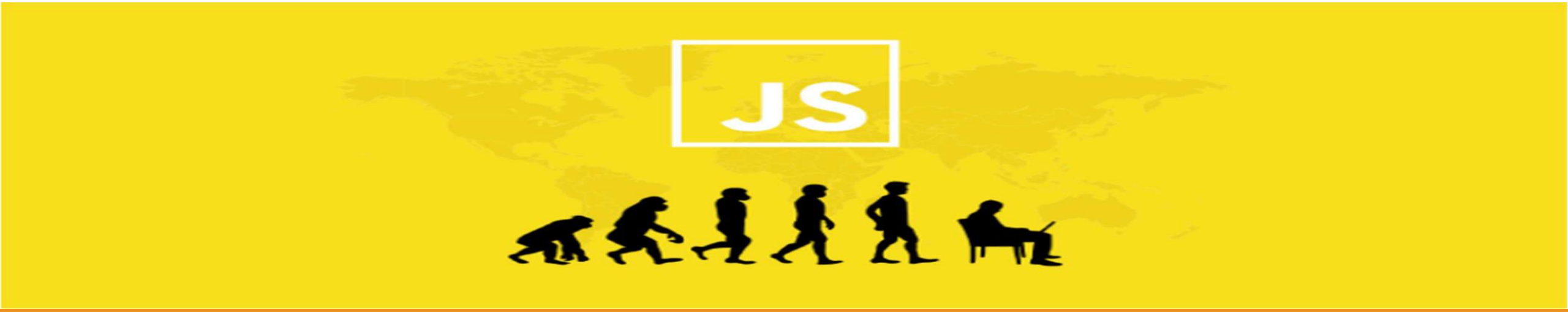
Aula 01

JavaScript - JS

O que é JS?

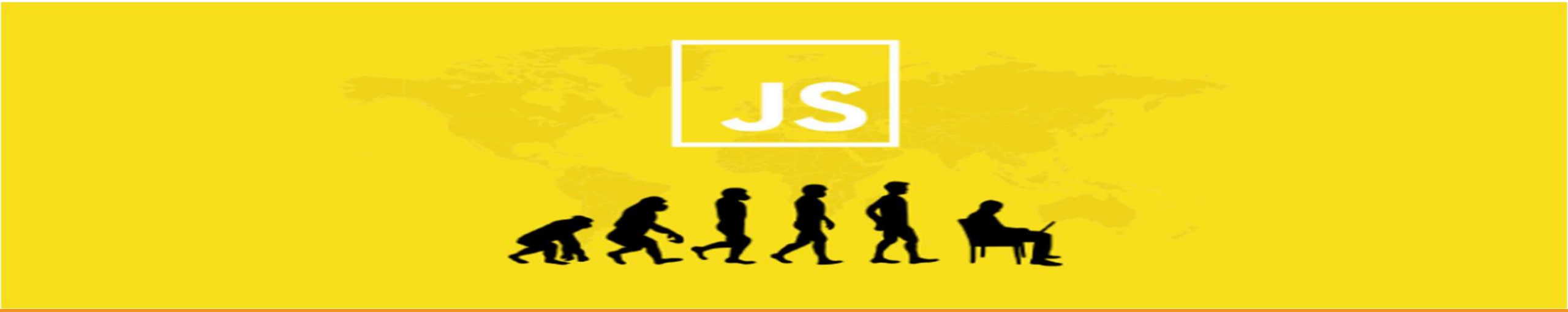
- Nasceu em 1996 pelo programador Brendan Eich.
- Linguagem de alto nível muito utilizada em páginas da web.
- Permite diversas interações com os usuários e é de fácil aplicação, compatível com todos os navegadores.
- Conhecido como linguagem de programação comportamental, criando conteúdos dinâmicos e animações, deixando as páginas mais interativas e interessantes.

Mas porque utilizar JavaScript nas páginas web e não outras linguagens?



Características do JS

- O JavaScript é uma linguagem que não precisa de um compilador, pois a interpretação é feita nos navegadores internet.
- É uma linguagem de fácil aprendizado.
- É totalmente compatível com várias plataformas e navegadores.
- É mais leve que as outras linguagens de programação.
- Os erros no JavaScript são mais fáceis de serem localizados, logo são mais fáceis de serem corrigidos.
- Com o JavaScript é possível fazer sites mais interativos e que segura a atenção do usuário.



Sintaxe Básica

- É uma linguagem case-sensitive (faz diferenciação entre letras minúsculas e maiúsculas).

- Por exemplo se uma variável se chama **Loja**, quando eu for usá-la se eu escrever **loja**, certamente vai dar erro, pois ela é case-sensitive.

- As instruções no JavaScript são chamadas de declaração

COMENTÁRIOS:

1. Os comentários em JavaScript são definidos de duas formas:
   a) a primeira é comentário de uma única linha (// comentário de uma linha);
   b) e a segunda é o comentário de múltiplas linhas (/* comentário de múltiplas linhas*/).

Sintaxe Básica

DECLARAÇÕES – Existem três tipos de declarações em JS, são elas:

1. Declaração de variáveis, onde opcionalmente, inicializando-a com um valor.
2. Declaração let, onde permite que declare várias variáveis limitando o escopo no bloco.
3. Declaração de constantes, onde possuem escopo de bloco. O valor de uma constante não pode ser alterado por uma atribuição ou ser redeclarada.

VARIAVEIS - As variáveis são como o sistema trata uma parte da memória para alocar/guardar informações, onde podem ser usadas posteriormente. Exemplo:

Preciso fazer uma conta aritmética simples e mostrar o resultado através de uma variável. Então vamos lá:

- Tenho a variável chamada **soma**, nela será armazenada a soma entre dois números, **numero1 + numero2 = soma;**
- Logo se preciso mostrar a soma dos números não preciso de um numero3, apenas eu chamo a variável **soma**.

Sintaxe Básica

As variáveis se dividem nos seguintes tipos:

- **String** – São variáveis de texto, quase sempre chamada de “cadeia de caracteres”. Os valores atribuídos a esse tipo normalmente usam aspas duplas (“ “) ou aspas simples (‘ ’).
- **Float** – Variáveis com números em casas decimais.
- **Boolean** – Tipos de variáveis com valores **true**(verdadeiro) e **false**(falso).
- **Int** – São variáveis com números inteiros.
- **Arrays** – Uma variável array faz a referência a vários espaços na memória, é um conjunto de valores organizados por um índice.

JavaScript na prática!

O JavaScript pode ser inserido na página HTML de duas formas:

INTERNO – Inserido no próprio documento em HTML utilizando as tags <script> e </script>, conforme na imagem abaixo:

```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <script type="text/javascript">
7               //código JavaScript
8           </script>
9       </head>
10
11          <body>
12
13          </body>
14  </html>
15
```

JavaScript na prática!

EXTERNO – Implantado em um documento externo (.js), em um documento separado do HTML. Deste modo devemos fazer a referência desse documento JavaScript na página HTML, conforme imagem abaixo:

```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <script type="text/javascript" src="meuArquivo.js"></script>
7       </head>
8
9           <body>
10
11          </body>
12  </html>
13
```

JavaScript na prática!

Usando as variáveis no JavaScript como vimos acima, vamos ver na prática o funcionamento da linguagem, sempre atento nas regras de declaração de variáveis.

Vamos ver um exemplo:

```
1    var nome;
2    nome = "fulano";
3    var idade = 30;
4    idade = 30 + 20 - 10*5;
```

JavaScript na prática!

O JavaScript por ser uma linguagem de programação, nos oferece métodos mais avançados do que outras linguagens.

As funções podem receber parâmetros e retornar valores, porém o tipo de retorno e o tipo dos parâmetros não precisa ser definido previamente.

```
2   function exibirMensagem()
3   {
4       alert('Olá, seja bem vindo(a)!')
5   }
```

Esse é um exemplo de uma função em JavaScript sem parâmetros e sem retorno.

JavaScript na prática!

Esse é um exemplo de uma função em JavaScript com parâmetro e com retorno.

Nota-se que para definirmos um retorno em uma função, devemos utilizar a palavra **return** seguida do valor ou expressão do resultado.

```
2    function somar(A, B)
3    {
4        return A + B;
5    }
6
```

Controle de Fluxo e Estruturas Condicionais

Estruturas de **controle de fluxo**, conhecidas também como **estruturas condicionais**, são estruturas onde se define condições em forma de **looping**, encadeando processos em que o resultado depende de ações predefinidas em uma das condições.

```
2   if(condicao1)
3   {
4       //ação se condição 1 verdadeira
5   }
6   else if(condicao2)
7   {
8       //ação se condição 2 verdadeira
9   }
10  else
11  {
12      //ação se nenhuma das condições for verdadeira
13  }
14
```

```
2   if(idade > 18)
3   {
4       alert('É maior de idade')
5   }
6   else
7   {
8       alert('É menor de idade')
9   }
10
```

Controle de Fluxo e Estruturas Condicionais

Em outros casos em que necessitamos de mais condições, usamos a estrutura **switch** para verificar o valor da variável e retornar o valor.

O comando switch faz a verificação do valor de uma variável, sendo que para cada opção executa um conjunto de ações.

Caso nenhum dos valores for verificado, os comandos do bloco default são executados.

O bloco default, porém, pode ser omitido caso não exista uma ação padrão a ser executada se nenhuma das opções for observada.

```
2  switch(variavel)
3  {
4      case valor1:
5          //ações caso valor1
6      break;
7      case valor2:
8          //ações caso valor2
9      break;
10     case valor3:
11         //ações caso valor3
12     break;
13     default:
14         //ações caso nenhum dos valores
15     break
16 }
17
```

Controle de Fluxo e Estruturas Condicionais

```
2   switch(dia)
3   {
4       case 1:
5           alert('Hoje é domingo')
6       break;
7       case 2:
8           alert('Hoje é segunda')
9       break;
10      case 3:
11          alert('Hoje é terça')
12      break;
13      default:
14          alert('Hoje não é nem domingo, nem segunda, nem terça')
15      break
16  }
17
```

O comando switch usado em outro exemplo, nesse caso a verificação de qual dia da semana.

Laços de repetição

Existem estruturas de repetições, onde comando executa o comando dentro do loop até que o retorno seja diferente do determinado para entrar em loop.

No JavaScript uma desses laços se chama **while** (enquanto – português).
Veja um exemplo ao lado.

```
2    while(condicao)
3    {
4        //ações
5    }
6
```

Laços de repetição

Essa estrutura de repetição **while** é muito usada para executar um conjunto de ações enquanto uma condição for verdadeira.

Quando a condição retorna o valor falso, o bloco de comando é finalizado.

Vamos ver um exemplo bastante usado em JavaScript:

```
2    var contador = 0;
3    while(contador < 5)
4    {
5        alert('Olá');
6        contador = contador + 1
7    }
8
```

Laços de repetição

Outra estrutura semelhante é a **do – while**, que executa um bloco de ações até que uma condição seja falsa.

Mas, essa condição é validada após a execução dos comandos, fazendo com que estes sejam executados pelo menos uma vez.

```
2  do
3  {
4      //ações
5  }
6  while(condicao)
7
```

Laços de repetição

Outra estrutura semelhante é a **do – while**, que executa um bloco de ações até que uma condição seja falsa.

Mas, essa condição é validada após a execução dos comandos, fazendo com que estes sejam executados pelo menos uma vez.

```
2    var contador = 0;
3    do
4    {
5        alert('Olá');
6        contador = contador + 1;
7    }
8    while(contador < 5)
9
```

Laços de repetição

Um exemplo semelhante ao comando **while** pode ser usado para representar a sintaxe do comando **do-while.**

Vamos ver o código ao lado:

```
2    var contador = 0;
3    do
4    {
5        alert('Olá');
6        contador = contador + 1;
7    }
8    while(contador < 5)
9
```

Laços de repetição

E a última estrutura de repetição usada no JavaScript é o **for**, ele usa um contador para executar um bloco de ações uma determinada quantidade de vezes.

Vejamos um exemplo da sintaxe:

```
2    for(inicializacao; condicao; complemento)
3    {
4        //ações
5    }
6
```

Laços de repetição

Fica mais fácil entender essa estrutura quando observamos um exemplo prático.

No exemplo abaixo, temos uma variável contador é inicializado com o valor zero, e enquanto for menor que 10, o laço deve ser executado.

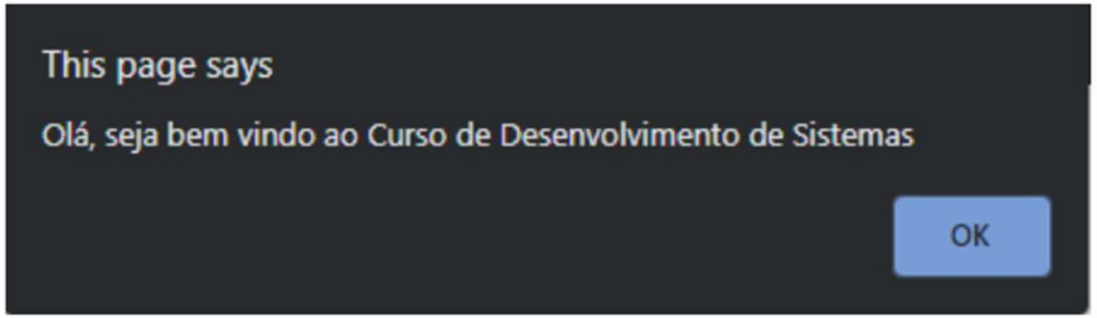
```
2    var contador;
3    for(contador = 0; contador < 10; contador++)
4    {
5        alert(contador);
6    }
7
```

Comando Alert

A função **alert**, exibe uma janela pop-up com uma mensagem definitiva, não sendo possível **digitar texto,** assim não é possível coletar dados por ela.

Vamos ao exemplo da sintaxe:

```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <meta charset="UTF-8">
7           <script type="text/javascript">
8               alert("Olá, seja bem vindo ao Curso de Desenvolvimento de Sistemas")
9           </script>
10      </head>
11
12          <body>
13
14          </body>
15  </html>
16
```

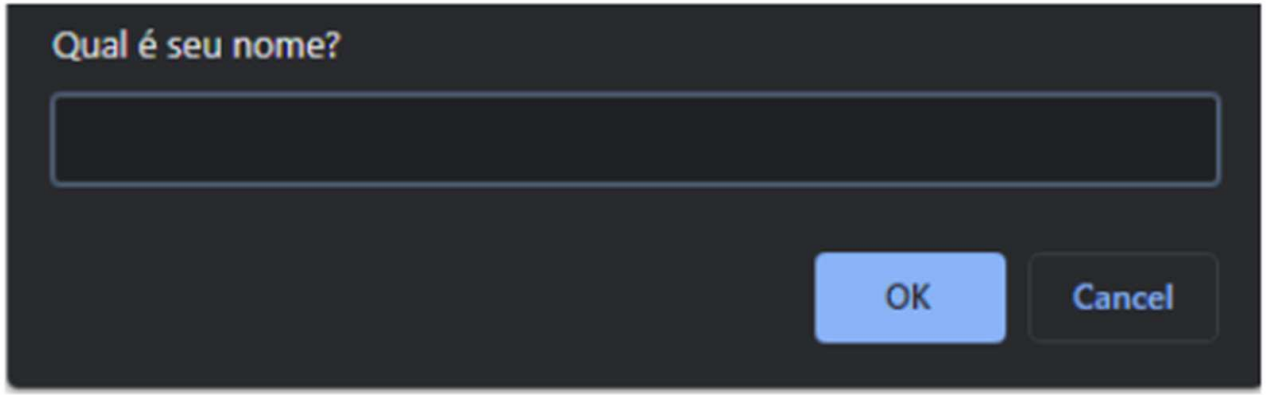


Comando Prompt

A função **prompt** é semelhante a função **alert**, a diferença, que na janela pop-up que essa função exibe é possível digitar texto, assim coletando dados digitados pelo usuário.

Vamos ao exemplo da sintaxe:

```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <meta charset="UTF-8">
7           <script type="text/javascript">
8               var nome;
9               nome = prompt("Qual é o seu nome?")
10              alert("Olá, " + nome)
11          </script>
12      </head>
13
14          <body>
15
16          </body>
17  </html>
18
```



Objeto Window

O **objeto window** é usado para fazer a manipulação das janelas do navegador.

Toda vez que abrirmos o navegador simultaneamente esse objeto é gerado, isso acontece porque esse objeto representa exatamente essa janela que foi aberta.

Para criarmos esse objeto, não precisamos usar nenhum tipo de linguagem, como HTML, porque isso é feito automaticamente pelo navegador.

Com esse objeto podemos criar e abrir novas janelas de maneiras diferentes.

Esses processos são possíveis através das propriedades e métodos que o objeto window possui.

Vamos ver a sintaxe desse objeto:

- **window.propriedade**
- **window.metodo**

Propriedades

As propriedades do **objeto window** tem como objetivo modificar os aspectos em relação à janela do navegador.

Closed – Esta propriedade retorna um valor booleano indicando se a janela foi fechada.

DefaltuStatus – Esta propriedade nos permite definir uma mensagem padrão que será exibida no status do navegador.

Document – Esta propriedade possui todas as características da página HTML, onde podemos integrar no meio de um comando JavaScript tag HTML.

Frames: Array de frames em uma janela.

History: Esta propriedade contém uma lista representando as URLs que o usuário já visitou.
Podemos acessar todas as URLs visitadas pelos usuários atribuindo os seguintes métodos a essa propriedade: current, next e previous.

InnerHeigth: Com esta propriedade podemos definir ou obter a altura da área onde o conteúdo é apresentado, não a altura do navegador em si.

Propriedades

InnerWidth: Com esta propriedade podemos definir ou obter a largura da área onde o conteúdo é apresentado, não a largura do navegador em si.

Length: Informa a quantidade de frames existentes em uma janela.

Location: Esta propriedade tem informações referentes ao endereço corrente. Um exemplo é a propriedade hostname, que retorna o nome do servidor que está hospedando a página carregada. Nessa propriedade temos mais dois métodos:

- **Reload**: Que força o navegador a carregar a página.
- **Replace**: Que carrega a URL informada.

Name: Podemos utilizar para definir ou obter o nome da janela.

Navigator: Esta propriedade contém informações sobre o navegador, nome, versão, e outras informações. O objeto navigator possui também um método interessante:

- **javaEnabled()**: Que informa se o navegador está com o Java habilitado.

Vamos utilizar o objeto ***document*** como exemplo para aprendermos como funciona a sintaxe do objeto window.

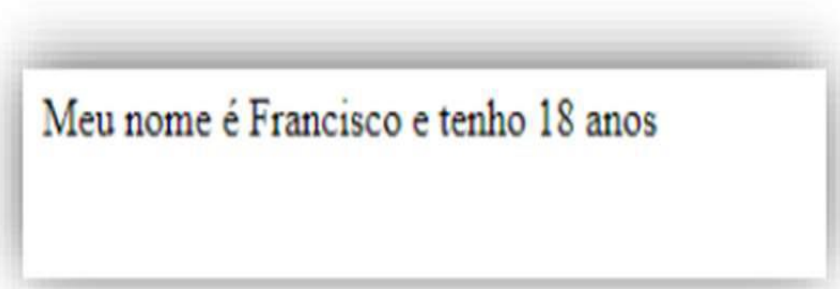
Document Write

O método **document.write** serve para escrever informações no documento HTML.

Ele é muito simples de usar, basta colocar dentro de parênteses ( ) o que deseja que apareça no documento.

Vamos definir duas variáveis, uma com um nome e outra com idade, então vamos escrever no documento a string (texto): "**Meu nome é [nome] e tenho [idade] anos**"

No código HTML fica assim:




```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <title>Curso Desenvolvimento de Sistemas</title>
7           <meta charset="UTF-8">
8
9           <script type="text/javascript">
10              var nome="Francisco";
11              var idade=18;
12
13              document.write("Meu nome é ", nome, " e tenho ", idade, " anos.");
14          </script>
15      </head>
16
17          <body>
18
19          </body>
20  </html>
21
```

Note que quando usamos texto para aparecer no documento, ele deve estar entre aspas " " , e quando usamos uma variável para mostrar o valor é somente o nome da variável, nesse caso nome e idade.

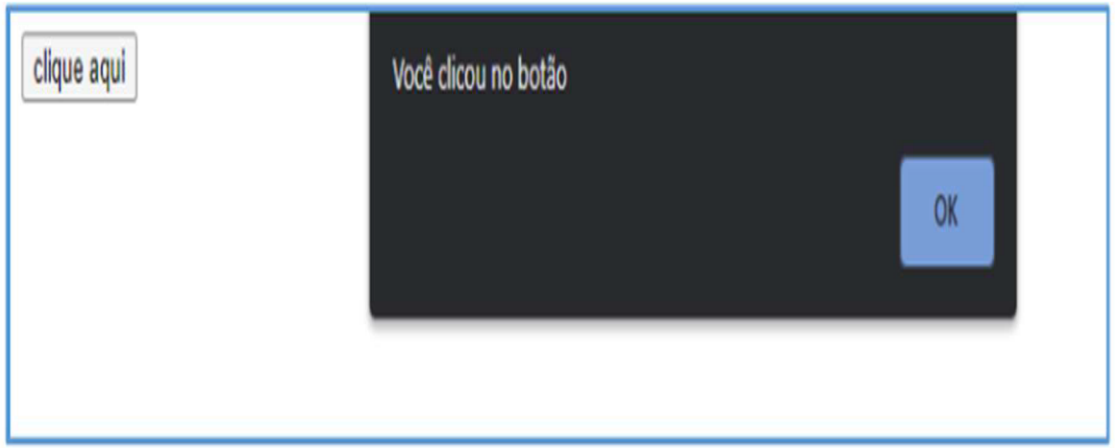
Manipulando eventos

Os eventos são disparados quando alguma ação é executada, como clique num botão, ou a digitação de valores em um input.

No código podemos atribuir valores aos eventos como se fossem propriedades, desde que o valor atribuído seja um código a ser executado.

Por exemplo na imagem, mostra o código de uma página com um botão, que, ao ser clicado, exibe uma mensagem (alert).

Resultado no navegador:




```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <meta charset="UTF-8">
7       </head>
8
9           <body>
10              <button onclick="alert('Você clicou no botão');"> Clique Aqui! </button>
11          </body>
12  </html>
13
```

Eventos

OnMouseover – O evento onmouseover executa um código em JavaScript quando o ponteiro do mouse é movido para um elemento ou para um de seus filhos.

OnMouseout – O evento executa um JavaScript quando o ponteiro do mouse é movido para fora de um elemento ou de seus filhos.

OnLoad – Esse evento carrega o elemento JavaScript.

OnBlur – Esse evento remove o foco do elemento.

OnChange – Esse evento altera o valor do elemento.

OnClick – Esse evento executa um comando quando é clicado pelo usuário.

OnKeyPress – Esse evento é executado quando o usuário pressiona uma tecla sobre o elemento.

OnSubmit - Esse evento é usado em formulários, onde é definida uma ação em JavaScript.

OnFocus – Esse evento é aplicado um foco no elemento.

Eventos

Os eventos são muito importantes no JavaScript, pois facilita a interação com o usuário e é carregado de modo instantâneo.

Usamos os eventos de duas maneiras:
INLINE – Dessa maneira o evento é definido diretamente na tag do elemento, vamos ao exemplo:

```
2   <!DOCTYPE html>
3   <html lang="pt-BR">
4
5       <head>
6           <title>Usando o evento onClick no Javascript</title>
7       </head>
8
9           <body>
10              <h1 onClick="this.innerHTML='Isso acontece quando usamos o evento onClick!'">
11                  Clique nesse link para testar o evento onClick </h1>
12          </body>
13  </html>
14
```

Eventos

EXTERNO – Para usarmos um evento externo, é preciso de um manipulador de evento, nesse exemplo vamos usar o event-listener que adiciona ou remove um evento sobre qualquer elemento.

Esse evento nos disponibiliza duas funções principais, são elas:

- **addEvent** – Adiciona uma função que será disparada quando ocorrer determinado evento no objeto.
- **removeEvent** – Remove um listener previamente adicionado em um objeto e retorna o valor true, em caso de sucesso.

Vamos ver um exemplo do uso desse evento:

```
2    <script type="text/javascript" src="event-listener.js"></script>
3
4    <form>
5        <input type="text" name="a">
6        <input type="submit">
7    </form>
8
```

Eventos

Vamos ver mais alguns exemplos de eventos mais usados no JavaScript:

Evento onload

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usando eventos no JavaScript</title>
</head>
<body onload="checkcookies()">

    <script>
        function checkcookies()
        {
            if(navigator.coockieEnabled==true)
            {
                alert("Coockies são permitidos")
            }
            else
            {
                alert("Coockies não são permitidos")
            }
        }
    </script>

    <p>Irá aparecer um alert dizendo se os coockies estão ou não liberados
        em seu navegador</p>
</body>
</html>
```

Eventos

Evento onchange

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usando eventos no JavaScript</title>
    <script>
        function myFunction()
        {
            var x = document.getElementById("fname")
            x.value = x.value.toUpperCase()
        }
    </script>
</head>
<body>
    Insira seu nome: <input type="text" id="fname" onchange="myFunction()">
    <p>Ao clicarmos fora do input text o texto escrito nele ficará todo em caixa alta.</p>
</body>
</html>
```

Eventos

onmouseover
onmouseout

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usando eventos no JavaScript</title>
</head>
<body>
    <div onmouseover="mOver(this)" onmouseout="mOut(this)" style="background-color: #D94A38; width:120px;height:20px;
    padding:40px;">Passe o mouse aqui</div>

    <script>
        function mOver(obj)
        {
            obj.innerHTML = "Obrigado!"
        }
        function mOut(obj)
        {
            obj.innerHTML = "Passe o mouse aqui."
        }
    </script>
</body>
</html>
```

Eventos

onmousedown
onmouseup

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Usando eventos no JavaScript</title>
</head>
<body>
    <div onmousedown="mDown(this)" onmouseup="mUp(this)"
    style="background-color: #f94f38; width:120px;height:20px;
    padding:40px;">Clique Aqui!</div>

    <script>
        function mDown(obj)
        {
            obj.style.backgroundColor = "#1ec5e5"
            obj.innerHTML = "Solte o click!"
        }
        function mUp(obj)
        {
            obj.style.backgroundColor = "#1ec5e5"
            obj.innerHTML = "Obrigado!"
        }
    </script>
</body>
</html>
```

Atividades

1. Crie um programa onde o usuário insira seu nome através de um prompt, e então o nome seja exibido em um alert.

2. Peça ao usuário para digitar seu ano de nascimento usando o prompt, calcule a idade e mostre o resultado com alert.

3. Solicite dois números ao usuário e mostre qual é o maior usando estruturas condicionais.

4. Peça ao usuário um número e calcule o fatorial desse número usando um laço for.

5. Crie um botão que, ao ser clicado, mostre uma mensagem com alert exibindo um cumprimento.

6. Peça ao usuário cinco números em sequência, some-os e mostre o resultado final com alert.

7. Peça ao usuário para digitar um número de 1 a 7 e mostre o dia da semana correspondente (exemplo: 1 = Domingo, 2 = Segunda...).

Atividades

8. Exiba a largura e altura da janela atual do navegador usando propriedades do objeto window.

9. Crie um formulário que tenha dois campos (nome e idade). Quando o botão "Enviar" for clicado, verifique se os campos estão preenchidos. Caso contrário, exiba um alerta avisando que o campo está vazio. Caso estejam preenchidos, mostre os dados em um alert.

10. Crie uma rotina em JavaScript em que o usuário digite um nome e a idade, e na janela do navegador apareça o nome e a idade digitados pelo usuário.

11. Implemente um jogo em que o programa escolha um número aleatório de 1 a 10, e o usuário deve adivinhar. O jogo deve continuar até que o usuário acerte, informando a cada tentativa se o número digitado é maior ou menor que o número secreto.

12. Neste exercício, você vai criar uma calculadora simples que realiza as quatro operações básicas (soma, subtração, multiplicação e divisão). O usuário deverá inserir dois números e escolher a operação desejada. O resultado será exibido em uma div após o clique no botão "Calcular".

13. O usuário deverá inserir uma senha, e o sistema verificará se a senha é forte. A senha será considerada forte se tiver pelo menos 8 caracteres, incluindo pelo menos uma letra maiúscula, uma minúscula, um número e um caractere especial. O resultado será exibido após clicar no botão "Verificar Senha".

Senac Fecomércio Sesc

Siga o Senac em Minas nas Redes Sociais: