

INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas

Fundamentos Web

Professor Sérgio Rodrigues

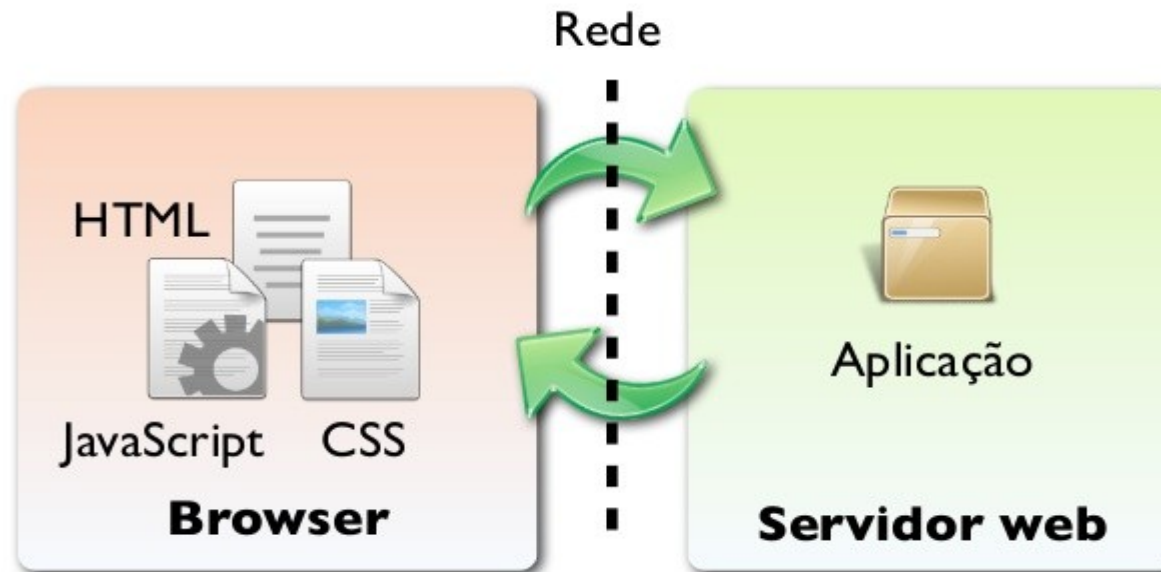


Como funciona a arquitetura cliente-servidor

O software da máquina cliente envia uma solicitação ao servidor, onde é processada e o resultado finalizado é enviado ao cliente.

Funções que são implementadas lado do cliente:

- gerar e enviar uma solicitação ao servidor;
- recebimento de resultados e atualização de informações na página.



Funções que são implementadas no lado do servidor:

- Armazenamento de dados;
- Processamento de uma solicitação de um cliente;
- Envio do resultado ao cliente.

Arquitetura cliente-servidor

HTTP é um protocolo que permite a obtenção de recursos, como documentos HTML. É a base de qualquer troca de dados na Web, é um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web.

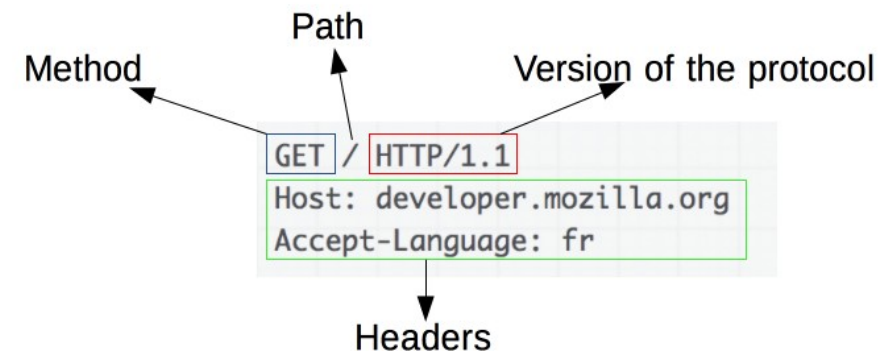
Clientes e servidores se comunicam trocando mensagens. As mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de **solicitações** (*requests*), ou também **requisições**, e as mensagens enviadas pelo servidor como resposta são chamadas (*responses*).

Arquitetura cliente-servidor

Quando o cliente quer comunicar com um servidor envia uma mensagem HTTP:

As requisições consistem dos seguintes elementos:

- Um método HTTP, geralmente é um verbo como GET, POST, DELETE, PUT, etc. Um cliente que pegar um recurso (usando GET) ou publicar dados de um formulário HTML (usando POST).
- O caminho do recurso a ser buscado; a URL do recurso; domínio (aqui como developer.mozilla.org); a porta TCP (aqui indicada pelo 80 que é oculto por ser o número da porta padrão)
- A versão do protocolo HTTP.
- Cabeçalhos opcionais que contém informações adicionais para os servidores.
- Ou um corpo de dados, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.

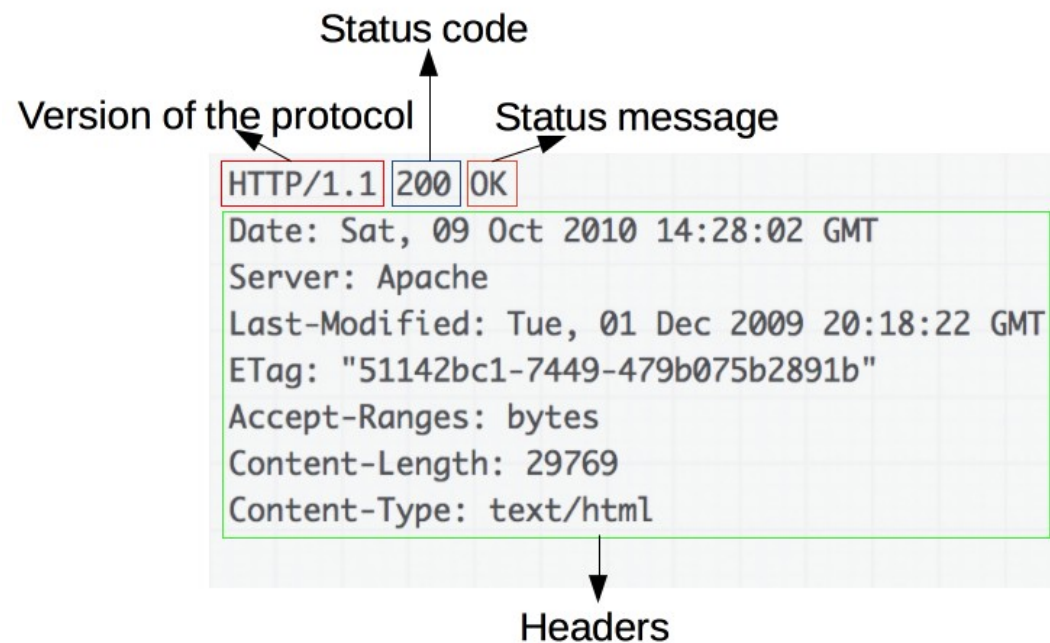


Arquitetura cliente-servidor

Do outro lado do canal de comunicação está o servidor que gera o documento requisitado devolvendo uma resposta para o cliente.

As respostas consistem dos seguintes elementos:

- A versão do protocolo HTTP que elas seguem.
- Um código de status, indicando se a requisição foi bem sucedida, ou não, e por quê.
- Uma mensagem de status, uma pequena descrição informal sobre o código de status
- Cabeçalhos HTTP, como aqueles das requisições.
- Opcionalmente, um corpo com dados do recurso requisitado.



Linguagens front-end e back-end

Linguagens **front-end** são executadas no computador do usuário, por meio do navegador — também conhecidas como linguagens **client-side**.



Linguagens **back-end** são linguagens que o **SERVIDOR** entende. O servidor vai processar o código e vai mandar a resposta para o navegador como HTML - também conhecidas como **server-side**.



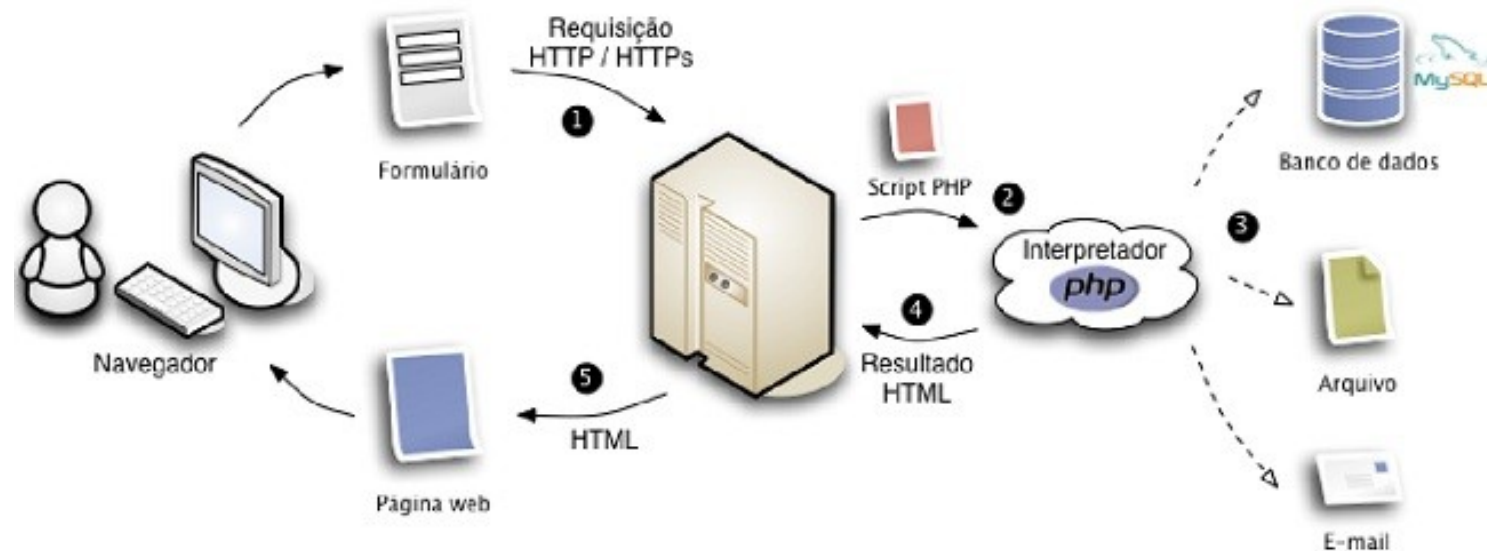
Linguagens front-end e back-end

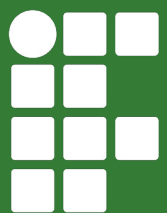
As linguagens client-side são linguagens onde apenas o seu NAVEGADOR vai entender. Quem vai processar essa linguagem não é o servidor, mas o seu browser.

Por exemplo: se criarmos um script em linguagem back-end (PHP, Asp, Java, Ruby, etc) que apenas calcula a soma de $2 + 2$, será o SERVIDOR (ou back, o server) que calculará este resultado. Se fizermos esse cálculo em alguma linguagem front-end, como o JavaScript, quem calculará essa conta é o BROWSER do usuário. Por isso o termo client ou front.

A linguagem server-side recebe o requerimento (Request) e faz o processamento. Depois, transforma o resultado final em um HTML e envia para o navegador. É a linguagem server-side que vai verificar se o usuário está logado, vai buscar informações no banco de dados etc...

Linguagens front-end e back-end





INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas

Java Script

Programação JavaScript

JavaScript é uma das linguagens de programação mais populares no desenvolvimento Web. Foi criada em 1995 (Netscape) e é responsável por praticamente qualquer tipo de dinamismo que queiramos em nossas páginas.

JavaScript é uma linguagem de programação que permite a implementação de itens complexos em páginas web e conteúdos que se atualizam dinamicamente.

É uma linguagem interpretada, ou seja, não depende de compilação para ser executada. O código é interpretado e executado conforme é lido pelo navegador, linha a linha, assim como o HTML

A linguagem JavaScript é constantemente atualizada.

(<https://www.w3schools.com/js/default.asp>)

Programação JavaScript

A linguagem JavaScript cresceu exponencialmente nos últimos anos e já possui um grande ecossistema com frameworks* para o Front-end, Back-end, Mobile e Desktop. Este crescimento facilita a utilização do JavaScript não apenas no front-end, mas também no back-end e em aplicações mobile.

Foi empregada por muito tempo como linguagem front-end, mas está cada vez sendo mais utilizada como uma linguagem server-side, por exemplo, no popular ambiente Node.js

*Os frameworks são templates usados pelos desenvolvedores para criar os seus projetos e gerenciá-los de maneira mais ágil e prática. Eles funcionam como uma plataforma de desenvolvimento, e trazem uma série de ferramentas especiais, guias e outros componentes. Exemplos: Laravel, React, Ruby, entre outros.

Programação JavaScript

De maneira equivocada, muitos "desconhecedores" e desinformados costumam afirmar que o JavaScript é o "Java interpretado". Isso é reflexo de uma completa falta de conhecimento não apenas da linguagem JavaScript mas também da linguagem Java.

Em resumo, o código em JavaScript é uma linguagem de script dinâmica, que reconhece a construção de objetos baseada em protótipos.

Com a intenção de reduzir o número de conceitos exigidos para se aprender a linguagem, a sintaxe básica do JavaScript é, simplesmente, muito similar à sintaxe das linguagens C++ e Java.

Dessa forma, as estruturas da linguagem JavaScript - como if, for, while, switch, entre outras - funcionam exatamente do mesmo modo (salvo algumas particularidades muito específicas da própria linguagem).

Programação JavaScript

Apesar da intensidade de seu uso no ambiente web, a linguagem JavaScript pode ser desenvolvida tanto como uma linguagem procedural quanto como uma linguagem orientada a objetos.

Apesar das vantagens do JavaScript algumas desvantagens são relevantes:

- **Exposição do código** - fácil visibilidade do código-fonte;
- **Tempo de espera** - o código script deve ser totalmente descarregado no navegador para que possa ser executado;
- **Configuração adequada** - o navegador (browser) precisa estar configurado para interpretar códigos script, caso contrário, apenas as instruções de marcação estruturais da página (HTML) serão interpretadas.

Funcionalidades básicas do JavaScript:

Manipular conteúdo e apresentação:

- Inserir marcação HTML
- Controlar de forma dinâmica a folha de estilo

Manipular o navegador:

- Criar janelas pop-up (alertas, confirmações)
- Abrir e fechar abas

Interagir com formulários:

- Acessar campos e valores digitados em formulários
- Validar dados

O que é necessário?

É necessário ter um navegador web moderno (como Chrome, Firefox, Edge ou Safari).

Também salvar o arquivo com a extensão “.html” ou “.js” para que o navegador execute o que está escrito no arquivo.

Para projetos mais complexos ou que envolvem servidores, o Node.js será necessário.

Estrutura Básica da Linguagem

A estrutura básica JavaScript consiste em simples comandos. No entanto, como os códigos não são compilados, mas sim interpretados diretamente pelos navegadores, qualquer erro de digitação ou falta de caractere para compor o parâmetro do código será suficiente para o código não ser interpretado pelo navegador.

Chamamos essa "sensibilidade" do código de "código sensitivo", ou seja, se houver um simples erro de digitação, o navegador interpretará apenas as marcações HTML, ignorando o código JavaScript.

Estrutura Básica da Linguagem

<code><script language>...</script></code>	São, respectivamente, delimitadores de início e fim da codificação JavaScript
<code><!-- //--></code>	São, respectivamente, indicadores de inicialização e de finalização do conteúdo das linhas do código JavaScript. São utilizados para informar ao navegador a possibilidade de não executar os comandos que estiverem entre esses elementos, caso o browser não suporte o código JavaScript.
<code>/* */</code>	É usado para definição de uma documentação e/ou um comentário interno do código.
<code>var</code>	É um parâmetro para definição de variáveis.
<code>;</code> (ponto-e-vírgula)	É utilizado para finalizar uma linha de comando.

Estrutura Básica da Linguagem

objeto	<p>São elementos ou sujeitos de uma ação, responsáveis por um comando ou por uma instrução. Por exemplo:</p> <p>Um documento web = uma página web carregada pelo browser Uma tela web = uma janela web carregada pelo browser. Exemplos de objeto: document, window.</p>
método	<p>É uma forma ou um recurso que indica como um objeto será manipulado ou processado. Podemos definir métodos de entrada ou de saída. Exemplos de método: prompt, form, alert, write</p>
propriedade	<p>É uma característica específica para tratar um objeto. Por exemplo: a aplicação de funções matemáticas, logarítmicas ou de variações. Exemplos de propriedade: PI, sqrt, random.</p>
evento	<p>É um recurso de ação aplicado ao objeto. Exemplos de eventos: onclick, onblur, onfocus, onload, onmousedown, onmousemove, onmouseout, onmouseover</p>

Primeiro Script

O script é enviado com o HTML para o navegador, mas como o navegador saberá diferenciar o script de um código html?

Para que essa diferenciação seja possível, é necessário envolver o script dentro da tag `<script> </script>`

```
1 <html>
2 <head>
3     <title>JavaScript</title>
4     <script> alert("Olá mundo!"); </script>
5 </head>
6 <body> ...
7 </body>
8 </html>
```

exemplo_1.html

Os exemplos deste material encontram-se na pasta compactada “exemplos JavaScript”

Separação em camadas

Recomenda-se trabalhar com o código JavaScript em um arquivo (Fora do HTML).

```
1 <html>
2 <head>
3     <title>JavaScript</title>
4     <script src="js/lib.js"></script>
5 </head>
6 <body> ...
7 </body>
8 </html>
```

Variáveis em JavaScript

Variáveis armazenam dados de diferentes tipos: inteiro, ponto-flutuante, string, etc.

∞ O nome da variável deve começar com:

∞ Uma letra

∞ Um caractere underscore (_)

∞ Dica: Não use o \$ para evitar confusão com códigos específicos de bibliotecas JavaScript

Tipos de variáveis

INTEIROS (INTEGER)

A=500

B=45

C=-32

REAL

NUM=2.34

BOOLEANOS

Este tipo de literal representa valores lógicos que podem ser:

TRUE ou 1

FALSE ou 0

LITERAIS STRING

Este literal representa qualquer cadeia de caracteres envolvida por aspas ou apóstrofo. Veja abaixo alguns exemplos:

“Adriano Lima” “Empresa ABC” ‘Maria Isabel’

Variáveis em JavaScript

- ⌘ JavaScript é uma linguagem de programação não tipificada.
- ⌘ O tipo de variável é dinâmico e pode mudar de acordo com o conteúdo

```
1 <html>
2 <head>
3   <title>JavaScript</title>
4   <script type="text/javascript">
5     minha_variavel = 235;
6     minha_variavel = "Alô Mundo!";
7
8     quant_produtos = 50;
9     nome_usuario = "João";
10
11     quantProdutos = 100;
12     nomeUsuario = "Ana";
13   </script>
14 </head>
15 <body>
16 </body>
17 </html>
```

Declaração de Variáveis em JavaScript

As variáveis podem ser declaradas separadas por vírgula, da seguinte maneira:

```
var nome, endereco;
```

```
let idade, telefone;
```

OU

```
var nome;
```

```
var endereco;
```

```
let idade;
```

```
let telefone;
```

Obs: Para a declaração de variáveis no JavaScript pode ser usada as palavras reservadas **var** ou **let**. A diferença reside no seu escopo **var** tem escopo global, enquanto **let** tem escopo de bloco.

Um exemplo prático de atribuição, é atribuir um mesmo valor a mais de uma variável, da seguinte maneira:

```
let campo1 = campo2 = campo3 = 5
```

No exemplo anterior, foi atribuído o número 5 nas variáveis campo1, campo2 e campo3.

Declaração de Variáveis em JavaScript

Em JavaScript, **const** é uma palavra-chave usada para declarar uma variável constante, ou seja, uma variável que não pode ser reatribuída depois de definida. Isso significa que, após a sua inicialização, o valor da variável **const** não pode ser alterado.

```
const PI = 3.14159;  
const nome = "João";  
const pessoa = {  
  nome: "Maria",  
  idade: 30  
};
```

Obs.: usa-se **const** para declarar variáveis que não serão alteradas, **let** para variáveis que precisam ser reatribuídas. O **var** é uma palavra-chave mais antiga, porém ainda empregada.

Operadores

Operadores Aritmético

%	Resto
+	Soma
-	Subtração
*	Multiplicação
/	Divisão

Operadores Lógico

&&	Lógico E
	Lógico Ou
!	Lógico Não

Operadores Relacionais

==	Igualdade
!=	Diferente
===	Identidade (estritamente igual)
!==	Não identidade (estritamente diferente)
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a

Exemplo com document.write

```
1 <script type="text/javascript">
2 valor=30;
3 document.write("Resultado do cálculo ",(10*2)+valor);
4 </script>
```

document.write: é uma função do JavaScript que escreve no html da página
*Utilizado para testes durante o desenvolvimento dos códigos

Exemplo com console.log

```
1 <script type="text/JavaScript">
2 valor=30
3 console.log("Resultado do calculo:",(10*2)+valor,"<br>")
4 </script>
```

console.log: é uma função do JavaScript que escreve uma mensagem no console da web.

*Esse console é exibido nas ferramentas de desenvolvedor do navegador (pressionando f12 no Firefox/Chrome). Normalmente, não é visível por usuários comuns, apenas desenvolvedores; Esta ferramenta Console serve também para mostrar os erros do código JavaScript

Caixas de Diálogo

Janela do tipo pop-up que se destina a fornecer informações ou coletar dados do usuário.

O JS possui três métodos (ou funções), para o objeto Window, para criar diferentes caixas de diálogo:

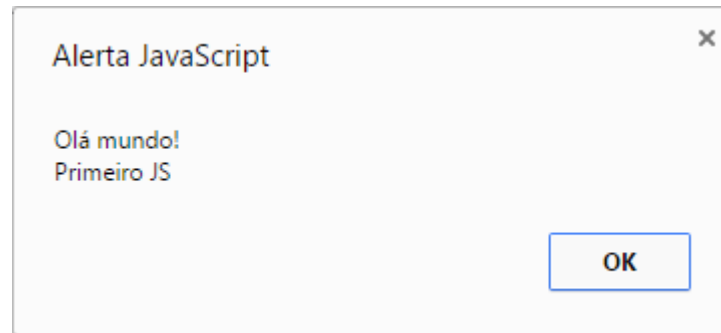
- ⌘ Alert box
- ⌘ Confirm box
- ⌘ Prompt box

Alert box

Coloca uma caixa de diálogo contendo uma mensagem ao usuário.

Sintaxe:

```
alert("Olá mundo!\nPrimeiro JS");
```



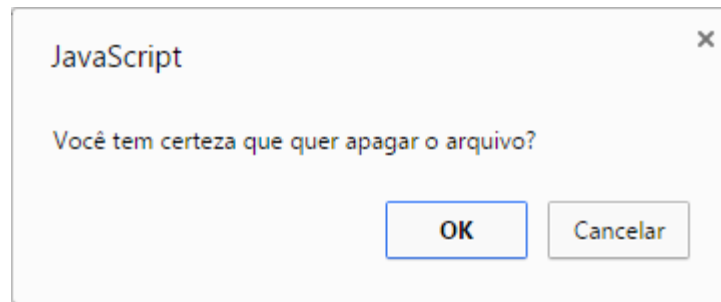
exemplo_2.html

Confirm box

Coloca uma caixa de diálogo contendo dois botões: Ok e Cancelar.

Sintaxe:

```
confirm("Você tem certeza que quer apagar o arquivo?");
```



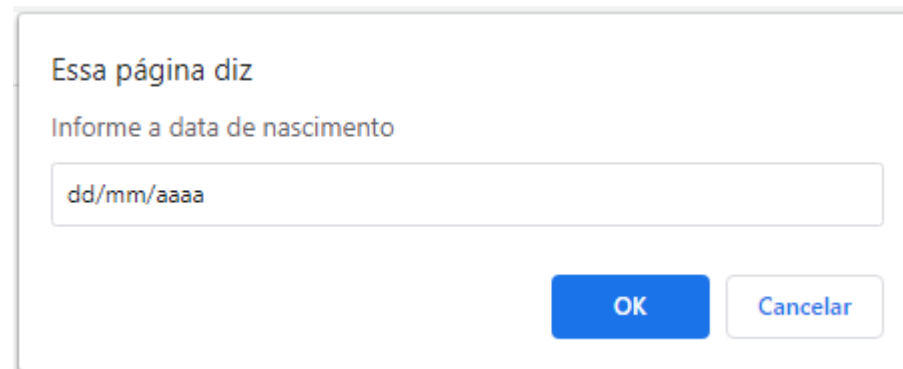
exemplo_3.html

Prompt box

Coloca uma caixa de diálogo contendo um campo para que o usuário digite uma string.

Sintaxe:

```
prompt("Informe a data de nascimento", "dd/mm/aaaa");
```

A screenshot of a JavaScript prompt box. The dialog box has a title bar and contains the text "Essa página diz" followed by "Informe a data de nascimento". Below this text is a text input field containing the placeholder "dd/mm/aaaa". At the bottom right of the dialog box are two buttons: a blue "OK" button and a white "Cancelar" button with a blue border.

exemplo_4.html

Estruturas de Controle

Assim como nas demais linguagens, no JS existem diversas estruturas de controle.

Estruturas condicionais:

if/then/else

switch/case

Estruturas de repetição:

while/do..while

for

Estruturas Condicionais

if... else if

∞ Tomada de decisão baseada numa expressão lógica.

```
1  <script type="text/javascript">
2
3  var hora = 15;
4  if (hora < 12)
5  alert("bom dia");
6  else if (hora >= 12 && hora < 18)
7      alert("boa tarde");
8  else
9      alert("boa noite");
10
11 </script>
```

exemplo_5.html

Estruturas Condicionais

switch

∞ Estrutura de seleção que apresenta várias possibilidades de escolha.

```
1  <script type="text/javascript">
2
3  switch(expressão) {
4  case valor1:
5      comando1;
6  break;
7  case valor2:
8      comando2;
9  break;
10 case valor3:
11     comando3;
12 break;
13 default:
14     comando4;
15 }
16
17
18 </script>
```

Estruturas de Repetição

while

∞ Estrutura de repetição que executará as instruções enquanto uma condição seja verdadeira.

```
while(condição) {  
    comando 1;  
    comando 2;  
    ...  
}  
  
do {  
    comando 1;  
    comando 2;  
    ...  
} while (condição);
```

Estruturas de Repetição

for

∞ Estrutura de repetição com variável de controle e limites fixos.

```
for(valor_inicial; condição; atualiza_valor) {  
... comandos  
}
```

Um exemplo prático de utilização do laço for que conta valores de 1 até 10, acrescentando um valor de cada vez:

```
1 <script type="text/javascript">  
2  
3 for (i=1 ; i<=10 ; i++){  
4 document.write("número: "+ i + "<br>");  
5 }  
6  
7 </script>  
8
```

exemplo_6.html

Funções

```
function nomeFunção() {  
  Comandos  
}
```

A sintaxe geral de uma função traz a palavra reservada `function` e o nome da função seguido de parênteses. O corpo da função deverá ser colocado entre chaves que indicarão o início do bloco de instruções e o fim do bloco de instruções.

A chamada da função compreende o nome da função seguido de parênteses sem argumentos.

```
<script type="text/JavaScript">  
function exemplo(){  
    alert("Curso de JavaScript");  
}  
exemplo();  
</script>
```

Esta função sempre apresentará em sua chamada o alert da mensagem “Curso de Java Script”

exemplo_7.html

Funções

```
function nomeFunção(parâmetros) {  
  Comandos  
}
```

Se a função possuir parâmetros, estes deverão estar colocados entre parênteses após o nome da função. Na chamada da função será passado o conteúdo (argumento) que será atribuído ao parâmetro da função.

```
<script type="text/JavaScript">  
function exemplo(texto){  
  alert(texto);  
}  
exemplo("Curso de JavaScript");  
exemplo("Bem-vindo");  
</script>
```

exemplo_8.html

Esta função apresentará a mensagem que for passada na chamada da função.
A função atribui para a variável texto a mensagem “Curso de JavaScript” na primeira chamada. E “Bem-vindo” na segunda chamada.

Funções

Para retornar um valor de uma função basta fazer o uso da instrução return mais o valor que queira retornar.

```
1 <script type="text/JavaScript">
2 function mostraNome(nome){
3     if (nome=="")
4         return "erro";
5     else
6         return nome;
7 }
8 texto=prompt("digite o nome: ");
9 var retorno=mostraNome(texto);
10 alert(retorno);
11 </script>
12
```

exemplo_9.html

A chamada da função passa como argumento o texto digitado no campo de prompt, que está sendo armazenado na variável “texto”.

```
var texto = prompt("digite o nome: ");
var retorno = mostraNome(texto);
```

A palavra “erro” ou o próprio nome digitado, (conforme a lógica do corpo da função) é atribuído para a variável “retorno” . E o conteúdo desta variável será retornado no alert.

```
alert(retorno);
```

ou seja, o alert mostra a “erro” se o conteúdo do campo estiver vazio ou o próprio nome digitado.

ATIVIDADE

Marque V para Verdadeiro e F para Falso, justifique as falsas:

- 1 () Os códigos JavaScript precisam ser compilados, assim como os códigos Java.
- 2 () JavaScript é conhecida como uma linguagem de programação leve, interpretada e orientada a objetos.
- 3 () JavaScript é executada no lado do servidor web, podendo ser utilizada para esboçar o comportamento da página web.
- 4 () O código-fonte do JavaScript apresenta difícil visibilidade.
- 5 () O JavaScript ao ser executado no navegador, pode ser descarregado em blocos.