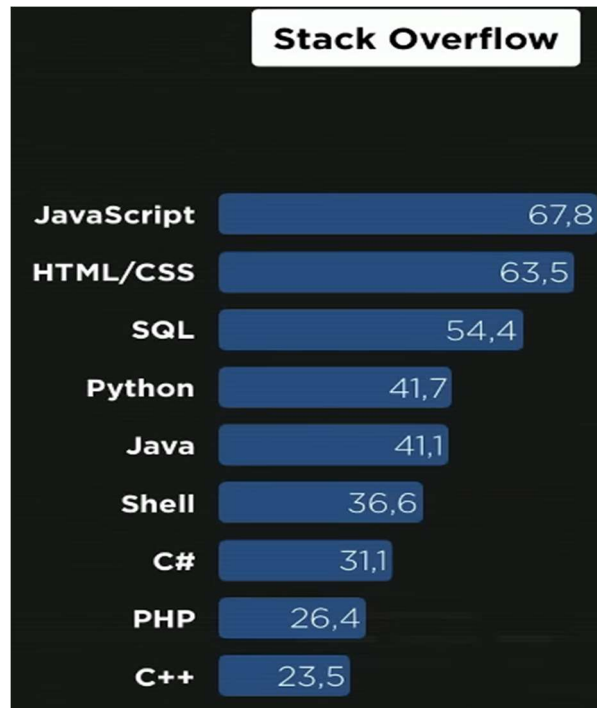


- Qual é a linguagem mais popular?
- Quais são as linguagens mais utilizadas no mundo?

⇒ Esse é o resultado de uma pesquisa de levantamento da **Stack Overflow**, uma das maiores plataformas de reunião de programadores do mundo.



E ela mostra que, **JavaScript** é uma das linguagens com maior quantidade de material e de pessoas interessadas por ela. E o motivo disso é muito fácil de entender. Basicamente, todo site ou aplicação que você utiliza no seu celular ou no seu computador utiliza **JavaScript** na sua essência.

E quando você se tornar um programador mais experiente e quiser aprender novas tecnologias como:

- JQuery;
- Angular;
- React;
- Vue.js;
- Electron;
- Ionic;
- Cordova.



Saiba todas elas têm uma coisa em comum, todas essas tecnologias utilizam na sua essência a linguagem **JavaScript**.

Vamos aprender os fundamentos da linguagem **JavaScript**, aprender a trabalhar no modo gráfico utilizando seu navegador e construções de páginas web interativas utilizando a linguagem, vamos trabalhar um pouco com o Node.js que executa o **JavaScript** fora dos navegadores.

Serão aulas básicas que tratam dos fundamentos da linguagem, vamos trabalhar com as versões mais recentes da especificação **ECMAScript**, que é a padronização internacional da linguagem **JavaScript**.

Para iniciar precisamos entender bem um conceito muito importante, temos que entender o conceito de **CLIENTE X SERVIDOR**.

HTTP (Hypertext Transfer Protocol) é o protocolo base para as aplicações web e clientes (como aplicativos mobile, entre outros) que se conectam a web APIs. Ele permite a comunicação entre clientes e servidores através da internet.

Cliente:

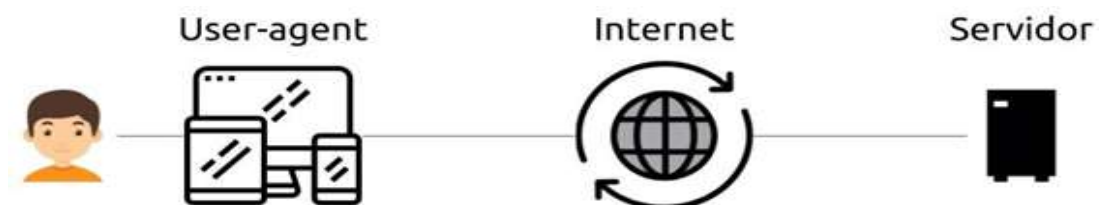
É o cliente quem sempre inicia a comunicação com o servidor.

O cliente mais conhecido é o navegador.

Quando acessamos um site pelo navegador, uma requisição é disparada pela internet para o computador no qual esse site está hospedado.

Chamamos as aplicações que agem em nome do usuário, assim como o navegador, de **user-agent**.

A Figura 1 demonstra esse fluxo de dados.



Comunicação entre o usuário e o servidor através da internet e por meio de um **user-agent**

Após enviar uma requisição, o **user-agent** aguarda até que haja uma resposta do servidor e então a comunicação se encerra.

Para obter um novo recurso do servidor o cliente deve iniciar uma nova requisição.

Servidor:

O papel do servidor web é receber uma requisição e devolver uma resposta para o cliente.

Geralmente o servidor não envia dados se o cliente não disparar primeiro uma requisição.

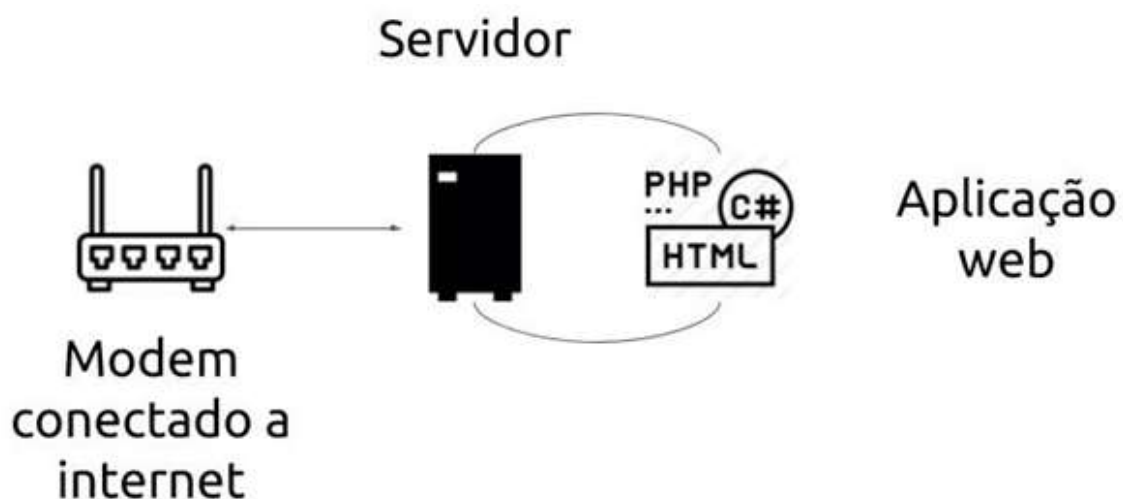
O servidor é um ou mais computadores com um único endereço como `//www.senai.com.br`. Nestes computadores podem ser hospedados documentos, imagens, áudio, vídeo, etc.

Geralmente o servidor também hospeda programas que atendem as requisições dos clientes e as processam. Chamamos esses programas de **web server**, ou servidor web.

É com um web server que nos comunicamos, sendo o computador no qual ele está instalado parte da infraestrutura que o conecta a internet.

O web server geralmente é capaz de executar aplicações escritas em linguagens como PHP, C#, Java, etc.

Na Figura 2 vemos um gráfico que ilustra isso.



Comunicação após o servidor web

Por exemplo, um servidor executando o PHP pode delegar a esse programa a tarefa de atender as requisições criando páginas HTML a partir de scripts escritos nessa linguagem.

Quando você acessa a internet, seja o que for, WhatsApp, Facebook, Instagram, o site do Google, o site do YouTube, o seu Gmail, no momento que você usa um dispositivo, ele é um **cliente** e que precisa de dados de um servidor. Às vezes de até mais de um servidor.

Agora você consegue entender que existe tecnologias para clientes, e existem tecnologias para servidores.

O JavaScript surgiu como uma tecnologia para clientes, é uma tecnologia client-side, ou seja, JavaScript funciona muito mais do lado do cliente.

Uma outra coisa que é muito importante de entender junto com JavaScript são as tecnologias relacionadas.

Sem elas o website não vai conseguir existir.

O conteúdo, o design e as interações.

E quando a gente está falando de desenvolvimento de sites, temos três tecnologias, **HTML**, quem dá todo o texto, imagens, sons e vídeos, o **CSS**, que deixa tudo bonito e o **JavaScript** que é o responsável pelas integrações e interações.

O **JavaScript** permite você fazer quase tudo, até mesmo modificar o documento na sua parte HTML e CSS. Esse é o poder da linguagem.

HTML e CSS não são linguagens de programação, são tecnologias de construção de sites, marcação e estilo, aqui você se diz desenvolvedor.

Mas JavaScript sim, é linguagem de programação, aqui você se diz programador.

Assim que vamos aprofundando os estudos vem as seguintes perguntas:

JavaScript e Java são a mesma coisa? São diferentes? Por que eles têm um nome tão parecido se eles são diferentes? Ou se eles são iguais? Porque tem um script no final de um e não tem um script no final do outro? JavaScript e ECMAScript, tem alguma relação entre eles? O que é esse ECMAScript?

A resposta é simples.

Como o HTML era muito estático, em 1995, um ex-funcionário da Silicon Graphics, que trabalhava para o Jim Clark, passou da Silicon Graphics para dentro da Netscape. Brandon Naick, era um desenvolvedor e a tarefa dele era a seguinte, criar uma linguagem que desse mais funcionalidades ao simples HTML que estava na época.

O Brandon Naick tinha um projeto e batizou a sua linguagem inicialmente de **Mocha**, que é um tipo de cafezinho. E nessa mesma época, estava surgindo uma linguagem lá da **Sun Microsystems**, chamada **Java**, em 1995, o Bryden Naick estava criando a linguagem **Mocha** e estava nascendo ao mesmo tempo a linguagem **Java**. E essa linguagem **Java** ganhou um sucesso muito grande na mídia, que estava chamando a linguagem de: **linguagem do futuro**. E aí a Netscape pensou, nós estamos criando uma linguagem, **Java** é linguagem do futuro, nossa linguagem não vai nem se chamar **Mocha**, na verdade ela tinha até um outro nome, **Mocha** foi só um nome inicial, ela se chamava **LiveScript**. E o pessoal da Netscape falou assim, sucesso, o nome **Java** faz sucesso? Não vamos chamar nossa linguagem de **LiveScript**, nós vamos chamar nossa linguagem de **JavaScript**.

O nome JavaScript não vem porque a linguagem se parece com Java, vem porque na época estava fazendo sucesso falar a linguagem Java, estava muito falada, e aí os caras da Netscape, numa grande jogada de marketing, falaram assim tá aí, vou usar esse nome Java na minha linguagem também, e acabaram lançando a linguagem JavaScript. Na verdade, se você analisar as linguagens Java e JavaScript, são bem diferentes entre si.

É claro que alguns comandos se parecem bastante. E nesse momento você pensa, eu aprendi Java e JavaScript, o if é igualzinho. Na verdade, o if não é o if do JavaScript nem o if do Java. As duas linguagens foram baseadas numa outra linguagem chamada linguagem C. Então é por isso que alguns comandos se parecem bastante, mas são linguagens completamente diferentes. JavaScript é uma coisa, Java é outra. Não confunda.

JavaScript foi uma linguagem que fez bastante sucesso e o sucesso atraiu a atenção de algumas empresas, e uma dessas foi a Microsoft, a criadora do Windows. A ideia da Microsoft foi a seguinte, JavaScript é uma linguagem legal, vamos nós mesmos criar a nossa própria linguagem. Isso porque ela tinha acabado de lançar um outro navegador que basicamente

era o código base do Mosaic, o Mosaic ficou com a NCSA, ele foi passado para uma empresa, que foi comprada pela Microsoft, que pegou o código do Mosaic e lançou um navegador chamado Internet Explorer.

O Internet Explorer foi lançado muito perto do ano de 95, junto com o Windows 95. E aí a Microsoft não deixou para trás, vamos criar nosso próprio JavaScript com os mesmos comandos, resolvendo alguns problemas que a galera já tinha identificado e vamos batizar nossa linguagem não de JavaScript, e sim de JScript, porque pode rolar um processo. A Netscape obviamente pensou, nossa, mas nós criamos a linguagem JavaScript, veio a Microsoft, meio que copiou utilizando os mesmos comandos, deu uma melhorada em outra, vai acabar virando bagunça. E a Netscape teve uma sacada muito boa, que foi padronizar a linguagem.

E isso aconteceu em 1997, quando a Netscape procurou uma empresa de padronização. Nesse caso, uma empresa europeia, a ECMA, que é a Associação Europeia de Fabricantes de Computadores. A ECMA na Europa é tipo uma ISO nos Estados Unidos, ela padroniza as coisas. E foi aí, com a Netscape cedendo o código do JavaScript para a ECMA, para uma padronização, que surgiu a linguagem ECMAScript.

Basicamente, ECMAScript é a linguagem JavaScript padronizada. Hoje, quando se estuda JavaScript, as pessoas pensam que vão estudar JavaScript, mas você vai estudar ECMA, isso significa que você vai estudar a versão padronizada do JavaScript.

E a evolução do JavaScript não foi só a evolução da linguagem, várias ferramentas surgiram com a evolução do JavaScript. E foram ferramentas que surgiram e que levam o JavaScript para outro patamar.

- **jQuery;** - O jQuery não é um framework, ele é um conjunto de bibliotecas. Foi muito famoso durante muitos anos, foi criado por desenvolvedores da Mozilla. Era uma biblioteca que facilitava muito o uso de interatividades em JavaScript.
- **Angular;** - O Angular é criado e mantido pelo Google ele facilita a criação de aplicações web, é uma linguagem menos imperativa, é uma linguagem mais declarativa.

- **React;** - O React tem mais ou menos a mesma funcionalidade do Angular, mas ele tem algumas flexibilidades maiores, Existe o React Native para criação de aplicações em celular.
- **Vue.js;** - O VUE.JS foi criado por um ex-programador do Google que segundo ele mesmo, já estava cansado de usar o Angular e criou uma versão bem melhor.
- **Electron;** - O Electron, hoje é mantido pela GitHub, é especializado para a criação de interfaces gráficas, para programas de interface gráfica. Você com certeza já utilizou ou vai utilizar um programa feito em Electron. O próprio Visual Studio Code da Microsoft, foi feito em Electron. Aqueles programas que você instala em seu computador, para acessar o WhatsApp, o Discord.
- **Ionic;** - O Ionic é um SDK para criar aplicações para dispositivos móveis, utiliza o próprio Node.js, como outros desses aqui usam o Node.js para rodar o JavaScript fora do navegador.
- **Cordova.** - O cordova é a evolução de uma tecnologia antiga chamada de PhoneGap, que hoje evoluiu, está na mão da Apache.

Existem vários outros frameworks, API, bibliotecas, principalmente também para criação de jogos.

São exemplo:

- **Phaser**
- **PixJS**
- **Impact**
- **Melon.js**
- **Crafty.js,**



existem várias bibliotecas e várias tecnologias para criação de jogos em JavaScript.

1. Framework (Estrutura):

- Definição: Um framework é uma estrutura ou ambiente de desenvolvimento que fornece uma base para criar software. Ele define a

estrutura e as diretrizes sobre como o código deve ser organizado e executado.

- Exemplo: Ruby on Rails é um framework popular para desenvolvimento web que fornece ferramentas e estruturas pré-definidas para criar aplicativos web de forma eficiente.

2. SDK (Kit de Desenvolvimento de Software):

- Definição: Um SDK é um conjunto de ferramentas, bibliotecas e documentação que facilitam o desenvolvimento de software para uma plataforma específica.

- Exemplo: O Android SDK é usado para desenvolver aplicativos para dispositivos Android, fornecendo APIs e ferramentas para compilar, depurar e testar aplicativos.

3. API (Interface de Programação de Aplicativos):

- Definição: Uma API é um conjunto de regras e protocolos que permite que diferentes softwares se comuniquem e interajam entre si.

- Exemplo: A API do Google Maps permite que desenvolvedores integrem mapas e funcionalidades de geolocalização em seus aplicativos, utilizando métodos e endpoints específicos para acessar esses recursos.

4. Biblioteca:

- Definição: Uma biblioteca é um conjunto de códigos e funcionalidades pré-definidos que podem ser reutilizados em diferentes projetos para realizar tarefas específicas.

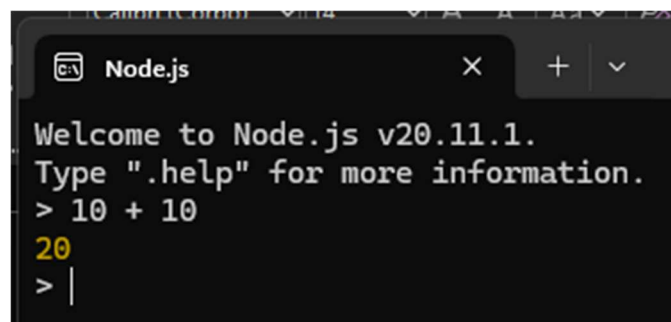
- Exemplo: A biblioteca jQuery é amplamente utilizada para simplificar o desenvolvimento de aplicativos web, oferecendo funções para manipulação de DOM, animações, requisições AJAX, entre outras funcionalidades.

Resumindo, um framework é uma estrutura que orienta o desenvolvimento de software, um SDK é um conjunto de ferramentas para desenvolver em uma plataforma específica, uma API define como os

softwares se comunicam e uma biblioteca oferece funcionalidades específicas para serem reutilizadas em projetos. Cada um desses conceitos desempenha um papel importante no processo de desenvolvimento de software, oferecendo ferramentas e recursos para os desenvolvedores.

Antes de iniciar o desenvolvimento em JavaScript, é necessário saber:

- 1- Instalar e organizar pastas com Visual Studio Code;
- 2- Instalar o NODE.JS;
 - 2.1 – <https://nodejs.org/en>
 - 2.2 – Baixar a versão Recommended
 - 2.3 – Siga a instalação até o final
 - 2.4 – Abra o Node.Js no menu iniciar
 - 2.5 – Para testar digite `10 + 10` e de Enter, ele deverá retornar o resultado 20.



```
Node.js
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> 10 + 10
20
> |
```

- 2.6 – Para sair digite `.exit`
- 3- Saber identificar o HTML5, CSS3 dentro de código;
- 4- Fazer comentários, para isso temos as simbologias:
 - 4.1– HTML
`<!--Comentário aqui-->`
 - 4.2– CSS
`/*Comentário aqui*/`
 - 4.3 - JavaScript
`// comentários em uma única linha.`
`/* */ Comentar trechos de um código.`

Dando inicio ao Desenvolvimento, vamos começar a desenvolver nossos primeiros scripts em JavaScript e aprender a disparar Janelas Simples.

- 1- Criar uma pasta para exercitar o JavaScript dentro de **Documentos** com nome **Javascript**;
- 2- Abrir a pasta e clicar com o direito do mouse abrir o terminal do Git Bash e chamar o **Visual Studio Code**;
- 3- Criar uma pasta chamada **aula1** e dentro dela criar um arquivo chamado **exercicio1.html**
- 4- Iniciar o corpo de um **HTML** simples, como aprendemos no inicio da unidade curricular **Codificação para Front-End**, colocar o Titulo **Exercício 1 JavaScript**, criar um título **<h1> Meu primeiro Script** e um ** Exercício 1** .

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device
6      <title>Exercício 1 JavaScript</title>
7  </head>
8  <body>
9      <h1>Meu primeiro Script</h1>
10     <span>Exercício 1</span>
11 </body>
12 </html>
```

Vamos aprender a colocar CSS dentro do HTML, para simplificar as aulas quando o código for pequeno.

- 1- Dentro do **<head>** vamos criar a **tag <style>** que será onde daremos estilo a esse nosso **exercicio1.html**;
- 2- Aplicar os estilos no **body**;

```
background-color: aqua;
color: blue;
text-align: center;
font-size: 25px;
```

- 3- Antes do fechamento do corpo do Site na **tag </body>**, para que os scripts sejam carregados depois do código base, vamos criar uma **tag <script>** (lembrando que tem como criar scripts externos, igual ao css, mais para frente iremos fazer isso, agora estamos apresentando um exemplo básico);

```
<body>
  <h1>Meu primeiro Script</h1>
  <span>Exercício 1</span>

  <script>

  </script>
</body>
</html>
```

4- Dentro da tag <script> vamos programar em JavaScript.

- ⇒ Não iremos dizer que programamos em HTML e nem em CSS, mas podemos dizer que iremos programar em JavaScript, porque ela sim é uma linguagem de programação. HTML é uma linguagem de marcação de conteúdo, e CSS é uma linguagem de estilos, são folhas de estilo, então você não diz que você programe nenhuma delas, a não ser o JavaScript.
- ⇒ Aqui dentro, podemos escrever o comando em JavaScript, sempre em letras minúsculas, na maioria das vezes, quando não for, irei explicar, porque tem diferença no caso do JavaScript.

SINTAXE BÁSICA

No JavaScript, as instruções (ou comandos) são chamadas de declaração. Uma instrução pode conter várias linhas, que serão lidas (linha por linha) e interpretadas pelo navegador, sequencialmente.

1. Vamos começar mostrando um alerta na tela:

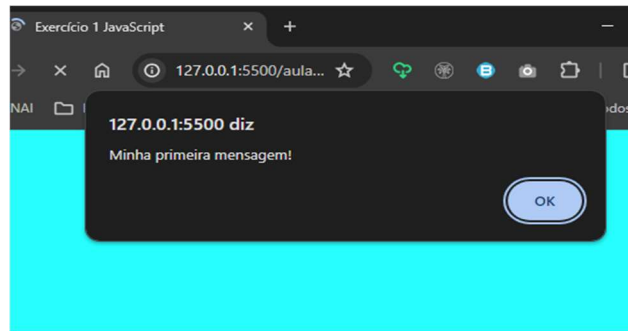
1.1 Alerta:

Exercicio1.html

⇒ **window.alert('Minha primeira mensagem!')**

- Antigamente era preciso colocar ponto e vírgula no final de cada comando, hoje em dia ele não é mais obrigatório.

⇒ Mostra uma janela de alerta!



- Aqui podemos perceber que a parte de estilo já foi exibido, só que o conteúdo ainda não apareceu, o Meu primeiro Script! E o Exercício1 não apareceu ali atrás, mas ele já disse aqui, Minha primeira mensagem. Então esse window.alert é o seu primeiro comando em JavaScript, que podemos até simplificar para somente alert, não tem problema.

Mesmo colocando o <script> no final, ele executou o comando, sem que o conteúdo apareça efetivamente na tela. Mesmo se atualizar o site, o conteúdo não aparece, mas a minha primeira mensagem apareceu, só quando clicar em OK, o conteúdo será exibido.

Essa é a interatividade que iremos criar.

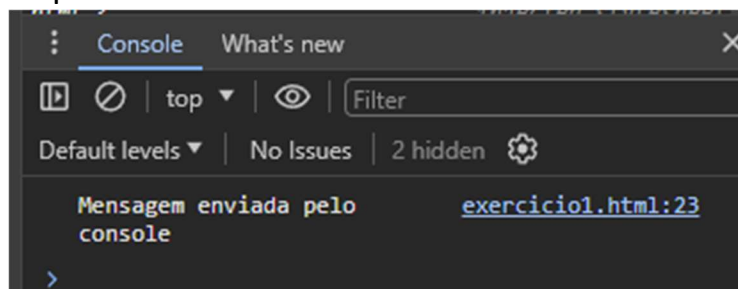
1.2 Console.

Exercicio1.html

⇒ `console.log('Mensagem enviada pelo console')`

```
21 <script>
22   window.alert('Minha primeira mensagem!')
23   console.log('Mensagem enviada pelo console')
24 </script>
```

⇒ `console.log` é uma mensagem visível apenas ao ser inspecionada.



⇒ A mensagem, no caso, é “Mensagem enviada pelo console”.

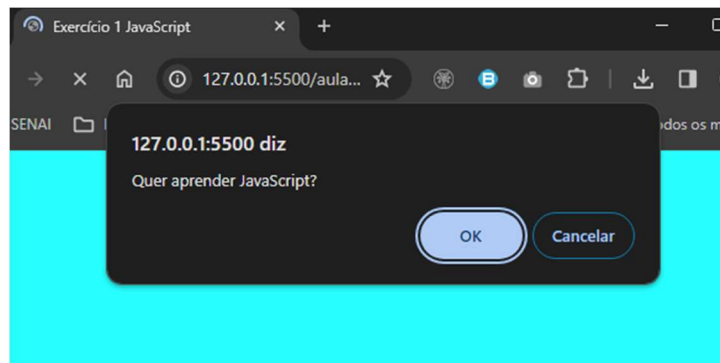
- Agora vamos pedir uma confirmação:

1.3 Confirmações:

Exercicio2.html

⇒ `window.confirm('Quer aprender JavaScript')`

⇒ Mostra uma janela de confirmação!



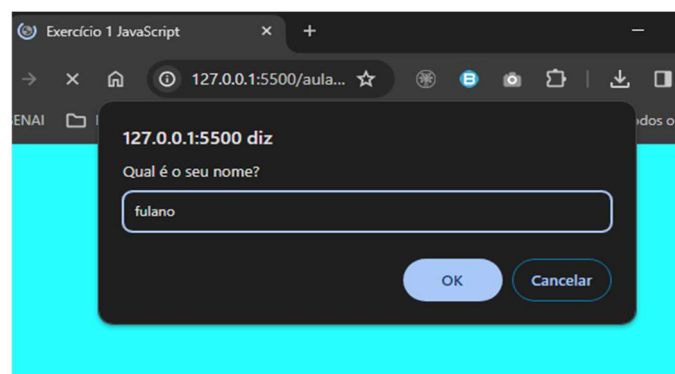
- O primeiro comando a ser executado é o meu `window.alert`. Aparece, Minha primeira mensagem. Depois de clicar no OK, será exibido o `window.confirm`, que está perguntando, Quer aprender JavaScript? com um Ok ou Cancelar.

1.4 Perguntas.

Exercicio2.html

⇒ `window.prompt('Qual seu nome?')`

⇒ Mostra uma janela com uma pergunta!



- Qual é o seu nome? Vou responder Fulano e clicar OK. Agora vem a pergunta, mas o nome não apareceu na tela. Não apareceu porque não mandamos ainda. Na verdade, não tem nem como testar no

momento em que está esse código, para isso é necessário saber qual foi o botão que a pessoa apertou, se ela apertou OK, se ela apertou Cancela. No prompt eu não sei qual foi o nome que a pessoa digitou, isso porque temos que utilizar aqui na frente dessas funcionalidades algumas variáveis.

2. Tipos de Dados e Variáveis;

VARIÁVEIS

Variáveis são elementos que armazenam dados de diferentes tipos (que serão utilizados em um algoritmo) na memória do computador.

Para criar uma variável em JavaScript, basta declará-la, digitando:

- A palavra "var" com o sinal de igual (=);
- O identificador;
- O valor que queremos atribuir a ela.

Exemplos:

⇒ Precisamos sempre nos lembrar que um único sinal de = será visto como **recebe**.

Vamos guardar 3 números:

n1	1	- Aqui falamos que n1 recebe 1
n2	8.5	- Aqui falamos que n2 recebe 8.5
n3	12	- Aqui falamos que n3 recebe 12

Assim teremos:

```
var n1 = 1
```

```
var n2 = 8.5
```

```
var n3 = 12
```

⇒ Precisamos sempre nos lembrar que um único sinal de = será visto como **recebe**.

⇒ **Quando colocamos os valores dentro das variáveis chamamos isso de atribuição, quer dizer que iremos atribuir um valor a uma variável.**

Agora vamos criar uma nova variável de tamanho maior para guardar outro tipo de dados, pois existem variáveis de tamanhos diferentes, para armazenar dados de tamanhos diferentes.

Vamos guardar 3 cadeias de caracteres - strings:

S1	sexta-feira	- Aqui falamos que n1 recebe 1
S2	Estou feliz	- Aqui falamos que n2 recebe 8.5
S3	JavaScript	- Aqui falamos que n3 recebe 12

Assim teremos:

```
var s1 = "sexta-feira"
```

```
var s2 = 'Estou feliz'
```

```
var s3 = `JavaScript`
```

Como podemos perceber, as cadeias de caracteres, nossas palavras, estão entre aspas, e podemos usar em JavaScript 3 tipos de aspas, sendo as duplas, a simples e a crase. São os três tipos de forma de delimitar um string dentro da linguagem JavaScript

Agora, podemos ver que temos 6 espaços, onde cada um tem seu nome definido, para evitar confundi-los. E o nome de cada uma dessas variáveis se chama Identificadores.

E para dar nome a essas variáveis existem algumas regras:

- Podem começar com: letra, \$ ou _
Ex: n1, s1, muito raro começar com \$ ou _, mas podemos utilizar.
- Não podem começar com números
EX: 1S, não podemos fazer.
- Podemos usar letras ou números
Ex: n1, n2, n10
- Podemos usar acentos e símbolos
Ex: média com acento, utilizar símbolo de π para representar o pi
- Não podem conter espaços
EX: Nota 1

- Não podem ser palavras reservadas
EX: function, alert, ou outras palavras que são comandos de JavaScript

Dicas na hora de criar variáveis:

- Maiúsculas e Minúsculas fazem diferença.
- Tente escolher nomes coerentes para as variáveis.

Representação dos tipos de dados:

Inteiros => 10, 2, -30

Reais => 0.2, -8.5, 8.4 – números com ponto flutuante, ou float.

No JavaScript não há a diferença entre esses tipos de dados, todos são vistos como do **tipo number**.

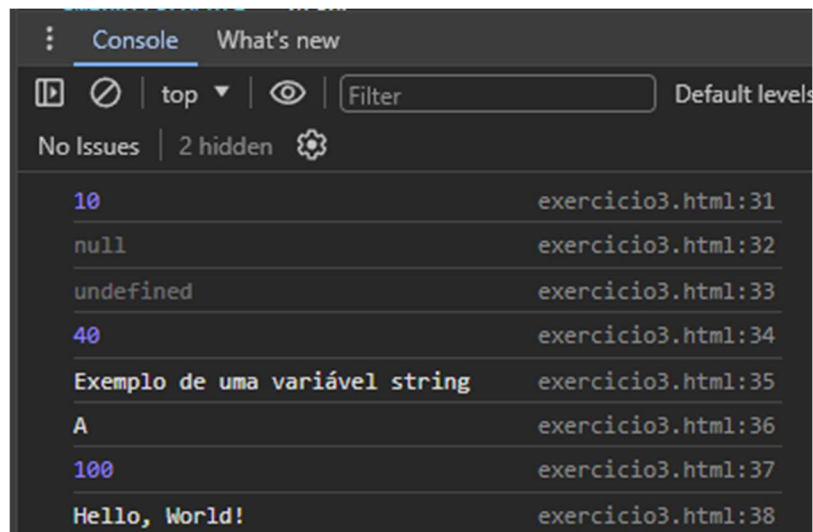
String => “Hoje”, ‘Java’, `tem` - são cadeias de caracteres, um string também pode ser um conjunto de números, por exemplo: o número do cpf, identidade, telefone. Como eles possuem . e -, são considerados strings, por serem um conjunto de caracteres compostos por números, pontos e traços.

Boolean => true – false, verdadeiro e falso

Exercicio3.html

```
21 <script>
22   var exemploBoolean = 10
23   var exemploNull = null
24   var exemploUndefined
25   var exemploNumber = 40
26   var exemploString = "Exemplo de uma variável string"
27   var exemploCaractere = "A"
28   var $ = 100
29   var $$ = "Hello, World!"
30
31   console.log(exemploBoolean)
32   console.log(exemploNull)
33   console.log(exemploUndefined)
34   console.log(exemploNumber)
35   console.log(exemploString)
36   console.log(exemploCaractere)
37   console.log($$)
38   console.log($$)
39
40 </script>
```


- Ao inspecionar o Site, na área do console podemos ver os resultados.



- Vamos ver o resultado no NODE.JS

```

Node.js
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> var exemploBoolean = 10
undefined
> var exemploNull = null
undefined
> var exemploUndefined
undefined
> var exemploNumber = 40
undefined
> var exemploString = "Exemplo de uma variável string"
undefined
> var exemploCaractere = "A"
undefined
> var $ = 100
undefined
> var $$ = "Hello, World!"
undefined

```

Como podemos perceber, após declarar a variável, recebemos em todas esse nome undefined, que significa não definido. Precisamos definir nosso **typeof**.

Nossos tipos de dados ou **data types** são:

Tipos Primitivos

1. String (Cadeia de Caracteres)

A primeira categoria de tipos de dados em JavaScript são as strings. Elas representam texto e são definidas usando aspas simples ou duplas.

Exemplo:

```
var nome = "João";
```

```
var mensagem = 'Olá, Mundo!';
```

2. Number (Número)

Os números são usados para representar valores numéricos. Isso pode ser um número inteiro ou um número de ponto flutuante.

2.1 - **Infinity** - é uma propriedade do *objeto global*, ou seja, é uma variável no escopo global. O valor inicial de Infinity é Number.POSITIVE_INFINITY. O valor Infinity (positivo) é maior do que qualquer outro número. Este valor se comporta matematicamente como infinito; por exemplo, qualquer número positivo multiplicado por Infinity é Infinity, e qualquer coisa dividida por Infinity é 0.

2.2 - **Nan** – Not a Number - é normalmente encontrado quando o resultado de uma operação aritmética não pode ser expresso como um número.

Exemplo:

```
var idade = 30;
```

```
var altura = 1.75;
```

3. Boolean (Booleano)

Os valores booleanos representam verdadeiro ou falso e são usados em lógica condicional.

Exemplo:

```
var aprovado = true;
```

```
var reprovado = false;
```

4. Undefined e Null

- Undefined representa uma variável que foi declarada, mas não foi inicializada.
- Null é usado para representar a ausência de valor.

Exemplos:

```
var valorNaoInicializado;
```

```
var valorNulo = null;
```

5. Symbol (Símbolo)

Símbolos são valores únicos e imutáveis, frequentemente usados como chaves de propriedades em objetos.

Exemplo:

```
var simbolo1 = Symbol('chave');
```

```
var simbolo2 = Symbol('chave');
```

Tipos de Referência

6. Object (Objeto)

Os objetos são estruturas de dados complexas que podem conter várias propriedades e métodos. Eles são fundamentais em JavaScript.

Exemplo:

```
var pessoa = {  
  nome: "Maria",  
  idade: 25,  
  cidade: "Lisboa"  
};
```