

```
mirror_mod = modifier_ob.  
#set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is n
```

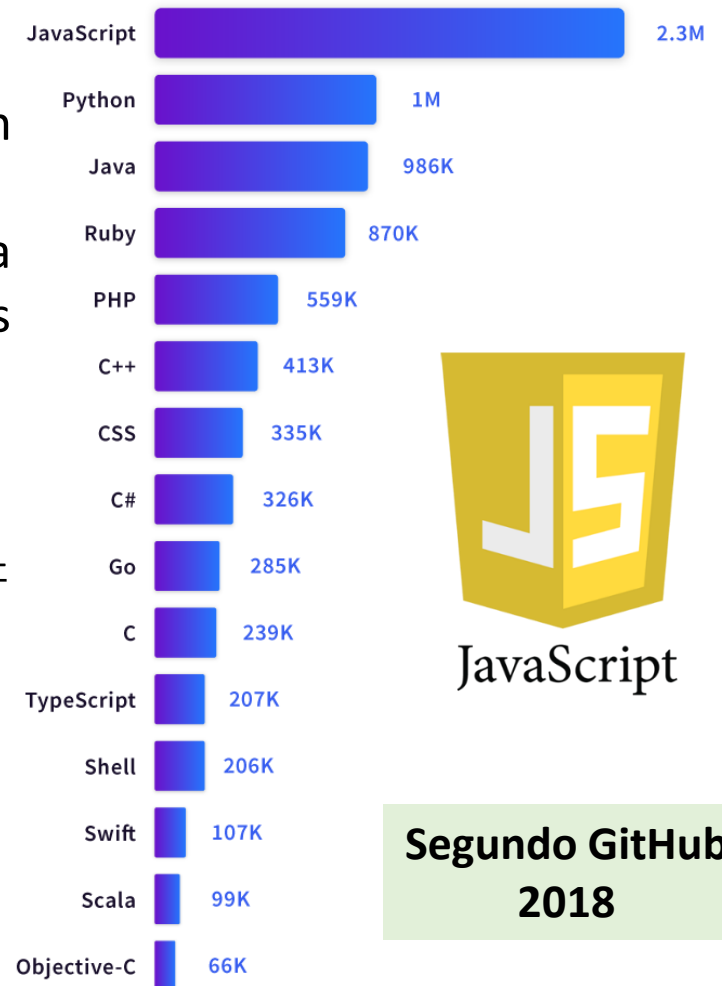
Programação de computadores

Prof. Dr. Josenalde B. Oliveira

josenalde.oliveira@ufrn.br

Linguagem JAVASCRIPT

- Netscape e Sun lançam em 1995
- Interpretada no navegador (browser), porém problemas de incompatibilidade!
- Em 1997 lançada primeira versão gerenciada pela ECMA (European Computers Manufacturers Association)
- Por isso, JavaScript também é conhecida como ECMAScript – versão 10 em Junho 2019:
<https://www.ecma-international.org/publications/standards/Ecma-262.htm>
- Embora originalmente web e usada em conjunto com o HTML, também pode ser utilizada como linguagem de console (terminal) interpretada sem navegadores, usando o **NODE JS**.
- <https://nodejs.org>



**Segundo GitHub
2018**

Tríade no desenvolvimento web...

Os códigos escritos em JavaScript (.js), também chamados de scripts, são interpretados diretamente pelos navegadores web. É utilizada para definir o comportamento de elementos na página web, por exemplo, o usuário pode interagir com formulários, acessar e modificar o conteúdo e as características de uma página, salvar informações no navegador, exibir opções de compra de acordo com o perfil do usuário, jogos interativos que rodem no browser etc.

Atualmente na versão 5, o HyperText Markup Language serve para descrever o conteúdo de uma página web e definir a marcação semântica (significado) dos elementos que compõem a página

O Cascading Style Sheets determina os estilos e a formatação dos elementos, ou seja, define a aparência do site – cores, bordas, espaçamentos etc. É a apresentação da página em si, cuja implementação fica geralmente aos cuidados do Web Designer.



JavaScript

HTML



CSS



JavaScript não é Java!

- Embora possua igualmente sintaxe herdada do C/C++, é uma linguagem própria, independente de Java, embora criada pela Sun em Janeiro 1996, depois vendida a Oracle anos depois.
- Embora possa ser criado código .js no próprio notepad, se utilizam editores online ou offline com mais recursos, como o Visual Studio Code com suporte a Javascript e extensões para complementação inteligente, como o Eslint
- Com o node.js instalado, uma simples execução como **node notepad.js**, interpreta e exhibe no terminal o resultado:

notepad.js

```
var scanf = require('scanf'); // importa biblioteca
var n = scanf('%d'); // %d indica que será lido um número inteiro
console.log('feito no notepad');
console.log(n);
```



Saída no terminal (stdout, standard output):

feito no notepad

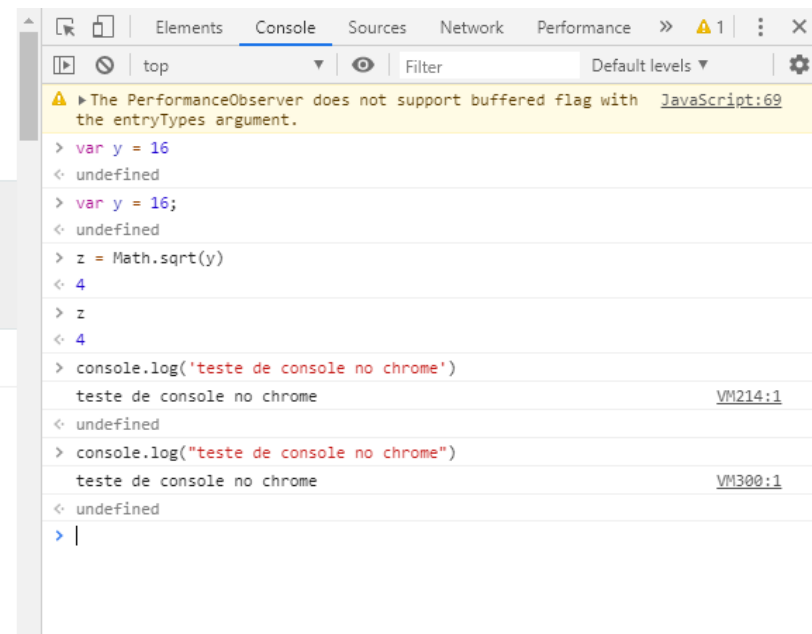
Para ler dados via console, método usual é o comando **scanf**, similar a C

Nos navegadores:

- Os browsers possuem o interpretador nativo JS, portanto, podem ser executados comandos para testes no console



The screenshot shows the MDN web docs homepage for JavaScript. At the top, there's a search bar with the text "Pesquisar na MDN" and a button "Entrar". Below the search bar, there are navigation links: "Tecnologias", "Referências e guias", and "Comentários". The main heading is "JavaScript". Below the heading, there's a breadcrumb "Aprendendo a Web > JavaScript" and a language selector "Português (do Brasil)". On the left, there's a section "Tópicos relacionados" with the text "Completos iniciantes, comecem por aqui!" and a link "Iniciando na Internet". On the right, there's a yellow box with a pencil icon and the text "Esta tradução está incompleta. Ajude a traduzir este artigo em inglês". Below this, there's a paragraph about JavaScript: "JavaScript é uma linguagem de programação que permite implementar funcionalidades mais complexas em páginas web. A cada momento uma página web faz mais do que apenas mostrar informações estáticas para você -".



The screenshot shows a browser's developer console. The console is open, and the "Console" tab is selected. There's a warning message at the top: "The PerformanceObserver does not support buffered flag with the entryTypes argument." Below the warning, there's a series of JavaScript commands and their outputs:

```
> var y = 16
< undefined
> var y = 16;
< undefined
> z = Math.sqrt(y)
< 4
> z
< 4
> console.log('teste de console no chrome')
teste de console no chrome VM214:1
< undefined
> console.log("teste de console no chrome")
teste de console no chrome VM300:1
< undefined
> |
```

Ferramentas online (repl.it, lpo.ufrn.br):

- w3schools.com, da Refsnes Data



Result Size: 668 x 459

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Na EAJ"'>Click Me!</button>

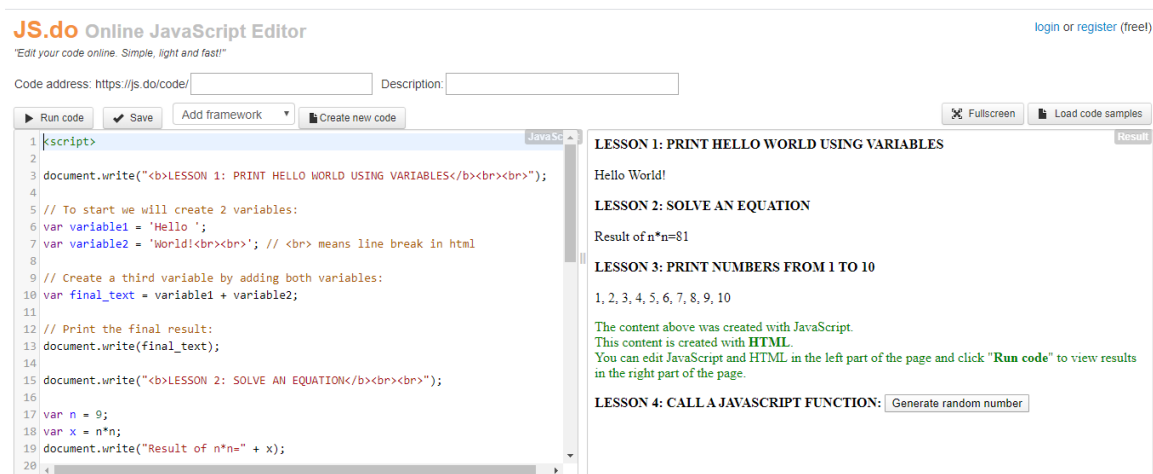
</body>
</html>
```

What Can JavaScript Do?

Na EAJ

Click Me!

- js.do



JS.do Online JavaScript Editor

login or register (free!)

Code address: Description:

Run code Save Add framework Create new code

```
1 <script>
2
3 document.write("<b>LESSON 1: PRINT HELLO WORLD USING VARIABLES</b><br><br>");
4
5 // To start we will create 2 variables:
6 var variable1 = 'Hello ';
7 var variable2 = 'World!<br><br>'; // <br> means line break in html
8
9 // Create a third variable by adding both variables:
10 var final_text = variable1 + variable2;
11
12 // Print the final result:
13 document.write(final_text);
14
15 document.write("<b>LESSON 2: SOLVE AN EQUATION</b><br><br>");
16
17 var n = 9;
18 var x = n*n;
19 document.write("Result of n*n= " + x);
20
```

LESSON 1: PRINT HELLO WORLD USING VARIABLES

Hello World!

LESSON 2: SOLVE AN EQUATION

Result of n*n=81

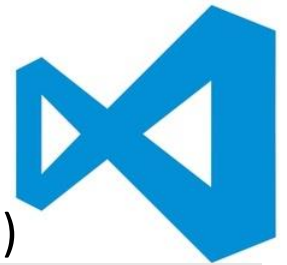
LESSON 3: PRINT NUMBERS FROM 1 TO 10

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

The content above was created with JavaScript.
This content is created with HTML.
You can edit JavaScript and HTML in the left part of the page and click "Run code" to view results in the right part of the page.

LESSON 4: CALL A JAVASCRIPT FUNCTION:

Modalidades de integração



- Simples arquivo .html com TAG (marcação) <script></script> (s1.html)

```
<!DOCTYPE html> <!--mínimo para informar que é arquivo .html-->
<meta charset="UTF-8"> <!--unicode (para acentos)-->
<script>
    //aqui é código JavaScript, alert exibe janela simples
    alert('primeiro javascript que faço');
    //pode ser aspa duplas
    /*
        implementações recentes do ECMAScript dispensam o ;porém no
        terminal com uso do node obrigatório
    */
</script>
```

- A entrada de dados pode ser pelo método (comando, função) **prompt**

```
<!DOCTYPE html>
<meta charset="UTF-8">
<script>
    //declara variável (não precisa do tipo)
    var nome = prompt('Qual é o seu nome?');
    alert('Oi ' + nome); //concatena texto (string) e variável
</script>
```

Modalidades de integração

- Declaração de variáveis e constantes

```
<!DOCTYPE html>
<meta charset="UTF-8">
<script>
    /* variáveis em JS não podem conter espaços, começar por
    número, conter caracteres especiais e utilizar palavras reservadas
    da linguagem. Podem começar com letra e _ (underscore) */
    var nome, _x, x1, tempAmbiente, notaAlunoInformatica;
    const MAXALUNOS = 50;
    _x = 20; //permitido pois é variável
    MAXALUNOS = 10; // ERRO, pois é constante
</script>
```

- Uso do **confirm** e escrita simples na página html com document.write

```
<!DOCTYPE html>
<meta charset="UTF-8">
<script>
    var temCerteza = confirm("Confirma os dados:");
    if (temCerteza) document.write('ok, gravando no banco de dados...');
    else console.log('operação cancelada');
</script>
```


Tipos de dados e conversões

```
<!DOCTYPE html>
<script>
  var a = "20";
  var e = a + 2; // 202, pois concatena 20 com 2
  alert('e: ' + e);
  var b = a * 2;
  var r = a % 2; // resto da divisão
  var c = a - 2;
  var d = a / 2;
  var p = a**2; //operador potência
  document.write(p);
  //ideal é converter números por padrão
  var s = Number(a); // ou parseInt(a)
  alert('s+2: ' + (s+2));
  var num = Number(prompt('Numero: '));
  var dobro = 2*num;
</script>
```

Uma boa prática é converter sempre dados numéricos recebidos de campos texto

Exercícios

1. Escreva script que leia a duração de uma viagem em dias e horas. Calcule e informe a duração total da viagem em número de horas
2. Numa pizzaria, leia o valor total de uma conta e quantos clientes vão pagá-la. Calcule e exiba o valor pago por cliente, com duas casas decimais e o símbolo de R\$ antes do valor
3. Elabore um programa para uma loja, que leia o preço de um produto e informe as opções de pagamento da loja. Calcule e informe o valor para pagamento à vista, com 10% de desconto e o valor em 3x sem juros

Uma boa prática é converter sempre dados numéricos recebidos de campos texto

Modalidades de integração com HTML

- Estrutura básica HTML (usar o ! No VS CODE para auto preenchimento)

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie-edge">
  <title>Título da Página</title>
</head>
<body>
  Aqui entra o corpo da página
</body>
</html>
```

Head – cabeçalho e inclui as **metatags** e o título

Modalidades de integração com HTML

- Vamos assumir este código digitado dentro do BODY

```
<body>
  <h1>Meu primeiro formulário</h1>
  <p>
    Nome: <input type="text" id="nome">
    <input type="button" value="Mostrar">
  </p>
  <p id="resposta"></p>
</body>
```

- <p>: cria parágrafo
- <h1>: cria texto formatado de título hierarquia 1
- <input>: cria controles para o formulário (text, button, ...)
- Observe que o campo de texto possui um ID (nome de variável), o botão possui um VALUE e o segundo parágrafo um ID
- Logo, podem ser acessados e manipulados via JavaScript

Modalidades de integração com HTML

- Entendendo EVENTOS e FUNÇÕES: o que acontece ao clicar no botão

```
<body>
  <h1>Meu primeiro formulário</h1>
  <p>
    Nome: <input type="text" id="nome">
    <input type="button" value="Mostrar" onclick=...>
  </p>
  <p id="resposta"></p>
</body>
```

- <p>: cria parágrafo
- <h1>: cria texto formatado de título hierarquia 1
- <input>: cria controles para o formulário (text, button, ...)
- Observe que o campo de texto possui um ID (nome de variável), o botão possui um VALUE (caption ou rótulo) e o segundo parágrafo um ID
- Logo, podem ser acessados e manipulados via JavaScript

```
...onclick="document.getElementById('resposta').textContent='oi'">
```

Modalidades de integração com HTML

- Entendendo EVENTOS e FUNÇÕES: o que acontece ao clicar no botão

```
<body>
  <h1>Meu primeiro formulário</h1>
  <p>
    Nome: <input type="text" id="nome">
    <input type="button" value="Mostrar" onclick=mostrarOla()>
  </p>
  <p id="resposta"></p>
</body>
<script>
  function mostrarOla() {
    var inNome = document.getElementById("nome");
    var nome = inNome.value;
    document.getElementById("resposta").innerHTML="Oi " + nome;
  }
</script>
```

- No evento (on)click, mas também existem (on)change, (on)submit, (on)blur, (on)load, etc. chama a execução do método ou função **mostrarOla()**

DOM: Document Object Model

- No HTML, os componentes possuem propriedades e métodos que podem ser acessadas e controladas via SCRIPT. Define-se uma variável associada ao componente com o ID – ideia: estrutura hierárquica, como árvore genealógica

```
<body>
  <h1>Meu primeiro formulário</h1>
  <p>
    Nome: <input type="text" id="nome">
    <input type="button" value="Mostrar" id="mostrar">
  </p>
  <p id="resposta"></p>
</body>
<script>
  function mostrarOla() {
    var inNome = document.getElementById("nome");
    var nome = inNome.value;
    document.getElementById("resposta").textContent="Oi " + nome;
  }
  var mostrar = document.getElementById("mostrar");
  mostrar.onclick = mostrarOla;
</script>
```

DOM: Document Object Model

- A boa prática é separar a aparência (HTML e CSS) – frontend, do script que roda por “trás” a lógica do script em si, chamado BACKEND. Pode-se criar uma pasta SRC (src) dentro do projeto para guardar os arquivos .js

```
<body> <!--arquivo index.html -->
  <h1>Meu primeiro formulário</h1>
  <p>
    Nome: <input type="text" id="nome">
    <input type="button" value="Mostrar" id="mostrar">
  </p>
  <p id="resposta"></p>
  <script src="src/backend.js"></script>
</body>
```


DOM: Document Object Model

- No arquivo backend.js

```
function mostrarOla() {  
    var inNome = document.getElementById("nome");  
    var nome = inNome.value;  
    document.getElementById("resposta").innerHTML="Oi " + nome;  
}  
var btMostrar = document.getElementById("mostrar");  
//registra para o botão "mostrar" um ouvinte para o evento click  
//que ao ser clicado irá chamar a função mostrarOla  
btMostrar.addEventListener("click", mostrarOla);
```

- O recomendado é utilizar os chamados listeners (ouvintes) de eventos ou modelos de eventos DOM nível 2. Pode-se passar várias funções para um mesmo elemento HTML num mesmo evento. Cria-se `elemento.addEventListener('evento', função);`