# Tema 01 – Introdução ao JavaScript

## Introdução

Olá meus queridxs alunxs, sejam bem vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você ver uma introdução sobre JavaScript e fazer os primeiros códigos utilizando essa linguagem.

## JavaScript

JavaScript (ou apenas JS) é uma linguagem de programação, que está de acordo com a especificação ECMAScript. O JS é uma linguagem de alto nível interpretada com tipagem dinâmica fraca e mutiparadigma (protótipos, orientado a objeto, imperativo e, funcional).

O JS, além do HTML e do CSS, é uma das tecnologias bases da World Wild Web. E, atualmente é uma das linguagens mais utilizadas do lado do cliente, mas também pode ser utilizada do lado do servidor por meio de ambiente como o NodeJS.

### Lado do cliente e lado do servidor

Programação baseada na internet possui dois principais lados: programação do lado do servidor e programação do lado do cliente. Na programação do lado do servidor, o código executa em um servidor web. E nesse caso, os navegadores comunicam-se com web servers utilizando requisições HTTP (*HyperText Transfer Protocol*). Por exemplo, quando você clica em um link em uma página da web, seja para enviar um formulário ou para fazer uma pesquisa, uma HTTP *request* (solicitação HTTP) é enviada do seu navegador para o servidor de destino. Na programação do lado do cliente, os programas são executados no computador do usuário utilizando scripts, que são carregados juntos com os arquivos HTML e CSS (Figura 1).

Diagram

Description automatically generated

Figura - Lado do servidor e lado do cliente

### Características

O JS é uma linguagem imperativa e estruturada, que suporta os elementos de sintaxe de programação estruturada da linguagem C como, por exemplo, if, while, switch, etc.

O JS também possui tipagem dinâmica e baseada em objetos. Objetos JavaScript são arrays associativos, potencializados com um protótipo e cada chave fornece o nome para uma propriedade de objeto. Propriedades e seus valores podem ser adicionadas, mudadas, ou deletadas em tempo de execução.

### Vanilla JavaScript

Vanilla JavaScript ou Vanilla JS se referem ao JavaScript desenvolvido puramente, sem o suporte de qualquer estrutura ou biblioteca adicional. *Scripts* escritos em Vanilla JS são códigos JavaScript simples. O que vamos aprender primeiramente aqui é programar códigos em Vanilla JS e depois aprenderemos como utilizar uma biblioteca/framework para auxiliar no desenvolvimento das aplicações web.

### JavaScript e HTML DOM

O HTML DOM (Document Object Model) permite o JavaScript acessar e modificar todos os elementos HTML em um documento (página web). Quando uma página web é carregado, o navegador cria o DOM da página com a estrutura de elementos e a árvore de objetos dessa página (Figura 2). O JS pode adicionar, alterar ou remover os elementos, os atributos e estilos CSS da página, bem criar ou reagir a eventos HTML da página.



Figura - Exemplo de uma árvore de objetos do DOM.

### Extensões do VS Code

Seguem algumas sugestões de extensão para você ter no seu VS Code, que podem te ajudar no desenvolvimento de códigos em JavaScript.

A extensão **ES Lint** analisa estaticamente seu código para encontrar problemas rapidamente e muitos problemas encontrados pelo ESLint podem ser corrigidos automaticamente.

Graphical user interface, text

Description automatically generated

A extensão **JavaScript (ES6) code snippets** contém snippets para a sintaxe ES6 do JavaScript. Por exemplo, o snippet **imp** importa módulos inteiros (ex.: **import fs from 'fs'**);

Graphical user interface, text, website

Description automatically generated

A extensão **Bracket Pair Colorizer 2** colore pares de parênteses, colchetes ou chaves com a mesma cor, facilitando a visualização de blocos de comando.

Graphical user interface, text, website

Description automatically generated

A extensão **Prettier - Code formatter** formata e organiza o seu código, que impõe um estilo consistente ao analisar seu código e imprimi-lo novamente com suas próprias regras que levam em consideração o comprimento máximo da linha, agrupando o código quando necessário.

Graphical user interface, text

Description automatically generated

Ao instalar as extensões **ESLint** e **Prettier - Code formatter**, você precisa configurá-las no seu VS Code para que a extensão possa realizar as organizações automaticamente. Vamos aos passos para fazer as configurações:

1. Você deve abrir o arquivo **settings.json** da sua IDE VS Code. Acesse o Menu Arquivo → Preferências → Configurações ou utilize o atalho do teclado **Ctrl+,**

Graphical user interface

Description automatically generated

1. Na aba configurações que será aberta, role até a opção **Code Actions On Save** e clique em Editar em **settings.json**.

Text

Description automatically generated

Você pode também buscar por **Code Actions On Save** na barra de **Configurações de pesquisa**.

A screenshot of a computer

Description automatically generated

1. O arquivo settings.json irá ser aberto. Insira o trecho de configuração das extensões Prettier – Code formatter e ESLint.

|  |
| --- |
| "editor.defaultFormatter": "esbenp.prettier-vscode",  "[javascript]": {  "editor.defaultFormatter": "esbenp.prettier-vscode",  "editor.formatOnSave": true  },  "editor.codeActionsOnSave": {  "source.fixAll.eslint": true  },  "eslint.validate": ["javascript", "typescript"],  "prettier.tabWidth": 4,  "prettier.singleQuote": true,  "prettier.bracketSpacing": true, |

1. Agora o seu código em JavaScript será formatado e verificado sempre que você o salvar.

## Gramática do JavaScript

### Comentários no JavaScript

Comentários são anotações inseridas no código fonte com o objetivo de descrever alguma lógica, instrução ou um lembrete TO-DO. Por exemplo, você pode usar comentários para: lembrar algo importante quando o código foi desenvolvido, criar secções para organização e ainda para criar um cabeçalho do código. Comentários são ignorados pelo compilador na verificação da sintaxe do código. Os comentários mais comuns em JavaScript são:

* Comentário de linha, que é iniciado por //

// Texto do comentário

* Comentário de bloco, que é iniciado por /\* e finalizado por \*/

|  |
| --- |
| /\*  Esse é um comentário tradicional. Ele  pode ser dividido em várias linhas  \*/ |

Observação: Comentários em uma linguagem de programação é muito importante e deve ser usado com consciência. Quando estamos ensinando uma nova linguagem de programação inserimos muitos comentários para que o aluno possa entender o comando ou instrução que está executada, mas no ambiente profissional, você deve primeiro ver a política de documentação da empresa e códigos exemplos do sistema que você irá trabalhar.

### Espaços em brancos

O espaço em branco geralmente é insignificante, mas ocasionalmente é necessário usar o espaço em branco para separar sequências de caracteres que, de outra forma, seriam combinadas em um único token. Por exemplo:

|  |
| --- |
| let num = 3; |

O espaço em branco entre a palavra reservada **let** e o nome da variável **num** é necessário e não deve ser removido, mas ou outros espaços em branco são opcionais.

### Nomes

Os nomes de variáveis ou funções em JavaScript podem conter letras, dígitos ou underline e não pode coincidir com uma das palavras reservadas da linguagem.

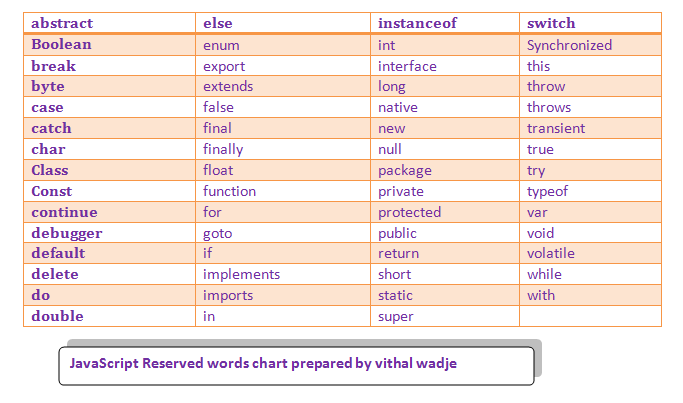


Figura - Palavras reservadas em JavaScript.

### Números

O JavaScript tem um único tipo de número. Internamente, é representado como ponto flutuante de 64 bits, o mesmo que o double do Java. Portando, em JS, não há diferença entre 1 e 1.0, esses números são interpretados como mesmo valor. Esta é uma conveniência significativa, porque problemas de estouro em inteiros curtos são completamente evitados e tudo que você precisa saber sobre um número é que ele é um número. Uma grande classe de erros de tipo numérico é evitada.

### Strings

Strings são sequência de caracteres, que no JavaScript, devem ser envolvidas utilizando aspas simples ou aspas duplas, mesmo se elas contêm zero ou mais caracteres. Todos os exemplos mostrados abaixo são válidos para serem usados na linguagem JavaScript.

|  |  |  |
| --- | --- | --- |
| let nome = "Irmão do Jorel";  let dados = "";  const frutas = ["maçã", "manga", "pêra"]; |  | let nome = 'Irmão do Jorel';  let dados = '';  const frutas = ['maçã', 'manga', 'pêra']; |

Se você fez a configuração da extensão Preittier como mostrada anteriormente, a configuração **"prettier.singleQuote": true** irá padronizar o uso de aspas simples em toda string no seu código JavaScript.

### Instruções

Uma unidade de compilação contém um conjunto de instruções executáveis. Em navegadores web, cada marcação **<script>** fornece uma unidade de compilação que é compilada e executada imediatamente. Desse modo o JS possui uma sintaxe específica para interpretar corretamente as suas instruções. Você deve ficar atento a essas sintaxes para não inserir códigos errados no seu script. Abaixo seguem algumas sintaxes usada pelo JavaScript:

|  |  |
| --- | --- |
| **Instrução var** | **Instrução do comando if** |
| Diagram  Description automatically generated | Diagram  Description automatically generated |
| **Instrução do comando switch-case** | **Instrução do comando for** |
|  |  |
| **Instrução do comando while** | **Instrução do comando do-while** |
|  |  |
| **Instrução do comando try-catch** | **Instrução do comando return** |
|  |  |

### Expressões

As expressões mais simples são um valor literal (como uma string ou número), uma variável, um valor embutido (verdadeiro, falso, nulo, indefinido, NaN ou Infinito), uma expressão de invocação/chamada precedida pela palavra reservada **new**, uma expressão de refinamento precedida pela palavra reservada **delete**, uma expressão entre parênteses, uma expressão precedida por um operador de prefixo ou uma expressão seguida por:

* Um operador infixo e outra expressão
* O operador ternário ? seguido por outra expressão, então por **:** , e então por outra expressão
* Um refinamento

Todas as expressões mostradas abaixo são de expressões na linguagem JavaScript:

|  |  |  |
| --- | --- | --- |
| 2 + 3 \* 5 === 17  (2 + 3) \* 5 === 25  const numbers = new Array(1, 2, 3, 4, 5);  const s1 = 'technology, computer, it, code';  const color = h > 10 ? 'red' : 'blue'; |  | const person = {  firstName: 'John',  lastName: 'Doe',  age: 30,  hobbies: ['music', 'movies', 'sports'],  address: {  street: '50 main st',  city: 'Boston',  state: 'MA'  }  } |

### Literais

Literais de objeto são uma notação conveniente para especificar novos objetos. Os nomes das propriedades podem ser especificados como nomes ou strings. Os nomes são tratados como nomes literais, não como nomes de variáveis, portanto, os nomes das propriedades do objeto devem ser conhecidos no momento da compilação. Os valores das propriedades são expressões.

Diagram

Description automatically generated

### Funções

Um literal de função define um valor de função. Ele pode ter um nome opcional que pode ser usado para se chamar recursivamente. Ele pode especificar uma lista de parâmetros que atuarão como variáveis inicializadas pelos argumentos de invocação. O corpo da função inclui definições e instruções de variáveis.

Diagram

Description automatically generated

## Trabalhar com o elemento <script>

Um código JavaScript é anexado ao arquivo HTML através da marcação **<script>**. O elemento **<script>** exige marcações de abertura e fechamento, sendo assim, você deve inserir a marcação de abertura **<script>**, que indica o início do código JavaScript, e, também, deve inserir a marcação de fechamento **</script>**, que indica o final do código JavaScript. Você pode inserir o código JavaScript no documento HTML de duas formas:

* Embutir o código JavaScript no documento HTML através do trecho de código:

|  |
| --- |
| <script>  // Aqui virá o código JavaScript  </script> |

* Vincular um arquivo de script externo com o código JavaScript através do atributo src:

|  |  |  |
| --- | --- | --- |
|  |  | Exemplo: |
| <script src="url"></script> |  | <script src="main.js"></script> |

Daremos alguns exemplos de como usar o código JavaScript embutido no documento HTML, mas é interessante você já se acostumar a criar um arquivo externo para deixar separado as duas partes, como é feito com o estilo CSS.

### Mensagem de alerta

O método **alert()** mostra uma caixa de alerta com uma mensagem específica e um botão de OK. Por exemplo:

|  |
| --- |
| alert('Vamos aprender JavaScript?'); |

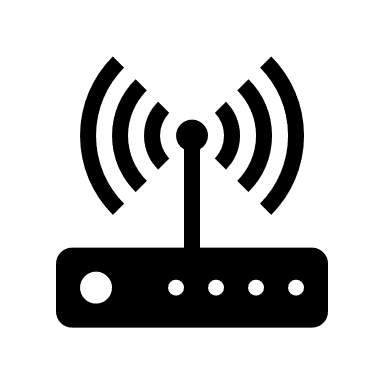
### Vamos praticar

Vamos criar nossa primeira aplicação web com o comando mais simples em JavaScript utilizando o método alert(). Esse método é o equivalente ao “Olá Mundo” em outras linguagens de programação. Siga os passos para criar a nossa primeira aplicação:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 01\_Intro\_JavaScript.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE *html*>  <html *lang*="pt-br">      <head>          <meta *charset*="UTF-8" />          <title>Primeiro exemplo em JavaScript</title>      </head>      <body>          Corpo da página web          <script>  **alert**('Vamos aprender JavaScript?');          </script>      </body>  </html> |

Esse código mostra a marcação <script>, que contém uma instrução em JavaScript. A instrução é a chamada do método alert() com a string “Vamos aprender JavaScript?”.

1. Salve (Ctrl+S) o arquivo e clique no botão **Go Live** da extensão Live Server. Assim, você poderá visualizar a página index.html exibindo a mensagem de alerta.

Graphical user interface, text, application, chat or text message

Description automatically generated

## Glossário

Aqui estão algumas definições e conceitos que você precisa conhecer para entender melhor (Fonte: Wikipedia e outras fontes da internet).

* Document Object Model (DOM): é uma convenção multiplataforma e independente de linguagem de programação, fiscalizada pelo entidade World Wide Web Consortium (W3C), para representação e interação com objetos em documentos HTML, XHTML e, XML.[1][2][3] Onde os elementos/nós de cada documento são organizados em uma estrutura de árvore, chamada de Árvore DOM, que endereça e manipula via uso de funções/métodos (interface pública) sobre os objetos, especificada de acordo com a interface de programação de aplicações (API) utilizada, que oferece uma maneira padrão de se acessar cada elemento de um documento, criando páginas altamente dinâmicas.
* Linguagem de script ou scripting: é uma linguagem de programação que suporta scripts, programas escritos para um sistema de tempo de execução especial que automatiza a execução de tarefas que seriam executadas, uma de cada vez, por um operador humano.
* Snippets: em programação é um termo usado para indicar uma pequena região de código reutilizável. Esses trechos fornecem uma maneira fácil inserir automaticamente códigos ou funções comumente usados ​​em um código.
* Paradigma de programação: é a forma de se classificar determinada linguagem de programação com base em seu funcionamento e sua estruturação. Alguns exemplos de paradigmas de programação são a Programação orientada a objetos, Programação Estruturada e a Programação Imperativa.
* Tipagem: um sistema de tipos é um conjunto de regras que atribuem uma propriedade chamada de tipo para as várias construções - tais como variáveis, expressões, funções ou módulos - que um programa de computador é composto.
* Tipagem dinâmica: é uma característica de determinadas linguagens de programação, que não exigem declarações de tipos de dados, pois são capazes de escolher que tipo utilizar dinamicamente para cada variável, podendo alterá-lo durante a compilação ou a execução do programa.
* Tipagem fraca: Em linguagens fracamente tipadas, as variáveis declaradas, poderão intercambiar seus tipos a qualquer momento, ou seja, poderão receber valores de tipos de dados diferentes e fazer operações entre eles sem a necessidade de uma conversão (casting) explicita do tipo.