

# Webdesign

## Introdução a JavaScript

Maurício Manoel  
mauriciomanoel@gmail.com  
<http://github.com/mauriciomanoel>

11 de dezembro de 2017

- Introdução
- Características
- Formas de Uso
- Display
- Sintaxe
- Variáveis
- Operadores
- Tipos de Dados

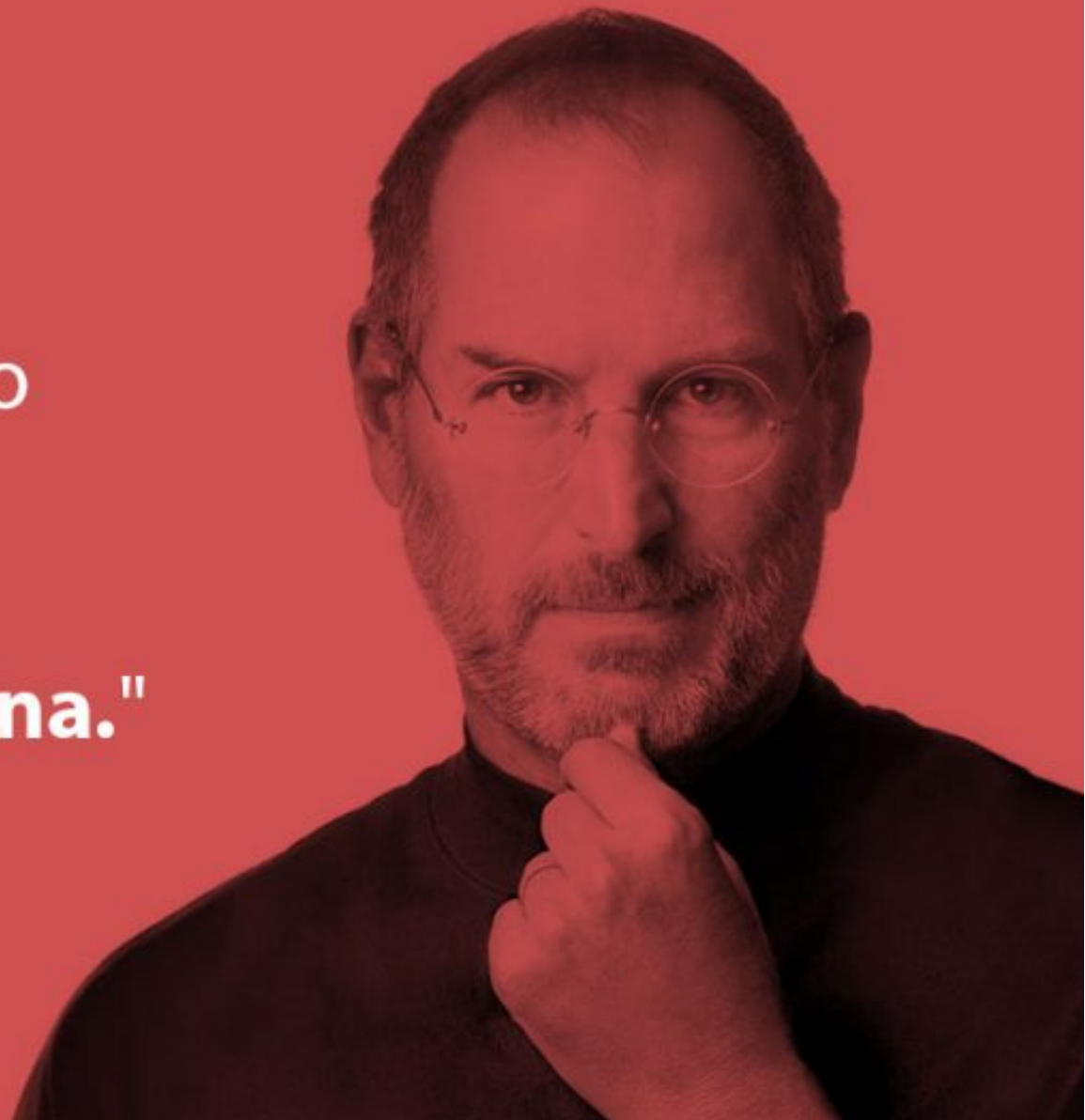
# Agenda

---



- Conceitos Importantes
- Funções
- Eventos
- Formulário

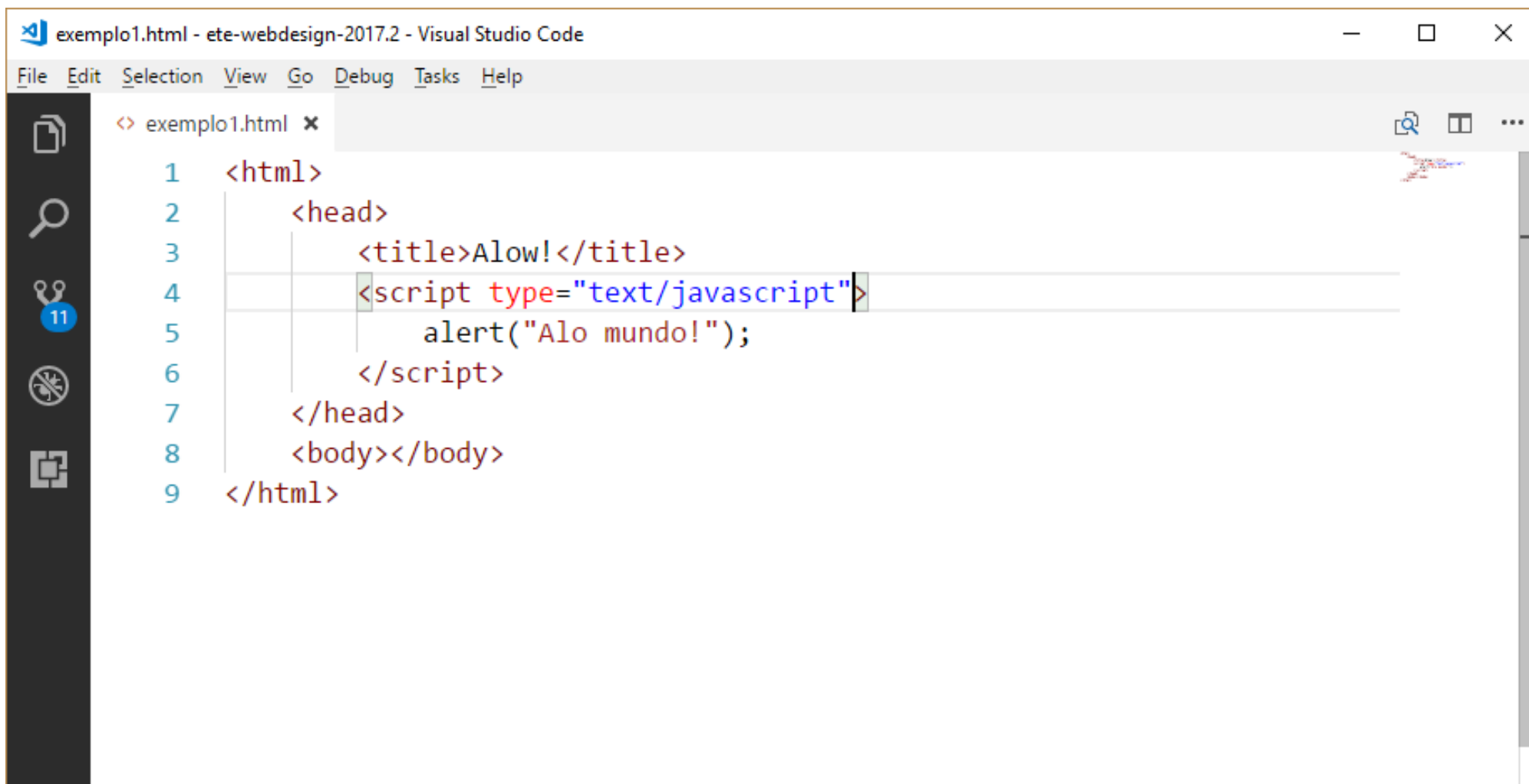
"Design não é apenas o  
que parece e o  
que se sente.  
Design é **como funciona.**"



- O código em uma página pode ser concebido em três visões distintas:
  - Estrutura e conteúdo: HTML
  - Apresentação: CSS
  - Comportamento: JavaScript
- Vantagens:
  - Reuso de partes do projeto
  - Modularidade
  - Flexibilidade e facilidade de manutenção
  - Legibilidade

- É uma linguagem poderosa, com sua grande aplicação do lado cliente (browser);
- É uma linguagem de scripts que permite interatividade nas páginas web;
- É incluída na página HTML e interpretada pelo navegador;
- É simples, porém pode-se criar construções complexas e criativas;
- JavaScript **NÃO** é Java. São linguagens com aplicações e recursos totalmente distintos

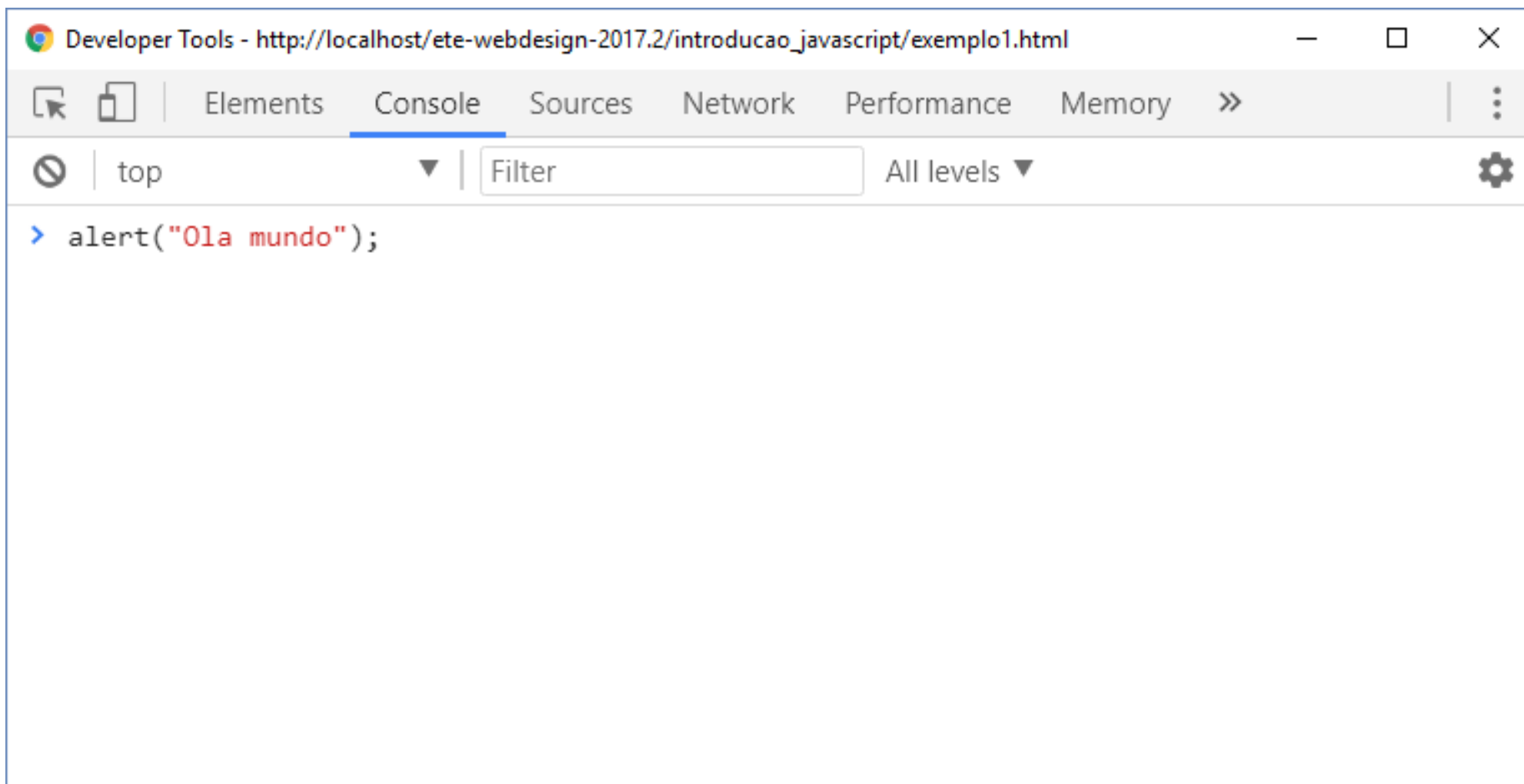
# Meu Primeiro JavaScript



The screenshot shows the Visual Studio Code editor interface. The title bar reads "exemplo1.html - ete-webdesign-2017.2 - Visual Studio Code". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Debug", "Tasks", and "Help". The left sidebar contains icons for Explorer, Search, Run and Debug, Extensions, and a panel with 11 items. The main editor area displays the content of "exemplo1.html" with the following code:

```
<> exemplo1.html x
1 <html>
2   <head>
3     <title>Aloow!</title>
4     <script type="text/javascript">
5       alert("Alo mundo!");
6     </script>
7   </head>
8   <body></body>
9 </html>
```

# Meu Primeiro JavaScript





## Dentro próprio código HTML (In Line)

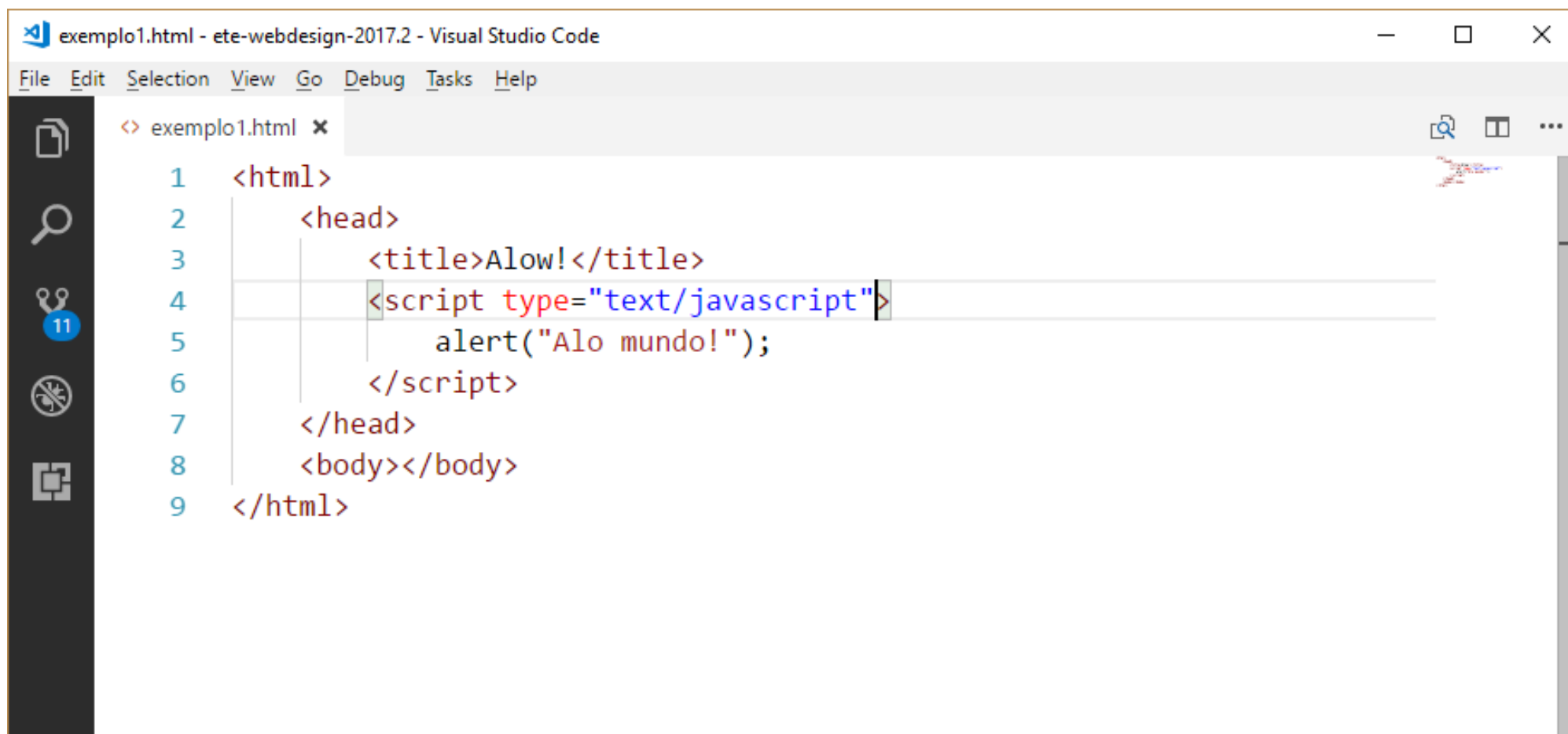


```
exemplo2.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

<> exemplo2.html x
1 <html>
2 <head>
3   <title>Olá Mundo</title>
4 </head>
5 <body>
6   <a href="#" onclick="alert('Olá mundo!')">Click Aqui</a>
7 </body>
8 </html>
```

Fonte: [https://www.w3schools.com/js/js\\_where.asp](https://www.w3schools.com/js/js_where.asp)

## Dentro da Tag Script (<header>)



```
exemplo1.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo1.html x
1 <html>
2   <head>
3     <title>Aloow!</title>
4     <script type="text/javascript">
5       alert("Alo mundo!");
6     </script>
7   </head>
8   <body></body>
9 </html>
```

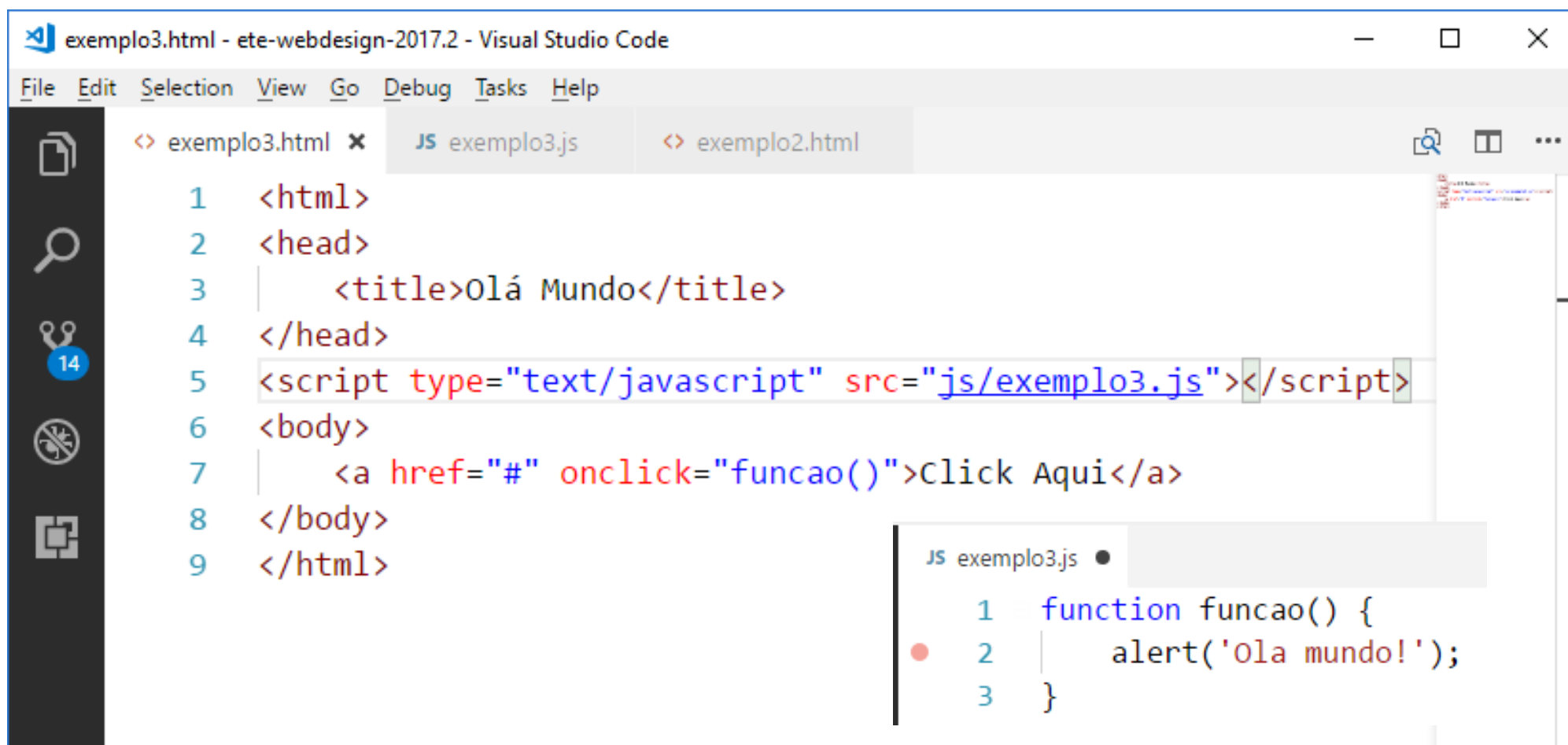
# Formas de Uso

## Dentro da Tag Script (<body>)



```
exemplo1.1.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo1.html <> exemplo1.1.html x
1 <html>
2   <head>
3     <title>Aloow!</title>
4   </head>
5   <body>
6     <script type="text/javascript">
7       alert("Alo mundo!");
8     </script>
9   </body>
10  </html>
```

## Referencia Externa (Local)



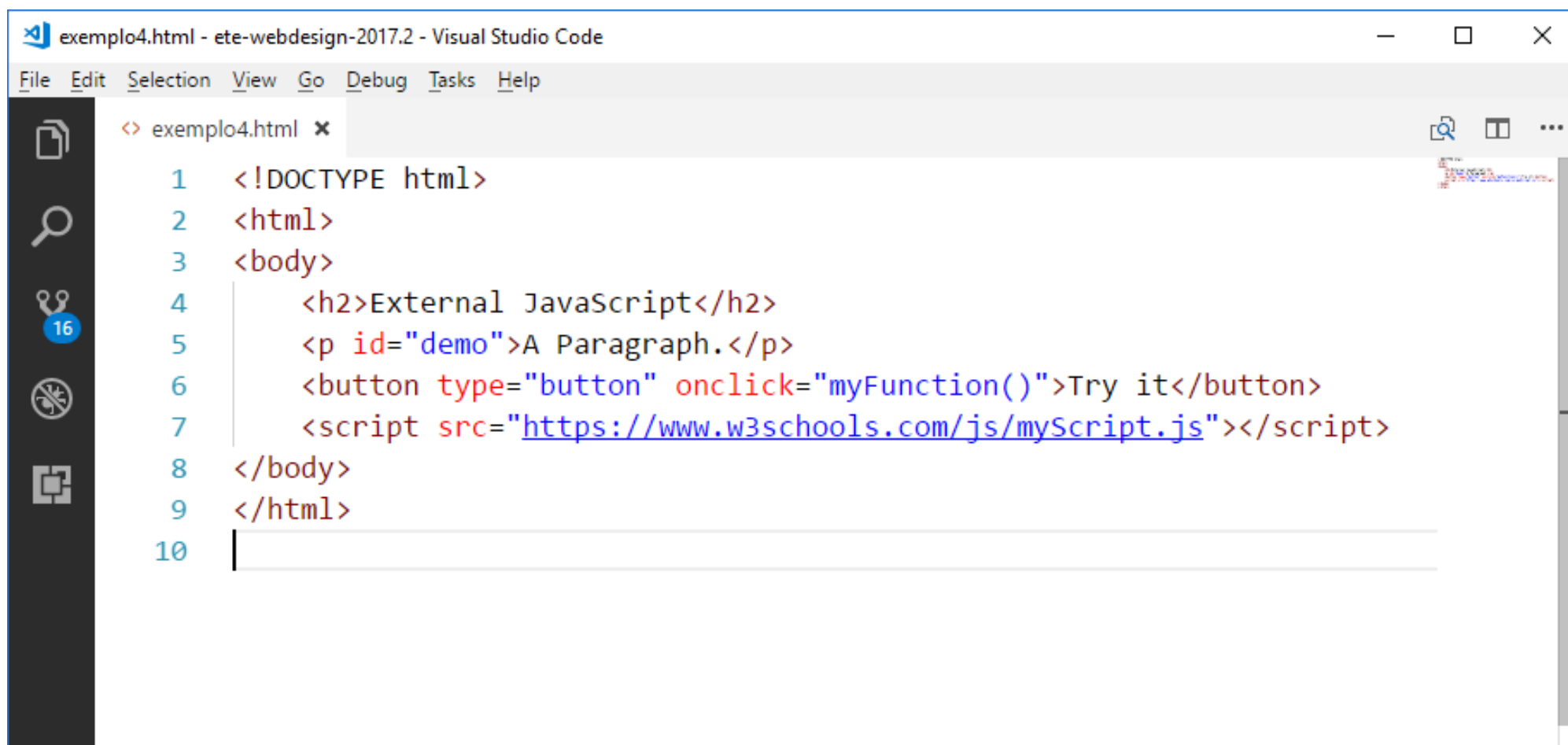
The screenshot shows the Visual Studio Code editor with a file named `exemplo3.html` open. The editor displays the following HTML code:

```
1 <html>
2 <head>
3   <title>Olá Mundo</title>
4 </head>
5 <script type="text/javascript" src="js/exemplo3.js"></script>
6 <body>
7   <a href="#" onclick="funcao()">Click Aqui</a>
8 </body>
9 </html>
```

The `src="js/exemplo3.js"` attribute in the `<script>` tag is highlighted, indicating a local reference to the JavaScript file. The file explorer on the left shows the project structure, and the output console on the right displays the JavaScript code from `exemplo3.js`:

```
1 function funcao() {
2   alert('Ola mundo!');
3 }
```

## Referencia Externa



```
exemplo4.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo4.html x
1 <!DOCTYPE html>
2 <html>
3 <body>
4     <h2>External JavaScript</h2>
5     <p id="demo">A Paragraph.</p>
6     <button type="button" onclick="myFunction()">Try it</button>
7     <script src="https://www.w3schools.com/js/myScript.js"></script>
8 </body>
9 </html>
10
```

## Vantagens da Referência Externa

- Separa HTML e código;
- Isso torna o HTML e o JavaScript mais fáceis de ler e manter;
- Arquivos de JavaScript em cache podem acelerar cargas de página;
- Para adicionar vários arquivos de script a uma página - use várias tags de script

```
<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>
```

- O JavaScript pode "exibir" dados de diferentes maneiras:
  - Escrevendo em um elemento HTML, usando `innerHTML`;
  - Escrevendo na saída HTML usando `document.write()`;
  - Escrevendo em uma caixa de alerta, usando `window.alert()`;
  - Escrevendo no console do navegador, usando `console.log()`.


## InnerHTML

- Para acessar um elemento HTML, o JavaScript pode usar o método `document.getElementById(id)`;
- O atributo `id` define o elemento HTML;



# Display

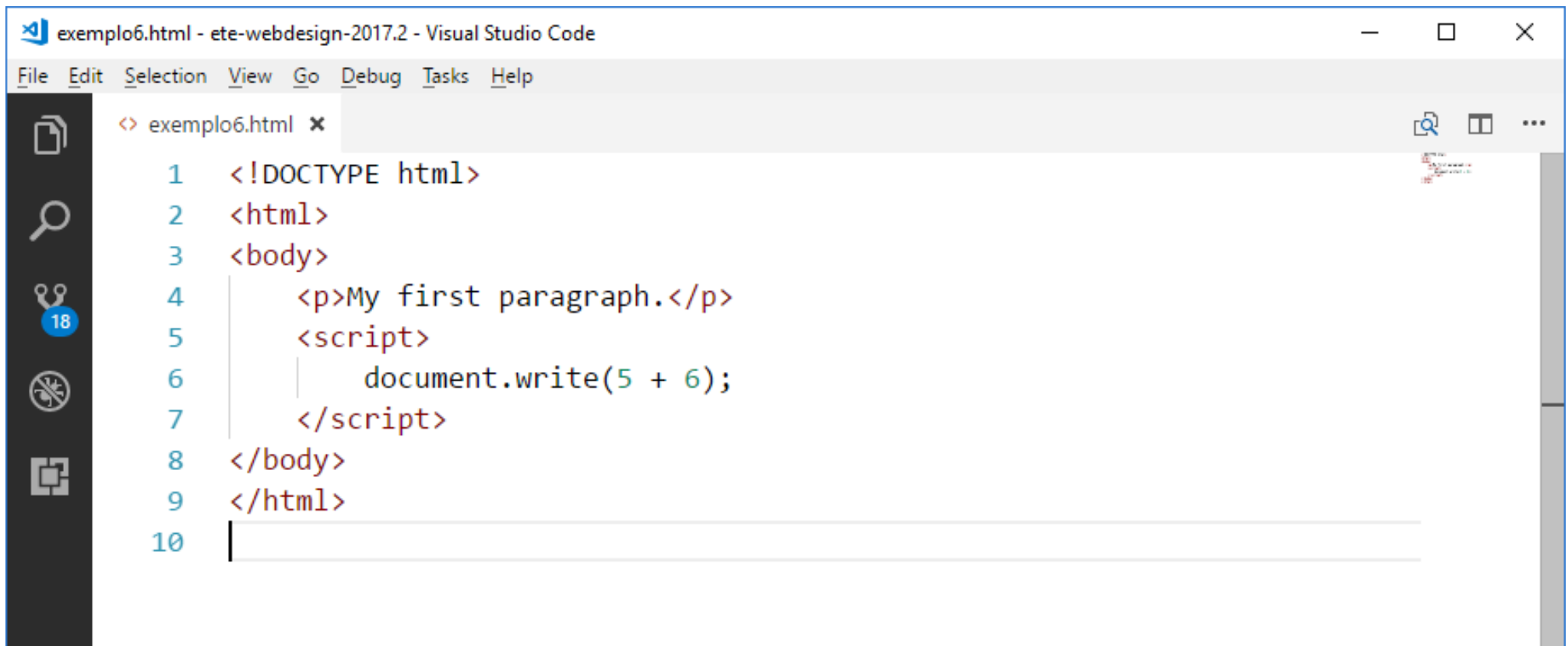
## InnerHTML



```
exemplo5.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo4.html <> exemplo5.html x
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p>My First Paragraph</p>
5   <p id="demo"></p>
6   <script>
7     document.getElementById("demo").innerHTML = 5 + 6;
8   </script>
9 </body>
10 </html>
11
12
```

## document.write

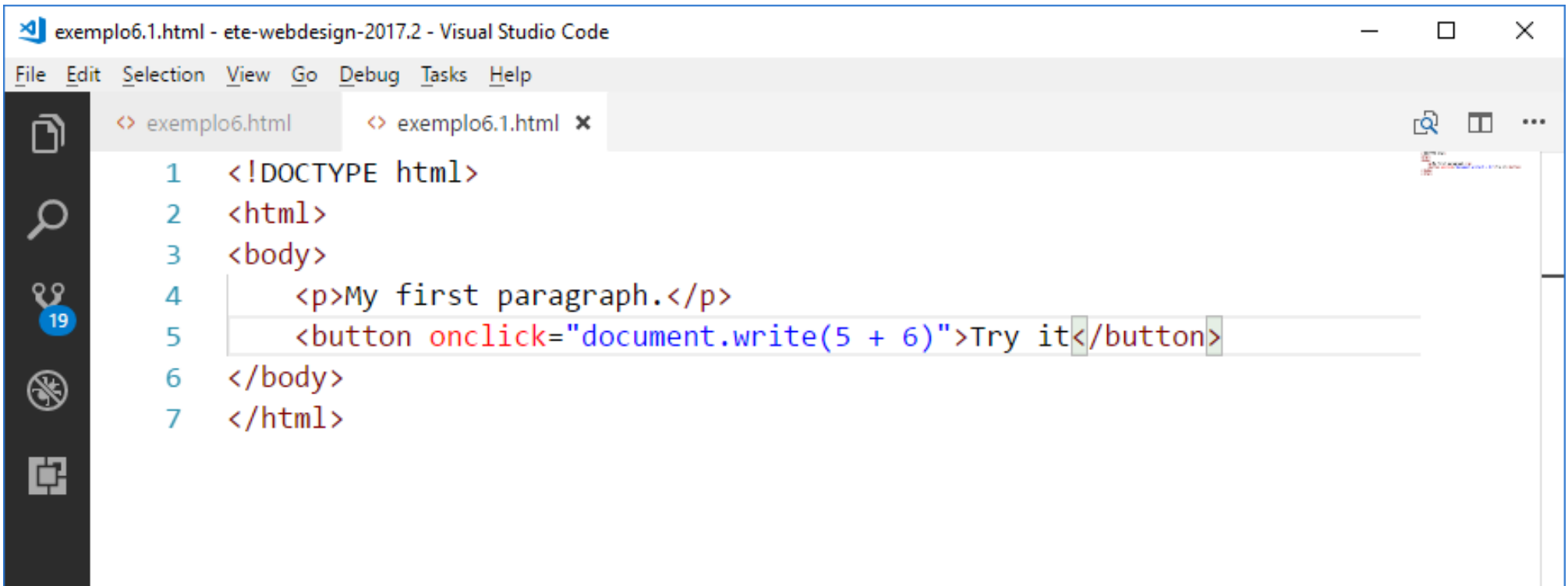
- Para fins de teste, é conveniente usar document.write():



```
exemplo6.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo6.html x
1 <!DOCTYPE html>
2 <html>
3 <body>
4     <p>My first paragraph.</p>
5     <script>
6         document.write(5 + 6);
7     </script>
8 </body>
9 </html>
10
```

## document.write

- Usando document.write() depois de um documento HTML estar totalmente carregado, irá excluir todos os HTML existentes;

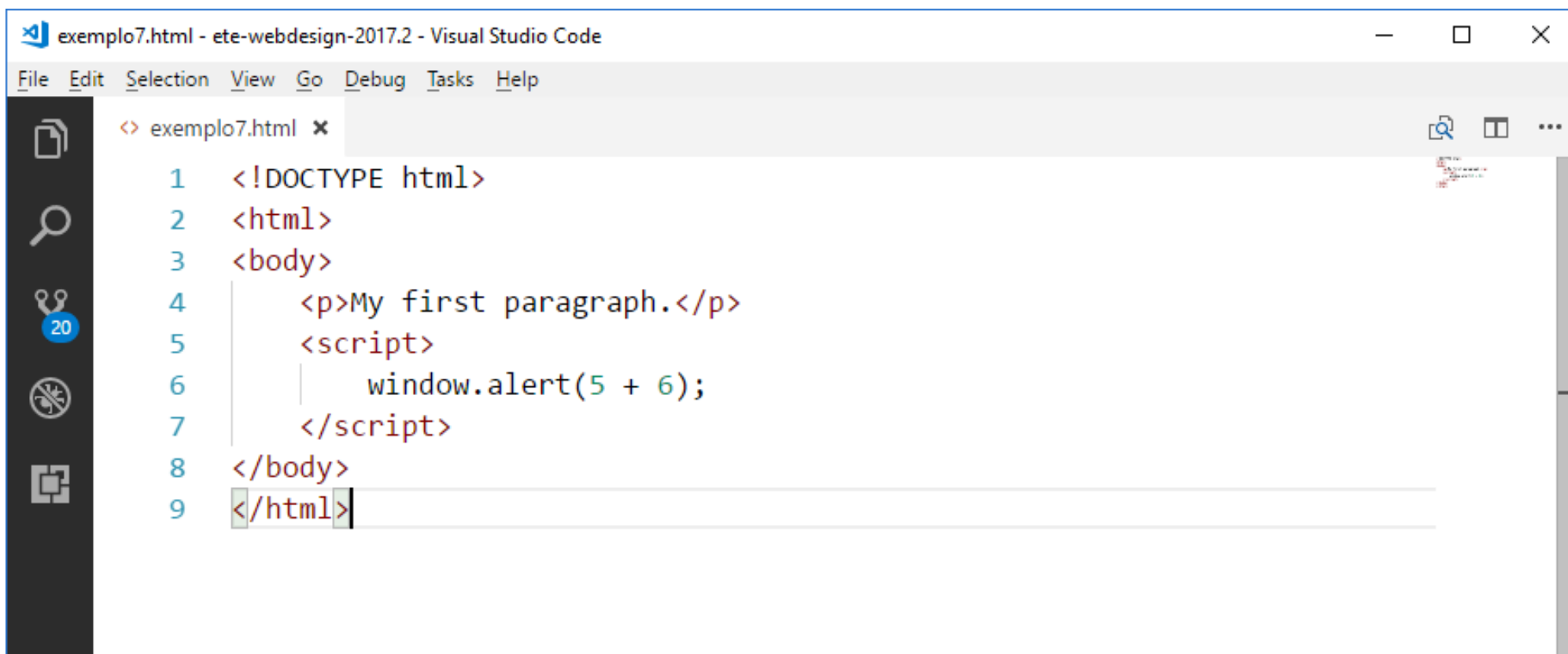


```
exemplo6.1.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo6.html <> exemplo6.1.html x
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p>My first paragraph.</p>
5   <button onclick="document.write(5 + 6)">Try it</button>
6 </body>
7 </html>
```

# Display

## window.alert

- Caixa de alerta para exibir os dados;



```
exemplo7.html - ete-webdesign-2017.2 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
<> exemplo7.html x
1 <!DOCTYPE html>
2 <html>
3 <body>
4     <p>My first paragraph.</p>
5     <script>
6         window.alert(5 + 6);
7     </script>
8 </body>
9 </html>
```

## console.log

- Para fins de depuração, você pode usar o método `console.log()` para exibir dados.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the file structure of a project named 'ete-webdesign-2017.2'. The main editor area shows the file 'exemplo7.1.html' with the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <script>
5     console.log(5 + 6);
6   </script>
7 </body>
8 </html>
```

- Tudo é case-sensitive, ou seja: **teste** é diferente de **Teste**;
- Construções simples: após cada instrução, finaliza-se utilizando um ponto-e-vírgula:
  - Instrução1;
  - Instrução2;
- Ex:
  - `alert("alo");`
  - `alert("mundo");`

- Comentários de uma linha:
  - `alert("teste"); // comentário de uma linha`
- Comentário de várias linhas:  
`/* este é um comentário  
de mais de uma linhas */`

- Variáveis são usadas para armazenar valores temporários;
- Usamos a palavra reservada **var** para defini-las;
- Em JS, as variáveis são **Fracamente Tipada**, ou seja, o tipo não é **definido explicitamente** e sim a partir de uma atribuição ( = );

**var** x = 4;      ← Declaração e atribuição de valor

**var** y;      ← Declaração sem atribuição

y = 2;      ← Atribuição

alert (x + y);



## Atribuição

```
var x = 5;           // assign the value 5 to x
var y = 2;           // assign the value 2 to y
var z = x + y;       // assign the value 7 to z (x + y)
```

# Operadores

## Aritmético

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

```
alert(1+1);  
alert(1-1);  
alert(2*2);  
alert(2/2);  
alert(5%2);  
var x = 1;  
alert(x);  
x++;  
alert(x);  
var x = 2;  
alert(x);  
x--;  
alert(x);
```

- As variáveis de JavaScript podem conter muitos tipos de dados: números, Strings, objetos e muito mais;
- Na programação, os tipos de dados são um conceito importante.
- Sem tipos de dados, um computador não pode resolver com segurança operações matemáticas entre outras;

### Números: Inteiros e Decimais

```
var i = 3;  
var peso = 65.5;  
var inteiroNegativo = -3;  
var realNegativo = -498.90;  
var expressao = 2 + (4*2 + 20/4) - 3;
```

## Strings ou Cadeia de Caracteres

```
var nome = "jósé";  
var endereco = "rua" + " das flores"; // concatenação  
alert(endereco); // imprimindo na tela  
nome = nome + " maria"; // concatenação  
alert(nome); // imprimindo na tela  
endereco = "rua a, numero " + 3; // concatenação com conversão numérica implícita  
alert(endereco);|
```

## Arrays

- Alternativa para o armazenamento de conjuntos de valores

```
var cidades = [];  
cidades[0] = "Recife";  
cidades[1] = "Carpina";  
cidades[2] = "Lagoa do Carro";  
cidades[3] = "Limoeiro";  
  
alert("Uma cidade de Pernambuco é " + cidades[2]);
```

## Arrays

- Tamanho de um array: usamos a propriedade **length** do próprio array:

```
alert(cidades.length);
```

- Primeiro item de um array:

```
alert(cidades[0]);
```

- Último item de um array:

```
alert(cidades[cidades.length-1]);
```

### Arrays Associativos

- Baseados também na ideia `array[indice] = valor`;
- O índice/chave de um array associativo é geralmente uma string

```
var idades = [];  
idades["ely"] = 29;  
idades["Gleison"] = 20;  
idades["maria"] = 20;  
alert("A idade de Maria é: " + idades["maria"]);
```



## Typeof

- Inspecionar o tipo de uma variável ou valor

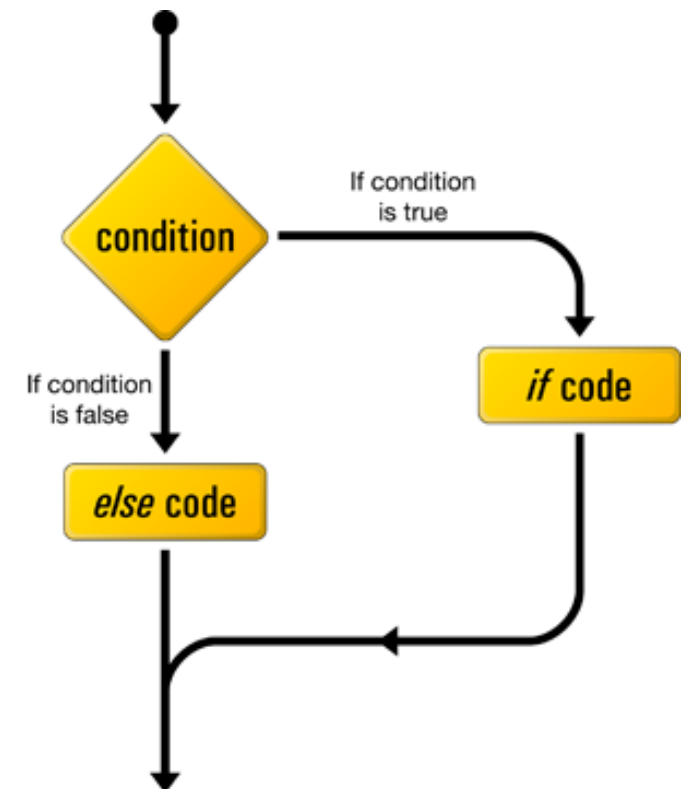
```
var a = "teste";  
alert( typeof a);      // string  
alert( typeof 95.8);   // number  
alert( typeof 5);      // number  
alert( typeof false); // boolean  
alert( typeof true);   // boolean  
alert( typeof null);   // object  
var b;  
alert( typeof b);      // undefined
```

## Typeof

- Utilizando typeof podemos ter os seguintes resultados:
  - **undefined**: se o tipo for indefinido;
  - **boolean**: se o tipo for lógico;
  - **number**: se for um tipo numérico (inteiro ou ponto flutuante);
  - **string**: se for uma string (cadeira de caracteres);
  - **object**: se for uma referência de tipos (objeto) ou tipo nulo.

## Estruturas de decisão – if e else

```
var idade = 17;  
if (idade >= 16 && idade < 18) {  
    alert("voto facultativo");  
}
```



## Operadores condicionais e lógicos

>	A > B
>=	A >= B
<	A < B
<=	A <= B
==	A == B
!=	A != B

← A é igual a B

← A é diferente de B

### Janelas de diálogo - Confirmação

- Nos permite exibir uma janela pop up com dois botões: **OK** e **Cancelar**;
- Como funciona: Se o usuário clicar em **OK**, ela retorna **true**; em **Cancelar** retorna **false**;

```
var email = confirm("Deseja enviar o E-mail?");
if (email == true) {
    alert("E-mail Enviado com Sucesso.");
}
else {
    alert("E-mail não Enviado.");
}
```

### Janelas de diálogo - Prompt

- Nos permite exibir uma janela pop up com dois botões (OK e Cancelar) e uma caixa de texto;
- Como funciona: se o usuário clicar em OK e preencher a caixa de texto, ela retorna o valor do texto; em Cancelar retorna null;
- O segundo parâmetro pode ser preenchido como uma sugestão;

```
var email = prompt("Digite seu e-mail", "");  
alert("O email " + email + " será usado para spam.");
```

### Janelas de diálogo - Prompt

- O que lemos da janela prompt é uma string;
- Podemos converter strings para inteiro utilizando as **funções** pré-definida **parseInt** e **parseFloat**
  - **parseInt(valor, base)**: converte uma string para inteiro. O valor será convertido para inteiro e base é o número da base (vamos usar base 10);
  - **parseFloat(valor)**: converte uma string para um valor real/ponto flutuante;

### Janelas de diálogo - Prompt

```
var notaStr = prompt("Qual a sua nota?","");  
var trabStr = prompt("Qual o valor do trabalho?","");  
var nota = parseFloat(notaStr,10);  
var trab = parseFloat(trabStr,10);  
nota = nota + trab;  
  
alert("Sua nota é: " + nota );
```



# Duvidas?

- Elabore scripts usando a função prompt que:
  - Leia um valor e imprima os resultados: “É maior que 10” ou “Não é maior que 10” ou ainda “É igual a 10”
  - Some dois valores lidos e imprima o resultado;
  - Leia 2 valores e a operação a ser realizada (+, -, \* ou /) e imprima o resultado;

- Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica;
- Uma função JavaScript é executada quando "algo" invoca (chama isso).

exemplo8.html ●

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5      <h2>JavaScript Functions</h2>
6
7      <p>Exemplo de chamada de uma função para calcular a multiplicação:</p>
8
9      <p id="demo"></p>
10
11     <script>
12         function myFunction(p1, p2) {
13             return p1 * p2;
14         }
15         document.getElementById("demo").innerHTML = myFunction(4, 3);
16     </script>
17
18 </body>
19 </html>
```

- São ações que podem ser detectadas por um Script;
- São disparos que chamam as funções dentro de seus Scripts;
- O tratamento desses eventos podem ser a chamada de funções do Script;
- Exemplo:
  - O clique que o usuário dá em um botão (**onclick**);
  - Outro tipo de evento é **onload**, que dispara quando a pagina acaba de ser carregada.

### onClick

- A ação é capturada quando o usuário clica em qualquer elemento html;

## onClick

exemplo9.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf8">
5      <title>onclick</title>
6      <script >
7          function mensagem(){
8              window.alert("Você clicou no botão!");
9          }
10     </script>
11 </head>
12 <body>
13     <form>
14         <input type="button" value="Clique Aqui" onclick="mensagem()">
15     </form>
16 </body>
17 </html>
```

## onClick

exemplo9.1.html x

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p>Clique no botão para chamar uma função que emitirá "Hello World"
5      em um elemento p com id="demo".</p>
6      <button onclick="myFunction()">Click me</button>
7      <p id="demo"></p>
8      <script>
9          function myFunction() {
10              document.getElementById("demo").innerHTML = "Hello World";
11          }
12      </script>
13  </body>
14  </html>
```



### onChange

- Ocorre quando o valor de um elemento foi alterado;
- Para botões de rádio e caixas de seleção, o evento onchange ocorre quando o estado verificado foi alterado;

## onChange

exemplo10.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head> ...
5  </head>
6  <body>
7  <p>Selecione um carro da lista.</p>
8
9  <select id="mySelect" onchange="myFunction()">
10    <option value="Audi">Audi
11    <option value="BMW">BMW
12    <option value="Mercedes">Mercedes
13    <option value="Volvo">Volvo
14  </select>
15  <p>Quando você selecionar um novo carro, a função é ativada, imprimindo o carro selecionado.</p>
16  <p id="demo"></p>
17  <script>
18  function myFunction() {
19    var x = document.getElementById("mySelect").value;
20    document.getElementById("demo").innerHTML = "You selected: " + x;
21  }
22  </script>
23  </body>
```

### Onmouseover / onmouseout

- O evento **onmouseover** ocorre quando o ponteiro do mouse é movido para um elemento ou para um de seus filhos;
- O evento **onmouseout** ocorre quando o ponteiro do mouse é movido para fora de um elemento ou fora de um dos seus filhos;

## Onmouseover / onmouseout

exemplo11.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head> ...
4  </head>
5  <body>
6  
7  <p>The function bigImg() é acionado quando o usuário move o ponteiro do mouse sobre a imagem.</p>
8  <p>The function normalImg() é acionado quando o ponteiro do mouse é movido para fora da imagem.</p>
9
10
11  <script>
12  function bigImg(x) {
13      x.style.height = "64px";
14      x.style.width = "64px";
15  }
16
17  function normalImg(x) {
18      x.style.height = "32px";
19      x.style.width = "32px";
20  }
21  </script>
22
23  </body>
24  </html>
```

### onkeydown

- Ocorre quando o usuário pressiona uma tecla (no teclado);

## onkeydown

exemplo12.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf8">
5  </head>
6  <body>
7      <p>Uma função é desencadeada quando o usuário pressiona uma tecla no campo de entrada.</p>
8      <input type="text" onkeydown="myFunction()">
9      <script>
10         function myFunction() {
11             alert("Você pressionou o teclado dentro do campo de entrada");
12         }
13     </script>
14 </body>
15 </html>
```

### onload

- Ocorre quando um objeto foi carregado;
- É mais frequentemente usado no elemento `<body>` para executar um script uma vez que uma página da Web carregou completamente todo o conteúdo (incluindo imagens, arquivos de script, arquivos CSS, etc.);
- O evento **onload** pode ser usado para verificar o tipo de navegador e a versão do navegador e carregar a versão correta da página da web com base nas informações;

## onload

exemplo13.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf8">
5  </head>
6  <body onload="myFunction()">
7
8      <h1>Hello World!</h1>
9
10     <script>
11         function myFunction() {
12             alert("Página esta carregada.");
13         }
14     </script>
15
16 </body>
17 </html>
```



### onSubmit

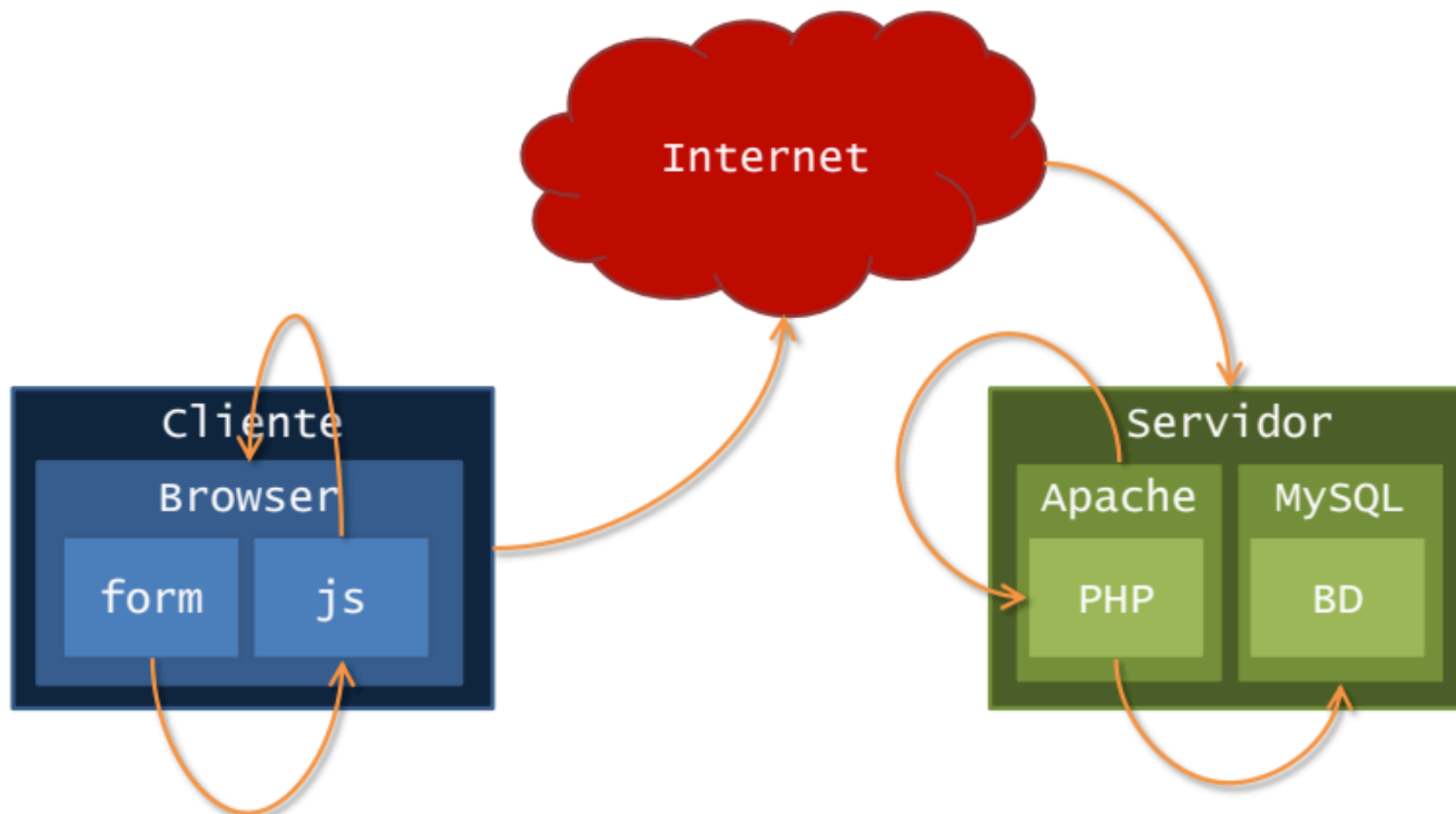
- Ocorre quando um formulário é enviado;

## onSubmit

exemplo14.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf8">
5  </head>
6  <body>
7      <p>Quando você envia o formulário, é ativada uma função que alerta algum texto.</p>
8      <form action="p" onSubmit="myFunction()">
9          Enter name: <input type="text" name="fname">
10         <input type="submit" value="Submit">
11     </form>
12     <script>
13         function myFunction() {
14             alert("The form was submitted");
15         }
16     </script>
17
18 </body>
19 </html>
```

# Formulário



## HTML

- Ao desenvolver aplicativos para internet, dados serão informados pelo usuário;
- Antes de enviar estes dados ao servidor, é possível validar/verificar se eles tem coerência em relação ao que é solicitado:
  - O usuário esqueceu campos em branco?
  - O e-mail digitado é válido?
  - A data digitada é válida?
  - Num campo numérico, foi digitado um número?

# Formulário

exemplo15.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head> ...
5  </head>
6  <script type="text/javascript" src="js/exemplo15.js"></script>
7  <body>
8      <form name="meuForm" action="xt_exemplo15.php" onsubmit="return validarForm();" method="post">
9          Nome: <input type="text" name="nome">
10         e-mail: <input type="text" name="email">
11         <input type="submit" value="Enviar">
12     </form>
13     <p id="valido">Preencha o formulário e clique em Enviar</p>
14 </body>
15 </html>
```

# Formulário

exemplo15.js ✕

```
1  function validarForm() {
2      var val = document.getElementById("valido");
3      try {
4          var nome = document.forms["meuForm"]["nome"].value;
5          if (nome == null || nome == "") {
6              throw "O Nome deve ser preenchido!";
7          }
8          var email = document.forms["meuForm"]["email"].value;
9          var atpos = email.indexOf("@");
10         var dotpos = email.lastIndexOf(".");
11         if (atpos < 1 || dotpos < atpos + 2 || dotpos + 2 >= email.length){
12             throw "Digite um e-mail válido!";
13         }
14         return true;
15     } catch (err) {
16         val.style.color = "#FF0000";
17         val.innerHTML = "Erro: " + err;
18         return false;
19     } //Fim catch
20 } //Fim function
--
```

### Adicionando

- É possível adicionar novos elementos HTML;
- Qualquer tipo de elemento, definindo qualquer propriedade;
- Tudo através do JavaScript;

## Adicionando

exemplo16.html x

```
1  <!DOCTYPE html>
2  <html>
3  ⊕ <head> ...
5  </head>
6  <body id="corpo">
7      <h1>Adicionar Elementos</h1>
8      <p>Digite o texto: <input type="text" id="texto"></p>
9      <p><button onclick="adicionar()">Adicionar</button></p>
10     <script>
11         function adicionar() {
12             var texto = document.getElementById("texto").value;
13             var para = document.createElement("p");
14             para.innerHTML = texto;
15             var corpo = document.getElementById("corpo");
16             corpo.appendChild(para); // Adiciona um nó ao final da lista de filhos de um nó pai especificado
17         }
18     </script>
19 </body>
20 </html>
21
```



### Removendo

- É possível remover elementos HTML;
- Qualquer tipo de elemento, com a condição de que conheçamos também o seu pai;
- Tudo através do JavaScript;

## Removendo

exemplo17.html x

```
1  <!DOCTYPE html>
2  <html>
3  <head> ...
5  </head>
6  <body id="corpo">
7      <h1>Remover Elemento</h1>
8      <p><button onclick="remover()">Remover</button></p>
9      <p id="texto">Texto que será removido...</p>
10     <script>
11         function remover() {
12             var pai = document.getElementById("corpo");
13             var filho = document.getElementById("texto");
14             pai.removeChild(filho);
15         }
16     </script>
17 </body>
18 </html>
```

# Bibliografia



- SILVA, M. S. Construindo sites com CSS e (X)HTML. São Paulo: Novatec, 2007
- FREEMAN, E.; FREEMAN, E. Use a cabeça! HTML com CSS e XHTML. Rio de Janeiro: Alta Books, 2006.
- CARRION, W. Design para web designers: princípios do design para web. Rio de Janeiro: Alta Books, 2006
- DEITEL, Harvey M.; DEITEL, Paul J. Java: como programar. 8. ed. São Paulo: Pearson/Prentice-Hall, 2010.
- FUGIERI, Sérgio. Java 2 – Ensino Didático. 6ª Edição. São Paulo: Érica, 2006.
- PAMPLONA, Vitor Fernando. Tutorial Java: o que é Java? Disponível em: <http://javafree.uol.com.br/artigo/871498/>. Acesso em: 22 out. 2017.
- SILVA, M. S. Web Design Responsivo. São Paulo: Novatec, 2014
- Universidade XTI. Curso de Java SE Completo - <https://www.youtube.com/channel/UCWqFtO6PsCsMuhK1XoqpWtA> Acesso em: 22 out. 2017.
- <http://www.w3c.br/Cursos/CursoHTML5>
- <https://www.w3schools.com/>