



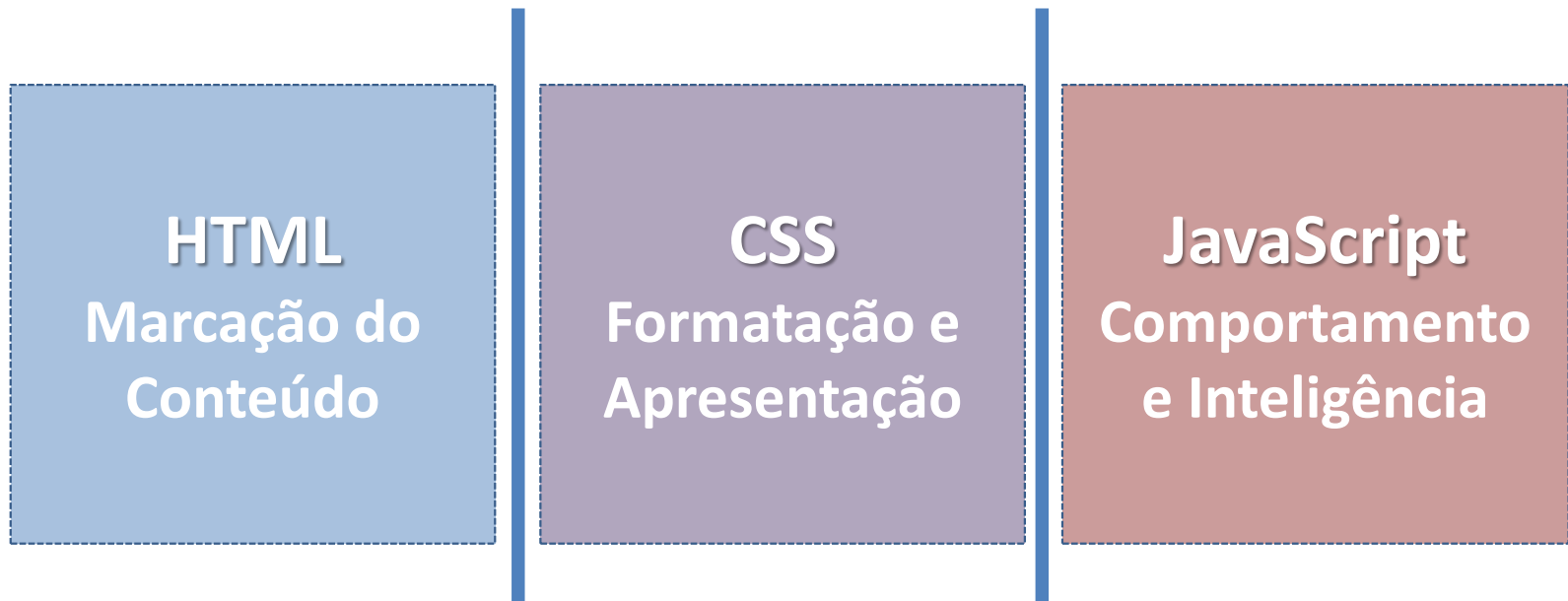
Desenvolvimento de Aplicações WEB

JavaScript

Profa. Joyce Miranda

JAVASCRIPT

- ▶ Linguagem de programação *front-end* que interage com o HTML e o CSS para adicionar comportamento e dinamismo à página
- ▶ Minimiza a comunicação entre o cliente e o servidor.



JAVASCRIPT

▶ Características

▶ Linguagem interpretada

- ▶ Scripts são embutidos nas páginas HTML e interpretados pelo navegador

▶ Tipagem dinâmica

- ▶ Tipos de variáveis não são definidos, seu conteúdo é que as define.

▶ Linguagem baseada em objetos

- ▶ Os elementos de uma página são tratados como objetos que podem ter propriedades, métodos e responder a eventos.

▶ Programação dirigida por eventos

- ▶ A página deixa de ser um documento estático e passa a interagir com as ações do usuário.

JAVASCRIPT

- ▶ Criação de Código
 - ▶ Programas Necessários
 - ▶ Editor de texto
 - ▶ Navegador
 - ▶ Os códigos JavaScript podem ser incluídos na página:
 - ▶ Bloco de código: **<SCRIPT> </SCRIPT>**
 - ▶ Arquivo externo (.js)

JAVASCRIPT

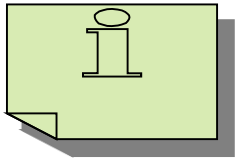
- ▶ DESENVOLVENDO SCRIPTS COM A TAG <SCRIPT>

```
1 <html>
2   <head>...</head>
3   <script type="text/javascript">
4       alert("Seja bem-vindo(a)!");
5   </script>
6   <body>
7       ...
8   </body>
9 </html>
```

JAVASCRIPT

- ▶ DESENVOLVENDO SCRIPTS ATRAVÉS DE UM ARQUIVO EXTERNO
 - ▶ Crie um arquivo com a extensão **.js** e inclua o código JavaScript nele.
 - ▶ Inclua a referência do arquivo na página HTML.

funcoes.js

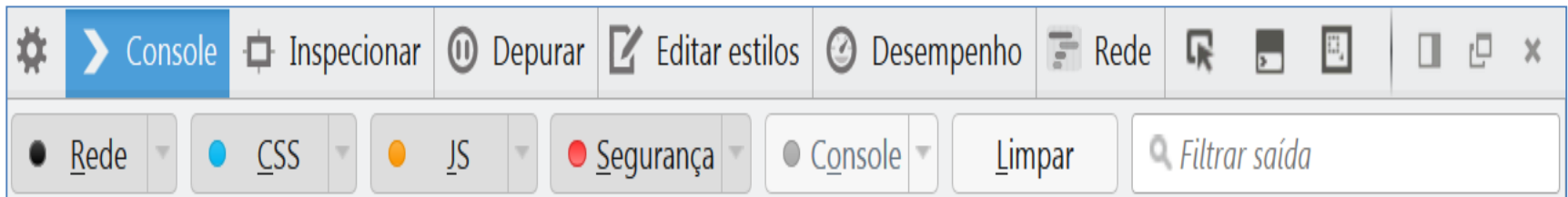


```
1 <html>
2   <head>...</head>
3   <script type="text/javascript" src="funcoes.js"/>
4   <body>
5       ...
6   </body>
7 </html>
```

JAVASCRIPT

► Console do Navegador

- Alguns navegadores dão suporte à entrada de comandos pelo console.
- **Botão direito: Inspecionar Elemento**



```
» alert('Interagindo com o navegador!!!')
```

JAVASCRIPT

▶ Sintaxe Básica

▶ Variáveis

`var nomeVariavel;`

▶ Suporte

- ▶ String
- ▶ Number
- ▶ Boolean

```
var texto = "Uma String deve ser envolvida em aspas simples ou duplas.";  
var numero = 2012;  
var verdade = true;
```


JAVASCRIPT

► Sintaxe Básica

► Funções

```
function nomeFuncao{  
    /* codigo */  
}
```

```
function nomeFuncao(arg1, arg2) {  
    /* codigo */  
    return valor;  
}
```

JAVASCRIPT

► Objeto Date

Método	Descrição
getDate/setDate	Dia do mês.
getDay/setDay	Dia da semana (0=Domingo – 6 =Sábado)
getHours/setHours	Horas (0 a 23).
getMinutes/setMinutes	Minutos (0 a 59).
getMonth/setMonth	Mês do ano (0=janeiro – 11=dezembro).
getSeconds/setSeconds	Segundos (0 a 59).
getFullYear/setFullYear	Ano contendo quatro dígitos.
toLocaleString	Converte a data do objeto <i>Date</i> para uma string nas configurações locais.

JAVASCRIPT

► Objeto Date

► Sintaxe

```
> data = new Date()  
< Mon Mar 20 2017 15:43:49 GMT-0300 (Hora oficial do Brasil)  
  
> data.toLocaleString()  
< "20/03/2017 15:43:49"  
  
> data = new Date(2017,2,20)  
< Mon Mar 20 2017 00:00:00 GMT-0300 (Hora oficial do Brasil)  
  
> data.getDay()  
< 1
```

JAVASCRIPT

► Objeto String

Propriedade/Método	Exemplo
length	<pre>var nome = "Joyce"</pre> <u><code>nome.length</code></u> Resultado = 5
indexOf	<pre>var nome = "Joyce"</pre> <u><code>nome.indexOf("c")</code></u> Resultado = 3 <i>Caso não encontre : -1</i>
substr	<pre>var nome = "Joyce"</pre> <u><code>nome.substring(0,3)</code></u> Resultado = "Joy"
replace	<pre>var nome = "Joyce"</pre> <u><code>nome.replace("e", "inha")</code></u> Resultado = "Joycinha"

JAVASCRIPT

► Objeto Array

```
var pessoas = ["João", "José", "Maria", "Sebastião", "Antônio"];
```

```
    alert(pessoas[0]);  
    alert(pessoas[1]);  
    alert(pessoas[4]);
```

```
for (var i = 0; i < pessoas.length; i++) {  
    alert(pessoas[i]);  
}
```

JAVASCRIPT

► Objeto Array

Método	Descrição	Exemplo
split	Divide uma string em diversas partes.	<pre>var s = "asp, php, java" <u>linguagem = s.split(",")</u> Resultado linguagem[0] = "asp" linguagem[1] = "php" linguagem[2] = "java"</pre>
join	Contrário do split. Junta em uma string os dados presentes em uma matriz.	<pre>local[0] = "RJ" local[1] = "SP" local[2] = "AM" <u>s = local.join(",")</u> Resultado: "RJ, SP, AM"</pre>
sort	Retorna uma versão array ordenada.	<pre><u>so = local.sort()</u> Resultado so[0] = "AM" so[1] = "RJ" so[2] = "SP"</pre>

JAVASCRIPT

- ▶ Objetos inerentes às páginas HTML
 - ▶ Todos os elementos de uma página são tratados como objetos.
 - ▶ Os objetos podem ter propriedades, métodos e responder a certos eventos.

Objeto	Descrição
window	Contém propriedades que se aplicam a toda a janela.
location	Contém as propriedades da URL atual.
history	Contém as propriedades das URLs visitadas anteriormente.
document	Contém as propriedades do documento contido na janela.

Alguns Métodos de Objetos

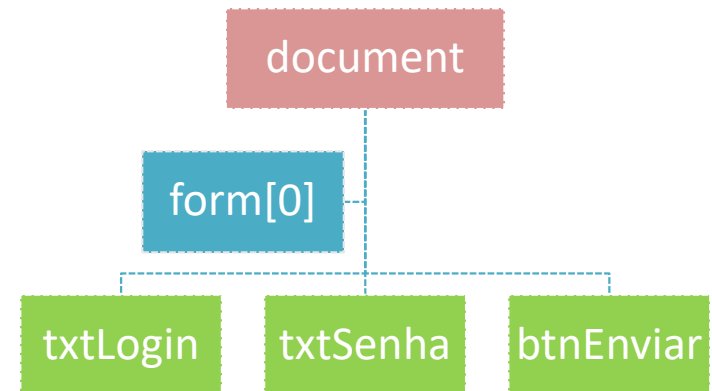
alert	Ex: <code>window.alert('Esta é uma janela de alerta!')</code>
confirm	Ex: <code>retorno = window.confirm('Deseja prosseguir?')</code>
reload	Ex: <code>location.reload()</code>
open	Ex: <code>window.open("URL")</code>
back	Ex: <code>history.back()</code>
forward	Ex: <code>history.forward()</code>
write	Ex: <code>document.write('Olá')</code>

JAVASCRIPT

- ▶ DOM – Document Object Model
 - ▶ Esses objetos estão organizados em um hierarquia

```
<form method="post" action="cadastra.html">  
Login:  
<input type="text" id="txtLogin" name="txtLogin" />  
<br/>  
Senha:  
<input type="password" id="txtSenha" name="txtSenha" />  
<br/>  
<input type="submit" id="btnEnviar" value="Cadastrar" />  
</form>
```

Árvore DOM



```
> document.forms[0].txtLogin  
< <input type="text" id="txtLogin" name="txtLogin">
```

JAVASCRIPT

▶ DOM – Document Object Model

▶ Recuperando Elementos

▶ getElementById

```
var elemento = document.getElementById("conteudo");
```

```
> document.getElementById("txtLogin")  
< <input type="text" id="txtLogin" name="txtLogin" size="50" required>
```

JAVASCRIPT

▶ DOM – Document Object Model

▶ Recuperando Elementos

▶ **getElementsByName**

```
var array = document.getElementsByName("categoria");
```

```
> document.getElementsByName("txtLogin")[0]  
< <input type="text" id="txtLogin" name="txtLogin" size="50" required>
```

JAVASCRIPT

▶ DOM – Document Object Model

▶ Recuperando Elementos

▶ Seletores CSS

❑ **querySelector**

- ❑ Retorna o primeiro elemento compatível com o seletor

❑ **querySelectorAll**

- ❑ Retorna um array de elementos compatíveis com o seletor

```
var elemento = document.querySelector("div.aprovado");  
var array = document.querySelectorAll("div.aprovado");
```

JAVASCRIPT

► DOM – Document Object Model

► Propriedade: *innerHTML*

- Retorna o conteúdo presente entre as tags de abertura e de encerramento de um elemento HTML.

```
var conteudo = elemento.innerHTML;
```

```
<h1 class="principal">Meu Título</h1>
```

```
elemento = document.querySelector("h1");  
alert(elemento.innerHTML);
```

JAVASCRIPT

► DOM – Document Object Model

► Propriedade: *value*

- ❑ Retorna o valor de um elemento HTML.

*Nome:

```
<form action="processa.php" method="get" >
  <label for="text_id" > *Nome: </label> </td>
  <input type="text" id="txtNome" name="txtNome" size="58" required >
  <input id="botao_id" type="submit" value="Enviar">
</form>
```

```
alert(document.getElementById("txtNome").value);
```

▶ Eventos

- ▶ Quaisquer ações iniciadas por parte do usuário.
 - ▶ Eventos reconhecidos pelos navegadores

Evento
onsubmit
onclick
ondblclick
onmouseover
onkeydown
onChange

JAVASCRIPT

▶ Tratamento de Eventos

▶ Sintaxe básica:

<tag atributos evento="código javascript">

```
<h1 class="principal" onclick="alert(this.innerHTML)">Meu Título</h1>
```


JAVASCRIPT

- ▶ Exemplos
 - ▶ onsubmit

Nome:	<input type="text"/>
Login:	<input type="text"/>
Senha:	<input type="password"/>
<input type="submit" value="Cadastrar"/>	

```
<form method="post" action="cadastra.html" novalidate onsubmit="return validar()">
<table>
<tr> <td>Nome:</td>
  <td><input type="text" id="txtNome" name="txtNome" size="50" required></td> </tr>
<tr> <td>Login:</td>
  <td><input type="text" id="txtLogin" name="txtLogin" size="50" required></td> </tr>
<tr> <td>Senha:</td>
  <td><input type="password" id="txtSenha" name="txtSenha" size="50" required></td> </tr>
<tr> <td colspan="2"> <input type="submit" value="Cadastrar" > </td> </tr>
</table>
</form>
```

JAVASCRIPT

- ▶ Exemplos
 - ▶ onsubmit

Função
JavaScript

```
function validar(){  
    var nome = document.getElementById('txtNome').value;  
    var login = document.getElementById('txtLogin').value;  
    var senha = document.getElementById('txtSenha').value;  
    if(nome == ""){  
        alert("Informe o nome.");  
        return false;  
    }else if(login == ""){  
        alert("Informe o login.");  
        return false;  
    }else if(senha == ""){  
        alert("Informe a senha.");  
        return false;  
    }else{  
        return true;  
    }  
}
```

JAVASCRIPT

▶ Exemplos

▶ onchange

Escolha sua cor de preferência:

```
79 <form method="post" action="cadastra.html" name="form">
80 <table border=1>
81   <tr>
82     <td>Cor:</td>
83     <td>
84       <select name="cbCor" onchange="mudaCor(this)">
85         <option value="white">Padrão</option>
86         <option value="blue">Azul</option>
87         <option value="red">Vermelho</option>
88         <option value="pink">Rosa</option>
89       </select>
90     </td>
91   </tr>
92 </table>
93 </form>
```

```
function mudaCor(element){
    document.body.style.backgroundColor=element.value;
}
```

JAVASCRIPT

► Conversão de Valores

Somando Valores	
Valor 1	<input type="text"/>
Valor 2	<input type="text"/>
Resultado	<input type="text"/>
<input type="button" value="Somar"/>	

```
function somar() {  
  
    var valor1 = parseFloat(document.getElementById('valor1').value);  
    var valor2 = parseFloat(document.getElementById('valor2').value);  
    var soma = valor1 + valor2;  
    document.getElementById('resultado').value = soma;  
  
}
```

JAVASCRIPT

- ▶ Associando Funções a Eventos
 - ▶ Outras formas
 - ▶ **addEventListener**(*evento*, *funcao*)

```
<h1>Título da Página</h1>  
=<script>  
  document.querySelector("h1").addEventListener("click", msg);  
=<function msg() {  
  alert("Seja Bem vindo(a)");  
  }  
</script>
```

JAVASCRIPT

- ▶ Associando Funções a Eventos
 - ▶ Outras formas
 - ▶ **addEventListener**(*evento*, *funcao*)

```
<h1>Título da Página</h1>

<script>
document.querySelector("h1").addEventListener(
    "click",
    function() {
        msgText("Volte Sempre!");
    });

function msgText(texto) {
    alert(texto);
}
</script>
```

JAVASCRIPT

▶ jQuery

▶ Biblioteca JavaScript

▶ Vantagem

- ▶ Maior compatibilidade de um mesmo código com diversos navegadores
- ▶ Biblioteca padrão na programação *front-end*
- ▶ Sintaxe mais fluida

```
<script src="js/jquery.js"></script>
```

▶ jQuery

▶ Document Ready Event

- ▶ Previne que um código seja executado antes da finalização do carregamento do documento no navegador.

```
$(document).ready(function(){  
  
    // jQuery methods go here...  
  
});
```


JAVASCRIPT

▶ jQuery

▶ Sintaxe Básica

▶ A Função \$

\$(selector).action()

```
$('#form').css('background', 'black');
```

```
$('.headline').hide();
```

```
$('p').text('alô :D');
```

```
$("#txtNome").val()
```

JAVASCRIPT

- ▶ jQuery
 - ▶ **Eventos**

```
$("#p").on("click", function(){  
    alert("The paragraph was clicked.");  
});
```

JAVASCRIPT

► jQuery

► Eventos

```
<nav id="menu"></nav>

<script>
  $(document).ready( function() {
    $("#menu").load("menuCSS.html");
  });
</script>
```

JAVASCRIPT

► jQuery

► Eventos

```
<div id="conteudo"></div>
```

```
<a href="#" onclick="carregaTabela()">Tabela</a>
```

```
function carregaTabela(){  
    var tabela = "<table class='table table-striped'>";  
    tabela = tabela + "<tr><td>Maria</td></tr>";  
    tabela = tabela + "<tr><td>José</td></tr>";  
    tabela = tabela + "<tr><td>Pedro</td></tr>";  
    tabela = tabela + "</table>";  
    $("#conteudo").html(tabela);  
}
```

JAVASCRIPT

▶ jQuery

▶ Exercício

- ▶ Verificar se todos os campos foram preenchidos antes de exibir o resultado
- ▶ Usar arquivo externo (.js)
- ▶ Usar sintaxe JQuery

Somando Valores	
Valor 1	<input type="text"/>
Valor 2	<input type="text"/>
Resultado	<input type="text"/>
<input type="button" value="Somar"/>	

JAVASCRIPT

▶ Armazenamento na Web

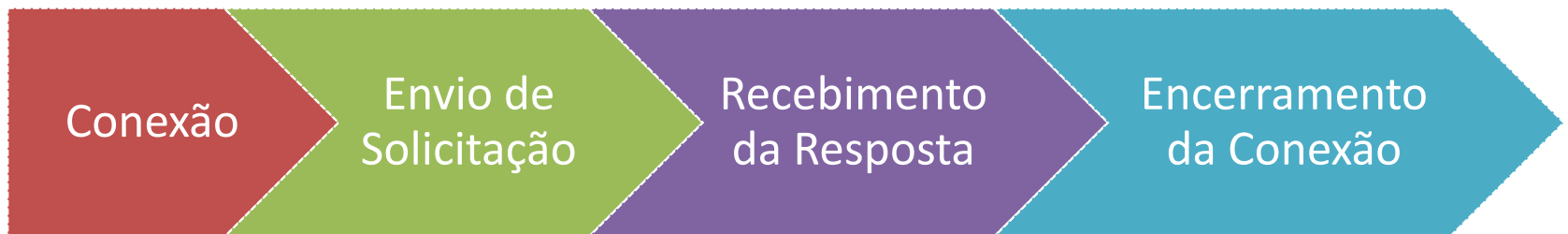
▶ Requisição HTTP

▶ Ciclo bem definido

- Considera cada requisição como uma transação independente.

▶ *Stateless*

- Não prevê armazenamento de dados entre as requisições.
 - O servidor web não identifica se duas requisições vieram de um mesmo navegador.
 - O navegador não faz gerenciamento em memória para que dados/mensagens sejam compartilhados entre requisições.



JAVASCRIPT

► Armazenamento na Web

► *Cookies x Sessions*

- Permitem que o servidor web troque informações de estado com o navegador do usuário
 - ❑ Assim, dados podem ser armazenados/recuperados entre diferentes requisições.
- Exemplo
 - ❑ Carrinho de compras, preferências do usuário



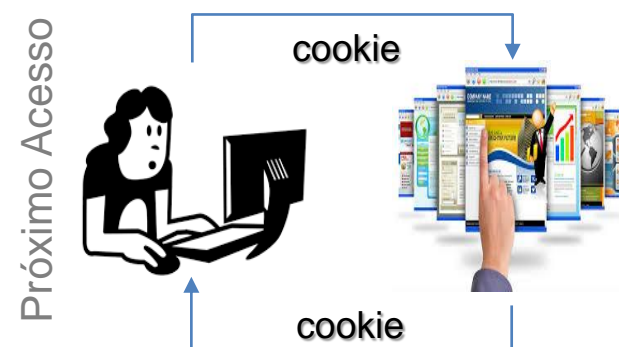
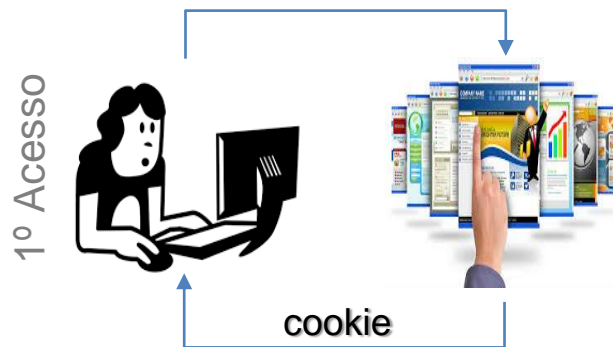
JAVASCRIPT

► Armazenamento na Web

► Cookies

► Armazenamento do lado do **cliente (disco)**.

- Pequenos arquivos que são gravados na máquina do usuário ao acessar um site, e são reenviados a este site quando novamente visitado.



- A recuperação se dá por origem (domínio e protocolo)
 - ❑ Pode ser compartilhado em subdomínios

JAVASCRIPT

▶ Armazenamento na Web

▶ Cookies

- ▶ Armazenamento temporário ou permanente.

▶ Observações

- ☐ Os cookies são inseridos no cabeçalho HTTP a cada nova requisição.
 - ☐ Pode comprometer a performance da aplicação
 - ☐ Vulnerabilidade de Segurança
- ☐ Você não deve armazenar dados sensíveis no cookie.
 - ☐ Diferentes usuários no mesmo computador podem ler/utilizar o mesmo cookie.
- ☐ Limitações dos Navegadores:
 - ☐ A maioria não é amistosa ao uso de cookies
 - ☐ Quantidade (20 por domínio – 300 no geral)
 - ☐ Espaço de armazenamento (4 KB – 4000 caracteres)

JAVASCRIPT

▶ Armazenamento na Web

▶ Sessions

- ▶ Armazenamento do lado do **servidor (memória)**.
- ▶ É possível guardar dados sensíveis
 - Não há armazenamento físico de dados
 - Cada navegador tem sua própria sessão
- ▶ Armazenamento temporário
 - Os dados são perdidos após o fechamento do navegador.
- ▶ Se o HTTP é *stateless*, como o servidor reconhece de quem de fato é uma sessão?
 - Ao iniciar uma sessão, é enviado um cookie para o navegador, com um valor único que corresponde à sessão aberta no servidor.

▶ HTML 5 WebStorage

- ▶ Alternativa para armazenar dados no cliente.
 - ▶ Mais seguro e confiável do que o uso de cookies
 - ❑ Dados não são incluídos a cada requisição HTTP
 - ❑ Dados não são compartilhados por subdomínios
 - ❑ Permite armazenar dados com maior sigilo
 - ▶ Maior capacidade de armazenamento
 - ❑ Até 10 MB por domínio

JAVASCRIPT

▶ HTML 5 WebStorage

▶ Objetos para armazenar dados no cliente

▶ *window.localStorage*

- Armazena dados sem expiração de data

▶ *window.sessionStorage*

- Armazena dados para uma sessão do usuário
 - *Os dados são perdidos quando o usuário fecha o navegador*

JAVASCRIPT

▶ HTML 5 WebStorage

- ▶ Verificar compatibilidade do navegador

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

```
if (typeof(Storage) !== "undefined") {  
    // Code for localStorage/sessionStorage.  
} else {  
    // Sorry! No Web Storage support..  
}
```

JAVASCRIPT

▶ HTML 5 WebStorage

▶ *window.localStorage*

▶ *localStorage.setItem(chave, valor)*

- ❑ Armazena um valor no local storage.
- ❑ Cada valor é referenciado por uma chave (nome).

▶ *localStorage.getItem(chave)*

- ❑ Recupera um valor armazenado a partir da sua chave.

JAVASCRIPT

▶ HTML 5 WebStorage

▶ *window.localStorage*

▶ Criando/Recuperando/Removendo

```
// Cria um item "usuario" com valor "Joyce"
window.localStorage.setItem('usuario', 'Joyce');

// Em outra página ou aba, recupera esse item
var usuario = window.localStorage.getItem('usuario');

// Remove o item
window.localStorage.removeItem('usuario');
```

JAVASCRIPT

▶ HTML 5 WebStorage

▶ *window.sessionStorage*

▶ Criando/Recuperando/Removendo

```
// Cria um item "usuario" com valor "Joyce"
window.sessionStorage.setItem('usuario', 'Joyce');

// Em outra página ou aba, recupera esse item
var usuario = window.sessionStorage.getItem('usuario')

// Remove o item
window.sessionStorage.removeItem('usuario')
```


JAVASCRIPT

► HTML 5 WebStorage

- O WebStorage não armazena objetos e *arrays* em JavaScript.
 - Ele converte o objeto/array para String.

```
produtos = [];  
produtos[0] = "Livro 1";  
produtos[1] = "Livro 2";  
  
sessionStorage.setItem("listaProdutos", produtos);  
produtos = sessionStorage.getItem("listaProdutos");
```

```
> produtos  
↵ "Livro 1,Livro 2"  
  
> typeof(produtos)  
↵ "string"
```



▶ Linguagens de Representação de Texto

▶ São utilizadas como modelos para:

- ▶ Armazenamento de informações no formato texto
- ▶ Transmissão de informações em diferentes plataformas

▶ Exemplos

▶ XML: *eXtensible Markup Language*

- Linguagem de Marcação extensível

▶ JSON: *JavaScript Object Notation*

- Notação de Objetos JavaScript



▶ Linguagens de Representação de Texto

▶ XML == JSON

- ▶ Os dois modelos representam informações no formato texto.
- ▶ Ambos possuem natureza auto-descritiva
- ▶ Concentração na estrutura e não na aparência
- ▶ São independentes de linguagem
- ▶ Ambos são capazes de representar informação complexa
 - ❑ Arrays
 - ❑ Objetos Compostos
 - ❑ Relações de Hierarquia
 - ❑ Atributos Multi-Valorados



► XML

- Baseado em tags
- Segue um modelo hierárquico de formatação
- Pode ser validada por um esquema
 - DTD ou XML Schema
- Estrutura
 - Cabeçalho
 - Raiz
 - Filhos

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<produtos>
  <produto>
    <codigo>1</codigo>
    <nome>Livro A</nome>
    <valor>100.00</valor>
  </produto>
  <produto>
    <codigo>2</codigo>
    <nome>Livro B</nome>
    <valor>150.00</valor>
  </produto>
</produtos>
```



▶ JSON

- ▶ Mais simples e compacto que o XML
 - ▶ Se baseia na notação **ATRIBUTO : VALOR**

```
{ nome : "Livro A" }
```

▶ Objetos

Produto

```
{  
  "codigo" : 1,  
  "nome" : "Livro A",  
  "valor" : 100.00  
}
```

Lista de Produtos

```
[{  
  "codigo" : 1,  
  "nome" : "Livro A",  
  "valor" : 100.00  
},  
{  
  "codigo" : 2,  
  "nome" : "Livro B",  
  "valor" : 150.00  
}]
```



▶ Linguagens de Representação de Texto

▶ XML != JSON

- ▶ JSON não é uma linguagem de marcação. XML é.
- ▶ XML pode ser validado por um esquema. JSON não.
- ▶ XML permite execução de instruções de processamento. JSON não.
- ▶ JSON é tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações.
 - Bancos de dados inteiros armazenados em XML e estruturados em SGBD's XML nativo.



▶ JSON

▶ *JSON.stringify(objeto)*

- ▶ Transforma um objeto em String com formato reconhecido pelo JSON.

▶ *JSON.parse(objeto)*

- ▶ Transforma um item no formato JSON para o seu formato original.

JAVASCRIPT

► HTML 5 WebStorage

► JSON

```
produtos = [];  
produtos[0] = "Livro 1";  
produtos[1] = "Livro 2";  
  
sessionStorage.setItem("listaProdutos", JSON.stringify(produtos));  
produtos = JSON.parse(sessionStorage.getItem("listaProdutos"));
```

```
> typeof(produtos)  
< "object"  
  
> produtos  
< ["Livro 1", "Livro 2"]  
  
> produtos[0]  
< "Livro 1"
```


JAVASCRIPT

▶ HTML 5 WebStorage

▶ JSON

```
produto = JSON.stringify({  
    Codigo    : 10001,  
    Nome      : "Livro 1",  
    Valor     : 100.00  
});  
  
sessionStorage.setItem("produto", produto);  
produto = JSON.parse(sessionStorage.getItem("produto"));
```

```
> produto  
< ▶ Object {Codigo: 10001, Nome: "Livro 1", Valor: 100}  
  
> produto.Nome  
< "Livro 1"
```

JAVASCRIPT

► HTML 5 WebStorage

► JSON

```
produto = JSON.stringify({  
    Codigo    : document.getElementById("codigo").value,  
    Nome      : document.getElementById("nome").value,  
    Valor     : document.getElementById("valor").value  
});
```

JAVASCRIPT

► HTML 5 WebStorage

► *Estudo de Caso: Simulando Carrinho de Compras*

Adicionar Produto ao Carrinho

Esvaziar Carrinho

Carrinho Vazio!!

JAVASCRIPT

► Estudo de Caso: *Simulando Carrinho de Compras*

```
<body>
  <input type="button" value="Adicionar Produto ao Carrinho" id="btnAdicionar" />
  <input type="button" value="Esvaziar Carrinho" id="btnExcluir" />
  <br><br>
  <div id="itensCarrinho"></div>
</body>
```

```
<script type="text/javascript">

  document.getElementById("btnAdicionar").addEventListener("click", adicionarProduto );
  document.getElementById("btnExcluir").addEventListener("click", esvaziarCarrinho);

  listarProdutos();

</script>
```

JAVASCRIPT

► Estudo de Caso: *Simulando Carrinho de Compras*

```
function listarProdutos() {  
    if(!localStorage.getItem("carrinho")){  
        document.getElementById("itensCarrinho").innerHTML = "Carrinho Vazio!!";  
    }else{  
        itens = "";  
        //exibindo informações do carrinho  
        carrinho = JSON.parse(localStorage.getItem("carrinho"));  
        for(i=0; i < carrinho.length; i++){  
            itens = carrinho[i] + "<br>" + itens;  
        }  
        document.getElementById("itensCarrinho").innerHTML = itens;  
    }  
}
```

```

function adicionarProduto(){
    do{
        produto = "";
        if(!localStorage.getItem("carrinho")){
            //incluindo produto no carrinho (array)
            produto = prompt("Informe o produto ou FIM:");
            if(produto == "FIM") break;
            //criando carrinho (array) de produtos
            carrinho = [];
            carrinho[0] = produto;
            //incluindo carrinho (array) na sessão
            localStorage.setItem("carrinho", JSON.stringify(carrinho));
        }else{
            //-- atualizando carrinho --//
            //adicionando novo produto ao carrinho
            produto = prompt("Informe o produto ou FIM:");
            if(produto == "FIM") break;
            //recuperando carrinho sessão
            carrinho = JSON.parse(localStorage.getItem("carrinho"));
            //recuperando quantidade produtos no carrinho
            qtde = carrinho.length;
            carrinho[qtde] = produto;
            //atualizando carrinho da sessão
            localStorage.setItem("carrinho", JSON.stringify(carrinho));
        }
    }while(produto != "FIM");
    listarProdutos();
}

```

JAVASCRIPT

► *Estudo de Caso: Simulando Carrinho de Compras*

```
function esvaziarCarrinho() {  
    localStorage.removeItem("carrinho");  
    listarProdutos();  
}
```

JAVASCRIPT

- ▶ jQuery + JSON
 - ▶ Criando conteúdo JSON

```
produto = JSON.stringify({  
    Codigo    : $("#codigo").val(),  
    Nome      : $("#nome").val(),  
    Valor     : $("#valor").val()  
});
```


JAVASCRIPT

► jQuery + JSON

► Processando conteúdo JSON

```
function loadDoc() {  
    listaProdutos = ' [{  
        "codigo" : 1,  
        "nome" : "Livro A",  
        "valor" : 100.00  
    },  
    {  
        "codigo" : 2,  
        "nome" : "Livro B",  
        "valor" : 10.00  
    } ]';  
    listaProdutos = $.parseJSON(listaProdutos);  
    produtos = "";  
    $.each(listaProdutos, function(indice, produto) {  
        var nome = produto.nome;  
        produtos = produtos + nome + " <br> ";  
    });  
    $("#lista").html(produtos);  
}
```

JAVASCRIPT

► jQuery + JSON

► Processando conteúdo JSON externo

```
function loadDocExternal() {  
    $.getJSON('arquivo_JSON.json', function(JSONDoc){  
        var produtos = "";  
        $.each(JSONDoc, function(indice, produto){  
            var nome = produto.nome;  
            produtos = produtos + nome + " <br> ";  
        });  
        $("#lista").html(produtos);  
    });  
}
```

```
[{  
  "codigo" : 1,  
  "nome"   : "Livro A",  
  "valor"  : 100.00  
},  
{  
  "codigo" : 2,  
  "nome"   : "Livro B",  
  "valor"  : 150.00  
}]
```

arquivo_JSON

JAVASCRIPT

- ▶ MISSÃO: Usando jQuery e JSON
 - ▶ (I) Agenda Telefônica
 - ▶ (II) Carrinho de Compras