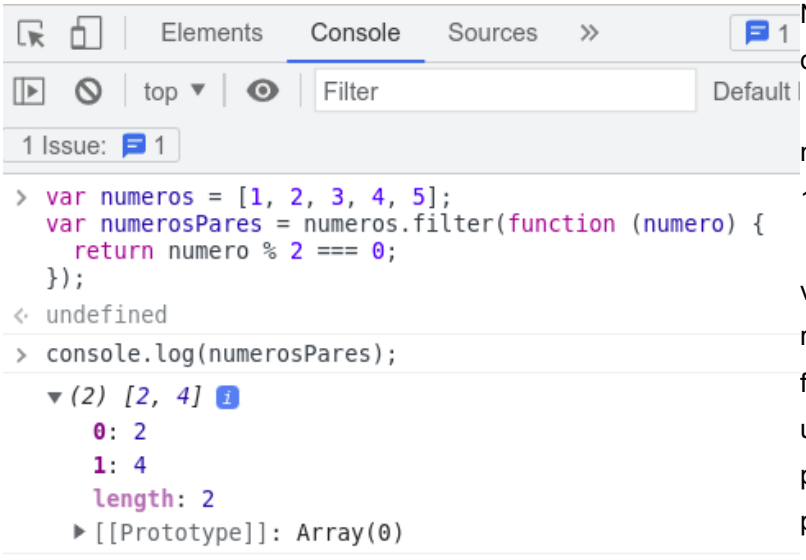
Exemplo: Manipulação de array no ES3:



2009: ECMAScript 5 (ES5)

O ES5 trouxe diversas melhorias importantes, como suporte a getters e setters, a criação de objetos com propriedades não enumeráveis, a introdução do modo estrito ("strict mode") e melhorias nas funções de manipulação de arrays, como forEach, map, filter e reduce.

Exemplo: Uso do método filter no ES5:



No exemplo o código JS é executado diretamente no console e contem,

Primeiro, cria-se um array chamado numeros, que contém cinco números inteiros: 1, 2, 3, 4 e 5.

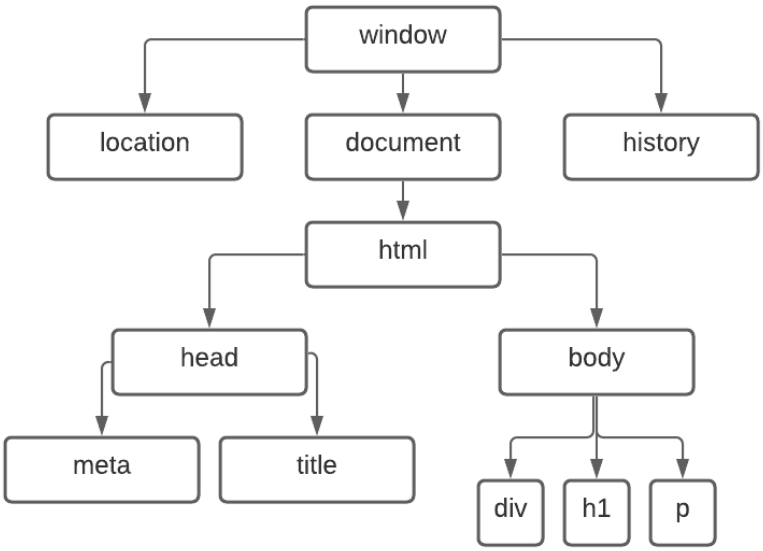
Na próxima linha é declarado uma variável “numerosPares” que recebera o retorno da execução de numeros.filter(..), ela faz parte dos arrays em js e nela é declarada um função, que será chamada para cada posição do array, passado como parâmetro o próprio conteúdo da posição. Se a função de callback retornar true, o elemento é incluído

no novo array que está sendo criado. Se retornar false, o elemento é ignorado. Assim temos o array de resultado [2, 4] que é atribuído a **numeroPares** e impresso no console.

2 O que é a Window, arvore DOM, location, alert...

Já sabemos que o JS é uma linguagem de uso geral que executa sobre o navegador por meio de interpretadores , agora vamos compreender que estrutura é essa DOM Document Object Model (DOM). No ambiente do interpretado JS no navegador existe um objeto global **'window',** que representa a janela do navegador, dentro desse objeto podemos acessar várias propriedades entre elas a arvore de componentes DOM.
O objeto window possui várias propriedades e métodos úteis para o desenvolvimento web, como por exemplo:

- ***window.location:*** contém informações sobre a localização atual do documento, como a URL, o host, a porta e assim por diante.
- ***window.document:*** representa o documento atual na janela do navegador.
- ***window.alert():*** exibe uma caixa de diálogo com uma mensagem para o usuário.
- ***window.setTimeout():*** agenda a execução de uma função para ser executada após um determinado intervalo de tempo.
- ***window.addEventListener():*** registra um ouvinte de eventos para um determinado elemento da página.



document é um objeto que representa o DOM (Document Object Model) da página. O DOM é uma representação hierárquica em forma de árvore de um documento HTML ou XML, que permite que o código JavaScript interaja com os elementos da página, manipulando-os e alterando sua aparência e comportamento. Os atributos dos elementos são representados por nós de atributos que são filhos dos nós de elementos correspondentes.

No console podemos visualizar o conteúdo da window, ou window.document ou window.location facielmente, pois ao escrever esses objetos o console exibe a estrutura dos mesmos. Ao digitarmos window no console podemos visualizar diversas objetos e suas pripriedades.

- >*window.alert(“”)* exibe um alerta na janela do navegador.
- >*window.document* é um objeto com dezenas de filhos que representa todo HTML Arvore DOM.
- >*window.history* é um objeto com array de histórico de navegação da janela.
- >*windows.location* é um objeto com dezenas de propriedades sobre a localização da atual janela do navegador, como protocolo, endereço porta.
- >*window.console* é um objeto que representa o console JS, inclusive a sua sub função log() imprime no terminal e clear() imprime no terminal.

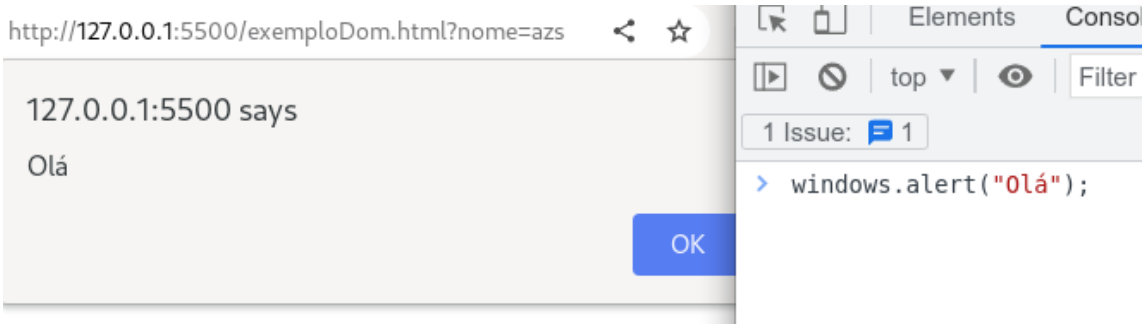


Figure 4: Exemplo de uso do console para exibir uma alerta pela janela do navegador.

Nos nossos códigos JS na maioria das vezes acessaremos elementos da Arvore Dom e modificaremos estes para interagir com o usuário. Já o console utilizaremos para debugar nossos códigos. Por fim é importante conhecer o ambiente de execução do JS, mas logo começaremos a compreender como acessar os elementos da DOM.

2.1 Selecionando elementos do dom

Para altear os elementos da arvore dom com JS precisamos acessá-los embora possamos colocar os caminhos completos como window.document.body.titulo isso seria pouquíssimo eficiente e prático, oque fazemos é buscar os elementos com base no id, tipo, seletor

`document.getElementById()`: seleciona um elemento da página usando o valor do atributo "id" do elemento.

```
var elemento = document.getElementById("exemplo");
```

`document.getElementsByClassName()`: seleciona um conjunto de elementos da página que possuem a mesma classe.

```
var elementos = document.getElementsByClassName("exemplo");
```

`document.getElementsByTagName()`: seleciona um conjunto de elementos da página que possuem a mesma tag.

```
var elementos = document.getElementsByTagName("p");
```

`document.querySelector()`: seleciona o primeiro elemento da página que corresponde a um seletor CSS específico.

```
var elemento = document.querySelector("#exemplo .filho");
```

`document.querySelectorAll()`: seleciona um conjunto de elementos da página que correspondem a um seletor CSS específico.

```
var elementos = document.querySelectorAll("p.exemplo");
```

3 Variáveis no JS

A declaração de variáveis no JS consuma causar um pouco de confusão, existem três diferentes formas de declarara-as, mas preste atenção que é bem simples.

vars.js > ...

```
1  // Declarando uma variável com var
2  var nome = "João";
3
4  // Declarando uma variável com let
5  let idade = 30;
6
7  // Declarando uma constante
8  const pi = 3.14159;
9
10 // Imprimindo o valor de uma variável
11 console.log(nome + " tem " + idade + " anos.");
12
```

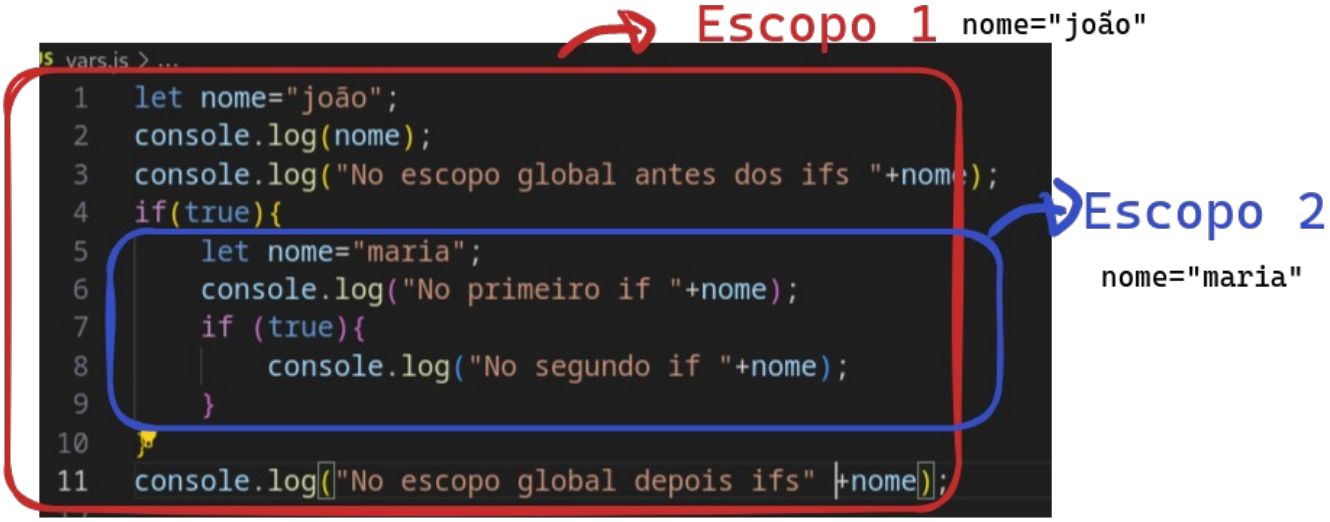
Qual a diferença das de se declarar uma variável com var, let e const.

var: A palavra-chave var foi a maneira original de declarar variáveis em JavaScript. As variáveis declaradas com var são "hoisted" (içadas), o que significa que elas são movidas para o topo do escopo e podem ser acessadas antes de sua declaração.

let: A palavra-chave let, é uma forma mais moderna de declarar variáveis no JS evitando conflitos e colisão de nomes. Declara-se uma variável por bloco.

const: A palavra-chave const, também introduzida no ECMAScript 6 (ES6), é usada para declarar constantes

Uma variável var é global e igual em todo local em que for acessada, entretanto variáveis definida com let pertence e é acessível pelo escopo atual e descendentes, como no exemplo abaixo.



Resultado:

```
joão
No escopo global antes dos ifsjoão
No primeiro ifmaria
No segundo ifmaria
No escopo global depois ifsjoão
```

Sugiro a leitura e exemplos de :

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Grammar_and_types

4 Tipos de variáveis no JS

No java script os tipos das variáveis são definidos dinamicamente, no momento em que atribuímos algo ela o interpretador analisa o conteúdo e define o tipo da variável como um dos tipos abaixo:

Strings: Uma String

Numbers: São os números, seja eles integer, float, double etc.

Booleans: São os operadores booleanos (true ou false)

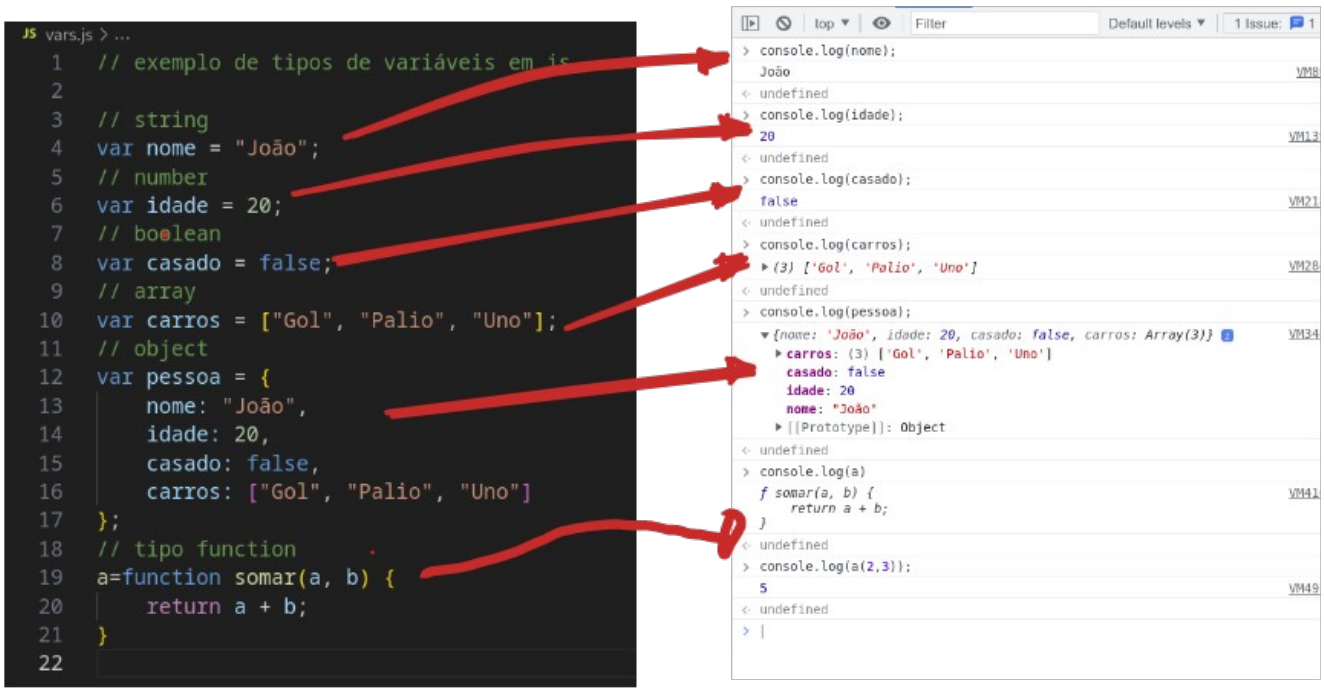
Arrays: É uma estrutura de dado para armazenar uma coleção de valores, sendo eles de qualquer tipo, um após o outro em uma estrutura de vetor.

Functions: Em JavaScript é possível declarar uma variável como uma função, podendo fazer operações e retornando o valor para a variável de declaração. Obs: muito utilizado no paradigma

Objects: Um objeto é uma ocorrência de um tipo personalizado para o programa ou contexto, é nada mais do que um conjunto de atributos aninhados a uma variável denomina-se um objeto.

de programação funcional.

No exemplo a seguir um código com diferentes tipos de variáveis é impresso no log.



5 Estrutura de seleção e repetição JS

A sintaxe básica do JS é muito similar do java e não deve haver dificuldade, a seguir são explicados alguns exemplos

```
//exemplo if com js
if (idade >= 18) {
    console.log("Maior de idade");
}

//exemplo if else com js
if (idade >= 18) {
    console.log("Maior de idade");
}
else {
    console.log("Menor de idade");
}
```

IF e IF and Else

No exemplo de uso do if temos a comparação de se uma variável previamente definida é igual ou maior a 18.

```
//exemplo com while javascript
var i = 0;
while (i < 10) {
    console.log(i);
    i++;
}
```

While

Inicialmente uma variável é declara com valor 0, em seguida inicia-se o laço de repetição “while” que permanece em execução até que i>=10;

```
//exemplo com for javascript
for (var i = 0; i < 10; i++) {
    console.log(i);
}
```

For O laço for é igual em java, c e c++ em que uma variável i é iterada até que alcance uma condição.

```
//exemplo com switch javascript
var dia = 1;
switch (dia) {
    case 1:
        console.log("Domingo");
        break;
    case 2:
        console.log("Segunda");
        break;
    case 3:
        console.log("Terça");

        break;
    case 4:
        console.log("Quarta");
        break;
    case 5:
        console.log("Quinta");
        break;
    case 6:
        console.log("Sexta");
        break;
    case 7:
        console.log("Sábado");
        break;
    default:
        console.log("Dia inválido");
        break;
}
```

Switch case: Dada uma variável inicial ‘dia’ será executado o bloco case que com cide com valor de ‘dia’. Caso não haja é executado valor *default*.

```
//exemplo com for of javascript
var numeros = [1, 2, 3, 4, 5];
for (var numero of numeros) {
    console.log(numero);
}
```

For each ou for a cada é uma estrutura que permite percorrer um array de forma facilitada, a cada iteração a variável numero assume um dos valores do *array*. Assim, evitando definição de variáveis de índice auxiliares

Eventos do navegador e JS

Eventos são ações ou ocorrências resultado da interação do usuário, como cliques, movimentos do mouse, pressionamento de teclas, ou também eventos automáticos, como o carregamento de uma página. No JavaScript, os eventos são usados para executar código quando algo específico acontece, como por exemplo no código abaixo, quando usuário clicar no botão é executada uma função que lê o campo texto input id nome exibe um alert.

exemploHelloWorldDialog.html > html > body > button#meuBotao

```html
<!DOCTYPE html>
<html >
<head>
  <title>Exemplo de Evento</title>
</head>
<body>
  <input type="text" id="nome" placeholder="Digite seu nome">
  <button id="meuBotao">Exibir Alerta</button>
</body>
<script src="scriptHello.js"></script>
</html>
```

scriptHello.js > ...

```js
// Seleciona o elemento com o ID "meuBotao"
const botao = document.getElementById("meuBotao");
// Cria a função "listener" do evento
function exibirAlerta() {
  // Seleciona o elemento com o ID "nome"
  const campoNome = document.getElementById("nome");
  // Obtém o valor do campo de texto
  const nome = campoNome.value;
  // Exibe um alerta com o nome inserido
  alert("Nome: " + nome);
}
// Adiciona a função "listener" ao evento "click" do botão
botao.addEventListener("click", exibirAlerta);
```

Observe que podemos associar um função a um evento por meio da chamada da função addEventListener(evento, função) ou pela declaração do atributo do evento *inline* como abaixo:

```html
<button onclick="exibeAlerta();" id="meuBotao">Exibir Alerta</button>
```

Conclusão da aula:

Nesta aula remota estudamos diversos conteúdos base para JS, vamos relembra-los.

Introdução e história do JavaScript (JS): Entendemos a fonte e a evolução do JavaScript como uma das principais linguagens de programação do desenvolvimento da Web e entendemos sua versão principal e a importância da linguagem na situação atual.

JavaScript e DOM: discutimos como o navegador interpreta e executa o código JavaScript, bem como a estrutura das árvores dom, bem como a interação dos elementos JavaScript e HTML.

Variáveis e tipos de dados: resolvemos a declaração e a alocação de variáv e os principais tipos de dados JavaScript, como string, números e valores booleanos. Também expleisoramos exemplos de operações variáveis e tipos de dados.

Sintaxe básica, Estudamos as principais estruturas condicionais em JavaScript (if , if eles) e estruturas de repetição como (for, do-while).

Evento do navegador: Finalmente, aprendemos como JavaScript é relacionado com eventos que ocorrem no navegador, podendo assim responder ações especificas do usuário.

Atividades Propostas: