



= JavaScript =

Profa. Me. Patrícia Raia Nogueira Cavoto
patricia.nogueira@puc-campinas.edu.br



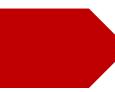
Agenda

- Visão Geral
- O que é JavaScript?
- Conceitos Básicos



Visão Geral

- JavaScript surgiu há muito tempo atrás e não funcionava em todos os browsers.
- Infelizmente carrega um pouco dessa má fama até hoje (embora já não exista mais este problema, pois, pode ser executado por todos os navegadores modernos).
- JavaScript serve pra fazer desde pequenas interações com o usuário até softwares complexos (com a utilização de bibliotecas e frameworks como: jquery, node, react, angular...).



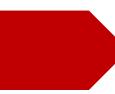
O que é JavaScript?

- JavaScript é uma linguagem de script, ou seja, de programação leve.
- Não é compilada.
- Pode ser inserido em páginas de HTML.
- O browser processa (assim como as próprias tags HTML).

O que é JavaScript?

- JavaScript é uma linguagem de script, ou seja, de programação leve.
- Não é compilada.
- Pode ser inserido em páginas de HTML.
- O browser processa (assim como as próprias tags HTML).

JavaScript
NÃO é Java!!!!



O que é JavaScript?

- Você pode incluir seu código JavaScript na sua página HTML das seguintes formas:

1. Diretamente na sua página HTML:

```
<script>  
    //seu código JavaScript aqui!  
</script>
```

O que é JavaScript?

2. Criando um arquivo .js e importando-o na sua página HTML:

```
<head>
    <script type="text/javascript" src="arquivo.js"></script>
</head>
```

Conceitos Básicos

- Instruções JavaScript são "comandos" para o navegador.
- O propósito das declarações é dizer ao navegador o que fazer.
- Cada instrução é executada pelo navegador na sequência em que foi escrita.



Conceitos Básicos

Estrutura da Linguagem:

- Programar em JavaScript é muito semelhante a programar em C ou Java.
- A estrutura da linguagem segue muitos dos padrões definidos para essas linguagens.

Conceitos Básicos

Ponto e vírgula:

- Ponto e vírgula separa instruções JavaScript.
- Normalmente você adicionar um ponto e vírgula no final de cada instrução executável.

Conceitos Básicos

Case Sensitive:

- JavaScript é case sensitive.
- Assim, a variável **minhaVariavel** não é a mesma que **MinhaVariavel**.
- Utiliza o padrão camel case:
 - Variáveis, atributos, métodos, funções começam sempre com uma letra minúscula. Se houver mais de uma palavra no nome, ela inicia com letra maiúscula. Exemplo: getElementById

Conceitos Básicos

Comentários:

- Comentários em JavaScript funcionam exatamente como em C/Java:
 - // comenta uma linha
 - /* */ permite comentar um bloco de linhas.

Conceitos Básicos

- A função alert() é utilizada para exibir uma janela de alerta com uma mensagem para o usuário.

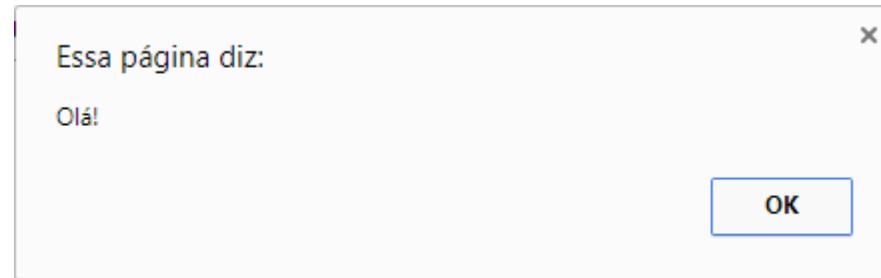
```
<script>
    alert('Mensagem a ser exibida!');
</script>
```



Conceitos Básicos

- A função alert() é utilizada para exibir uma janela de alerta com uma mensagem para o usuário.

```
<script>
    alert('Mensagem a ser exibida!');
</script>
```



Conceitos Básicos

`document.getElementById`

- Podemos utilizar JavaScript para manipular elementos HTML.
- Para acessar um elemento HTML diretamente do código JavaScript você pode usar:

```
document.getElementById('id do elemento')
```

Conceitos Básicos

innerHTML

- Permite que você inclua conteúdo em um elemento HTML via JS.
- Precisa ser utilizado junto com getElementById:

```
document.getElementById('id').innerHTML = 'Conteúdo!';
```

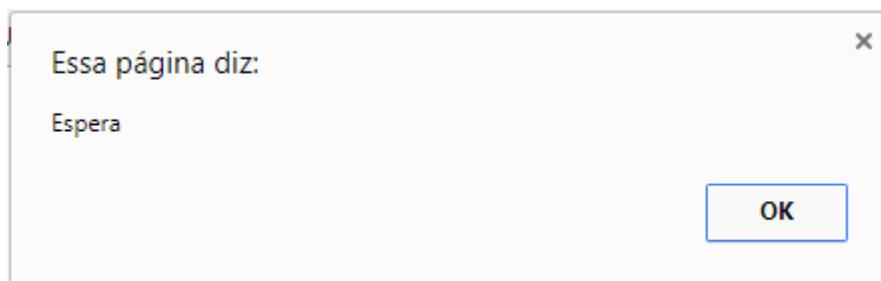
Conceitos Básicos

```
<body>
    <h1>Minha Página Web</h1>
    <p id="paragrafo">Meu parágrafo</p>
    <script>
        alert('Espera');
        document.getElementById('paragrafo').innerHTML = 'Teste JavaScript';
    </script>
</body>
```



Conceitos Básicos

```
<body>
    <h1>Minha Página Web</h1>
    <p id="paragrafo">Meu parágrafo</p>
    <script>
        alert('Espera');
        document.getElementById('paragrafo').innerHTML = 'Teste JavaScript';
    </script>
</body>
```



Conceitos Básicos

```
<body>
    <h1>Minha Página Web</h1>
    <p id="paragrafo">Meu parágrafo</p>
    <script>
        alert('Espera');
        document.getElementById('paragrafo').innerHTML = 'Teste JavaScript';
    </script>
</body>
```

Minha Página Web

Teste JavaScript



exemplo4.htm

Conceitos Básicos

Variáveis:

- Palavras reservadas do JS não podem ser utilizadas como nome de variáveis. Exemplos: break, else, new, var...
- Para declarar uma variável em JS devemos usar:

```
var nomeVariavel;
```

Conceitos Básicos

Tipos de Dados:

- Note que não indicamos o tipo da variável, apenas a declaramos:

```
var pi=3.14;  
var pessoa="João da Silva";  
var resposta='Sim!!!!';
```

Conceitos Básicos

Tipos de Dados:

- Quando você atribui um valor de texto a uma variável, pode utilizar aspas simples ou duplas, porém, lembre-se sempre de manter um padrão.
- Quando você atribui um valor numérico a uma variável, não deve colocar aspas. Se você colocar aspas em torno de um valor numérico, ele será tratado como texto.



Conceitos Básicos: Variáveis Nulas e Indefinidas

- Uma variável null é uma variável que você definiu e atribuiu valor null:

```
var nome = null;
```

- Uma variável é indefinida quando você a declara mas não atribui nenhum valor:

```
var nome;
```

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- Por exemplo: o que deveria ser impresso no código abaixo:

```
function imprimeIdade() {  
    var idade = 30;  
    console.log('Minha idade é:', idade);  
}  
imprimeIdade();
```

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- Por exemplo: o que deveria ser impresso no código abaixo:

```
function imprimeIdade() {  
    var idade = 30;  
    console.log('Minha idade é:', idade);  
}  
imprimeIdade();
```

Minha idade é: 30

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E agora?

```
function imprimeIdade() {  
    var idade = 30;  
}  
imprimeIdade();  
console.log('Minha idade é:', idade)
```

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E agora? Até aqui, tudo ok.

```
function imprimeIdade() {  
    var idade = 30;  
}  
imprimeIdade();  
console.log('Minha idade é:', idade)
```

```
console.log('Minha idade é:', ^  
idade)
```

ReferenceError: idade is not defined

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E agora?

```
function imprimeIdade() {  
    console.log('Minha idade é:', idade);  
    var idade = 30;  
}  
imprimeIdade();
```

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E agora?

```
function imprimeIdade() {  
    console.log('Minha idade é:', idade);  
    var idade = 30;  
}  
imprimeIdade();
```

Minha idade é: undefined

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E esse código?

```
function imprimeIdade() {  
    for (var idade = 30; idade <= 40; idade++) {  
        console.log('Idade dentro do for:', idade)  
    }  
    console.log('Idade fora do for:', idade)  
}  
imprimeIdade()
```

Conceitos Básicos: var e let

- A utilização de declarações utilizando var tem um problema de escopo.
- E esse código?

```
function imprimeIdade() {  
    for (var idade = 30; idade <= 40; idade++) {  
        console.log('Idade dentro do for:', idade)  
    }  
    console.log('Idade fora do for:', idade)  
}  
imprimeIdade()
```

```
Idade dentro do for: 30  
Idade dentro do for: 31  
Idade dentro do for: 32  
Idade dentro do for: 33  
Idade dentro do for: 34  
Idade dentro do for: 35  
Idade dentro do for: 36  
Idade dentro do for: 37  
Idade dentro do for: 38  
Idade dentro do for: 39  
Idade dentro do for: 40  
Idade fora do for: 41
```

Conceitos Básicos: var e let

- A solução é a utilização da palavra reservada `let` para limitar o escopo e permitir que o código funcione conforme o esperado.
- Desconsiderando o escopo, o restante do comportamento do `let` é igual ao comportamento do `var`.

```
let variavel;
```

Conceitos Básicos: Constantes

- Para definir um valor constante para um identificador usamos a palavra-chave const.
- Constantes são definidas com letras maiúsculas:

```
const PI = 3.141516;
```

Conceitos Básicos

```
<body>
  <script>
    const PI = 3.14;
    var valorReal = 1.78;
    var nome = "João da Silva";
    var resposta = 'Sim!';
    var x = 1, y = 2, z = x + y;
    alert('Valor da constante PI: ' + PI);
    alert('Valor de valorReal: ' + valorReal);
    alert('Valor de nome: ' + nome);
    alert('Valor de resposta: ' + resposta);
    alert('Valor de z: ' + z);
  </script>
</body>
```



Conversão de tipos

- Conversão de strings para números:

```
var idade = '23';
```

```
var idadeNumerica = parseInt(idade);
```

```
var peso = '67.65';
```

```
var pesoNumerico = parseFloat(peso);
```

Conversão de tipos

- Conversão de números para strings:

```
var idade = 23;
```

```
var idadeStr = String(idade);
```

```
var peso = 67.65;
```

```
var pesoStr = String(peso);
```

Tipos Primitivos de Dados Como Objetos

- Os tipos primitivos (int, float, char, etc) podem ser tratados como objetos em JavaScript. Desta forma, podemos acessar métodos destes objetos utilizando o operando “.” (ponto). Exemplo:

```
var meuNome = 'Joao Silva';  
alert(meuNome.length);
```

- Qual é o resultado na tela?

Tipos Primitivos de Dados Como Objetos

- Os tipos primitivos (int, float, char, etc) podem ser tratados como objetos em JavaScript. Desta forma, podemos acessar métodos destes objetos utilizando o operando “.” (ponto). Exemplo:

```
var meuNome = 'Joao Silva';  
alert(meuNome.length);
```

- Qual é o resultado na tela?

Objeto String

- Para criar um objeto String:

```
var nome = new String('João Silva');
```

Objetos

- Se for necessário acessar frequentemente os “métodos” e propriedades de uma variável, podemos criar a variável como um objeto:

```
var meuNome = new String('Patrícia');
```

- Se não for necessário, podemos criá-la como um tipo primitivo:

```
var meuNome = 'Patrícia';
```

Métodos do Objeto String

Método	Descrição	Argumentos
valueOf	Retorna a string literal que o objeto String armazena	Nenhum
length	O tamanho da string	Use sem parênteses
anchor	Cria uma âncora HTML	String com o título da âncora
big, blink, bold, italics, small, strike, sub, sup	Formatam e retornam o valor literal do objeto String como HTML	Nenhum
charAt, charCodeAt	Retornam um caracter (charAt) ou código do caractere inteiro (charCodeAt) em uma determinada posição.	Um número representa a posição. Inicia em 0;
indexOf	Retorna a posição inicial da primeira ocorrência de um substring	substring a ser procurada

Métodos do Objeto String

Método	Descrição	Argumentos
lastIndexOf	Retorna a posição final da primeira ocorrência de um substring	substring a ser procurada
link	Retorna HTML para o link	URL do atributo href
concat	Concatena strings	As strings a serem concatenadas
split	Divide uma string em partes com base em um separador	Separador e número máximo de divisões.
slice	Retorna o trecho de uma string	As posições inicial e final do trecho
substring, substr	Retorna uma substring	O início e o final da string
match, replace, search	Associa, substitue e pesquisa expressões	String com a expressão
toLowerCase, toUpperCase	Converte letras para maiúsculas ou minúsculas	Nenhum

Exemplo Objeto String

```
<p id="resultado"></p>
<script>
    var frase = String('Programação Web na PUC-Campinas');
    var resultado = frase + '<br>';
    resultado += frase.big() + '<br>';
    resultado += frase.blink() + '<br>';
    resultado += frase.sup() + '<br>';
    resultado += frase.sub() + '<br>';
    resultado += frase.strike() + '<br>';
    resultado += frase.bold() + '<br>';
    resultado += frase.italics() + '<br>';
    resultado += frase.small() + '<br>';
    resultado += frase.link('http://www.puc-campinas.edu.br');
    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Exemplo Objeto String

```
<p id="resultado"></p>
<script>
    var frase = String('Programação Web na PUC-Campinas');
    var resultado = frase + '<br>';
    resultado += frase.big() + '<br>';
    resultado += frase.blink() + '<br>';
    resultado += frase.sup() + '<br>';
    resultado += frase.sub() + '<br>';
    resultado += frase.strike() + '<br>';
    resultado += frase.bold() + '<br>';
    resultado += frase.italics() + '<br>';
    resultado += frase.small() + '<br>';
    resultado += frase.link('http://www.puc-campinas.edu.br');
    document.getElementById('resultado').innerHTML = resultado;
</script>
```

Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas

Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas
Programação Web na PUC-Campinas



Exemplos

```
var frase = String('Programação Web na PUC-Campinas');
document.getElementById('resultado').innerHTML = frase.charAt(3);
var codigo = frase.charCodeAt(3);
document.getElementById('resultado').innerHTML = codigo;
```

➤ Qual será o resultado?

Exemplos

```
var frase = String('Programação Web na PUC-Campinas');
document.getElementById('resultado').innerHTML = frase.charAt(3);
var codigo = frase.charCodeAt(3);
document.getElementById('resultado').innerHTML = codigo;
```

➤ Qual será o resultado?

g

103

Exemplos

```
var text = 'Isto é uma string de teste.';  
  
var subStringCom8 = text.subStr(5,8);          //retorna é uma st  
var subStringCom3 = text.substring(5,8);        //retorna é um  
var iVal = text.indexOf('t'); //retorna 2 (posição do primeiro 't')  
var value = text.valueOf(); //retorna o conteúdo da string
```

Objeto Boolean

- Quando criamos uma instância de um objeto tipo boolean seu valor inicial é false.

```
var desejaSair = new Boolean(false);
```

Objeto Boolean

`toString()`: converte o conteúdo booleano em string:

```
var desejaSairString = desejaSair.toString();
```

```
//copia o valor da variável desejaSair(Boolean) para a variável  
desejaSairString como String.
```

Objeto *Number*

- Para declarar um tipo NUMBER:

```
var numero = new Number(2.0);
```

Objeto *Number*

- `number.MAX_VALUE`: Retorna o maior número que poder ser representado em JS.
- `number.MIN_VALUE`: Retorna o menor número que poder ser representado em JS.
- Além disso os métodos abaixo também valem para objetos Number:
 - `valueOf()`
 - `toString()`

Objeto Number

Outros Métodos:

- `toExponential`: Retorna uma string representando o número usando notação exponencial.
- `toFixed`: Retorna uma string representando o número usando notação de ponto fixo.
- `toPrecision`: Retorna uma string representando o número usando uma determinada precisão.

Exemplo Objeto Number

```
<p id="resultado"></p>
<script>
    var numero = new Number(34.8896);
    var resultado = 'Valor inicial: ' + numero + '<br>';
    resultado += 'Exponencial(3): ' + numero.toExponential(3) + '<br>';
    resultado += 'Algarismos(3):' + numero.toPrecision(3) + '<br>'; // 3 algarismos.
    resultado += 'Precisão(6):' + numero.toFixed(6); //6 casas decimais.
    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Exemplo Objeto Number

```
<p id="resultado"></p>
<script>
    var numero = new Number(34.8896);
    var resultado = 'Valor inicial: ' + numero + '<br>';
    resultado += 'Exponencial(3): ' + numero.toExponential(3) + '<br>';
    resultado += 'Algarismos(3):' + numero.toPrecision(3) + '<br>'; // 3 algarismos.
    resultado += 'Precisão(6):' + numero.toFixed(6); //6 casas decimais.
    document.getElementById('resultado').innerHTML = resultado;
</script>
```

Valor inicial: 34.8896
Exponencial(3): 3.489e+1
Algarismos(3):34.9
Precisão(6):34.889600



Objeto *Date*

- Gera uma variável com a data e hora atual:

```
var dataAtual = new Date();
```

- Você também pode criar uma data pré-definida:

```
var data = new Date('March 12, 1980 12:20:25');
```

```
var data = new Date('March 12, 1980');
```

Objeto *Date*

- Você também pode gerar a data a partir do ano, mês e dia:

```
var data = new Date(1973, 08, 20);
```



Métodos do Objeto *Date*

- GetHours: obtém as horas.
- GetMinutes: obtém os minutos.
- GetSeconds: obtém os segundos.
- GetMilliseconds: obtém os milisegundos.

Métodos do Objeto *Date*

- `GetFullYear`: obtém o ano com 4 dígitos.
- `GetMonth`: obtém o mês na forma de um número entre 0 e 11.
- `GetDate`: obtém o dia do mês.
- `GetDay`: obtém um número representando o dia da semana, começando em 0 para domingo e terminando em 6 para sábado.



Métodos do Objeto *Date*

- Todos os métodos anteriores tem o método set que altera o valor ao invés de pegar o valor.
- O único método set que não tem método get equivalente é o getDay.

Métodos do Objeto *Date*

Métodos convertem a data para uma string formatada:

- **toString**: Resulta em uma string com horário local.
- **toGMTString**: Resulta em uma string usando padrões GMT.
- **toLocaleDateString** e **toLocaleTimeString**: Resultam na data e no horário, respectivamente, usando o local.
- **toLocaleString**: Converte a string usando o local corrente.

Exemplo Objeto Date

```
<p id="resultado"></p>
<script>
    var data = new Date(); //obtém data da máquina
    var resultado = 'Data obtida: ' + data.toString() + '<br>';
    resultado += 'Hora: ' + data.getHours() + '<br>';
    resultado += 'Dia: ' + data.getDate() + '<br>';
    resultado += 'Ano: ' + data.getFullYear() + '<br>';
    resultado += 'Dia da Semana: ' + data.getDay() + '<br>';
    resultado += 'GMT: ' + data.toGMTString() + '<br>';
    resultado += 'Local Data: ' + data.toLocaleDateString() + '<br>';
    resultado += 'Local Hora: ' + data.toLocaleTimeString() + '<br>';
    resultado += 'Padrão Local: ' + data.toLocaleString();
    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Exemplo Objeto Date

```
<p id="resultado"></p>
<script>
    var data = new Date(); //obtém data da máquina
    var resultado = 'Data obtida: ' + data.toString() + '<br>';
    resultado += 'Hora: ' + data.getHours() + '<br>';
    resultado += 'Dia: ' + data.getDate() + '<br>';
    resultado += 'Ano: ' + data.getFullYear() + '<br>';
    resultado += 'Dia da Semana: ' + data.getDay() + '<br>';
    resultado += 'GMT: ' + data.toGMTString() + '<br>';
    resultado += 'Local Data: ' + data.toLocaleDateString() + '<br>';
    resultado += 'Local Hora: ' + data.toLocaleTimeString() + '<br>';
    resultado += 'Padrão Local: ' + data.toLocaleString();
    document.getElementById('resultado').innerHTML = resultado;
</script>
```

Data obtida: Thu Aug 24 2017 18:41:05 GMT-0300 (Hora oficial do Brasil)
Hora: 18
Dia: 4
Ano: 2017
Dia da Semana: 4
GMT: Thu, 24 Aug 2017 21:41:05 GMT
Local Data: 24/08/2017
Local Hora: 18:41:05
Padrão Local: 24/08/2017 18:41:05



Conceitos Básicos

Funções:

```
function nomeFuncao(var1, var2) {  
    //comandos da função  
    return valorRetorno;  
    //return é opcional  
}
```

*note que não há necessidade de indicar o tipo da função, nem dos parâmetros e que o comando return é opcional.

Conceitos Básicos

onClick

- Permite que você execute uma função JavaScript quando o usuário clicar sobre um elemento HTML.

Conceitos Básicos

document.write

- Método utilizado para escrever elementos HTML via javascript.

```
<body>
  <h1>Minha Página Web</h1>
  <script>
    document.write("<p>Tag HTML via
                  JavaScript!</p>");
  </script>
</body>
```



Conceitos Básicos

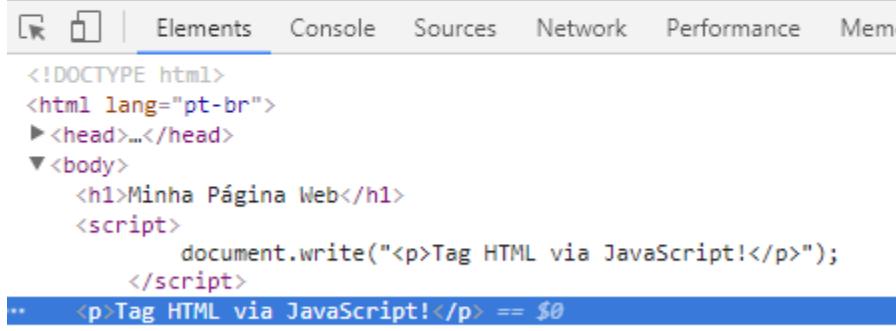
document.write

- Método utilizado para escrever elementos HTML via javascript.

```
<body>
  <h1>Minha Página Web</h1>
  <script>
    document.write("<p>Tag HTML via
                  JavaScript!</p>");
  </script>
</body>
```

Minha Página Web

Tag HTML via JavaScript!



```
<!DOCTYPE html>
<html lang="pt-br">
  <head>...</head>
  <body>
    <h1>Minha Página Web</h1>
    <script>
      document.write("<p>Tag HTML via JavaScript!</p>");
    </script>
    .. <p>Tag HTML via JavaScript!</p> == $0
  </body>
</html>
```



ATENÇÃO!

- Use `document.write()` para escrever no HTML enquanto a página está sendo carregada.
- Se você usá-lo após o documento ter sido completamente carregado, toda a página HTML será sobrescrita.
- Para você entender melhor, vamos verificar o código do exemplo2.htm exibido no início da aula.

Conceitos Básicos

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>JavaScript</title>
    <script>
        function funcaoLimpa() {
            document.write("... limpei o documento!");
        }
    </script>
</head>
<body>
    <h1>Minha página web</h1>
    <p>Parágrafo</p>
    <button onClick="funcaoLimpa () ">Teste</button>
</body>
</html>
```



exemplo2.htm

Conceitos Básicos

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>JavaScript</title>
    <script>
        function funcaoLimpa() {
            document.write("... limpei o documento!");
        }
    </script>
</head>
<body>
    <h1>Minha página web</h1>
    <p>Parágrafo</p>
    <button onClick="funcaoLimpa () ">Teste</button>
</body>
</html>
```

Minha página web

Parágrafo

Teste



exemplo2.htm

Conceitos Básicos

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>JavaScript</title>
    <script>
        function funcaoLimpa() {
            document.write("... limpei o documento!");
        }
    </script>
</head>
<body>
    <h1>Minha página web</h1>
    <p>Parágrafo</p>
    <button onClick="funcaoLimpa () ">Teste</button>
</body>
</html>
```

Minha página web

Parágrafo

Teste

... limpei o documento!



exemplo2.htm

Conceitos Básicos

Comando Condicional (if...else):

```
if (condicao) {  
    //comandos se true  
} else {  
    //comandos se false  
}
```

Conceitos Básicos

Operadores de Comparações:

➤ Para os exemplos, suponha $x=5$

Operador	Descrição	Exemplo
<code>==</code>	Igual	$(x == 8) \rightarrow \text{false}$
<code>!=</code>	Diferente	$(x != 3) \rightarrow \text{true}$
<code>></code>	Maior	$(x > 5) \rightarrow \text{false}$
<code><</code>	Menor	$(x < 5) \rightarrow \text{false}$
<code>>=</code>	Maior ou igual	$(x >= 5) \rightarrow \text{true}$
<code><=</code>	Menor ou igual	$(x <= 5) \rightarrow \text{true}$

Conceitos Básicos

Operadores Lógicos:

- Para os exemplos, suponha **x=6** e **y=3**.

Operador	Descrição	Exemplo
<code>&&</code>	and	$(x < 10 \&\& y > 1) \rightarrow \text{true}$
<code> </code>	or	$(x==5 \mid\mid y==5) \rightarrow \text{false}$
<code>!</code>	not	$!(x==y) \rightarrow \text{true}$

Conceitos Básicos

```
<script>
    function changeImage() {
        element = document.getElementById('lampada');
        if (element.src.match('bulbon')) {
            element.src = 'pic_bulboff.gif';
        }
        else {
            element.src = 'pic_bulbon.gif';
        }
    }
</script>
<body>
    
    <p>Clique na lâmpada para acender ou apagar a luz!</p>
</body>
```



Conceitos Básicos

```
<script>
    function changeImage() {
        element = document.getElementById('lampada');
        if (element.src.match('bulbon')) {
            element.src = 'pic_bulboff.gif';
        }
        else {
            element.src = 'pic_bulbon.gif';
        }
    }
</script>
<body>
    
    <p>Clique na lâmpada para acender ou apagar a luz!</p>
</body>
```



Clique na lâmpada para acender ou apagar a luz!



Conceitos Básicos

```
<script>
    function changeImage() {
        element = document.getElementById('lampada');
        if (element.src.match('bulbon')) {
            element.src = 'pic_bulboff.gif';
        }
        else {
            element.src = 'pic_bulbon.gif';
        }
    }
</script>
<body>
    
    <p>Clique na lâmpada para acender ou apagar a luz!</p>
</body>
```



Clique na lâmpada para acender ou apagar a luz!



Conceitos Básicos

Comando de Seleção (switch):

```
switch(variavel) {  
    case valor1: //comandos  
        break;  
    case valor2: //comandos  
        break;  
    default:      // comandos  
}
```

Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var data = new Date(); //obtém data da máquina
    var resultado;
    var diaSemana = data.getDay();
    switch(diaSemana){
        case 0: resultado = 'Domingo'; break;
        case 1: resultado = 'Segunda'; break;
        case 2: resultado = 'Terça'; break;
        case 3: resultado = 'Quarta'; break;
        case 4: resultado = 'Quinta'; break;
        case 5: resultado = 'Sexta'; break;
        case 6: resultado = 'Sábado'; break;
    }
    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Conceitos Básicos

prompt

- Permite que você obtenha um valor do usuário e armazene em uma variável.

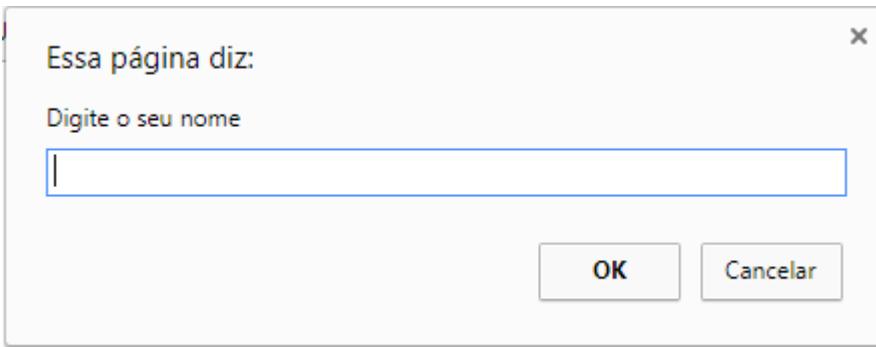
Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var nome = prompt('Digite o seu nome');
    document.getElementById('resultado').innerHTML = nome;
</script>
```



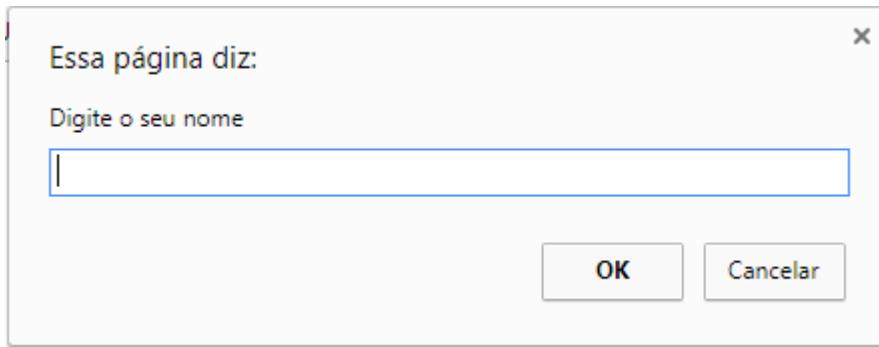
Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var nome = prompt('Digite o seu nome');
    document.getElementById('resultado').innerHTML = nome;
</script>
```



Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var nome = prompt('Digite o seu nome');
    document.getElementById('resultado').innerHTML = nome;
</script>
```



Conceitos Básicos

Comando de Repetição(for):

```
for(condicaoInicio; condicaoParada; atualizacao) {  
    //comandos  
}
```

Conceitos Básicos: Vetores

```
var vetor = new Array(10);
```

- Pode ser um vetor de 10 posições do tipo inteiro.

10	15	13	0	8	9	34	19	13	23
0	1	2	3	4	5	6	7	8	9

```
var x = new Array(3) ;
```

- Pode ser um vetor de 3 posições do tipo char.

A	@	x
0	1	2

Conceitos Básicos: Vetores

```
<script>
    //construção simples sem dimensionamento
    var meuVetor1 = new Array();

    //construção simples com dimensionamento
    var meuVetor2 = new Array(4);

    //construção inserindo valores
    var meuVetor3 = new Array("João", "Roberto", "José", "Maria");
</script>
```



Conceitos Básicos: Atribuição de Valor ao Vetor

```
var vetor = new Array(4);
```

```
vetor[0] = 1;
```

```
vetor[1] = 4;
```

```
vetor[2] = 5;
```

```
vetor[3] = 6;
```



Conceitos Básicos: Preenchendo um Vetor

```
var vetor = new Array(4);
var i;
for (i = 0; i < 4; i++)
    vetor[i] = prompt("Digite um valor:");
```

Conceitos Básicos: Exibindo um Vetor

```
for (i=0; i<10; i++)
document.write(vetor[i]+<br>);
```



Conceitos Básicos: Propriedade length do Vetor

- O objeto Array possui uma propriedade chamada `length`.
- Esta propriedade mostra quantos elementos o vetor possui.

Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var vetor = new Array(4);
    var i;
    for (i = 0; i < vetor.length; i++)
        vetor[i] = prompt("Digite um valor:");

    var resultado = '';
    for (i = 0; i < vetor.length; i++)
        resultado += vetor[i] + '<br>';

    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var text = 'Isto é uma string de teste.';
    var resultado = '';
    var palavras = text.split(' ');
    for (var i = 0; i < palavras.length; i++) {
        resultado += palavras[i] + '<br>';
    }
    document.getElementById('resultado').innerHTML = resultado;
</script>
```



Conceitos Básicos

```
<p id="resultado"></p>
<script>
    var text = 'Isto é uma string de teste.';
    var resultado = '';
    var palavras = text.split(' ');
    for (var i = 0; i < palavras.length; i++) {
        resultado += palavras[i] + '<br>';
    }
    document.getElementById('resultado').innerHTML = resultado;
</script>
```

Isto
é
uma
string
de
teste.



Conceitos Básicos

Comando de Repetição(while):

```
while (condicao) {  
    //comandos  
}
```

Conceitos Básicos

Comando de Repetição(do...while):

```
do {  
    //comandos  
} while (condicao);
```

Conceitos Básicos

Verificação de erro (try...catch):

```
try{  
    //comandos a testar  
} catch (erro) {  
    //tratamento do erro  
}
```

Conceitos Básicos

confirm

- A função `confirm()` é utilizada para exibir uma janela de alerta com as opções OK e CANCELAR para o usuário.
- É possível identificar qual das opções o usuários selecionou e realizar tratamentos específicos.

Conceitos Básicos

```
<p>Clique no botão para exibir uma janela CONFIRM.</p>
<button onClick="minhaFuncao()">Clique aqui!</button>
<p id="resultado"></p>
<script>
    function minhaFuncao() {
        var resultado;
        var opcao = confirm('Clique em um botão!');
        if (opcao == true)
            resultado = 'Você clicou em OK!';
        else
            resultado = 'Você clicou em Cancelar!';
        document.getElementById('resultado').innerHTML = resultado;
    }
</script>
```



Conceitos Básicos

Formulário com JavaScript:

- Para disparar um evento (submit/reset) de formulário:

```
document.nomeForm.evento()
```

- Para recuperar informações de um campo do formulário em JS utilizar:

```
document.nomeForm.nomeCampo.value
```

Conceitos Básicos

Formulário com JavaScript:

- O exemplo que veremos a seguir mostra a validação de um campo obrigatório via JavaScript e tem função apenas didática, uma vez que com HTML5 podemos fazer essa validação com o atributo required.

Conceitos Básicos

```
<script>
    function validaForm() {
        var nome = document.meuForm.nome.value;
        if (nome == null || nome == '') {
            alert('Nome deve ser obrigatório');
            return false;
        }
    }
</script>
<form name="meuForm">
    Nome: <input type="text" name="nome">
    <input type="button" value="Enviar" onClick="validaForm();">
</form>
```



Referências

1. W3C: <http://www.w3c.br/>
2. W3Schools JavaScript: <http://www.w3schools.com/js/default.asp>
3. Exemplos de input com atributo pattern no HTML5:
<http://html5pattern.com/>