



**GOVERNO DO
ESTADO DO CEARÁ**

Secretaria da Educação

**ESCOLA ESTADUAL DE
EDUCAÇÃO PROFISSIONAL - EEEP**
ENSINO MÉDIO INTEGRADO À EDUCAÇÃO PROFISSIONAL

CURSO TÉCNICO EM INFORMÁTICA

JAVA SCRIPT/ PHP



GOVERNO DO ESTADO DO CEARÁ

Secretaria da Educação

Governador
Cid Ferreira Gomes

Vice Governador
Domingos Gomes de Aguiar Filho

Secretária da Educação
Maria Izolda Cela de Arruda Coelho

Secretário Adjunto
Maurício Holanda Maia

Secretário Executivo
Antônio Idilvan de Lima Alencar

Assessora Institucional do Gabinete da Seduc
Cristiane Carvalho Holanda

Coordenadora da Educação Profissional – SEDUC
Andréa Araújo Rocha



GOVERNO DO ESTADO DO CEARÁ

Secretaria da Educação

Coordenação Técnica Pedagógica

Renanh Gonçalves de Araújo

Equipe de Elaboração

Adriano Gomes da Silva
Cíntia Reis de Oliveira
Fernanda Vieira Ribeiro
Francisco Aislan da Silva Freitas
João Paulo de Oliveira Lima
Liane Coe Girão Cartaxo
Mirna Geyla Lopes Brandão
Moribe Gomes de Alcântara
Niltemberg Oliveira Carvalho
Paulo Ricardo do Nascimento Lima
Renanh Gonçalves de Araújo
Renato William Rodrigues de Souza

Colaboradores

Maria Analice de Araújo Albuquerque
Maria Danielle Araújo Mota
Sara Maria Rodrigues Ferreira Feitosa

JavaScript / PHP



MANUAL DO (A) ALUNO (A)

Janeiro / 2013
FORTALEZA/CEA

Sumário

Apresentação	6
Aula 1. Entendendo o JavaScript	8
1.1. JavaScript -Introdução	8
1.2. JavaScript – Variáveis.....	10
1.3. JavaScript – Operadores	12
1.3.1. Operadores Matemáticos	12
1.3.2. Operadores de Comparaçāo.....	13
1.3.3. Operadores Lógicos	13
1.3.4. Operadores (Cálculo de IMC)	14
1.4. JavaScript - Estruturas de Controle.....	15
1.4.1. Estrutura Condicional (if/else).....	15
1.4.2. Estrutura Condicional (switch).....	17
1.4.3. Estrutura de Loop (for).....	18
1.4.4. Estrutura de Loop (for in).....	19
1.4.5. Estrutura de Loop (while).....	19
1.5. JavaScript – Funções.....	20
1.6. JavaScript - Objeto Array.	24
1.7. JavaScript – Eventos.	25
2. Introdução ao PHP	29
2.1. PHP –Introdução.....	29
2.2. Enviando Dados para o Servidor HTTP	30
2.3. PHP – Variáveis	35
2.4. PHP – Operadores	36
2.4.1. Operadores Matemáticos	36
2.4.2. Operadores de Comparaçāo.....	38
2.4.3. Operadores Lógicos	38
2.4.4. Operadores de Atribuição.....	38

2.4.5. Operadores (Média Aritmética)	40
2.5. PHP - Estruturas de Controle.....	41
2.5.1. Estrutura Condicional (if).....	41
2.5.2. Estrutura Condicional (switch).....	42
2.5.3. Estrutura de Loop (for).....	43
2.5.4. Estrutura de Loop (while).....	44
2.6. PHP – Definição de Funções.....	45
2.7. PHP - Arrays.....	48
3. Introdução a Framework JQuery.....	51
3.1. Instalação.....	51
3.2. Colunas e células de tabelas.....	52
3.3. Tooltips.....	55
3.4. Accordion.....	56
3.5. Datepicker.....	58
3.6. Auto-complete.....	59
3.7. Janela de dialogo modal.....	60
3.8. Menu.....	61
3.9. Abas.....	62
3.10. jQuery Mobile na prática.....	64
3.10.1. Eventos.....	69
3.10.2. Métodos e utilidades	72
3.10.3. Widgets.....	74
3.10.4. Twitter	80
3.10.5. Geolocalização.....	81
4. Formulários web.....	84
4.1. Formulário de cadastro de clientes.....	84
4.2. Formulário de Contato	88
5. Estudos de Caso.....	90

5.1.	Carrinho de compras.....	91
5.2.	Chat de atendimento.....	91
	Referências Bibliográficas	92
	Índice de Ilustrações	93

Apresentação

O manual apresenta aulas práticas e conceitos importantes para o entendimento na prática, está distribuído em cinco capítulos.

Elaborado no intuito de qualificar o processo de formação, este Manual é um instrumento pedagógico que se constitui como um mediador para facilitar o processo de ensino-aprendizagem em sala de aula.

1º Capítulo

Apresenta alguns fundamentos de JavaScript uma linguagem de programação para navegadores, onde será muito importante os conhecimentos adiquido em lógica de programação para facilitar o entendimento, neste capítulo veremos operadores da linguagem, estruturas de controle, funções, eventos e objetos array.

2º Capítulo

Introdução a linguagem de programação PHP, conhecendo seus operadores, estruturas de controle, definição de funções e arrays, no próximo semestre veremos a integração da linguagem com o banco de dados mysql.

3º Capítulo

Framework Jquery uma biblioteca JavaScript cross-browser desenvolvida para simplificar os scripts client side que interagem com o HTML, neste capítulo também abordaremos JqueryUI e Jquery Mobile conhecendo alguns widgets.

4º Capítulo

Formulários web, neste capítulo iremos desenvolver alguns formulários web, para aplicarmos os conhecimentos que estamos adquirindo no decorrer do processo de aprendizagem.

5º Capítulo

Neste capítulo vamos trabalhar com estudos de caso, que são situações em que precisamos reunir conhecimentos adquiridos para resolver situações do cotidiano, os projetos que serão desenvolvidos neste capítulo iremos continuá-los no próximo semestre.

Esperamos contribuir com a consolidação do compromisso e envolvimento de todos (professores e alunos) na formação desses profissionais.

Aula 1. Entendendo o JavaScript.

1.1. JavaScript -Introdução

JavaScript É uma linguagem de programação que roda do lado cliente, ou seja, no navegador do usuário, nos permitindo realizar determinadas ações dentro de uma página web.

Criada pela Netscape em 1995, se chamava LiveScript e visava atender necessidades de interação com a página web e validação de formulários do lado cliente.

O JavaScript tem sua sintaxe parecida com a linguagem Java que aprendemos na disciplina de lógica de programação, porém o JavaScript não descende e não tem nenhuma relação com a linguagem Java. A grande vantagem do JavaScript é a capacidade de interagir com uma página web.

É uma linguagem com tipagem dinâmica, os tipos de variáveis não precisam ser definidos no inicio do programa, é uma linguagem interpretada em vez de compilada, isso significa que os códigos são interpretados em tempo real e executados pelo navegador no momento que o usuário acessa a página web.

Escrevendo programas em JavaScript.

Para escrevermos programas em JavaScript necessitamos basicamente de um editor de texto e um navegador compatível com JavaScript, porém utilizando outros editores que nos oferecem mais facilidades na hora de escrever, como por exemplo marcar com cores diferentes as palavras reservadas, permitem que sejam abertos simultaneamente vários documentos, recursos de auto completar e outros que facilitam a vida do programador proporcionando mais agilidade na escrita do código

Abaixo temos uma tabela com a relação de alguns editores

Editor	Descrição	Link
 Dreamweaver	É um software para desenvolvimento web desenvolvido pela empresa Macromedia e comprado pela Adobe, fornece uma interface visual intuitiva para criar e editar sites em HTML e em linguagens de programação web. Nele é possível visualizar o designer da página antes da publicação com o recurso Visualização multitela, onde o desenvolvedor pode trabalhar visualizando o código e o	http://www.adobe.com/br

	designer.	
	UltraEdit é um editor HTML e editor avançado PHP, Perl, Java e JavaScript. UltraEdit é também um editor XML, incluindo um parser XML árvore estilo.	http://www.ultraedit.com/
	Bluefish é um editor livre e de código com uma variedade de ferramentas para programação em geral e para o desenvolvimento de sites dinâmicos. Suporta desenvolvimento em HTML, XHTML, CSS, XML, PHP, C, C++, JavaScript, Java, Go Google, Vala, Ada, D, SQL, Perl, ColdFusion, JSP, Python, Ruby e Shell. Bluefish está disponível para várias plataformas, incluindo Linux, Solaris e Mac OS X e Windows.	http://bluefish.openoffice.nl/index.html
	O netbeans é um ambiente de desenvolvimento integrado de código-fonte aberto gratuito para desenvolvedores de software. Todas as ferramentas necessárias para criar aplicações desktop profissionais, corporativas, Web e móveis com a plataforma Java, bem como C/C++, PHP, JavaScript e etc.	http://netbeans.org/

Agora que já conhecemos algumas IDE's para o desenvolvimento web, vamos começar a conhecer a sintaxe do JavaScript. Na disciplina de HTML/CSS aprendemos muitas coisas que serão utilizadas nesta disciplina, iremos iniciar com os documentos HTML onde serão inseridos a sintaxe da linguagem de programação JavaScript a utilização da mesma se dá sob forma de funções, onde são chamadas em determinadas situações ou em resposta a determinados eventos, estas funções podem estar localizadas em qualquer parte do código HTML, a única restrição é que devem iniciar com a declaração <**SCRIPT**> e termina com o respectivo </**SCRIPT**>, por convenção costuma-se colocar todas as funções no início do documento (entre as TAGs <**HEAD**> e </**HEAD**>, isso para garantir que o código JavaScript seja carregado antes que o usuário interaja com a Home Page), ou seja, antes do <**BODY**>.

Vamos ver o exemplo abaixo.

```

6  <html>
7   <head>
8     <title></title>
9     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10    <script type="text/javascript">
11
12    </script>
13  </head>
14  <body>
15    <script>
16
17    </script>
18  </body>
19 </html>

```

Figura 1 - Estrutura JavaScript

Neste exemplo temos a declaração JavaScript feita também na **<BODY>** para exemplificar que uma declaração JavaScript pode ser feita tanto nas TAGs **<HEAD>** e **</HEAD>** como na **<BODY>**.

1.2. JavaScript – Variáveis

Aprendemos em lógica de programação que uma variável pode assumir diferentes valores, porém ela só pode armazenar um valor a cada instante, aprendemos também que preciso definir um tipo de dados a esta variável em algumas linguagens de programação isso ocorre porque a mesma é fortemente tipada em JavaScript não é necessário declarar formalmente o tipo de dados o qual vai ser utilizado basta utilizar a instrução **var** e definir o nome da variável.

Existem dois tipos de abrangência para as variáveis:

- **Global** - Declaradas fora de uma função. As variáveis globais podem ser acessadas em qualquer parte do programa.
- **Local** - Declaradas dentro de uma função. Só podem ser utilizadas dentro da função onde foram criadas e precisa ser definida com a instrução Var.

Com relação à nomenclatura, as variáveis devem começar por uma letra ou pelo caractere sublinhado “_”, o restante da definição do nome pode conter qualquer letra ou número.

Outro ponto importante é que as variáveis são keysensitive há diferenciação entre maiúsculas e minúsculas, caracteres de acentuação e especiais.

Existem três tipos de variáveis: **Numéricas**, **Booleanas** e **Strings**, que são utilizadas da mesma forma que em lógica de programação, como já vimos que a diferença é que não precisamos declarar o tipo de dados, numéricas para armazenar números, booleanas para valores lógicos (True/False) e strings com sequência de caracteres.

As strings podem ser delimitadas por aspas simples ou duplas, a única restrição é que se a delimitação começar com as aspas simples, deve terminar com aspas simples, da mesma

forma para as aspas duplas. Podem ser incluídos dentro de uma string alguns caracteres especiais, como podemos ver na tabela abaixo;

Caracteres Especiais	Descrição
\t	Posiciona o texto a seguir, na próxima tabulação;
\n	Passa para outra linha;
\f	Insere um caractere de barra;
\b	back space;
\r	Insere um retorno.

O JavaScript reconhece ainda um outro tipo de contudo em variáveis, que é o **NULL**. Na prática isso é utilizado para a manipulação de variáveis não inicializadas sem que ocorra um erro no seu programa.

Quando uma variável possui o valor **NULL**, significa dizer que ela possui um valor desconhecido ou nulo. A representação literal para **NULL** é a string '**null**' sem os delimitadores. Quando referenciado por uma função ou comando de tela, será assim que **NULL** será representado. Observe que **NULL** é uma palavra reservada.

EXEMPLOS DE VARIÁVEIS



```

1 <html>
2   <head>
3     <title></title>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5     <script type="text/javascript">
6       <!-- Declaração de variável global
7       ano=1984;
8       v1_inicial=null;
9       //--&gt;
10
11       var idade=28;
12       var textoidade="Minha Idade é ";
13     &lt;/script&gt;
14   &lt;/head&gt;
15   &lt;body&gt;
16
17     &lt;script&gt;
18       document.write("Olá aqui ja é JavaScript");
19       document.writeln("\n"+textoidade);
20       document.write(idade);
21       document.write("\tAno de Nascimento: "+ano);
22     &lt;/script&gt;
23
24   &lt;/body&gt;
25 &lt;/html&gt;
</pre>

```

Figura 2 - Declaração de variáveis

Entendendo o código acima.

- Nas linhas 7 e 8 temos um exemplo da declaração de variáveis globais, em JavaScript não necessariamente precisamos declarar com a palavra reservada **var**, como vemos no exemplo acima.
- Nas linhas 11 e 12 utilizo a palavra reservada **var**, observamos que ao declaramos não utilizamos o tipo de dados, pois como já vimos não é necessário.

- Da linha 18 a 21 usamos o objeto **document** e o método **write()** para escrever na página HTML, passando dentro dos parênteses a variável declarada ou um texto.

1.3. JavaScript – Operadores

Os operadores são meios pelo qual incrementamos, decrementamos, comparamos e avaliamos dados dentro do computador realizamos atribuição e calculo com os valores das variáveis.

Temos os tipos de operadores abaixo:

- ✓ **Operadores Matemáticos**
- ✓ **Operadores de Comparaçāo**
- ✓ **Operadores Lógicos**

1.3.1. Operadores Matemáticos

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Incremento	++
Decremento	--
Resto da divisão	%

Exemplo 1: Neste exemplo declaramos os valores das operações em variáveis globais para serem utilizadas por todos os operadores, como vemos na imagem ao lado.

```
<script type="text/javascript">
    var valor1 =7;
    var valor2 =2;
</script>
```

```

18 <h1>Operadores Matemáticos</h1>
19 <div>
20     <h2>Soma</h2>
21     <script>
22         document.write("\nValor1: "+valor1+" + ");
23         document.writeln("\nValor2: "+valor2);
24         var resultado = valor1+valor2
25         document.writeln("\tResultado: "+resultado);
26     </script>
27 </div>

```

variáveis valor1 e valor2.

Agora estamos realizando a soma do valor das variáveis criados na tag **<head>**, este script foi criado na tag **<body>**, onde serão exibidos os valores das variáveis valor1 e valor2. Na linha 24 é declarada a variável resultado onde a mesma recebe a soma das

Exemplo 2: Neste exemplo fazemos as operações sem o uso de variáveis, utilizando os valores diretamente, na instrução `document.write()`, nas linhas 32 a 34 separamos os valores que são passados pelo método por vírgula, acrescentando tags HTML **** e a **
**, desta forma posso passar estas tags HTML e outras dentro do método junto com instruções JavaScript.

```

29 <div>
30     <h2>Operadores</h2>
31     <script>
32         document.write("A subtração de 5-2 é: ",<b>,5-2,</b>"")
33         document.write("<br>","A Multiplicação de 5*2 é: ",<b>,5*2,</b>")
34         document.write("<br>","A Divisão de 10 / 2 é: ",<b>,10/2,</b>")
35     </script>
36 </div>

```

O operador de incremento é representado pelo duplo sinal de adição “++”, já o operador de decremento é representado pelo duplo sinal de subtração “--”. Veja a seguir alguns exemplos:

Variável++ ou ++variável

Variável-- ou –variável

1.3.2. Operadores de Comparação

Os operadores na tabela abaixo comparam o conteúdo dos operandos e retornam um valor booleano **TRUE** ou **FALSE**, baseado no resultado da comparação. Abaixo a relação de operadores.

Operação	Operador
Atribuição de valores	=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=
Igualdade	==
Igual e mesmo tipo	====
Diferente	!=

Exemplos;

```

17 <script>
18     var v1_1=8;
19     var v1_1=5;
20
21     document.write("Maior que: ",v11,">",v12,"<br>Resultado: ",v11>v12);
22     document.write("<br>","Menor que: ",v11,"<",v12,"<br>Resultado: ",v11<v12);
23     document.write("<br>","Maior ou igual a: ",v11,">=",v12,"<br>Resultado: ",v11>=v12);
24     document.write("<br>","Menor ou igual: ",v11,"<=",v12,"<br>Resultado: ",v11<=v12);
25     document.write("<br>","Igual: ",v11,"==",v12,"<br>Resultado: ",v11==v12);
26     document.write("<br>","Igual e mesmo tipo: ",v11,"====",v12,"<br>Resultado: ",v11====v12);
27     document.write("<br>","Diferente: ",v11,"!=",v12,"<br>Resultado: ",v11!=v12);
28 </script>

```

1.3.3. Operadores Lógicos

São exigidos valores booleanos, como operandos, e será retornado um valor lógico, na tabela abaixo listamos a operação e o operador.

Operação	Operador
E (AND) Uma expressão E (AND) é verdadeira se todas as condições forem Verdadeiras	&&
OU (OR) Uma expressão OR (OU) é verdadeira se pelo menos uma condição for verdadeira	

NÃO (NOT)

Expressão ou condição, se verdadeira
inverte para falsa

!

Veremos exemplos de sua aplicação com as estruturas de controle, nas próximas aulas.

1.3.4. Operadores (Cálculo de IMC)

O IMC (Índice de massa corporal) é uma fórmula utilizada para verificar se um adulto ou criança está acima do peso, obeso ou abaixo do peso ideal. Para tal, necessitamos aplicar a seguinte fórmula $IMC = peso / (altura)^2$. Um especialista da área solicitou que fosse desenvolvida uma página que realize este cálculo.

```

1 <html>
2   <head>
3     <title></title>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5     <script>
6       var altura = 1.70;
7       var peso = 65;
8       var calimc = peso/(altura * altura);
9     </script>
10    </head>
11    <body>
12      <div>Calculo IMC</div>
13      <script>
14        document.write(calimc);
15      </script>
16    </body>
17  </html>
```

Inicialmente vamos praticar a utilização dos operadores em JavaScript, sem a interação com o usuário nas próximas aulas iremos iniciar esta interação.

Na imagem ao lado criamos um bloco de código na tag **<head>** onde declaramos as variáveis e usamos diversos operadores, a variável **calimc** recebe o $peso / (altura * altura)$, nesta linha 8 o que será calculado primeiro será o que está nos parênteses igual como fazemos nos cálculos na matemática, e depois a divisão do peso pela altura.



Exercício Prático



Todo o exercício deve ser feito cada um em uma página HTML, nesta etapa não utilizaremos interação com o usuário.

- 1) Crie um novo arquivo HTML e faça um script para calcular a média aritmética de 3 notas, na exibição na página deve-se apresentar as 3 notas e informar qual a média.

- 2) Faça um script para saber a idade de uma pessoa, através do ano atual e ano de nascimento, na pagina em HTML deve ser mostrado o nome de uma pessoa e sua idade.
- 3) Elabore um script para mostrar o consumo médio de um automóvel.

1.4. JavaScript - Estruturas de Controle.

Já sabemos a importância das estruturas de controle que estudamos em lógica de programação, em JavaScript também iremos utilizá-las, para controlar o fluxo de execução de blocos de instruções.

Também temos a necessidade de controlar um fluxo, que pode se repetir ou em determinadas circunstâncias nem mesmo precisar ser executado.

Na maioria das linguagens de programação temos as estruturas de controle que podem nos dar repetições simples, repetições condicionais e desvio de fluxo, que serão descritas e exemplificadas nas subseções seguintes.

Em JavaScript iremos conhecer alguns comandos para efetuar o controle de fluxo que são:

if else
Switch
For
for in
While
do while

1.4.1. Estrutura Condicional (if/else)

A estrutura de decisão “IF” normalmente vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando IF então execute determinado comando.

Na sintaxe abaixo, temos um conjunto de instruções que deve ser delimitado por chaves, quando a condição for verdadeira ele irá executar o bloco de comandos.

```
if (condição){
    bloco de comandos
}
```

Abaixo temos a sintaxe do **IF** se a condição não for verdadeira, ele irá executar o bloco de comando que está no **ELSE** que está delimitada por chaves.

```
if (condição){
    bloco de comandos
}else{
    bloco de comandos
}
```

Quando temos mais de uma condição a ser avaliada pode-se fazer o uso da instrução **ELSE IF**. Observe sua sintaxe:

```
if (condição) {
    comandos
} else if (condição2) {
    comandos
} else {
    comandos
}
```

Exemplo 1; Neste exemplo se o valor da variável **opc**, tornar verdadeira alguma condição do bloco de código, ele entra e escreve a opção escolhida.

```
13 <div>IF
14     <script>
15         var opc=1;
16
17         if(opc==1){
18             document.write("<br><b>", "Opção escolhida 1", "</b><br>");
19         }
20         if(opc==2){
21             document.write("<br><b>", "Opção escolhida 2", "</b><br>");
22         }
23         if(opc==3){
24             document.write("<br><b>", "Opção escolhida 3", "</b><br>");
25         }
26
27     </script>
```

Exemplo 2; Agora utilizamos o **IF/ELSE** onde se a condição não for verdadeira ele executa a instrução **ELSE**.

```
31 <div>IF ELSE
32     <script>
33         var idade=18;
34         var maiorIdade=18;
35         if (idade>=maiorIdade){
36             document.write("<br><p>",idade," anos, Você ja é maior de idade</p>");
37
38         }else{
39             document.write("<br><p>",idade," anos, Você ainda não é maior de idade</p>");
40         }
41
42     </script>
```

Exemplo 3; Neste exemplo usamos os operadores lógicos para avaliar as duas sentenças que são passadas no IF ELSE IF.

```

47 <script>
48     var altura = 1.70;
49     var peso = 65;
50     var calimc = peso/(altura * altura);
51
52     if (calimc < 18.5) {
53         document.write(calimc+"Classificação Magreza");
54     } else if ((calimc> 18.5) && (calimc < 24.9)) {
55         document.write(calimc+" Classificação Saudável");
56
57     } else if ((calimc > 25.0) && (calimc < 29.9)) {
58         document.write(calimc+" Classificação Sobre peso");
59
60     } else if ((calimc > 30.0) && (calimc < 34.9)) {
61         document.write(calimc+" Classificação Obesidade Grau I");
62
63     } else if ((calimc > 35.0) && (calimc < 39.9)) {
64         document.write(calimc+" Classificação Obesidade Grau II (severa)");
65
66     } else if (calimc >= 40) {
67         document.write(calimc+" Classificação Obesidade Grau III (morbida)");
68     }
69
70 </script>

```

1.4.2. Estrutura Condicional (switch).

Esta instrução é bem semelhante com uma estrutura IF, porém é mais eficiente em razão de ser mais simples sua utilização e seu entendimento. Veja a sintaxe utilizada para o uso de instruções **SWITCH**:

```

switch (expressão)
{
    case 1: [bloco de comandos];
    break;

    case 2: [bloco de comandos];
    break;

    case 3: [bloco de comandos];
    break;
    .....
    default: [bloco de comandos];
}

```

Exemplo 1; Neste exemplo caso o valor da variável **opc** seja igual há algum bloco do case ele entra e executa a instrução do bloco, caso não ele executa a instrução **default**.

```

12 <div>
13   <script>
14     var opc=3;
15
16     switch (opc){
17       case 1: document.write("<br><b>","Opção escolhida 1","</b></br>");
18         break;
19
20       case 2: document.write("<br><b>","Opção escolhida 2","</b></br>");
21         break;
22
23       case 3: document.write("<br><b>","Opção escolhida 3","</b></br>");
24         break;
25
26       default: document.write("<br><b>","Opção escolhida não existe","</b></br>");
27     }
28   </script>
29 </div>

```

1.4.3. Estrutura de Loop (for).

A instrução **for** realiza uma ação até que determinada condição seja satisfeita, abaixo sua sintaxe básica:

```

for (variável = valor inicial; condição; incremento)
{
  Bloco de instruções
}

```

Na primeira sentença do **for** determina o valor inicial do laço. Normalmente é 0 ou 1, porém poderá ser especificado qualquer outro valor.

O valor especificado é atribuído em uma variável, por exemplo, **i=0, j=1**.

A **condição** determina a expressão que irá controlar o número de repetições do laço. Enquanto esta expressão for verdadeira, o laço continuará sendo executado, caso seja falso, o laço terminará. No exemplo abaixo: **i<=10**. Enquanto o valor de **i** for menor ou igual a 10, a condição é verdadeira.

O incremento determina como o laço irá contar, de um em um, dois em dois, cinco em cinco e assim por diante.

No Exemplo: **i++**. Será aumentado o valor da variável **i** a cada repetição.

Em JavaScript, a instrução **for**, utiliza ponto e vírgula para separar os argumentos igual a linguagem Java, como fazíamos em lógica de programação.

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

Exemplo 1:

```

6 <html>
7   <head>
8     <title></title>
9     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10    </head>
11   <body>
12     <div>
13       <script>
14         for (i=1 ; i<=10 ; i++){
15           document.write("<b>número:</b> "+ i + "<br>");
16         }
17       </script>
18     </div>
19   </body>
20 </html>

```

1.4.4. Estrutura de Loop (for in).

É usado para saber os nomes de propriedades e conteúdos das propriedades de objetos no JavaScript.

Esta instrução é muito usada na depuração de códigos. Por exemplo, caso uma parte do código JavaScript não esteja funcionando corretamente e existe suspeita de que existe uma falha do objeto JavaScript, o usuário poderá usar **for...in** para examinar as propriedades do objeto usado. Sua sintaxe é mostrada abaixo:

```
for (variável in objeto) {
```

bloco de comandos;

```
}
```

Exemplo 1: Neste exemplo será exibido as propriedades do arquivo HTML que será exibido pelo método **alert()**.

```
13 |     <script>
14 |         for (teste in document){
15 |             alert(teste);
16 |         }
17 |     </script>
```

1.4.5. Estrutura de Loop (while).

A instrução while realiza uma ação enquanto determinada condição for satisfeita. Sua sintaxe básica é:

```
while (condição)
```

```
{
```

Bloco de comandos

```
}
```

Exemplo; No exemplo abaixo o laço **while** irá escrever de 1 até 10 enquanto a condição for verdadeira, e a cada passo na linha 17 vai incrementar 1 a variável **num**.

```
13 |     <script>
14 |         num=1;
15 |         while(num<=10){
16 |             document.write("Número: "+num+"<br>");
17 |             num++;
18 |         }
19 |     </script>
```

A instrução do while vai repetir um bloco de comandos enquanto a condição for falsa, nesta estrutura ele inicia com os comandos e depois avalia a condição, abaixo temos a sintaxe básica desta instrução.

```
do
```

```
{
```

Bloco de comandos

```
}while(condição)
```

Exemplo: Neste exemplo fazemos um contagem do maior para o menor, o bloco de comando é executado enquanto a condição for falsa, e a cada passo decrementa um valor.

```

24      <script>
25          num2=10;
26          do {
27      document.write("Número: "+num2+"<br>");
28      num2--;
29  }while (num2>=1)
30      </script>

```

1.5. JavaScript – Funções.

Uma função é um grupo de linhas de código de programação destinado uma tarefa bem específica e que podemos se necessário, utilizar várias vezes. A utilização de funções melhora bastante a leitura do script.

Em Javascript, existem dois tipos de funções:

- As funções próprias do Javascript. Que chamamos de "métodos". Elas são associadas a um objeto bem particular como o caso do método **Alert()** com o objeto **window**.
- As funções escritas por nós para executar o nosso script.

Declarando funções

Para declarar ou definir uma função, utiliza-se a palavra reservada **function**.

A sintaxe básica de uma de função é a seguinte:

```
function nome_da_função(argumentos) {
... código de instruções ...
}
```

O nome da função segue as mesmas regras que aprendemos na lógica e nesta disciplina, relembrando (**número de caracteres indefinido, começado por uma letra pode incluir números...**).

Todos os nomes de funções num um script devem ser únicos.

É graças ao parêntese que o interpretador Javascript distingue as variáveis das funções, ao definir uma função não quer dizer que ela será executada e junto às instruções que nela contém. Só é executada quando chamamos a função, abaixo veremos um exemplo.

Exemplo 1:

```

6  <html>
7    <head>
8      <title></title>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10     <script>
11       function msn(){
12         var textomsn="Olá, Bem vindo!!!!";
13         alert(textomsn);
14       }
15     </script>
16   </head>
17   <body onload="msn()">
18   </body>
19 </html>

```

A invocação de uma função se faz de forma simples, pelo nome da função com parênteses ().

No exemplo acima temos na linha 17 o nome da função **msn()**, que esta sendo chamada na **<body>** pelo evento **onload**, que será executado quando a página for carregada.

É aconselhado inserir todas as declarações de funções no cabeçalho da página, isto é entre as tags **<HEAD>...</HEAD>**. Assim terão a certeza que as funções já estarão interpretadas, caso haja necessidade de serem invocadas no **<BODY>**, é só criar o script na mesma.

Passando valor a uma função

Podem-se passar valores ou parâmetros as funções Javascript. Assim as funções podem utilizar valores.

Para passar um parâmetro a uma função, fornece-se um nome de uma variável dentro da declaração da função.

Um exemplo simples para compreender. Está escrito abaixo uma função que insere uma caixa de aviso em que o texto pode ser alterado.



Na declaração da função, escreve-se:

```

function Exemplo(Texto) {
  alert(Texto);
}

```

O nome da variável é Texto e é definida como um parâmetro da função.

Na invocação da função, fornece-se o texto: **Exemplo ("Bom dia a todos")**, é possível passar vários valores a uma função e vários parâmetros, a lógica de função em JavaScript e em outras linguagens é a mesma.

Os parâmetros são separados por vírgulas.



```
function nome_da_função(arg1, arg2, arg3) {
    ... código de instrução ...
}
```



A sintaxe da declaração de função:

```
function Exemplo2(Texto1, Texto2){...}
```

Invocação da função:

```
Exemplo2("Bem vindo a minha página", "Bom dia a todos");
```

Para retornar um valor, basta escrever a palavra chave **return**. Por exemplo:



```
function cubo(numero) {
    var cubo = numero*numero*numero
    return cubo;
}
```

A instrução **return** é facultativa e podemos encontrar vários **return's** na mesma função.

Para explorar este valor da variável reenviada pela função, utiliza-se uma formulação do tipo **document.write(cubo(5))**.

Exemplo 1: Neste exemplo criaremos um script para calcular o IMC de uma pessoa.

```

14 |         <script type="text/javascript">
15 |             function imc() {
16 |                 var altura = document.imcform.altura.value;
17 |                 var peso = document.imcform.peso.value;
18 |
19 |                 if ((altura == "") || (peso == "")) {
20 |                     alert("É necessário indicar o seu peso e sua altura.");
21 |                     document.imcform.altura.focus();
22 |                 } else {
23 |                     var calimc = peso/(altura * altura);
24 |                     document.imcform.result.value = calimc;

```

Nessa primeira parte do script, iniciamos criando uma função que será chamada quando o usuário mandar calcular o IMC, na linha 16 e 17 temos a declaração da variável altura e peso, e atribuição do valor da tag <input>, para acessar o valor do input temos que iniciar **document**(que é o documento HTML).imcform(o nome do formulário).altura(é o nome

do input).value(atributo da tag input) é dessa forma que acessamos objetos HTML em um formulário.

Na linha 23 é realizado o calculo do IMC com base nas informações passadas.

Na imagem abaixo temos o restante do script, utilizando a instrução IF/else.

```

26     if (calimc < 18.5) {
27         alert(calimc+" Classificação Magreza");
28     } else if ((calimc> 18.5) && (calimc < 24.9)) {
29         alert(calimc+" Classificação Saudável");
30
31     } else if ((calimc > 25.0) && (calimc < 29.9)) {
32         alert(calimc+" Classificação Sobre peso");
33
34     } else if ((calimc > 30.0) && (calimc < 34.9)) {
35         alert(calimc+" Classificação Obesidade Grau I");
36
37     } else if ((calimc > 35.0) && (calimc < 39.9)) {
38         alert(calimc+" Classificação Obesidade Grau II (severa)");
39
40     } else if (calimc >= 40) {
41         alert(calimc+" Classificação Obesidade Grau III (morbida)");
42     }
43 }
44
45 </script>

```

Na tag input usamos o evento **onclick** para chamar a função **imc()**.

The screenshot shows a web page with the URL `localhost/aula_php_js_nb/aula_js/calc_imc_1.html`. The page contains a form with the following fields:

- Altura:** A text input field containing `1.78`.
- Peso:** A text input field containing `87`.
- Calcular:** A blue button.
- IMC = a:** A text input field containing `27.45865421`.

Figura 3 - Visualização da página para calcular IMC



Exercício Prático

- 1) Elaborar um programa no qual o sejam informados: o valor da compra e o valor pago. O programa deve calcular e apresentar o troco. Se o valor pago não for suficiente, deve-se emitir um aviso.
- 2) Elaborar um programa no qual seja informado o valor de um produto em dólar e o valor da cotação do dia. O programa deve converter e apresentar o valor em reais.

- 3) Elaborar um programa para ler uma temperatura em graus °C, converter e apresentar seu valor em fahrenheit.

1.6. JavaScript - Objeto Array.

Um ARRAY é um grupo de itens que são tratados como uma única variável. Lembre-se dos vetores em lógica de programação tem o mesmo objetivo. Um exemplo que iremos tratar em JavaScript, é o grupo de meses do ano estarem dentro de um array chamado meses.

Os elementos de um array podem ser strings, números, objetos ou outros tipos de dados.

Para que se possa declarar um array, use a seguinte sintaxe:



NomeArray = new Array(numElementos)

Agora veremos como declarar um array chamado meses e seus elementos.



Meses = new Array(12);

Outra maneira, de atribuir os valores para um array. Vejamos a sintaxe abaixo:



Meses = new Array("janeiro", "fevereiro", "março", "abril", "maio", "...);

Quando atribuído o número de elementos no array, é necessário declarar os elementos que farão parte do mesmo. Vejamos o exemplo abaixo:

```

10     <script>
11         ListaFrutas[3];
12     Meses[0]=Goiaba;
13     Meses[1]=Jambo;
14     Meses[2]=Banana;
15 </script>

```

Exemplo 1; O script abaixo apresenta a data atual no navegador.

```

15     <script>
16 // Array com os dias da semana
17 hoje=new Date();
18 semana=new Array("Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta");
19 // Array com os meses do ano
20 meses=new Array("Janeiro", "Fevereiro", "Março", "Abril",
21 "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro");
22
23 document.write("Hoje é: ",semana[hoje.getDay()], "-feira, Estamos no mês de ",meses[hoje.getMonth()]);
24
25 </script>

```

Neste exemplo declaramos uma variável chamada hoje para receber os valores de data, criamos dois arrays, um chamado **semana** e o outro chamado **meses**.

Cada um dos arrays possui como conteúdo os dias da semana e os meses do ano, utilizamos o objeto **document.write** que apresentará a variável hoje com o array, correspondente da variável semana de acordo com seu método **getDay()**, apresentando o valor especificado do dia da semana, ocorrendo o mesmo com a variável meses para os meses do ano.

Exemplo 2; Neste outro exemplo de utilização dos arrays, faremos que seja criado vários campos de formulário, o programa pergunta ao usuário a quantidade de campos do tipo input, e depois retorna a criação da quantidade de campos informado.

```

13 <form name="form1">
14     <br>
15     <script>
16         nome=prompt("digite a quantidade","");
17         for(i=0;i<nome;i++){
18             document.write("<br>Nome", [i],":<input type=text name=campo"+[i], ">");
19         }
20     </script>
21     <br>
22     <input type="button" value="ok" onClick="alert(form1.campo3.value)">
23 </form>

```

Na linha 16: Declaramos a variável **nome** que vai receber o valor da entrada do usuário obtido pela função **prompt()**.

Na linha 17: Foi criado um laço **for** que caso o valor da variável **i**, for menor que a quantidade referenciada na variável **nome**, será incrementado o valor de **nome** dentro da variável **i**.

Para a execução do laço foi definido que será criado no documento atual um campo de formulário, do tipo texto e a variável de cada campo criado que aqui chamada de campo, receberá cada uma o valor de **i** cada vez que o laço se repete.

Com isto serão criados os campos cada um nomeado da seguinte forma:

Se o usuário informar 5 campos, serão criados cinco campos cada um chamado de: campo0, campo1, campo2, campo3, campo4. Lembre-se que um array sempre se inicia a partir de 0.

Na linha 22: Criamos fora do script um botão de formulário que ao clicar sobre ele, será exibido em um caixa de alerta o valor que o usuário digitou em um determinado campo.

1.7. JavaScript – Eventos.

Os eventos em JavaScript, são associados as funções, aos métodos e aos formulários, abrem uma grande porta para interatividade das páginas, EVENTOS são quaisquer ações iniciadas por parte do usuário.

Sua utilização se dá como atributos da linguagem HTML, ou seja, dentro das próprias Tag's HTML. Sua sintaxe tem a seguinte formação;

<TAG nomeEvento="Instruções JavaScript">

- ✓ Onde é apresentado **TAG** é uma instrução da linguagem HTML.
- ✓ Onde é **evento** é o nome do evento gerado da linguagem JavaScript.
- ✓ Onde “**Instruções JavaScript**” serão as instruções JavaScript à serem executadas. Elas estarão sempre entre aspas.

Quando há mais de um comando JavaScript a ser executado para o mesmo evento estes deverão estar separados por ponto e vírgula (;), conforme mostrado no exemplo abaixo:

```
<TAG nomeEvento="JavaScript1;JavaScript2;JavaScript3">
```

Abaixo veremos diferentes eventos implementados em JavaScript.

EVENTOS	MANIPULADOR	DESCRIÇÃO
Clik	onclik	Quando o usuário clica sobre um botão, um link ou outro elemento.
Load	onload	Quando a página é carregada pelo browser.
Unload	onunload	Quando o usuário sair da página.
MouseOver	onmouseover	Quando o usuário coloca o ponteiro do mouse sobre um link ou outro elemento.
MouseOut	onmouseout	Quando o ponteiro do mouse não está sobre um link ou outro elemento.
Focus	onfocus	Quando um elemento de formulário tem o foco, isto é, está ativo.
Blur	onblur	Quando um elemento de formulário perde o foco, isto é, quando deixa de estar ativo.
Change	onchange	Quando o valor de um campo de formulário é modificado.
Select	onselect	Quando o usuário selecionar um campo dentro de elemento de formulário.
Submit	onsubmit	Quando o utilizador clica sobre o botão Submit para enviar um formulário.
keyDown	onkeydown	Ocorre quando uma tecla é mantida pressionada.
keyPress	onkeypress	Ocorre quando uma tecla é pressionada.
keyUp	onKeyUp	Ocorre quando uma tecla é solta.

Vejamos alguns exemplos de eventos.

Neste exemplo vamos criar um script fora da página, abra o seu editor preferido para JavaScript, abaixo temos as três funções que vamos fazer em JavaScript puro.

```

1  function evento_entrar(){
2      alert("Seja Bem Vindo!!!");
3  }
4  function evento_sair(){
5      alert("Até a próxima visita!!!");
6  }
7
8  function eventoClique(){
9      document.getElementById('id2').style.color='red';
10 }
11

```

Figura 4 - Script em JavaScript

Acima temos três funções, a função **eventoClique** acessa um elemento de uma página HTML pelo id do elemento, para mudar a cor deste elemento.

Abaixo temos a estrutura da página HTML, observe a linha 10, nesta através do elemento **src** referencia o local onde o script está, neste exemplo o mesmo encontra-se no diretório **aula_funcoes/** e **js_eventos.js** é o nome do script.

```

6  <html>
7  <head>
8      <title></title>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10     <script src="aula_funcoes/js_eventos.js"></script>
11 </head>
12 <body onload="evento_entrar()" onunload="evento_sair()">
13     <h1 id="id2">Texto 1</h1>
14     <button type="button" onclick="eventoClique()">
15         Clique aqui!
16     </button>
17 </body>
</html>

```

Figura 5 - Estrutura HTML, chamando script.

onClick

Este evento é o mais clássico, o evento será disparado no momento do clique do botão do mouse.

Na linha 14 vemos a utilização deste evento, após referenciar o script na tag **<head>**, basta realizar a chamada do método que esta no arquivo de script.

onLoad e onUnload

O evento Load aparece quando a página acaba de ser carregada. O inverso, Unload aparece quando o usuário sai da página.

Os eventos onLoad e onUnload são utilizados sob forma de atributos da tags **<BODY>** ou **<FRAMESET>**. Pode-se assim escrever um script para desejar as boas vindas na abertura de uma página e uma mensagem de adeus ao sair desta, como vimos na imagem anterior.

Onchange

Neste exemplo quando o valor do campo **fname** que esta na linha 20, é modificado através deste evento chama a função **alter_txt()**, que vai alterar o valor do campo para letras maiúsculas.

```

10      <script>
11          function alter_txt()
12          {
13              var x=document.getElementById("fname");
14              x.value=x.value.toUpperCase();
15          }
16      </script>
17  </head>
18  <body>
19      <div>
20          Digite seu nome: <input type="text" id="fname" onchange="alter_txt()">
21          <p>Clique fora da caixa de texto para alterar o texto em maiúsculo</p>
22      </div>
23  </body>

```

Ao observar as linhas do nosso script, percebemos que na linha 13 é declarado uma variável chamada **x**, esta vai receber o elemento **fname**, depois na linha 14 vemos que a variável **x.value** vai receber o valor de **x.value.toUpperCase()**, o método **toUpperCase()** é responsável por passar o valor do elemento para maiúsculas.

onmouseOver e onmouseout

```

10      <script>
11          function mouse_in(){
12              document.getElementById("texto").style.color="blue";
13          }
14          function mouse_out(){
15              document.getElementById("texto").style.color="black";
16          }
17      </script>
18  </head>
19  <body>
20
21      <h1 id="texto" onmouseover="mouse_in()" onmouseout="mouse_out()"> Texto 1</h1>
22
23  </body>

```

O evento **onmouseOver** executa-se quando o cursor passa por cima (sem clicar) de um link, de uma imagem ou de outros elementos. Este evento é bastante prático, por exemplo, na imagem acima vemos um exemplo deste evento, ao passar o cursor do mouse sobre a palavra Texto 1 (sem clicar no link), é acionado a função **mouse_in()** na linha 21, este evento foi criado nas linhas 11 a 13, como podemos ver bem simples, o elemento com id **texto** foi criado dentro da tag **<h1>**, é através do id do elemento que é aplicado o evento.

No evento **onmouseout**, geralmente associado um **onmouseOver**, executa-se quando o cursor sair da zona sensível do elemento, a aplicação deste evento esta na imagem acima no elemento **texto**, o efeito neste exemplo foi, quando o cursor estiver na área do objeto a cor será mudada para azul, quando sair volta para a cor preta.

2. Introdução ao PHP

2.1. PHP –Introdução.

PHP é uma sigla que significa PHP HyperText Preprocessor. O PHP é uma linguagem de código-fonte aberta, muito utilizada na Internet e especialmente criada para o desenvolvimento de aplicativos Web, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links.

A diferença de scripts CGI escritos em outras linguagens como Perl ou C é que ao invés de escrever um programa com um monte de instruções para imprimir HTML, você escreve um arquivo HTML com algum código inserido, por exemplo, imprimir um texto. O código PHP é delimitado por tags iniciais e finais que lhe permitem pular pra dentro e pra fora do “modo PHP”, como vemos abaixo:

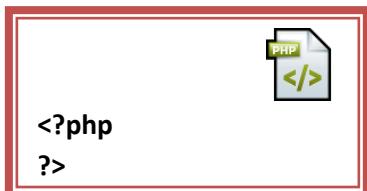


Figura 6 - Tag inicial do PHP

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php

    ?>
  </body>
</html>
```

Figura 7 - Tag PHP dentro do HTML

Podemos escrever script em PHP dentro da estrutura HTML ou com as instruções do PHP puro, para escrevermos iremos utilizar um editor, mais qual? Podemos utilizar qualquer um dos editores que conhecemos no inicio da disciplina, ou outro de sua preferência.

Para testar scripts PHP é necessário um servidor com suporte a esta tecnologia, Normalmente, o mais utilizado é o Apache. O banco de dados mais utilizado com os scripts PHP é o MySQL, alem deste a linguagem tem suporte a um grande número de bancos de dados, como dBase, Interbase, MS-SQL Server, Oracle, Sybase, PostgreSQL e vários outros, com relação a PHP e a banco de dados será abordado no próximo semestre onde faremos a integração da linguagem e o banco de dados. Além do suporte a banco de dados

podemos trabalhar com outros serviços através de protocolos tais como IMAP, SNMP, NNTP, POP3 e principalmente HTTP. Ainda é possível abrir *sockets* e interagir com outros protocolos.

Com relação ao ambiente para desenvolver script em PHP veremos mais adiante em ambiente de desenvolvimento em PHP.

História da Linguagem PHP

A linguagem PHP foi desenvolvida durante o outono de 1994 por **Rasmus Lerdorf**. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua *home-page* apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como “**Personal Home Page Tools**” (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava alguns macros e alguns utilitários que rodavam “por trás” das *home-pages*: um livro de visitas, um contador e algumas outras coisas.

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de **PHP/FI**, o “FI” veio de outro pacote escrito por Rasmus que interpretava dados de formulários HTML (**Form Interpreter**). Ele combinou os scripts do pacote *Personal Home Page Tools* com o FI e adicionou suporte a mySQL, nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

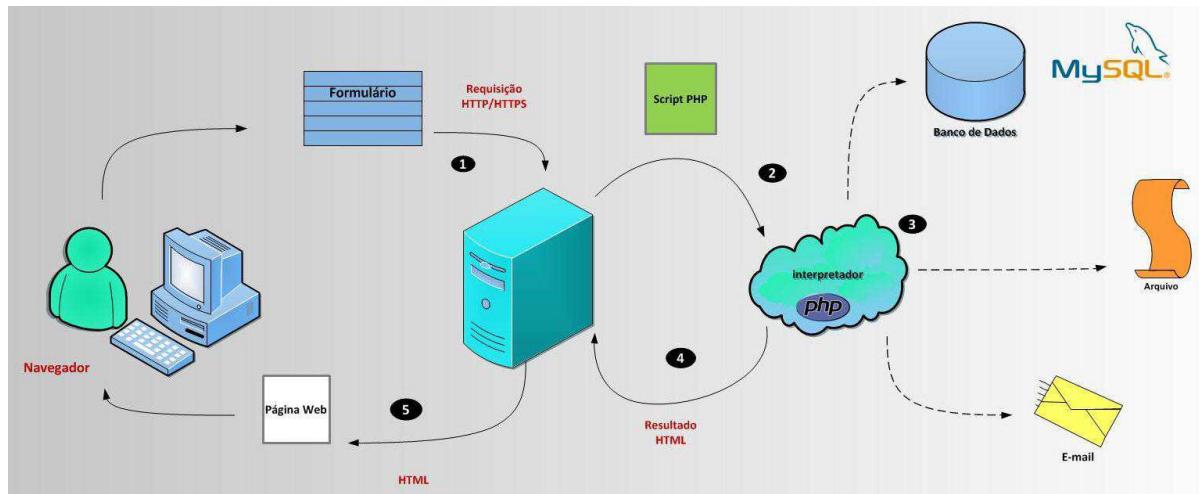
O interpretador foi reescrito por **Zeev Suraski** e **Andi Gutmans**, e esse novo interpretador foi a base para a versão 3, chamada de “**PHP: Hypertext Preprocessor**”.

O lançamento do PHP4, ocorrido no ano 2000, trouxe muitas novidades aos programadores de PHP. Uma das principais foi o suporte a sessões, bastante útil pra identificar o cliente que solicitou determinada informação.

A versão mais nova da linguagem é PHP 5, que da suporte a orientação a objetos, o PHP não é uma linguagem orientada a objetos e sim da suporte a orientação a objetos.

2.2. Enviando Dados para o Servidor HTTP

Desenvolver para web consiste basicamente em receber dados de usuário processá-los e enviar a resposta dinâmica, após ser enviada ao servidor onde ficam armazenadas as páginas em PHP e outras linguagens web é encerrado o contato entre o servidor e o cliente. Vamos aprender um pouco mais de como funciona, o que precisaremos para montar nosso servidor e testar as páginas em PHP que iremos escrever.

**Figura 8 - Enviando dados ao servidor**

1. O protocolo HTTP/HTTPS provê dois principais métodos para enviar informações para o servidor web, além da URL referente ao arquivo solicitado. Esses métodos são o POST e o GET, o usuário através do formulário faz a requisição ao servidor como vemos na imagem acima.
2. O servidor que tem suporte ao PHP vai neste ponto interpretar o script PHP que veio na requisição.
3. Após o script PHP ser interpretado, o que foi solicitado na requisição vai ser processado, a requisição pode ser uma interação como o banco de dados, com um arquivo ou o envio de um email, ou ate mesmo estas operações ao mesmo tempo.
4. O resultado em HTML será enviado ao servidor.
5. O servidor por vez sua monta a resposta e apresenta ao usuário no navegador em HTML.

O método GET

O protocolo HTTP utiliza a definição do método GET, utilizado pelo browser para solicitar um documento específico.

Por exemplo: a seguinte requisição HTTP retornaria o documento "index.html", localizado no diretório do servidor chamado "meusite":

```
GET /meusite/index.html CRLF
```

Ao observar a requisição GET inicia com a palavra GET, inclui o documento solicitado e encerra com a combinação dos caracteres ***carriage return e line feed (CRLF)***.

Vamos entender melhor, você pode fazer uma requisição GET conectando diretamente em algum servidor WEB, conectando através de um programa chamado telnet, geralmente o

servidor HTTP utiliza a porta 80. A resposta será o código da página solicitada, como vemos abaixo.

```
telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
GET /index.php3
(... página solicitada ...)
Connection closed by foreign host.
```

No caso do browser é ele que trata as informações recebidas e exibe a página já formatada.

Com o método GET também é possível passar parâmetros da requisição ao servidor, que pode tratar esses valores e até alterar a resposta a depender deles, como no exemplo abaixo:

```
telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
GET /index.php3?id=0024horas&tipo=Taxi
(... página solicitada ...)
Connection closed by foreign host.
```

No exemplo são passados dois parâmetros: **id** e **tipo**. Esses parâmetros estão no formato conhecido por URLencode.

Agora já sabemos que podemos passar parâmetros utilizando o método GET, e com isso gerar páginas dinamicamente, porém este método tem pelo menos dois pontos que em determinadas circunstâncias devem ser analisados conforme a aplicação web que se está desenvolvendo:

- **1º O método GET permite uma quantidade de dados passados limitada a 1024 caracteres, o que pode gerar perda de informações em certos casos.**
- **2º É pelo fato de que as informações fazem parte da URL, todos os dados podem ser vistos pelo usuário. Isso pode ser extremamente perigoso quando informações sigilosas estão envolvidas, por exemplo, senhas.**

O método POST

É usado quando queremos enviar dados a serem gravados em um banco de dados ou uma pesquisa cujos dados sejam grandes o suficiente para não caber na URL da página.

Uma conexão ao servidor HTTP utilizando o método POST seria algo semelhante ao que segue:

```

telnet www.guia-aju.com.br 80
Trying 200.241.59.16...
Connected to www.guia-aju.com.br.
Escape character is '^]'.
POST /index.php3
Accept */
Content-type: application/x-www-form-urlencoded
Content-length:22
id=0024horas&tipo=Taxi
(... página solicitada ...)
Connection closed by foreign host.

```

Ambiente de desenvolvimento em PHP.

Agora que já sabemos como funciona a comunicação entre o servidor e o cliente, em um servidor web vamos implementar um servidor web para testar nossas aplicações, abaixo temos nossas opções que podem ser utilizadas neste manual, apresentaremos três porém existem outras que não serão abordadas.

Pacotes prontos para execução de um ambiente Apache + PHP + MySQL.



As páginas PHP devem ser salvas no diretório raiz do servidor. Para testes locais deve-se abrir o browser de internet, acessar a URL com o endereço local (<http://127.0.0.1>) ou nome do domínio (<http://localhost/>), se na sua maquina possuir um servidor web instalado será exibido uma pagina, caso contrario será exibido um página de erro, informando que não consegui estabelecer uma conexão com o servidor.

Abaixo vamos conhecer os pacotes prontos para execução do ambiente.



EasyPHP

EasyPHP é Ferramenta para web que simula um servidor um pacote pronto para execução de um ambiente Apache + PHP + MySQL, pode ser baixado no link <http://www.easypht.org>, o diretório raiz para testar o seu projeto é **c:\Arquivos de programas\EasyPHP\www**. Para acessar a página, deve-se abrir o browser de Internet e digitar-se o nome do domínio (<http://127.0.0.1>) e o nome da página com extensão **.php**. Quando o EasyPHP está sendo executado, aparece um ícone com uma letra “e” ao lado do relógio do Windows.



Wamp Server

Wamp Server é um ambiente de desenvolvimento web para Windows, permitindo que você tenha um ambiente Apache + PHP + MySQL, para instalar primeiramente teremos que fazer o download no site <http://www.wampserver.com/en/>, o Diretório raiz do servidor para testar as páginas encontra-se em **c:\Arquivos de programas\wamp\www**, este caminho é utilizado na instalação padrão.



XAMPP

No Linux podemos utilizar o Xampp que é um pacote de instalação do ambiente Apache + PHP + MySQL, podendo fazer o download no link <http://www.apachefriends.org/en/xampp-linux.html>.

No Linux podemos instalar separadamente o apache, php e o mysql com o comando apt-get via terminal.

Outra solução em pacote de instalação é o LAMP – Linux- Apache –MySQL- PHP, para instalar basta abrir um terminal e digitar **sudo apt-get install lamp-server^**.

No Linux o diretório raiz do servidor apache é em **var/www**, todos os seus diretórios de sites podem ser colocado neste diretório, depois é só chamar via navegador.

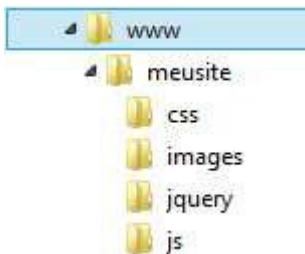
Exemplo de Script PHP

Para criar o primeiro exemplo, digite o código-fonte abaixo no seu editor e salve com o nome de **teste.php** dentro do diretório raiz do seu servidor.

```
<?php
    phpinfo();
?>
```

A função **phpinfo()**, exibe uma tabela contendo todas as constantes pré-definidas, assim como configurações da máquina, sistema operacional, servidor http e versão do PHP instalada.

Agora que já temos que organizar a nossa estrutura de diretório dos nossos sites, então vamos a uma dica muito importante, ao criar o diretório do seu site é aconselhável criar alguns diretórios, abaixo temos um exemplo de uma estrutura de diretórios que iremos adotar.



- **meusite:** Diretório do seu site, este é o raiz do seu site.
- **css:** Neste vamos armazenar as nossas folhas de estilo.
- **images:** Aqui iremos colocar as imagens que utilizaremos em nossos projetos.
- **jquery:** Neste diretório vamos armazenar todos os scripts da biblioteca jquery, que será usado nas próximas aulas.
- **js:** Este diretório será o local onde vamos guardar os script que iremos desenvolver na próximas aulas.

2.3. PHP – Variáveis

Em PHP também utilizamos variáveis, porém é diferente de como fazíamos em lógica de programação, onde aqui não precisamos declarar o tipo da variável, vamos ver como declaramos uma variável em PHP, na imagem abaixo.

```

12 <body>
13     <?php
14     // declaração de variáveis
15     $idade;
16     // Atribuição de valor a variável criada
17     $idade = 28;
18
19     $nome = "João Paulo";
20     echo 'Olá, meu nome é ', $nome, ' Minha idade é ', $idade;
21     ?>
22 </body>
```

Figura 9 - Declaração de Variáveis

Na linha 20 usamos o **echo** para escrever em uma página, podendo ser passado texto e variáveis como observaram separados por vírgulas, com ele também é possível atribuir tags HTML dentro das aspas simples.

As variáveis podem ser globais onde podemos ter acesso à mesma em qualquer parte do código, ou serem declaradas dentro de uma função, sendo que se a variável for declarada dentro de uma função ela só estará disponível dentro deste trecho de código.

Apesar de não precisarmos declarar o tipo de variável o PHP suporta vários tipos de dados:

Inteiro – Números inteiros (isto é, números sem ponto decimal).

Números de dupla precisão – Números reais (isto é, números que contêm um ponto decimal).

String – Texto entre aspas simples (' ') ou duplas ("").

Booleanos – armazenam valores verdadeiros ou falsos, usados em testes de condições.

Array – Grupo de elementos do mesmo tipo.

Objeto – Grupo de atributos e métodos.

Recurso – Uma origem de dados externa.

Nulo – Nenhum valor.

Constantes

São identificadores para valores simples. O seu conteúdo não muda durante a execução do código. Para declararmos utilizamos a função define e, por convenção, são escritas com letras maiúsculas e não usam o cifrão no início.

Declaração de uma constante

```

7 |<body>
8 |<?php
9 |define("CONSTANTE_ALO", "Alô mundo.");
10|echo CONSTANTE_ALO;
11|?>
12|</body>
```

Figura 10 - Exemplo de uma constante

As variáveis não podem ter o mesmo nome de instruções do PHP, que são chamadas de palavras chaves, abaixo uma tabela com as palavras chaves.

Palavras-chave do PHP						
and	do	for	default	include	require	true
break	else	foreach	list	virtual	return	var
case	extends	elseif	function	new	class	static
global	not	switch	xor	continue	false	if
or	while	this				

2.4. PHP – Operadores

São usados para efetuarem operações sobre as variáveis e constantes. Os tipos de operadores do PHP são:

- ✓ **Operadores Matemáticos ou aritméticos**
- ✓ **Operadores de Comparação**
- ✓ **Operadores Lógicos**
- ✓ **Operadores de Atribuição**

2.4.1. Operadores Matemáticos

São utilizados quando os operandos são números (integer ou float). Caso sejam de outro tipo, terão seus valores convertidos antes da realização da operação.

Operador	Nome
+	Adição
-	Subtração
*	Multiplicação
/	Divisão

%	Módulo ou resto da divisão
---	----------------------------

Exemplo:

```

11 <body>
12     <?php
13     $x = 2;
14     echo ($x + 2);
15     echo "<br>";
16     $x = 2;
17     echo (5 - $x);
18     echo "<br>";
19     $x = 4;
20     echo ($x * 5);
21     echo "<br>";
22     $x = 15;
23     echo ($x / 5);
24     echo "<br>";
25     $x = 10;
26     echo ($x % 8);
27     echo "<br>";
28     ?>
29 </body>
```

Incremento e Decremento

No caso de string só há um operador exclusivo:

++	Incremento
--	Decremento

Podem ser utilizados: antes ou depois da variável. Quando utilizado antes, retorna o valor da variável antes de incrementá-la ou decrementá-la. Quando utilizado depois, retorna o valor da variável já incrementado ou decrementado.

Exemplo:

```

11 <body>
12     <?php
13     $y = 5;
14     $y++;
15     echo ($y);
16     echo "<br>";
17     $y = 5;
18     $y--;
19     echo ($y);
20     echo "<br>";
21     $y = 8;
22     echo ($y);
23     ?>
24 </body>
```

Concatenação:

No caso de string só há um operador exclusivo:

.	Concatenação
---	---------------------

2.4.2. Operadores de Comparação

As comparações são feitas entre os valores contidos nas variáveis, observamos durante o curso que a lógica dos operadores é a mesma, o que esta mudando à medida que avançamos no curso o que muda em alguns casos é a sintaxe da linguagem, os operadores de comparação sempre retornam um valor booleano.

Operador	Nome	Exemplo	Resultado
<code>==</code>	Igual	<code>\$a == \$b</code>	Verdadeiro se \$a for igual a \$b
<code>!=</code>	Diferente	<code>\$a != \$b</code>	Verdadeiro se \$a não for igual a \$b
<code><></code>	Diferente	<code>\$a <> \$b</code>	Verdadeiro se \$a não for igual a \$b
<code>==</code>	Idêntico	<code>\$a === \$b</code>	Verdadeiro se \$a for igual a \$b e for do mesmo tipo.
<code>!==</code>	Não idêntico	<code>\$a !== \$b</code>	Verdadeiro se \$a não for igual a \$b, ou eles não são do mesmo tipo.
<code><</code>	Menor que	<code>\$a < \$b</code>	Verdadeiro se \$a for menor que \$b
<code>></code>	Maior que	<code>\$a > \$b</code>	Verdadeiro se \$a for maior que \$b
<code><=</code>	Menor ou igual	<code>\$a <= \$b</code>	Verdadeiro se \$a for menor ou igual a \$b.
<code>>=</code>	Maior ou igual	<code>\$a >= \$b</code>	Verdadeiro se \$a for maior ou igual a \$b.

2.4.3. Operadores Lógicos

Os operadores lógicos são utilizados para combinar expressões lógicas entre si, realizando testes condicionais.

Operador	Nome	Exemplo	Resultado
<code>AND</code>	E	<code>(10 > 7) AND (9 == 9)</code>	Verdadeiro se 10 for maior que 7 e 9 for igual a 9
<code>OR</code>	Ou	<code>(10 > 7) OR (9 == 9)</code>	Verdadeiro se 10 for maior que 7 ou 9 for igual a 9
<code>XOR</code>	Ou exclusivo	<code>(10 > 7) XOR (9 == 9)</code>	Verdadeiro se 10 for maior que 7 ou 9 for igual a 9, mas não se ambos forem verdadeiro
<code>!</code>	Negação	<code>! (10 > 7)</code>	Verdadeiro se 10 for menor que 7
<code>&&</code>	E	<code>(10 > 7) && (9 == 9)</code>	Verdadeiro se 10 for maior que 7 e 9 for igual a 9
<code> </code>	Ou	<code>(10 > 7) (9 == 9)</code>	Verdadeiro se 10 for maior que 7 ou 9 for igual a 9

2.4.4. Operadores de Atribuição

Existe um operador básico de atribuição (`=`) e diversos derivados. Sempre retornam o valor atribuído. No caso dos operadores derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado para o primeiro. A atribuição é sempre por valor, e não por referência, abaixo tabela com os operadores.

Operador	Descrição
<code>=</code>	Atribuição simples
<code>+=</code>	Atribuição com adição
<code>-=</code>	Atribuição com subtração
<code>*=</code>	Atribuição com multiplicação
<code>/=</code>	Atribuição com divisão

%=	Atribuição com módulo
.=	Atribuição com concatenação

Precedência de operadores

A tabela seguinte mostra a precedência dos operadores, da maior precedência no começo para os de menor precedência de operadores.

Operador	Descrição
- ! ++ --	Negativo, negação, incremento e decremento
* / %	Multiplicação, divisão e resto da divisão
+ - .	Adição, subtração e concatenação
> < >= <=	Maior que, menor que, maior ou igual e menor ou igual
== != <>	Igual e diferente
&&	E
	Ou
= += -= *= /= %=	Operadores de atribuição
AND	E com menor prioridade
XOR	Ou exclusivo
OR	Ou com menor prioridade

Comentários de código

É muito importante documentar o código, o código bem documentado auxilia bastante na hora de uma manutenção, ou em um trabalho de equipe.

Comentários de uma linha:

Marca como comentário até o final da linha ou até o final do bloco de código PHP – o que vier antes. Pode ser delimitado pelo caractere “#” ou por duas barras “//”.

Exemplo 1:

```
19 |     <?php echo "teste"; #isto é um teste ?>
20 |     <?php echo "teste"; //este teste também é um comentário ?>
```

Comentários de mais de uma linha:

Temos como delimitadores os caracteres “/*” para o início do bloco e “*/” para o final do comentário.

Exemplo 2:

```
22 |
23 |     <?php
24 |         echo "teste"; /* Isto é um comentário com mais
25 |             de uma linha
26 |             */
?>
```

2.4.5. Operadores (Média Aritmética).

Nesta aula vamos criar um script em PHP para calcular a média aritmética, o mesmo vai receber duas notas recebidas do formulário HTML e irá retornar a sua média.

Para esta prática crie uma página web com a extensão ***.php**, esta página deve conter os elementos iguais a página abaixo;

```

2  <html>
3    <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5      <title>Calculo de Média</title>
6    </head>
7    <body>
8      <form method="post" action="calc_media.php">
9        Nota 1: <input type="text" size="20" name="np1">
10
11       Nota2: <input type="text" size="20" name="np2">
12
13       <input type="submit" value="Calcular">
14     </form>
15   </body>
16 </html>

```

Observe que na linha 8 dentro da tag **<form>** definimos o método **post** para enviar as informações, e no **action** o script PHP que será responsável pelo calculo da média, então temos 2 arquivos um responsável pela apresentação ao usuário e outro para calcular.

Agora vamos codificar o script **calc_media.php**.

```

1 <?php
2
3 require 'form_IMC.php';
4 $np1 = $_POST['np1'];
5 $np2 = $_POST['np2'];
6 $media = ($np1 + $np2) / 2;
7 echo "Sua média é: <b>$media</b> <br>";
8 ?>

```

Entendo o Script:

Na linha 3: usamos uma função própria do PHP o **require**, ele é responsável por requerer ou retornar um script, neste exemplo usamos para a mensagem que esta na linha 7 seja escrita no formulário de apresentação, como é mostrado na imagem seguinte, sem esta instrução quando o script fosse executado o retorno seria apenas a mensagem dentro do **calc_media.php**.

Nas Linhas 4 e 5: Estamos declarando as variáveis que vão receber os valores passados pelo formulário, onde elas recebem através **\$_POST**, **np1** e **np2** são os nomes dos inputs do formulário, que os valores dos mesmos estão sendo passados para as variáveis **\$np1** e **\$np2**.

Quando definimos **\$_POST** estamos dizendo que os dados do formulário estão sendo passados através deste método, logo se no **<form>** do meu formulário esta **method="post"** para receber o post no outro script uso **\$_POST**.

Na linha 6: Realiza-se o calculo $\$media = (\$np1 + \$np2) / 2$, que na linha 7 é passado para ser devolvido a página que foi solicitada.

Abaixo vemos o resultado final desta prática.

Peso: Altura: Calcular

Sua média é: 10

Exercício Prático

Agora vamos praticar um pouco, abaixo temos duas situações, para cada uma delas temos que criar formulários web e scripts ao final exibir o resultado ao usuário. **Obs.:** Lembre-se que teremos um formulário em PHP para receber os dados e um script em PHP para os cálculos.

- 1) Crie um formulário na web que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, depois de receber estes dados informar o seu nome, o salário fixo e o salário no final do mês.
- 2) O Sr. João necessita saber o consumo médio de um automóvel, e solicitou para você desenvolver uma página web que sendo fornecida a distância total percorrida pelo automóvel e o total de combustível gasto, mostrar o consumo do automóvel. (formula Distância /combustível).

2.5. PHP - Estruturas de Controle.

As estruturas que veremos a seguir são comuns para as linguagens de programação, iremos observar que a estrutura lógica é a mesma que vimos em lógica de programação, bastando, portanto, descrever a sintaxe de cada uma delas, resumindo o funcionamento, abaixo tem alguns exemplos de cada uma dessas estruturas de controle.

Um bloco consiste de vários comandos agrupados com o objetivo de relacioná-los com determinado comando ou função. Em comandos como **if**, **for**, **while**, **switch** e em declarações de funções os blocos podem ser utilizados para permitir que um comando faça parte do contexto desejado.

Os blocos em PHP são delimitados pelos caracteres “{” e “}”, assim como na linguagem Java que iniciamos na disciplina de lógica de programação. A utilização dos delimitadores de bloco em uma parte qualquer do código não relacionada com os comandos citados ou funções não produzirá efeito algum, e será tratada normalmente pelo interpretador.

2.5.1. Estrutura Condicional (if).

Ele testa a condição e executa o comando indicado se o resultado for **verdadeiro (true)**. Abaixo veremos os exemplos em PHP:

Exemplo 1; Exemplo da instrução IF em PHP.

```

14 |         <?php
15 |         $x = 20;
16 |         if ($x > 10) {
17 |             echo("O valor da variável é maior que 10.");
18 |         }
19 |     ?>

```

O **else** é um complemento opcional para o **if**. Se utilizado, o comando será executado se a expressão retornar o valor falso. Veja o exemplo abaixo:

Exemplo 2; Exemplo da instrução IF /ELSE em PHP.

```

14 |         <?php
15 |         // put your code here
16 |
17 |         $x = 5;
18 |         if ($x > 10)
19 |         {
20 |             echo("O valor da variável é maior que 10.");
21 |         }
22 |         else
23 |         {
24 |             echo("O valor da variável é menor que 10.");
25 |         }
26 |
27 |     ?>

```

Exemplo 3; Exemplo da instrução ELSE IF encadeado.

```

12 |         <?php
13 |         // put your code here
14 |         $cor = "branco";
15 |         if ($cor == "vermelho")
16 |         {
17 |             echo("A variável contém o valor 'vermelho'.");
18 |         }
19 |         else if ($cor == "azul")
20 |         {
21 |             echo("A variável contém o valor 'azul'.");
22 |         }
23 |         else if ($cor == "amarelo")
24 |         {
25 |             echo("A variável contém o valor 'amarelo'.");
26 |         }
27 |         else
28 |         {
29 |             echo("O valor da variável não foi identificado.");
30 |         }
31 |     ?>

```

ELSE IF. Ele tem a mesma função de um **ELSE** e um **IF** usados sequencialmente, como no exemplo ao lado. Num mesmo IF podem ser utilizados diversos **ELSEIF's**, ficando essa utilização a critério do programador.

2.5.2. Estrutura Condicional (switch).

A instrução switch atua de maneira semelhante a uma série de comandos if na mesma expressão. É utilizado quando se deseja comparar uma variável com diversos valores, e executar um código diferente dependendo de qual valor é igual ao da variável.

Quando isso for necessário, deve-se usar o comando switch. O exemplo abaixo mostra a sua utilização.

Em outras linguagens que implementam o comando switch, ou similar, os valores a serem testados só podem ser do tipo inteiro. Em PHP é permitido usar valores do tipo string como elementos de teste do comando switch.

Exemplo 1; Neste exemplo é declarado à variável **\$d** que vai receber a data do navegador,

```

11 <body>
12   <?php
13     // a variável $d recebe a data do navegador
14     $d = getdate();
15
16     switch ($d['wday']) {
17       case 5:
18         echo("Finalmente Sexta");
19         break;
20       case 6:
21         echo("Super Sábado");
22         break;
23       case 0:
24         echo("Domingo Sonolento");
25         break;
26       default:
27         echo("Estou esperando pelo fim da semana");
28     }
29   ?>
30 </body>

```

na **linha 16** é passado somente o dia usando um array, com relação à array iremos estudar sobre este assunto nas próximas aulas, após obter o dia em formato numérico vai ser feita a comparação com os cases, o array é o vetor que conhecemos em lógica de programação, então como já sabemos ele inicia em 0. Por isso que o domingo tem o valor 0, pois domingo é o primeiro dia da semana.

2.5.3. Estrutura de Loop (for).

Esta estrutura de loop já é uma velha conhecida nossa, é basicamente composta de três pontos principais ou sentenças separadas por ponto e vírgula.

1. **Inicialização:** Deve conter a variável que inicia o laço.
2. **Condição:** Expressão booleana que define se os comandos que estão dentro do laço serão executados ou não. Enquanto a expressão for verdadeira os comandos serão executados.
3. **Incremento:** Executado ao final de cada execução do laço.

Exemplo 01: Neste exemplo foi criado um contador simples de 1 a 10.

```

12 <?php
13   // put your code
14   for ($i = 1; $i <= 10; $i++)
15   {
16     print $i;
17   }
18   ?>
... ...

```

FOREACH

É um laço de repetição para interações em arrays, é um for de forma simplificado, ele decompõe o vetor ou matriz em cada um de seus elementos por meio da cláusula **AS**, veremos exemplos de sua aplicação na aula de arrays.

Sua sintaxe é;



```
foreach ($array) as $valor)
{
instruções
}
```

\$array será o nome do array.

as \$valor esta passando os valores do array para a variável \$valor, as instruções podem ser por exemplo, um echo ou print, para exibir na página o resultado.

Tudo que esta entre as chaves as instruções serão repetidas o numero de vezes das posições do vetor.

2.5.4. Estrutura de Loop (while).

É o comando de repetição (laço) mais simples. Ele testa uma condição e executa um comando, ou um bloco de comandos, enquanto a condição for verdadeira.

Exemplo: O exemplo abaixo mostra o uso do while para imprimir de 1 até 9.

```
12      <?php
13          // put your code here
14          $a = 1;
15          while ($a < 5) {
16              print $a;
17              $a++;
18          }
19      ?>
```

do... while

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a expressão é testada ao final do bloco de comandos. Abaixo temos um exemplo usando o do while.

Exemplo: Neste exemplo abaixo usamos o do while para escrever de 9 a 10, decrementando a cada passo do laço.

```
11      <body>
12          <?php
13          $i = 10;
14          do {
15              print --$i;
16          } while ($i > 0);
17
18      </body>
```



Exercício Prático

- 1) Construir um programa em PHP que solicite que um valor inteiro seja digitado e imprima o quadrado do valor digitado se este valor for maior do que 50.
- 2) Construir um programa em PHP que peça para que um valor inteiro e positivo seja digitado e imprima/escreva uma mensagem informando se este número é par ou não.

- 3) Construir um programa em PHP que leia cinco valores inteiros e calcule a média aritmética desses cinco valores. Imprimir a média calculada com a mensagem: “A média calculada é:”
- 4) Construir um programa em PHP que leia as notas de AV1, AV2 e AV3 de um aluno e imprima a média calculada, com a mensagem pertinente, ou seja, se o aluno foi aprovado, está em prova final ou foi reprovado.
- 5) Construir um programa em PHP que simule uma máquina de calcular e execute as quatro operações sobre os dois números digitados.

2.6. PHP – Definição de Funções.

Uma função é um bloco de código reutilizável que é executado devido a um evento ou pela chamada de outra função. Deve-se usar a declaração **function** para criar uma função. Os parâmetros usados pela função são declarados entre parênteses. Os comandos a serem executados pela função devem estar entre chaves.

A sintaxe básica para definir uma função é:

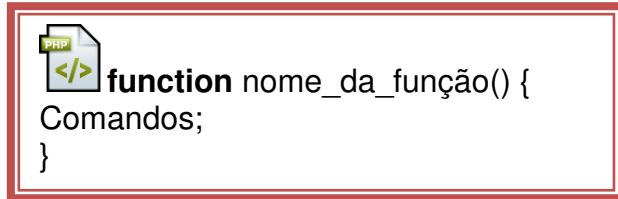


Figura 11 - Sintaxe de uma função

Exemplo: No código abaixo temos a declaração de uma função chamada **msn()** que escreve um texto, na linha 21 esta função é chamada.

```

12 <body>
13     <?php
14
15     function msn() {
16         $texto = "Olá, Bem vindo às Funções no PHP";
17         echo '<h2>', $texto, '</h2>';
18     }
19
20
21     <?php msn(); ?>
22     </body>
  
```

A declaração return

A declaração **return** retorna um valor quando a função é chamada. Esta declaração não é necessária se a função não retorna nenhum valor.

Para se chamar uma função, deve-se escrever seu nome e indicar os parâmetros entre parênteses, abaixo a sintaxe de uma função com **return** que pode ter ou não argumentos.



```
function nome_da_função([arg1, arg2, arg3]) {
Comandos;
[return <valor de retorno>];
}
```

Toda função pode opcionalmente retornar um valor, ou simplesmente executar os comandos e não retornar valor algum. Uma função não pode retornar mais de um valor, mas é permitido fazer com que uma função retorne um valor composto, como listas ou arrays.

```
20      <?php
21
22      function soma() {
23          $valor1=10;
24          $valor2=7;
25          $resultado = $valor1 + $valor2;
26          return ($resultado);
27      }
28      echo soma();
29      ?>
```

Exemplo: Função que retorna o resultado de uma soma, na linha 28 é feita a chamada do método para imprimir o resultado da soma.

Argumentos

É possível passar argumentos para uma função. Eles devem ser declarados logo após o nome da função, entre parênteses, e tornam-se variáveis pertencentes ao escopo local da função. A declaração do tipo de cada argumento é utilizada apenas para efeito de documentação.



```
function nome_da_função([arg1,arg2,arg3]) {
Comandos;
}
```

Exemplo: A função imprime foi declarada, onde a mesma recebe como argumento a variável **\$texto**, na linha 7 é feita a chamada da função onde é passado o texto para a função imprimir.

```
1      <?php
2
3      function imprime($texto) {
4          echo $texto;
5      }
6
7      imprime("teste de funções");
8
9      ?>
```

Passagem de parâmetros por referência

Normalmente, a passagem de parâmetros em PHP é feita por valor, ou seja, se o conteúdo da variável for alterado, essa alteração não afeta a variável original.

Exemplo:

```

1  <?php
2
3  [-] function mais5($numero) {
4      $numero += 5;
5  }
6
7  $a = 3;
8  echo mais5($a);
9
10 // $a continua valendo 3
11 ?>
```

verdade com uma cópia da variável. Porém, referência, toda alteração que a função realizar no valor passado como parâmetro afetará a variável que o contém.

No exemplo, como a passagem de parâmetros é por valor, a função mais5 é inútil, já que após a execução sai da função o valor anterior da variável é recuperado. Se a passagem de valor fosse feita por referência, a variável \$a teria 8 como valor. O que ocorre normalmente é que ao ser chamada uma função, o interpretador salva todo o escopo atual, ou seja, os conteúdos das variáveis. Se uma dessas variáveis for passada como parâmetro, seu conteúdo fica preservado, pois a função irá trabalhar na se a passagem de parâmetros for feita por referência.

Há duas formas de fazer com que uma função tenha parâmetros passados por referência: indicando isso na declaração da função, o que faz com que a passagem de parâmetros sempre seja assim; e também na própria chamada da função. Nos dois casos utiliza-se o modificador "&". Vejamos um exemplo que ilustra os dois casos:

```

1  <?php
2
3  [-] function mais5(&$num1, &$num2) {
4      $num1 += 5;
5      $num2 += 5;
6  }
7
8  $a = $b = 1;
9
10 [-] mais5($a, $b); /* Neste caso, só $num1 terá seu valor alterado, pois a
11     * passagem por referência está definida na declaração da função. */
12
13 [-] mais5($a, &$b); /* Aqui as duas v
14     * ariáveis terão seus valores alterados. */
15
16 ?>
```

Argumentos com valores pré-definidos (default)

Em PHP é possível ter valores default para argumentos de funções, ou seja, valores que serão assumidos em caso de nada ser passado no lugar do argumento. Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna-se opcional.

```

1  <?php
2
3  function teste($txt = "testando") {
4      echo $txt;
5  }
6
7  teste(); // imprime "testando"
8  teste("outro teste"); // imprime "outro teste"
9 ?>

```



Quando a função tem mais de um parâmetro, o que tem valor default deve ser declarado por último.

Contexto

O contexto é o conjunto de variáveis e seus respectivos valores num determinado ponto do programa. Na chamada de uma função, ao iniciar a execução do bloco que contém a implementação da mesma é criado um novo contexto, contendo as variáveis declaradas dentro do bloco, ou seja, todas as variáveis utilizadas dentro daquele bloco serão eliminadas ao término da execução da função.

Escopo

O escopo de uma variável em PHP define a porção do programa onde ela pode ser utilizada. Na maioria dos casos todas as variáveis têm escopo global. Entretanto, em funções definidas pelo usuário um escopo local é criado. Uma variável de escopo global não pode ser utilizada no interior de uma função sem que haja uma declaração.

2.7. PHP - Arrays.

Um array é uma variável que armazenam mais de um valor simultaneamente, são como contêineres, servindo para armazenar números, strings, objetos, dentre outros, de forma dinâmica, alem disso oferece muitas funções para manipulá-los.

Os arrays são acessados por uma posição, como o índice numérico, para criar um array, pode-se utilizar a função array, abaixo a sintaxe de criação do array.



`array([chave=>] valor, ...)`

Exemplo 1: neste exemplo na linha 14, estamos criando um array usando a função array, criamos a variável **\$cores** onde ela recebe a função array e dentro dos parênteses passando os valores.

```

12
13  <?php
14  // put your code here
15  $cores = array('vermelho -', 'azul -', 'verde -', 'amarelo');
16  echo $cores[0];
17
18  echo $cores[1];
19
20  echo $cores[2];
21
22  echo $cores[3];
?>

```

Nas linhas 15 a 21, exibimos os valores do array o que esta dentro de [] é o índice do array, então neste array temos 4 posições iniciando em 0 e vai até 3, de 0 a 3 temos 4 elementos.

Exemplo 2: Agora neste exemplo criamos um array, mais já especificando o índice que será usado, desta forma podemos criar um array que não inicia em zero, diferente da linguagem Java em que o índice sempre vai iniciar em 0.

```

13 <?php
14
15
16 $frutas = array(
17     1 => "Laranja",
18     2 => "Maçã",
19     3 => "Uva");
20 echo "<li> $frutas[1]<br>";
21 echo "<li> $frutas[2]<br>";
22 echo "<li> $frutas[3]<br>";
23
24 ?>

```

Arrays associativos

São chamados assim por possuírem uma chave de acesso para cada posição, abaixo iremos criar um array associativo, para exemplificar sua utilização.

Exemplo 1: Neste exemplo estamos criando um array, definindo qual será sua chave de acesso, isso mesmo em PHP ao invés de ter como acesso ao array um numero inteiro, posso definir o nome da chave de acesso, nas linhas 15 a 18 crio a variável **\$frutas** agora sem usar a função array, entre [] com aspas simples determino o nome da chave de acesso, e depois é atribuído o valor para aquela chave.

```

12 <?php
13 // put your code here
14
15 $frutas['cor'] = 'vermelha';
16 $frutas['sabor'] = 'doce';
17 $frutas['formato'] = 'redonda';
18 $frutas['nome'] = 'maçã';
19
20 foreach ($frutas as $chave => $frutas)
21 {
22     echo "$chave => $frutas <br>\n";
23 }
24 ?>

```

entre [] com aspas simples determino o nome da chave de acesso, e depois é atribuído o valor para aquela chave.

Na linha 20: Estou usando uma estrutura de repetição chamada **foreach**, ele é

responsável por decompor o array em cada posição (índice), no caso como o que esta dentro das chaves do **foreach** é uma instrução para imprimir na página, isso vai fazer com que todas as posições do vetor (array) seja mostrado na página(linha 22).

Exemplo 2: Neste exemplo estamos fazendo um acesso ao array, os mesmos podem ser acessados a qualquer momento e podem ser realizadas operações, como no exemplo abaixo.

A função **var_dump** gera uma saída do array, exibindo tamanho, índices, valores do array, como mostrado na imagem.

```

12 <?php
13 // put your code here
14 $minha_multa['carro'] = 'Pálio';
15 $minha_multa['valor'] = '178.00';
16
17 //Alteração de valores
18 $minha_multa['carro'] .= ' ED 1.0';
19 $minha_multa['valor'] += 20;
20
21 //Exibir Array
22
23 var_dump($minha_multa);
24
25 $comidas[] = 'lazanha';
26 $comidas[] = 'Pizza';
27 $comidas[] = 'Macarrão';
28
29 //Alteração de valores
30
31 $comidas[1] = 'Pizza Calabresa';
32
33 //Exibe o Array
34 var_dump($comidas);
35
36 ?>

```

```

array (size=2)
  'carro' => string 'PálioED 1.0' (length=12)
  'valor' => float 198

array (size=3)
  0 => string 'lazanha' (length=7)
  1 => string 'Pizza Calabresa' (length=15)
  2 => string 'Macarrão' (length=9)

```

Figura 12 - resultado gerado pela função var_dump

Array multidimensional

São arrays que podem em algumas de suas posições conterem outro array de forma recursiva, ele também pode ser criado pela função array().

No exemplo a seguir vamos criar um array multidimensional da tabela abaixo;

Modelo do carro (\$carros)	Cor	Potência	Opcionais
Palio	Azul	1.0	Ar condicionado
Corsa	Cinza	1.3	MP3
Gol	Branco	1.0	Metálica

O nosso array vai se chamar **\$carros** onde temos os modelos e suas características, duas dimensões [] [].

Exemplo: Neste exemplo temos um array multidimensional, onde a variável \$carros é o nosso array, nele como podemos observar temos três chaves (palio,Corsa,Gol) e em

cada chave adiciono outras que são relacionadas as características do carro como cor,potência e opcionais e a cada uma é atribuído um valor.

Para facilitar o entendimento o array multidimensional nada mais é que uma tabela com linhas e colunas, uma matriz

Na linha 24: Geramos uma saída porem o que vai ser mostrado na página será o valor “Ar condicionado”.

Na linha 28 a 35: Temos um novo bloco PHP, onde

usamos o **foreach** encadeado, ou seja, um dentro do outro para exibir os modelos (**\$modelo**) de carros e suas características (**\$características**) e seus valores (**\$valor**), gerando a saída abaixo no navegador;

```

12 <?php
13 // put your code here
14 $carros['palio']['cor'] = 'azul';
15 $carros['palio']['potência'] = '1.0';
16 $carros['palio']['opcionais'] = 'Ar condicionado';
17 $carros['Corsa']['cor'] = 'cinza';
18 $carros['Corsa']['potência'] = '1.3';
19 $carros['Corsa']['opcionais'] = 'MP3';
20 $carros['Gol']['cor'] = 'Branco';
21 $carros['Gol']['potência'] = '1.0';
22 $carros['Gol']['opcionais'] = 'Metalica';
23
24 echo $carros['palio']['opcionais'];
25 ?>
26
27
28 <?php
29 foreach ($carros as $modelo => $características) {
30     echo "> modelo $modelo<br>\n";
31     foreach ($características as $características => $valor) {
32         echo "características $características => $valor<br>\n";
33     }
34 }
35 ?>

```

```
=> modelo palio
características cor => azul
características potência => 1.0
características opcionais => Ar condicionado
=> modelo Corsa
características cor => cinza
características potência => 1.3
características opcionais => MP3
=> modelo Gol
características cor => Branco
características potência => 1.0
características opcionais => Metalica
```

3. Introdução a Framework JQuery

jQuery é uma poderosa biblioteca JavaScript criada para simplificar a criação de efeitos visuais e de interatividade em web sites.

jQuery propicia a criação de scripts de uma forma tão simples e intuitiva que consegue com meia dúzia de linhas os mesmos efeitos de um script de 30 a 40 linhas desenvolvido com JavaScript tradicional. Simplicidade foi à diretriz que norteou John Resig na criação da biblioteca. (Silva)

Nesta etapa da disciplina, iremos trabalhar apresentando a biblioteca e um estudo da sintaxe usando os seletores e comandos jQuery, iremos trabalhar desenvolvendo práticas que chamaremos de laboratórios empregando vários scripts, que poderão ser utilizados no desenvolvimento de web sites.

3.1. Instalação.

A instalação é bem simples basta fazer o download no site <http://jquery.com/> e depois fazer o download, você irá baixar um arquivo compactado descompacte-o no diretório raiz do seu site.

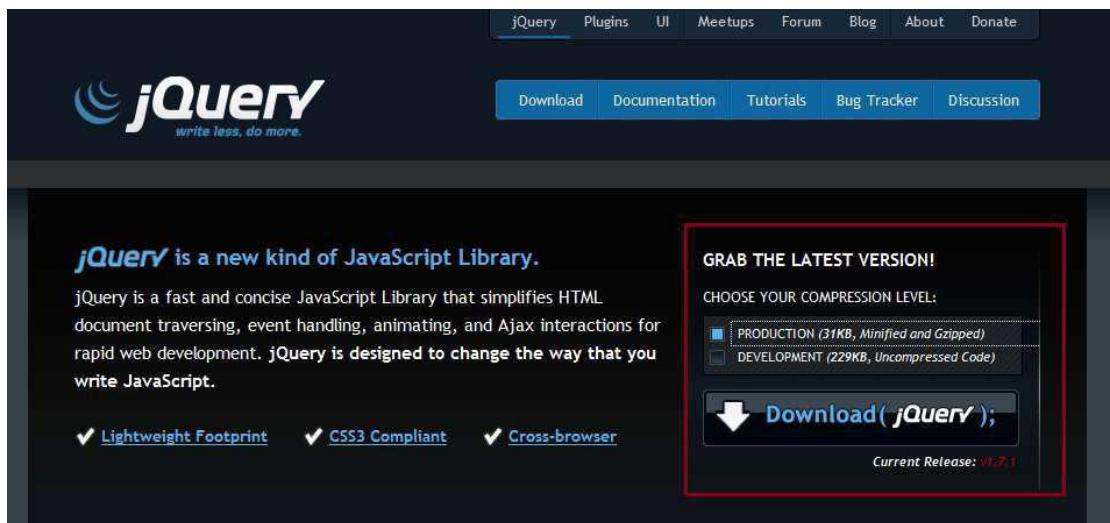


Figura 13 - Site Jquery

Para utilizar a biblioteca basta chamarmos o script na tag **<head>**, como vemos na imagem abaixo:

```

9 <title></title>
10 <script src="../jquery/jquery-1.8.3.min.js"></script>

```

Iniciamos a tag **<script>** usamos o elemento **src** e passamos o caminho do script Jquery que tem a extensão *.js, agora podemos acessar funções da biblioteca.

jQuery UI (User Interface)é um conjunto de funções em um script e estilos CSS com de interações de interface de usuário, efeitos, widgets, temas construídos em cima da Biblioteca JavaScript jQuery, para baixar acesse o link <http://jqueryui.com/>, coloque-o no mesmo diretório que o jquery, nas próximas aulas desenvolveremos alguns exemplos com ele.

No link <http://code.jquery.com/> podemos fazer o download de vários script de projetos jquery.

The screenshot shows the **jQuery CDN - provided by (mt) Media Temple** page. It has four main sections:

- Recent Stable Versions**: Links to [jquery.min.js](#) (most recent stable) and [jquery.js](#).
- jQuery Live Git Copy**: Link to [jquery-git.js](#).
- jQuery Mobile Stable**: Links to various files: [jquery.mobile-1.2.0.js](#), [jquery.mobile-1.2.0.min.js](#), [jquery.mobile-1.2.0.css](#), [jquery.mobile-1.2.0.min.css](#), [jquery.mobile.structure-1.2.0.css](#), [jquery.mobile.structure-1.2.0.min.css](#), [jquery.mobile-1.2.0.zip](#), and [jquery.mobile-1.2.0.docs.zip](#).
- jQuery Mobile Git Copy**: Links to [jquery-mobile.js](#), [jquery-mobile.css](#), [jquery-mobile.min.js](#), [jquery-mobile.min.css](#), and [jquery.mobile.zip](#).

Agora vamos a prática o lema do Jquery que é escreva menos e faça mais, por isso os temas das aulas a seguir são bem práticos.

3.2. Colunas e células de tabelas.

Nesta aula vamos trabalhar com tabelas, as páginas que vamos criar usando o Jquery terão a extensão ***.php**, abaixo temos o resultado final desta prática.

Exemplo Tabela 1

Viação Alfa - Horários					
Destino	Saída	Chegada	Classe	Tarifa	Frequência
Brusque	06:45	14:30	Convencional	R\$80,00	Diária
Joinville	12:30	21:00	Convencional	R\$100,00	Sábado
São Paulo	19:00	06:00	Leito	R\$150,00	Diária
Porto União	13:00	18:00	Convencional	R\$60,00	Diária
Caçador	00:30	23:00	Executivo	R\$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R\$280,00	Sábado
Blumenau	08:00	16:00	Convencional	R\$85,00	Diária
Itajai	23:00	06:00	Executivo	R\$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R\$50,00	Terça
Urussanga	14:30	20:00	Leito	R\$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R\$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R\$90,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R\$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R\$250,00	Sábado
Xanxeré	18:00	22:00	Executivo	R\$75,00	Diária
Chapecó	13:00	21:00	Executivo	R\$105,00	Qua. e Dom.
Xaxim	07:00	14:00	Executivo	R\$165,00	Dom.

Válida para o período de 02/10/2008 a 30/11/2008.

1º passo: É criar uma página com web com a extensão ***.php**, nela poderemos trabalhar com HTML e PHP.

2º passo: Agora vamos criar a nossa folha de estilo, criaremos um novo arquivo css chamado **tabela.css** para depois fazer um link com a página, abaixo temos a imagem do arquivo ***.css**

```

2  body {
3      width:600px;
4      font: 80%/1.2 Arial, Helvetica, sans-serif;
5      margin:30px auto;
6      padding:0;
7      color:#000;
8  }
9  table {
10     width:550px;
11     border-collapse: collapse;
12     border: 2px solid #000;
13     margin:0 auto;
14  }
15  caption {
16     text-align: right;
17     margin-bottom: 0.3em;
18     border-bottom: 1px solid #333;
19     padding-right: 0.3em;
20  }
21  thead tr th {
22     text-align:center;
23     border-bottom: 2px solid #000;
24     border-left: 2px solid #000;
25  }
26  tr td, tr th {
27     padding: 1px 5px;
28     text-align:left;
29     font-size: 0.9em;
30     border: 1px dotted #333;
31  }
32  tfoot tr td {
33     text-align:center;
34     border-top: 2px solid #000;
35  }
36  /* CSS para efeitos jQuery */
37  .impar {background:#add6ef;}
```

Nesta parte do css, temos as marcações das tag que serão usadas no HTML.

Continuações do css, na linha 37 têm a cor de fundo que será aplicada na tabela, para obter o efeito cor sim cor não.

3º passo: Neste passo vamos chamar o css e o script Jquery na página web.

```

9
10    <title></title>
11    <link type="text/css" href="../css/tabelas.css" rel="stylesheet">
<script type="text/javascript" src="../jquery/jquery-1.8.3.min.js"></script>
```

Na tag **<head>** estamos chamando a folha de estilo na **linha 10** que tem o nome de **tabela.css**.

Na linha: 11 chamamos o script Jquery, que esta no diretório jquery.

4º passo: Escrever o script JavaScript que será executado quando o documento for lido.

```

13     <script type="text/javascript">
14         $(document).ready(function() {
15             $('table#horario tbody tr:odd').addClass('impar');
16
17         });
18     </script>

```

Linha 14: Nesta linha quando o documento for lido (**\$(document).ready**) vai ser acionado uma função que tem como objetivo adicionar a formatação do atributo css **table**, que criamos anteriormente, na tabela chamada **horario**. De forma a adicionar a class **impar** que vai aplicar o **background:#add6ef**, criado no css, fazendo com que as linhas impares sejam adicionadas o background definido.

5º passo: Estrutura HTML de nossa página, na tag **<body>** criamos a nossa tabela, o id da tabela chama-se “**horario**” que faz referência ao script criado anteriormente.

```

19 <body>
20     <table id="horario">
21         <caption>
22             Viação Alfa - Horários
23         </caption>
24         <thead>
25             <tr id="horizontal">
26                 <th>Destino</th>
27                 <th scope="col">Saida</th>
28                 <th scope="col">Chegada</th>
29                 <th scope="col">Classe </th>
30                 <th scope="col">Tarifa </th>
31                 <th scope="col">Frequência</th>
32             </tr>
33         </thead>
34         <tfoot>
35             <tr>
36                 <td colspan="6">Válida para o periodo de 02/10/2008 a 30/11/ 2008.</td>
37             </tr>
38         </tfoot>
39     <tbody>
40

```

Na linha 26: Temos a tag **<tr>** com o id “horizontal” para identificar que esta é a linha horizontal, o restante do código é apenas a estrutura da tabela em HTML a qual já conhecemos.

Laboratório Tabela 2

Neste exemplo vamos adicionar um efeito de cores uma para cada linha, e um evento **hover** para destacar a linha onde o mouse passar, o resultado desta prática podemos visualizar na imagem abaixo.

Exemplo Tabela 2

Viação Alfa - Horários					
Destino	Saída	Chegada	Classe	Tarifa	Frequência
Brusque	06:45	14:30	Convencional	R\$80,00	Diária
Joinville	12:30	21:00	Convencional	R\$100,00	Sábado
São Paulo	19:00	06:00	Leito	R\$150,00	Diária
Porto União	13:00	18:00	Convencional	R\$60,00	Diária
Cacador	00:30	23:00	Executivo	R\$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R\$280,00	Sábado
Blumenau	08:00	16:00	Convencional	R\$85,00	Diária.
Itajaí	23:00	06:00	Executivo	R\$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R\$50,00	Terça
Urussanga	14:30	20:00	Leito	R\$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R\$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R\$90,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R\$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R\$250,00	Sábado
Xanxerê	18:00	22:00	Executivo	R\$75,00	Diária.
Chapecó	13:00	21:00	Executivo	R\$105,00	Qua. e Dom.
Xaxim	07:00	14:00	Executivo	R\$165,00	Dom.

Válida para o período de 02/10/2008 a 30/11/2008.

1º passo: Vamos voltar em nosso arquivo css onde temos o estilo da tabela anterior e adicionar a class **.par** com o background da linha par, a class **.destacar** que mudará a cor quando o mouse passar na linha da tabela.

```

39  /* continuação para o exemplo 2*/
40  .par {background:#d6e2e5;}
41  /* Este atributo será usado quando o mouse estiver sobre a linha da tabela */
42  .destacar {
43      background:#444;
44      color:#fff;
45  }

```

2º passo: Para este exemplo podemos apenas editar o script anterior acrescentando, a class **par**, o evento **hover** como na linha 18.

```

12  <script type="text/javascript">
13      $(document).ready(function() {
14          $('table#horario tbody tr:odd').addClass('impar');
15
16          //continuação para o exemplo 2
17          $('table#horario tbody tr:even').addClass('par');
18          $('table#horario tbody tr').hover(
19              function() {
20                  $(this).addClass('destacar');
21              },
22              function() {
23                  $(this).removeClass('destacar');
24              });//fim exemplo 2
25      });
26  </script>

```

Na linha 19 a 24: temos a função responsável por adicionar e remover a class css, que será mostrado quando o ponteiro do mouse estiver sobre uma linha ou sair da linha.
A estrutura HTML é a mesma do exemplo anterior, podendo apenas adicionar este efeito.

3.3. Tooltips.

É como são chamadas as legendas, neste laboratório vamos aprender como criar o efeito abaixo na legenda.

localhost/aula_php_js_nb/aula_jquery/jquery_toolTips.php

CPF
Somente números

Data

1º passo: Criar uma página em PHP como mostrado acima, com o campo CPF e Data.

2º passo: Criar o script responsável por apresentar a legenda com o efeito.

```

12 <script type="text/javascript" src="../jquery/jquery-1.8.3.min.js"></script>
13 <script type="text/javascript">
14     $(document).ready(function() {
15         $('.dica + span')
16             .css({display: 'none',
17                   border: '1px solid #add6ef',
18                   padding: '2px 4px',
19                   background: '#d6e2e5',
20                   marginLeft: '10px'
21             });
22         $('.dica').focus(function() {
23             $(this).next().fadeIn(1500);
24         }).blur(function() {
25             $(this).next().fadeOut(1500);
26         });
27     });
28 </script>

```

Como estamos trabalhando com Jquery sempre temos que adicionar o script Jquery.

Na linha 14: Como no exemplo anterior Iniciamos o script com o método ready do documento carregando a função.

Na linha 15 a 20: criamos a class **.dica** com os atributos do css para formatação do texto que será mostrado, onde o texto encontra-se na tag ****.

Na linha 22: Aciono o método **focus** e o **blur** chamando a função para aplicar o efeito **fadein** e **fadeout** com o valor 1500, que vai gerar um esmaecimento na entrada do foco no campo e na saída do campo.

3º passo: Na estrutura HTML da página na tag ****, colocaremos o texto da dica, na tag **<input>** vamos adicionar a class “**dica**” que foi criada no script acima.

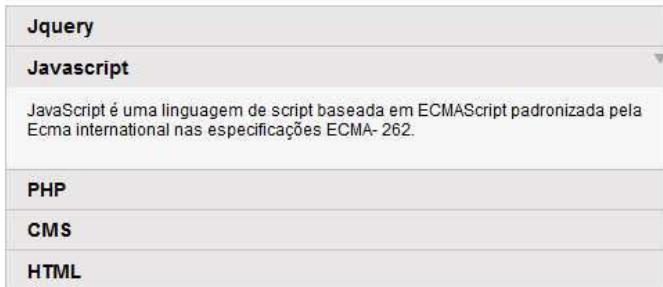
```

33 <form action="" method="get" id="formulario">
34     <label for="cpf">CPF</label><br />
35     <input name="cpf" type="text" id="cpf" class="dica" /> <span>Somente números</span><br />
36     <label for="data">Data</label><br />
37     <input name="data" type="text" id="data" class="dica"/> <span> Digite a data no formato dd/mm/yyyy</span>
38 </form>

```

3.4. Accordion.

Neste laboratório vamos cria um accordion, que é este efeito sanfona entre uma aba e outras, abaixo temos a imagem de como ficará este exemplo.



1º passo: Criar a folha de estilo com o nome accordion.css, nela vamos colocar alguns efeitos personalizados.

```

1  body {
2    margin: 10px auto;
3    width: 570px;
4    font: 75%/120% Arial, Helvetica, sans-serif;
5  }
6  .accordion2 {
7    width: 480px;
8    border-bottom: solid 1px #c4c4c4;
9  }
10 .accordion2 h3 {
11   background: #e9e7e7 url(..../images/seta.png) no-repeat right -51px;
12   padding: 7px 15px;
13   margin: 0;
14   font: bold 120%/100% Arial, Helvetica, sans-serif;
15   border: solid 1px #c4c4c4;
16   border-bottom: none;
17   cursor: pointer;
18 }
19 .accordion2 h3:hover {
20   background-color: #e3e2e2;
21 }
22 .accordion2 h3.active {
23   background-position: right 5px;
24 }
25 }

26 .accordion2 p {
27   background: #f7f7f7;
28   margin: 0;
29   padding: 10px 15px 20px;
30   border-left: solid 1px #c4c4c4;
31   border-right: solid 1px #c4c4c4;
32   display: none;
33 }
34

```

Na linha 11: Colocamos um ícone chamado seta.png, você pode baixar algum na internet com as dimensões 11x 7, porem o código pode ficar sem o ícone. Abaixo temos a continuação da folha de estilo.

2º passo: Criar o script para o accordion, o script abaixo vai fazer a interação com a folha de estilo aplicando os efeitos.

```

3  $(document).ready(function(){
4    $(".accordion2 h3").eq(2).addClass("active");
5    $(".accordion2 p").eq(2).show();
6    $(".accordion2 h3").click(function(){
7      $(this).next("p").slideToggle("slow")
8      .siblings("p:visible").slideUp("slow");
9      $(this).toggleClass("active");
10     $(this).siblings("h3").removeClass("active");
11   });
12 });

```

3º passo: Adicionar o estilo css e o script na página, não se esqueça de incluir também o script do Jquery.

```

11 <script type="text/javascript" src="..../jquery/jquery-1.8.3.min.js"></script>
12 <script type="text/javascript" src="..../js/accordion.js"></script>
13
14 <link type="text/css" href="..../css/accordion.css" rel="stylesheet"/>

```

4º passo: Abaixo temos a estrutura HTML de nossa página na linha 18 adicionamos a class **accordion2**, que vem da folha de estilo criada no inicio desta aula.

```

17 <body>
18   <div class="accordion2">
19     <h3>Jquery</h3>
20     <p>jQuery é uma biblioteca JavaScript cross-browser desenvolvida
21       para simplificar os scripts client side que interagem com o HTML.</p>
22     <h3>Javascript</h3>
23     <p>JavaScript é uma linguagem de script baseada em ECMAScript
24       padronizada pela Ecma international nas especificações ECMA-
25       262.</p>
26     <h3>PHP</h3>
27     <p>PHP (um acrônimo recursivo para "PHP: Hypertext
28       Preprocessor", originalmente Personal Home Page) é uma linguagem
29       interpretada livre e utilizada para gerar conteúdo dinâmico na World Wide
30       Web.</p>
31     <h3>CMS</h3>
32     <p>Content Management Systems (CMS) mas igualmente
33       designada como Conversational Monitor System [nota 1] - é um
34       sistema gestor de websites, e intranets que integra ferramentas
35       necessárias para criar, gerir (inserir e editar) conteúdos em tempo
36       real sem a necessidade de programação de código, cujo objetivo
37       é estruturar e facilitar a criação, administração, distribuição, publicação e
38       disponibilidade da informação.</p>
39     <h3>HTML</h3>
40     <p>HTML (acrônimo para a expressão inglesa HyperText Markup
41       Language, que significa Linguagem de Marcação de Hipertexto) é uma
42       linguagem de marcação utilizada para produzir páginas na Web.</p>
43   </div>
44 </body>

```

3.5. Datepicker.

Este widget apresenta um calendário quando o usuário coloca o foco no campo de data, abaixo o resultado do nosso calendário.



Este exemplo é bem simples e pode ser personalizado acessando o script JqueryUI, nesta aula vamos utilizar o JqueryUI que é um complemento do Jquery, trata especialmente da interface com o usuário, e para o programador deixa o trabalho bem mais simplificado.

1º passo: Adicionar scripts do Jquery e JqueryUI na linha 13 chamamos a folha de estilo que vem com o JqueryUI quando é feito o download.

```

10   <script src="../jquery/jquery-1.8.3.min.js"></script>
11
12   <script src="../jquery/jquery-ui.js"></script>
13   <link href="../jquery/jquery-ui.css" rel="stylesheet">

```

2º passo: Agora temos a penas que criar um script, para chamar o método datepicker através do elemento **#datepicker** que será o id do elemento HTML.

```

15      <script>
16      $(function() {
17          $('#datepicker').datepicker();
18      });
19  </script>

```

3º passo: Na estrutura HTML, precisamos apenas definir a partir de que elemento vai ser chamado o calendário, bastando informar o **id** com o mesmo nome do elemento criado no script anterior.

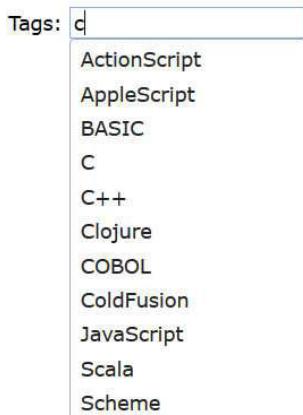
```

21      <body>
22          <p>Data: <input type="text" id="datepicker" value="" /></p>
23
24      </body>

```

3.6. Auto-complete.

Neste laboratório vamos criar um campo com auto-complete nesta aula também vamos utilizar o JqueryUI.



1º passo: Adicionar os scripts jquery.js e jqueryUi.js e a folha de estilo jquery-ui.css.

```

10      <script src="../jquery/jquery-1.8.3.min.js"></script>
11
12      <script src="../jquery/jquery-ui.js"></script>
13      <link href="../jquery/jquery-ui.css" rel="stylesheet">

```

```

13      <script>
14          $(function() {
15              var availableTags = [
16                  "ActionScript",
17                  "Asp",
18                  "BASIC",
19                  "C",
20                  "C++",
21                  "Clojure",
22                  "COBOL",
23                  "ColdFusion",
24                  "Erlang",
25                  "Fortran",
26                  "Groovy",
27                  "Java",
28                  "JavaScript",
29                  "Lisp",
30                  "Perl",
31                  "PHP",
32                  "Python",
33                  "Ruby",
34              ];
35              $('#tags').autocomplete({
36                  source: availableTags
37              });
38          });
39      </script>

```

2º passo: Criar o script para as palavras chaves, vamos criar uma função e dentro dela criar um arrays com as palavras chaves, na linha 15 declaramos a variável e atribuímos os valores.

Na linha 35 ainda dentro da função criamos o elemento **#tags** e chamamos a função **autocomplete** passando para o source a variável criada na linha 15.

3º passo: A estrutura HTML, nesta definimos a class da div **ui-widget** a mesma vem da jqueryUI.

Depois na linha 46 definimos o id do input como **tags** referenciando com **#tags** criado no código JavaScript acima.

```

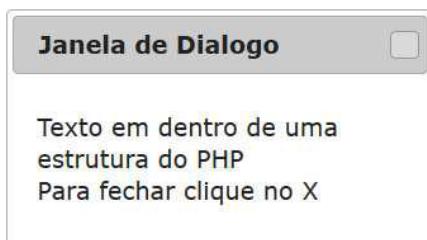
42 <body>
43     <!--A class ui-widget do JqueryUI -->
44     <div class="ui-widget">
45         <label for="tags">Tags: </label>
46         <input id="tags" />
47     </div>
48 </body>

```

3.7. Janela de dialogo modal.

Neste laboratório vamos aprender a criar uma janela de dialogo modal, este tipo de janela abre sobre o conteúdo da sua página como um popup, o resultado desta prática é apresentado abaixo.

Clique para abrir



1º passo: Adicionar script Jquery, JqueryUI e folha de estilo em nossa página.

```

10     <script src="../jquery/jquery-1.8.3.min.js"></script>
11     <script src="../jquery/jquery-ui.js"></script>
12     <link href="../jquery/jquery-ui.css" rel="stylesheet">

```

2º passo: Escrever o script responsável pela animação e estilo da janela.

```

14 <script>
15     // Aplica o padrão da animação e velocidade para exibição do efeito
16     $.fx.speeds._default = 1000;
17     $(function() {
18         $("#dialog").dialog({
19             autoOpen: false,
20             show: "blind",
21             hide: "explode"
22
23         });
24         $("#opener").click(function() {
25             $("#dialog").dialog("open");
26             return false;
27         });
28     });
29 </script>

```

Linha 16: Defini a velocidade da animação em ms.

Linha 17: Função que aplica o estilo “**#dialog**” e o efeito de abertura **show** e o de fechamento **hide**.

Linha 24: definição de que evento vai dispara a janela, e em que elemento “#opener” que é o id da tag <Button>.

3º passo: Na estrutura da página, linha 33 referenciou o id da div ao atributo criado no script anterior.

```

31 |<body>
32 |
33 |<div id="dialog" title="Janela de Dialogo">
34 |<p>
35 |<?php echo 'Texto em dentro de uma estrutura do PHP', '<br>Para fechar clique no X'; ?>
36 |
37 |</p>
38 |</div>
39 |<button id="opener">Clique para abrir</button>
40 |</body>

```

Linha 35: Inserimos um trecho de código PHP com uma mensagem, só para demonstrar que podemos entrar com instruções PHP.

Linha 39: referencio o id do button ao atributo #opener criado no JavaScript acima.

3.8. Menu.

Nesta aula vamos criar um menu para o nosso site, na biblioteca JqueryUI, podemos fazer um menu em pouco tempo já com um estilo predefinido, mais podemos criar menus com outros estilos, em alguns sites oferecem vários modelos de menu para download, para conferir acesse o site <http://apycom.com/>.

Este exemplo vamos usar alguns estilos próprios do JqueryUI, o resultado desta prática é mostrado abaixo.



1º passo: Adicionar os scripts e o estilo da JqueryUI

```

10 |<script src="../jquery/jquery-1.8.3.min.js"></script>
11 |<script src="../jquery/jquery-ui.js"></script>
12 |<link href="../jquery/jquery-ui.css" rel="stylesheet">

```

2º passo: Agora vamos criar o script para chamar o método menu(), criamos o atributo #menu para ser aplicado a lista da estrutura HTML.

```

13 |<script>
14 |$(function() {
15 |    $('#menu').menu();
16 |});
17 |</script>

```

3º passo: Alterar uma propriedade do estilo jqueryui, para o menu ficar projetado como a imagem acima.

```

18  <style>
19      .ui-menu { width: 150px; }
20  </style>

```

4º passo: Estrutura HTML, definimos na tag **** o **id="menu"** para referenciar ao atributo **#menu** criado no JavaScript.

Na linha 25: Adicionamos a **class ui-state-disabled**, para desabilitar este item do menu, só para fins didáticos.

```

22  <body>
23      <ul id="menu">
24          <li><a href="#">Home</a></li>
25          <li class="ui-state-disabled"><a href="#">Serviços</a></li>
26          <li><a href="#">Quem somos</a></li>
27          <li><a href="#">Contatos</a></li>
28          <li>
29              <a href="#">Produtos</a>
30              <ul>
31                  <li>
32                      <a href="#">Acessórios</a>
33                      <ul>
34                          <li><a href="#">Estabilizador</a></li>
35                          <li><a href="#">Teclado</a></li>
36                          <li><a href="#">Mouse</a></li>
37                      </ul>
38                  </li>
39                  <li>
40                      <a href="#">Computadores</a>
41                      <ul>
42                          <li><a href="#">Login</a></li>
43                      </ul>
44                  </li>
45                  <li class="ui-state-disabled"><a href="#">Login</a></li>
46              </ul>
47      </body>

```

3.9. Abas.

Nesta aula vamos aprender a criar um painel tabulado, como o que vemos abaixo.

JavaScript	JQuery Mobile	Formulário Web
<p>É uma linguagem de programação que roda do lado cliente, ou seja, no navegador do usuário, nos permitindo realizar determinadas ações dentro de uma página web.</p>		

1º passo: Adicionar scripts e estilo css da biblioteca Jquery.

```

10 |         <script src="../jquery/jquery-1.8.3.min.js"></script>
11 |         <script src="../jquery/jquery-ui.js"></script>
12 |         <link href="../jquery/jquery-ui.css" rel="stylesheet">

```

2º passo: Script que cria o painel com as abas, criou-se uma função e definimos um atributo **#tabs** que esta vinculado ao método **tabs()** do jquery, o atributo vai ser usado na estrutura da página.

```

13 |         <script>
14 |             $(function() {
15 |                 $("#tabs").tabs();
16 |             });
17 |         </script>

```

3º passo: Estrutura da página, nesta é criado uma div que representa o painel onde estarão as abas.

Depois criamos uma **** onde cada lista vai chamar um atributo **#tab-1**, **#tab-2** e **#tab-3**, este que representa as div's criadas da linha 26 a 43, dessa forma junto com o JavaScript vai mostrar todos os títulos das abas, e quando um for escolhida mostrará seu conteúdo.

```

19 |     <body>
20 |         <div id="tabs">
21 |             <ul>
22 |                 <li><a href="#tabs-1">JavaScript</a></li>
23 |                 <li><a href="#tabs-2">JQuery Mobile</a></li>
24 |                 <li><a href="#tabs-3">Formulário Web</a></li>
25 |
26 |             <div id="tabs-1">
27 |                 <p>É uma linguagem de programação que roda do lado cliente, ou seja,
28 |                     no navegador do usuário, nos permitindo realizar determinadas ações
29 |                     dentro de uma página web.</p>
30 |
31 |             <div id="tabs-2">
32 |                 <p>
33 |                     O jQuery Mobile é uma biblioteca JavaScript para criar aplicativos da web móveis.
34 |                     Baseado no jQuery e em sua interface com o usuário (UI), com ela é possível
35 |                     garantir aparência e comportamento consistentes em plataformas de
36 |                     dispositivos móveis diferentes. </p>
37 |
38 |             <div id="tabs-3">
39 |                 <p>
40 |                     Nesta aula vamos desenvolver formulários para aplicarmos os efeitos
41 |                     aprendidos em aulas anteriores.
42 |
43 |                 </p>
44 |             </div>
45 |         </div>
        </body>

```

3.10. jQuery Mobile na prática

Introdução

O jQuery Mobile é uma biblioteca JavaScript para criar aplicativos da web móveis. Baseado no jQuery e em sua interface com o usuário (UI), com ela é possível garantir aparência e comportamento consistentes em plataformas de dispositivos móveis diferentes. Os recursos básicos do jQuery Mobile são:

➤ **Simplicidade e flexibilidade gerais.**

O uso da estrutura é simples. É possível:

- ✓ Desenvolver páginas usando marcação acionada com pouco ou nenhum JavaScript.
- ✓ Usar eventos e JavaScript avançado.
- ✓ Usar um único documento HTML com várias páginas integradas.
- ✓ Dividir o aplicativo em várias páginas.

➤ **Aprimoramento progressivo e degradação suave.**

Embora o jQuery Mobile utilize o HTML5, CSS 3 e JavaScript mais recentes, nem todos os dispositivos móveis fornecem esse suporte. A filosofia do jQuery Mobile é suportar tanto os dispositivos de alto nível quanto os com menos recursos — como os que não suportam JavaScript.

➤ **Suporte para toque e outros métodos de entrada.**

Fornece suporte para diversos métodos de entrada e eventos: toque, mouse e métodos de entrada do usuário baseados em foco.

➤ **Acessibilidade**

O jQuery Mobile foi projetado tendo em mente a acessibilidade. Tem suporte para Accessible Rich Internet Applications (WAI-ARIA) para ajudar a tornar as páginas da web acessíveis para visitantes portadores de deficiência por meio de tecnologias assistidas.

➤ **Leve e modular.**

A estrutura é leve, e dividida em biblioteca JavaScript, CSS, além de alguns ícones.

Para fazer o download acesse o link <http://jquerymobile.com>, o arquivo baixado é compactado, para usar descompacte no diretório do seu site, para iniciar a utilizar e desenvolver suas páginas nas aulas seguintes.

The screenshot shows the official jQuery Mobile website. At the top, there's a navigation bar with links for 'jQuery', 'UI', 'Mobile' (which is underlined), 'Plugins', 'Meetups', 'Forum', 'Events', 'About', and 'Donate'. Below the navigation is a secondary menu with tabs for 'Docs', 'Download', 'Platforms', 'Themes', 'Resources', 'Forum', and 'Blog'. The main content area has a large title 'Download' in bold. Underneath it, there's a section titled 'New Download builder' with a brief description of the tool. It says: 'We now have a tool to let you build your own custom bundle that contains only the components you need. The builder will generate a custom JavaScript and both a full and structure-only theme stylesheet for production use.' Below this, there's a call to action: 'Try the [download builder tool](#) now (Alpha)'. To the right of this text is a screenshot of the 'jQuery Mobile Download Builder' interface, which is a web-based configuration tool with various checkboxes for selecting components like 'Core' and 'Device福地 (jQuery Mobile)'.

New Download builder

We now have a tool to let you build your own custom bundle that contains only the components you need. The builder will generate a custom JavaScript and both a full and structure-only theme stylesheet for production use.

Try the [download builder tool](#) now (Alpha).

Latest Stable Version: 1.2.0

We provide CDN-hosted versions of jQuery Mobile for you to include into your site. These are already minified and compressed – and host the image files as well. It'll likely be the fastest way to include jQuery Mobile in your site.

CDN-Hosted JavaScript:

- [Uncompressed: jquery.mobile-1.2.0.js](#) (useful for debugging)
- [Minified and Gzipped: jquery.mobile-1.2.0.min.js](#) (24KB, ready to deploy)

Para acessar a documentação da biblioteca e exemplos acesse:

<http://jquerymobile.com/demos/1.2.0/>

Estrutura básica de uma página no JqueryMobile

```

  6<html>
  7  <head>
  8    <title>Minha página no JqueryMobile</title>
  9    <meta charset="utf-8">
 10   1 <meta name="viewport" content="width=device-width, initial-scale=1">
 11   2 <link rel="stylesheet" href="css/jquery.mobile-1.1.0.min.css" />
 12   <script src="js/jquery-1.8.3.min.js"></script>
 13   <script src="js/jquery.mobile-1.2.0.min.js"></script>
 14 </head>
 15 <body>
 16
 17 <div data-role="page">
 18
 19  <div data-role="header">
 20    <h1>Minha página no JqueryMobile</h1>
 21  </div><!-- /header -->
 22
 23  <div data-role="content">
 24    <p>Olá Bem Vindo ao JqueryMobile</p>
 25  </div><!-- /content -->
 26
 27  <div data-role="footer"><!-- rodape -->
 28    rodapé
 29  </div><!-- rodape -->
 30
 31 </div><!-- /page -->
 32 </body>
 33 </html>

```

Figura 14 - Estrutura básica de uma página no JqueryMobile

- ✓ 1 – Aqui temos o cabeçalho de nossa página, onde na linha 9 definimos a codificação de caracteres.
- ✓ 2 – Para utilizarmos a biblioteca precisamos chamar o script do jquery e do jquery.mobile, além dos script precisamos chamar a folha de estilos, sempre que formos desenvolver uma página usando o jquery móbil, precisamos usar estes arquivos.
- ✓ 3 – Esta é a estrutura da página com jquery mobile, usando div's e adicionando os atributos.

Layout de uma página básica no JqueryMobile

Este é o layout básico da nossa página, não sendo obrigatório mais é uma estrutura recomendável, em nossas aulas vamos utilizar sempre este layout.

Cabeçalho (header) - ***data-role="header"***

Área de Conteúdo (content) - ***data-role="content"***

Rodapé (footer) - ***data-role="footer"***

Abaixo vemos como nossa página baseado no layout acima é apresentada, observe que com poucas linhas já criamos o designer atraente.



Um site jQuery Mobile que inicia com "doctype" o HTML5 pode tirar o máximo proveito de todos os recursos do framework. Porem dispositivos mais antigos com navegadores que não entendem HTML5 vai ignorar o "doctype", mas apresentara a página algumas funcionalidades é que poderão não ser reconhecidas.

Dentro da tag **<body>**, cada exibição ou "página" no dispositivo móvel é identificado com um elemento (geralmente uma div), com o atributo **data-role="page"**

Agora vamos conhecer os atributos mais comuns para utilizarmos em nossas páginas.

Atributos JqueryMobile.

Atributo	Descrição
data-role="page"	Define os estilos do css para página, com o papel de página.
data-role="header"	Define os estilos do css para cabeçalho da página, com o papel de cabeçalho.
data-role="content"	Define os estilos do css para o conteúdo da página, com o papel de conteúdo.
data-role="footer"	Define os estilos do css para o rodapé da página, com o papel de rodapé.
data-icon	Define um ícone para o elemento.
data-role	Atributo usado para definir vários widgets, como botões, páginas, barras de navegação, listas de visualização e outros.
data-transition=	Define o tipo de transição será usado.
data-rel=	Atributo para o link de âncora de uma página, podendo

	ser popup's e dialog.
data-inset="true"	Exibi um elemento listview com cantos arredondados quando true.
data-theme	Define um tema para o elemento.

Tema

Com o jQuery Mobile, é possível usar o atributo **data-theme** para aplicar um tema padrão ou cinco amostras adicionais, com os nomes de A até E, como vemos na tabela abaixo;

Tema	Estilo
Tema Padão	
Tema A - data-theme="a"	
Tema B - data-theme="b"	
Tema C- data-theme="c"	



Além destes temas podemos utilizar o ThemeRoller, no link <http://jquerymobile.com/themeroller/> onde podemos criar customizar um estilo para as nossas páginas, na imagem abaixo vemos a ferramenta do link anterior.

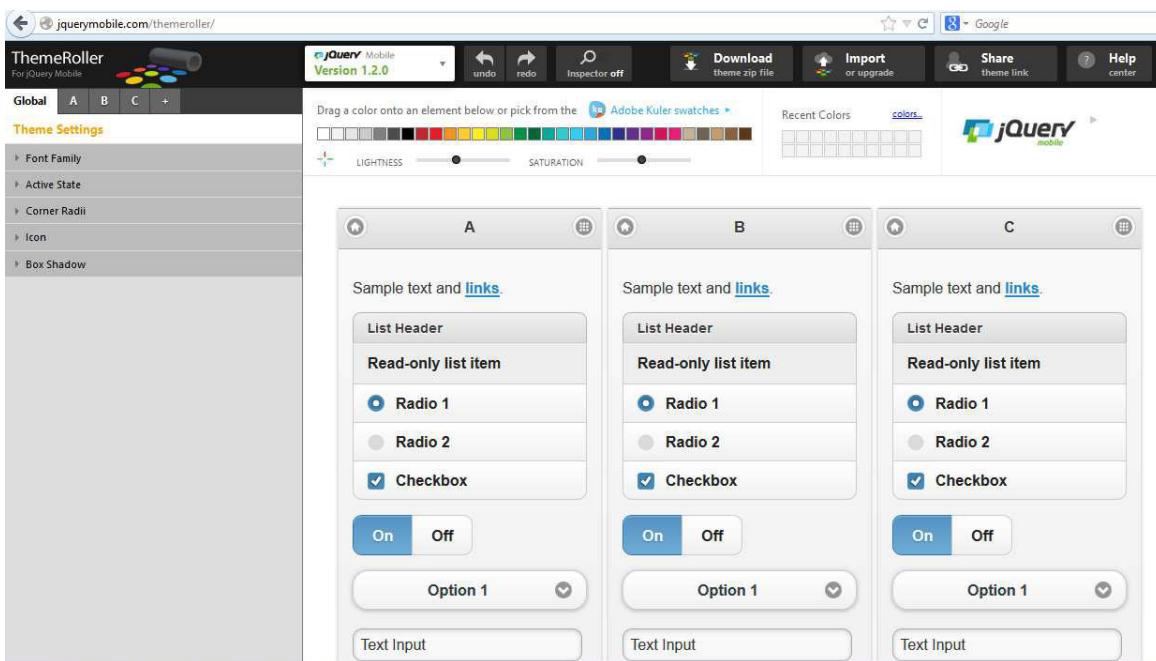


Figura 15 - Ferramenta ThemeRoller

3.10.1. Eventos.

No JavaScript, ao usar o jQuery, é possível fazer referência ao objeto jQuery em si como `\$` e obter acesso aos recursos do jQuery. A estrutura jQuery Mobile, que amplia o núcleo do jQuery, está disponível por meio do `\$.mobile`, que também dá acesso aos eventos e métodos específicos do jQuery Mobile.

Assim como acontece com outros eventos do jQuery, os eventos do jQuery Mobile são ligados usando as funções **live()** e **bind()**. Por exemplo, os eventos de toque incluem dar um toque, dar um toque e manter e vários eventos de deslizar os dedos e de mouse virtual. É possível ligar a mudanças de orientação e eventos de rolagem, como scrollstart e scrollstop. Os eventos de página permitem receber notificações:

- Antes da criação de uma página
- Quando uma página é criada
- Logo antes de mostrar ou ocultar uma página
- Quando uma página é mostrada e oculta

Abaixo tipos de eventos:

- tap/clica
- taphold/clica-segura
- swipe/clica-arrasta
- swipeleft/clica-arrasta-esquerda
- swiperight/clica-arrasta-direita

Mais exemplos e informações disponíveis na documentação da biblioteca, no link <http://jquerymobile.com/demos/1.0rc1/docs/api/events.html>.

Exemplo: Criamos uma página para simular alguns eventos, abaixo passo para o desenvolvimento deste exemplo.

1º passo: Cria uma página na estrutura JqueryMobile, como foi feito na aula inicial deste conteúdo.

2º passo: Adicionar os scripts jquery e jquery.mobile e o estilo css, caso não tenha feito ainda, lembre-se que o caminho dos scripts no diretório raiz do seu site.

```

12 | <link rel="stylesheet" href="css/jquery.mobile-1.1.0.min.css" />
13 | <script src="js/jquery-1.7.2.min.js"></script>
14 | <script src="js/jquery.mobile-1.1.0.min.js"></script>
```

3º passo: Criar alguns atributos css personalizados para aplicar no texto que será exibido, este estilo é criado na tag <head>, como é feito em aulas anteriores.

```

16     <style>
17         .ui-header .ui-title { white-space: normal; }
18         #eventos {
19             width: 300px;
20             min-height: 200px;
21             line-height: 100px;
22             background: white;
23             border: 2px solid black;
24         }
25         #eventos p {
26             font: bold 14px Sans-serif;
27             margin: 2px 5px;
28             color: #800000;
29         }
30         #eventos ul {
31             margin: 3px;
32         }
33         #eventos ul li {
34             color: red;
35             line-height: normal;
36         }
37         .ui-content { padding-left: 3px; }
38     </style>

```

4º passo: Criar script para os eventos utilizados, este será um script externo em JavaScript, também pode ser feito na própria página, mais optei por fazer uma chamada externa para ficar mais organizado.

```

1  $("#home").live("pageinit", function() {
2      var texto = '<p>Dispare aqui os seguintes eventos:\n</p><ul><li>tap/clica</li><li>taphold/clica-segura</li><li>swipe/clica-arrasta</li>\n<li>swipeleft/clica-arrasta-esquerda</li><li>swiperight/clica-arrasta-direita</li></ul></p>';
3
4
5
6      $("#eventos").html(texto);
7      //evento tap, toque ou clique
8      $("#eventos").bind("tap", function(e) {
9          $(this).append('<p>Disparado o evento: ' + e.type);
10     });//evento taphold/clica-segura
11     $("#eventos").bind("taphold", function(e) {
12         $(this).append('<p>Disparado o evento: ' + e.type);
13
14     });//evento swipe, clica-arrasta
15     $("#eventos").bind("swipe", function(e) {
16         $(this).append('<p>Disparado o evento: ' + e.type);
17     });//evento swipeleft, clica-arrasta-esquerda
18     $("#eventos").bind("swipeleft", function(e) {
19         $(this).append('<p>Disparado o evento: ' + e.type);
20     });//evento swiperight, clica-arrasta-direita
21     $("#eventos").bind("swiperight", function(e) {
22         $(this).append('<p>Disparado o evento: ' + e.type);
23     });
24     $("#eventos")
25     .next()
26     .bind("tap", function() {
27         $("#eventos").html(texto);
28     });
29 });

```

Linha 1: os eventos serão aplicados no atributo **#home**, que deverá ser o mesmo nome do id da **<div data-role="Page"** , carregados par a função no evento **live("pageinit")**.

Linha 2: declarar uma variável texto contendo o texto e algumas tag HTML, para serem inseridos dentro da div **eventos**. Nas outras linhas, quando os eventos especificados ocorrerem será mostrado o conteúdo da variável texto referente ao evento usado, não se esqueça de depois do script feito adicioná-lo na pagina.

5º passo: Estrutura da página, a diferença desta para a da estrutura inicial é; A identificação da div Page com o **id="home"**, e a criação da div com o **id="eventos"**, onde ocorrerá os eventos.

```

44 44 <body>
45 45
46 46 <div data-role="page" id="home">
47 47
48 48     <div data-role="header">
49 49         <h1>JqueryMobile Eventos</h1>
50 50     </div><!-- /header -->
51 51
52 52     <div data-role="content">
53 53         <p>JqueryMobile Eventos</p>
54 54
55 55         <div id="eventos">
56 56             </div>
57 57             <button type="button" data-role="button" data-inline="true">Limpar</button>
58 58         </div><!-- /content -->
59 59
60 60     <div data-role="footer"><!-- rodape -->
61 61     rodapé
62 62 </div><!-- rodape -->
63 63
64 64 </div><!-- /page -->
65 65
66 66 </body>

```

3.10.2. Métodos e utilidades.

No Jquery mobile podemos utilizar alguns métodos da própria API, ou criar funções próprias com base em eventos e métodos da API jquery, abaixo temos uma tabela com alguns métodos da API JqueryMobile.

Tabela 1 - Métodos do jQuery Mobile

Método	Uso
\$.mobile.changePage	Para passar por meio de programação de uma página a outra. Por exemplo, para acessar a página teste.php usando uma transição de slide, use \$.mobile.changePage("teste.php", "slide") .
\$.mobile.loadPage	Para carregar uma página externa.
\$.mobile.showPageLoadingMsg	Para mostrar a mensagem de carregamento de página.
\$.mobile.hidePageLoadingMsg	Para ocultar a mensagem de carregamento de página.
\$.mobile.path.isSameDomain	Um método utilitário para comparar o domínio de duas URLs.
\$.mobile.activePage	Uma referência à página que está em visualização no momento.

Mais informações e outros exemplos podem ser consultados no link: <http://api.jquery.com/>

Exemplo: Neste exemplo vamos passar de uma página para outra usando o método **\$.mobile.changePage**, passando um tipo de transição.



1º passo: Criar a página principal, uma segunda e a terceira página, depois adicionar os scripts e o estilo css do jquery.mobile.

```

12 <link rel="stylesheet" href="css/jquery.mobile-1.1.0.min.css" />
13 <script src="js/jquery-1.7.2.min.js"></script>
14 <script src="js/jquery.mobile-1.1.0.min.js"></script>
```

2º passo: Script dentro da tag <head> na página principal.

```

15 <script>
16     $("#um").live("pageinit", function() {
17         $("#p2").bind("click", function(e) {
18             $.mobile.changePage("tema_B.html", {
19                 transition: "flip"
20             });
21         });
22         $("#p3").bind("click", function(e) {
23             $.mobile.changePage("eventos_jquerymobile.html", {
24                 transition: "slideup",
25                 changeHash: false
26             });
27         });
28     });
29 </script>
```

Linha 16: Definimos que o evento será aplicado no atributo **#um**, que será o id da div **page** da página principal. O evento usado é o **pageinit**, na inicialização da página.

Linha 18: Usamos o método **changePage** e passamos como argumento a página que será chamada, na linha 19 o efeito de transição usado.

3º passo: Estrutura da página principal, as tags <a> recebem o id **p2** e **p3**, correspondente aos atributos criados no script no inicio da página, o passo anterior.

```

33 <body>
34
35     <div data-role="page" id="um">
36         <div data-role="header">
37             <h1>JqueryMobile Métodos</h1>
38         </div><!-- /header -->
39         <div data-role="content">
40             <h1>Página um<br /><small>id desta página #um</small></h1>
41             <p>Clique o botão para acionar um método da API.</p>
42             <div data-role="controlgroup" id="botões">
43                 <a href="tema_B.html" data-role="button" id="p2">pagina dois</a>
44                 <a href="eventos_jquerymobile.html" data-role="button" id="p3">pagina tres</a>
45             </div>
46         </div><!-- /content -->
47         <div data-role="footer"><!-- rodape -->
48             rodapé
49         </div><!-- rodape -->
50     </div><!-- /page -->
51
52 </body>

```

Transições baseadas em CSS

No exemplo anterior usamos algumas transições na chamada de algumas páginas, na tabela abaixo temos outros efeitos de transição. As transições podem ser utilizadas na passagem de página dentro dos atributos de página, nas próximas aulas usaremos algumas.

Transição	Uso
fade	Efeito de transição fade in/fade out
pop	Efeito de transição pop
flip	Efeito de transição de inversão
turn	Efeito de transição de giro
flow	Efeito de transição de fluxo (semelhante ao slide)
slide	Efeito de transição de slide (horizontal)
slideup	Mostra a página ou diálogo deslizando de baixo para cima na página
slidedown	Mostra a página ou diálogo deslizando de cima para baixo na página
none	Sem efeito de transição

Tabela 2- Transições baseadas em CSS

3.10.3. Widgets.

O atributo **data-role** é usado para definir os diversos widgets. Entretanto, nem todos os widgets da UI são acionados pelo atributo **data-role**, abaixo vamos conhecer alguns widgets.

➤ Botão (button)

Este widget é um botão básico, é iniciado na tag e passados os atributos do Jquery Mobile, podendo usar ícones e temas da biblioteca.

```

24     <div data-role="content">
25         <a href="index.html" data-role="button" data-icon="info">Botão</a>
26     </div><!-- /content -->

```

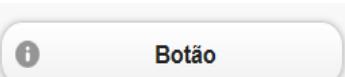
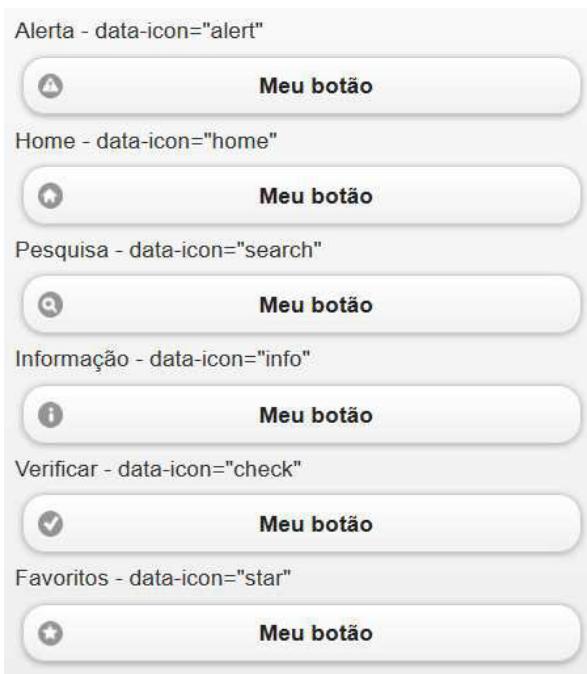


Figura 16 - Widget button

O atributo **data-icon** pode ser referenciado para criar os ícones mostrados abaixo:



➤ Entrada de texto (input)

Para usar este widget basta usar a tag **<input>** normalmente assim como no HTML, bastando os scripts e o estilo terem sido adicionados na página.

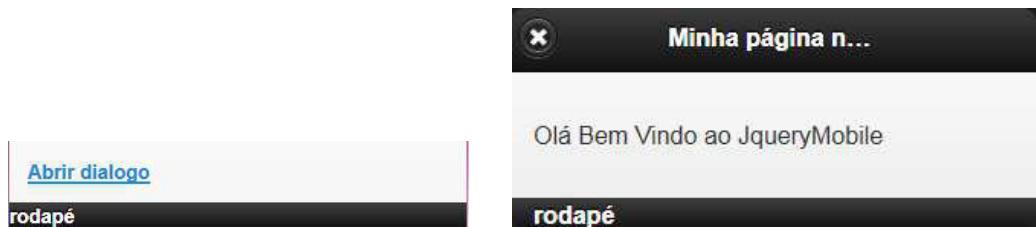
Olá Bem Vindo ao JqueryMobile
 Entrada de texto:
 rodapé

```

24 <div data-role="content">
25   <p>Olá Bem Vindo ao JqueryMobile</p>
26   <label for="fld_txt">Entrada de Texto:</label>
27   <input type="text" name="name" id="fld_txt" value="" />
28 </div><!-- /content -->
  
```

➤ Dialogo (dialog)

Este widget exibe uma caixa de dialogo, como mostrado na imagem abaixo.



Vamos criar uma página na estrutura básica do JqueryMobile, na linha 22 adicionamos a tag <a> nela passamos a página que será chamada dentro do dialogo, para este exemplo crie uma página chamada **teste.html** com uma estrutura básica JqueryMobile e digite uma mensagem de boas vindas.

```

16 <body>
17     <div data-role="page">
18         <div data-role="header">
19             <h1>Minha página no JqueryMobile</h1>
20         </div><!-- /header -->
21         <div data-role="content">
22             <a href="teste.html" data-rel="dialog" data-transition="pop">Abrir dialogo</a>
23         </div><!-- /content -->
24         <div data-role="footer"><!-- rodape -->
25             rodapé
26         </div><!-- rodape -->
27     </div><!-- fim page 1 -->
28 </body>

```

Nesta mesma linha é adicionado o atributo **data-rel**, onde pode ser passado o valor **dialog** ou **popup**, em outro atributo o data **transition** será responsável pelo efeito de transição.

➤ Lista (listview)

As listas de visualização(ListView) utilizam o atributo **data-role**, elas podem ter o formato quadrado como na imagem abaixo, para tornar cantos arredondados utiliza o atributo **data-inset="true"**.



Figura 18 - ListView com data-inset=false



Figura 17 - ListView com data-inset=true

Exemplo: Listview.

Linha 29: Adicionar o atributo **data-role="listview"**.

```

27 <div data-role="content">
28     <p>Meio de transporte</p>
29     <ul data-role="listview" >
30         <li><a href="#carro" >Carro</a></li>
31         <li><a href="index.html">Avião</a></li>
32         <li><a href="index.html" data-rel="dialog" data-transition="pop">Aeroplano</a></li>
33     </ul>
34 </div><!-- /content -->

```

➤ Lista redutível

Este widget usamos o atributo **data-role="collapsible"**, para utilizá-lo devemos criar uma div e nela aplicar o atributo, para definir o que será o título da lista use a tag **<h3>**

```

24 <div data-role="content">
25   <div data-role="collapsible">
26     <h3>ThemeRoller</h3>
27     <p>
28       Além destes temas podemos utilizar o ThemeRoller, no link
29       <a href="http://jquerymobile.com/themeroller/">http://jquerymobile.com/themeroller/</a>
30       onde podemos criar customizar um estilo
31       para as nossas páginas, na imagem abaixo vemos a ferramenta do link anterior.
32     </p>
33   </div>
34 </div><!-- /content -->

```

➤ CheckBox

Este widget não precisa do atributo **data-role**, basta ser inserido na página de conteúdo, como é mostrado no código abaixo.

```

24   <div data-role="content">
25     <input type="checkbox" name="checkbox-1" id="checkbox-0" checked="" />
26     <label for="checkbox-0">Checkbox</label>
27   </div><!-- /content -->

```

➤ Botão de opções

Neste exemplo criamos inputs do tipo radio, como é feito em HTML, e colocamos o atributo **data-role="controlgroup"**, o restante do código são tags em HTML.

```

Widget Botão de Opção

Escolha um animal:

 Gato
 Cachorro
 Rato

```

Abaixo código da estrutura da página.

```

24 <div data-role="content">
25     <p>Widget Botão de Opção</p>
26
27         <fieldset data-role="controlgroup">
28             <legend>Escolha um animal:</legend>
29             <input type="radio" name="radio-choice-1" id="radio-choice-1"
30             value="choice-1" />
31             <label for="radio-choice-1">Gato</label>
32             <input type="radio" name="radio-choice-1" id="radio-choice-2"
33             value="choice-2" />
34             <label for="radio-choice-2">Cachorro</label>
35             <input type="radio" name="radio-choice-1" id="radio-choice-3"
36             value="choice-3" />
37             <label for="radio-choice-3">Rato</label>
38         </fieldset>
39
40     </div><!-- /content -->

```

➤ Fomulários (form)

Vamos agora criar um formulário de contato, nesta aula vamos utilizar a linguagem PHP junto com o JqueryMobile, o resultado final desta aula é mostrado na imagem abaixo.

1º passo: Criar duas páginas uma vai ser o formulário de contato **form_contatos.php**, outra para apresentar os dados **dados_contato.php**, inicialmente criaremos duas páginas com a estrutura básica para JqueryMobile. No formulário de contato podemos adicionar os elementos como na abaix, no outro serão enviadas as informações via código PHP.

2º passo: Formulários de contato, nesta página deveram dar uma atenção especial para o nome dos elementos.

Linha 22: Na tag `<form>` estamos definindo qual será o **action** do formulário, ou seja sua ação, o arquivo **dados_contato.php** será responsável por exibir em uma nova página os dados digitados no formulário. Depois definimos o método **post**, onde os dados que estão sendo enviados serão mostrados na página.

```

16  <body> |
17   <div data-role="page">
18     <div data-role="header">
19       <h1>Formulário de contato</h1>
20     </div><!-- /header -->
21     <div data-role="content">
22       <form action="dados_contato.php" method="post" id="form_contatos" data-transition="slideup">
23         <p>Formulário de contato</p>
24         <label>
25           Nome para Contato
26           <input name="nome" type="text" size="40" />
27         </label>
28         <label>
29           E-mail<br>
30           <input type="text" size="40" value="" id="e_mail" name="e_mail"/>
31         </label>
32         <label>
33           Telefone
34           <input type="text" name="tel" value="" size="40" />
35         </label>
36         <input type="submit" value="Enviar" />
37       </form>
38     </div><!-- /content -->
39     <div data-role="footer"><!-- rodape -->
40       Rodapé
41     </div><!-- rodape -->
42   </div><!-- /page -->
43 </body>

```

3º passo: A página que receberá os dados do contato.

```

23   <div data-role="content">
24     <p>Dados de Contato</p>
25     <?php
26       echo '<b>Nome: </b>', $nome = $_POST['nome'];
27       echo '<br><b>Email: </b>', $email = $_POST["e_mail"], '</b><br>';
28       echo '<b>Telefone: </b>', $tel = $_POST["tel"], '</b>';
29     ?>
30   </div><!-- /content -->

```

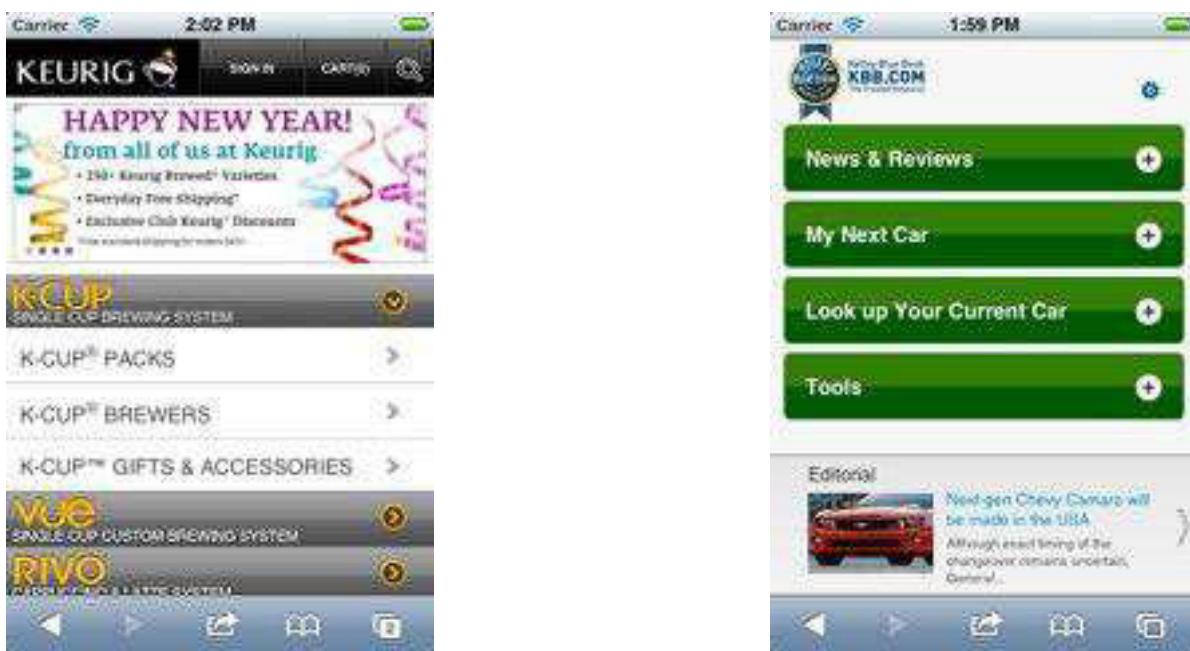
Este trecho de código do arquivo **dados_contato.php**, será responsável por pegar o valor dos inputs do formulário através do método **post**. Nesta página o trecho em PHP foi inserido na tag de conteúdo como apresentada na imagem acima.

Na linha 26 temos a instrução **echo** que gera a escrita na página com o conteúdo que esta sendo passado, dentro das aspas simples passo tags HTML para estruturar a página e apresentar o resultado como na imagem abaixo.



Definimos a variável **\$nome** que vai receber **\$_POST['nome']**, onde **\$_POST** representa obviamente o método que esta sendo passado, o qual foi definido no formulário, o valor '**nome**' representa o nome do input que esta no formulário.

A seguir temos alguns exemplos de páginas usando o JqueryMobile.



Exercício Prático

- 1) Criar um formulário aplicando os widgets aprendidos, com a transição de páginas.
- 2) Criar um formulário de cadastro de contatos.
- 3) Formulário de cadastro de cliente.

3.10.4. Twitter

Nesta aula iremos desenvolver uma pagina para receber as atualizações do Twitter, utilizaremos Jquery plugin twitter que se encontra no link

<http://coda.co.za/content/projects/jquery.twitter/>

jquery.twitter.js
Created by [Damien du Toit](#), Powered by [jQuery](#)

[Download](#) | [View JavaScript](#) | [View CSS](#)

Latest Tweets

Behold, the jQuery Plugin Registry is open for business! Check it out & publish your plugins! <http://t.co/GENvFUU3> | <http://t.co/aKv0qnkR>
[3 days ago](#)

Today's the day! jQuery 1.9 & jQuery Migrate are officially released, and the first beta of jQuery 2.0 is available! <http://t.co/OXgvDFoR>
[6 days ago](#)

It's been 7 years! As we begin 2013, what's the State of jQuery? Find out from @davemethvin: <http://t.co/mw4sXTdo>
[8 days ago](#)

Desenvolvido por Damien du Toit, clique no link download para baixar o arquivo compactado com os scripts que iremos utilizar, depois descompacte-o e coloque em um diretório do raiz do seu site ou em um diretório, neste exemplo criamos o diretório **tw**.

<http://coda.co.za/content/projects/jquery.twitter/>

Na imagem abaixo vemos os scripts que utilizaremos nesta prática, nas linhas 17 e 18 temos o script do plugin para Twitter e a folha de estilo, os outros scripts já são utilizados na estrutura básica de nossas páginas em aulas anteriores.

```

12 |     <link rel="stylesheet" href="css/jquery.mobile-1.1.0.min.css" />
13 |
14 |     <script src="js/jquery-1.8.3.min.js"></script>
15 |     <script src="js/jquery.mobile-1.1.0.min.js"></script>
16 |     <!-- JavaScript para o Twitter -->
17 |     <script src="tw/jquery.twitter.js"></script>
18 |     <link rel="stylesheet" href="tw/jquery.twitter.css"/>

```

Na **<head>** vamos criar a função **carregar_tw()**, neste script definido na linha 25 a marcação **#twitter** vai pegar as atualizações do twitter, a propriedade **username** na linha 26 recebe o nome do twitter, neste exemplo usamos o do g1.

```

22 |         <script type="text/javascript">
23 |             function carregar_tw() {
24 |                 $(document).ready(function() {
25 |                     $("#twitter").getTwitter({
26 |                         userName: "g1",
27 |                         numTweets: 5,
28 |                         loaderText: "Carregando tweets...",
29 |                         slideIn: true,
30 |                         showHeading: true,
31 |                         headingText: "Últimos Tweets",
32 |                         showProfileLink: true
33 |                     });
34 |                 });
35 |
36 |             }
37 |         </script>

```

As linhas 28 e 31 definem o texto que será apresentado quando estiver carregando as atualizações do twitter, e o texto do cabeçalho da área onde serão mostradas as atualizações.

➤ Estrutura da página

Em nossa página na estrutura HTML só precisamos na tag **<body>** usar o evento **onload** e chamar a função **carregar_tw()**, que criamos anteriormente.

```

40 |         <body onload="carregar_tw()">
41 |

```

3.10.5. Geolocalização.

Nesta aula vamos fazer uma página usando a API do Google para geolocalização, no link <https://code.google.com/p/jquery-ui-map/>, podemos obter vários outros exemplos e baixar os arquivos que serão utilizados neste exemplo, este plugin pode ser usado para Jquery e JqueryMobile.



Na imagem ao lado temos o resultado final desta prática, este exemplo é bem simples e especialmente utilizado para iniciantes, no site do projeto obtemos toda a documentação do plugin, então vamos iniciar.

Clique em [Download the plugin with examples](#), para baixar o plugin, conforme a imagem abaixo.

jquery-ui-map
Google map v3 plugin for jQuery and jQuery Mobile

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

[Summary](#) [People](#)

Project Information

+68 Recommend this on Google
[Project feeds](#)

Code license
MIT License

Labels
jQuery, UI, map, JavaScript, GMapV3, jqm, Mobile

Members
johansallarsson
1 committer
8 contributors

Featured

[Downloads](#)
[jquery-ui-map-3.0-rc.zip](#)
Show all 2

Links

External links
[Twitter](#)

Groups
[Get help in the forum](#)

Google maps v3 plugin for jQuery and jQuery Mobile

The Google Map version 3 plugin for jQuery and jQM takes away some of the head aches from working with the Google Map API. Instead of having to use Google event listeners for simple events like click, you can use jQuery click events on the map and markers.

It is also very flexible, highly customizable, lightweight (3.2kB or 3.9kB for the full) and works out of the box with jQuery mobile. But one of its best features (atleast for SEO people) is that you can populate a map from microformats. RDFa or microdata on your site, which can be used as a fallback when a user doesn't have javascript enabled.

[Download the plugin with examples](#) or [get the minified file here](#)

Donate

[Donate](#)

[Entities which have donated](#)

jQuery mobile demo

- [jQuery Mobile example](#)

Demo

- [Benchmark](#)
- [Basic example](#)
- [Streetview example](#)
- [Geolocation example](#)

Neste exemplo vamos baixar o arquivo zip da versão 3 (jquery-ui-map-3.0-rc.zip). Após o download descompacte o arquivo no raiz do seu projeto, ou onde preferir.

[code.google.com/p/jquery-ui-map/downloads/list](#)

jquery-ui-map
Google map v3 plugin for jQuery and jQuery Mobile

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

[Search](#) Current downloads [for](#) [Search](#)

Filename ▾	Summary + Labels ▾
jquery-ui-map-3.0-rc.zip	jQuery UI Google map v3 version 3.0-rc1 Featured
jquery-ui-map-3.0-beta.zip	jQuery UI Google map v3 version 3.0-beta
jquery-ui-map-3.0-alpha.zip	jQuery UI Google map v3 version 3.0-alpha
jquery-ui-map 2.0.2.zip	jQuery UI Google map v3 version 2.0.2

➤ Agora vamos chamar os arquivos baixados do site.

Na linha 12 estou chamando o script responsável pela API básica e passo como parâmetro a ativação do sensor, neste caso esta ativado, pois nossa página será acessada tanto em desktops como em dispositivos móveis, este exemplo precisa de internet para carregar o Google Maps.

Na linha 19 estou chamando o script responsável pelas funções que o plugin implementa, como localizar por coordenadas, nível de zoom e outras.

```

7   <head>
8     <title>Minha página no JqueryMobile</title>
9     <meta charset="utf-8">
10    <meta name="viewport" content="width=device-width, initial-scale=1">
11    <!-- Carrega a API do google e ativa ou desativa o sensor de localização--&gt;
12    &lt;script type="text/javascript" src="https://maps.google.com/maps/api/js?sensor=true"&gt;
13    &lt;/script&gt;
14    &lt;!-- Folha de estilo utilizada--&gt;
15    &lt;link rel="stylesheet" href="css/jquery.mobile-1.1.0.min.css" /&gt;
16    &lt;!-- Scripts Jquery e JqueryMobile--&gt;
17    &lt;script src="js/jquery-1.7.2.min.js"&gt;&lt;/script&gt;
18    &lt;script src="js/jquery.mobile-1.1.0.min.js"&gt;&lt;/script&gt;
19    &lt;script src="jquery-ui-map-3.0-rc/ui/min/jquery.ui.map.full.min.js"&gt;&lt;/script&gt;
</pre>

```

➤ Script

Este script foi escrito dentro da tag **<head>**, nele declaramos a variável **LatLng** onde será armazenada as informações de latitude e longitude, através dela que na imagem abaixo encontra-se na linha 26, a API retorna o mapa baseado nas coordenadas, o formato de coordenadas esta decimal,mas podem ser convertidas em varias formas de apresentação no site <http://www.sunearthtools.com/dp/tools/conversion.php?lang=pt>, encontra-se uma ferramenta que alem de pegar as coordenadas ainda apresenta a conversão em vários formatos.

Na linha 27 criamos a marcação **#map_canvas** que será usado para mostrar o mapa.

```

23   <script type="text/javascript">
24     $(function() {
25
26       var LatLng = new google.maps.LatLng(-4.9158328, -39.6386719);
27       $('#map_canvas').gmap({ 'center': LatLng });
28     });
29   </script>

```

➤ Estrutura da página

Como vemos na imagem abaixo para carregar o mapa em nossa página, neste exemplo criamos uma div onde será carregado o mapa na linha 42 fazemos a referência ao atributo **map-canvas** que foi criado na **<head>** no script anterior, e com a propriedade style passamos o tamanho da área do mapa.

```
32 <body>
33     <div data-role="page">
34
35         <div data-role="header">
36             <a href="index.html" data-role="button" data-icon="home">Home</a>
37             <h1>Geolocalização</h1>
38         </div><!-- /header -->
39
40         <div data-role="content">
41             <p>Olá Bem Vindo ao JqueryMobile</p>
42             <div id="map_canvas" style="width:250px;height:250px"></div>
43         </div><!-- /content -->
44
45         <div data-role="footer" ><!-- rodape -->
46             Rodapé
47         </div><!-- rodape -->
48     </div><!-- /page -->
49 </body>
```

No site esta disponível a documentação completa do plugin http://code.google.com/p/jquery-ui-map/wiki/jquery_ui_map_v_3_api

Outros exemplos podem ser encontrados no link <http://jquery-ui-map.googlecode.com/svn/trunk/demos/jquery-google-maps-mobile.html>

4. Formulários web.

Nesta aula vamos desenvolver formulários para aplicarmos os efeitos aprendidos em aulas anteriores.

Para criamos formulários usamos a tag **<form></form>**, nela podemos usar alguns atributos, por exemplo, **action**, **method** e **id**, no atributo **method** vão ser passados o método **get** ou **post**, no **action** uma ação que será executada pelo formulário quando o mesmo enviar as informações para o servidor, este dois pontos abordamos no inicio da aula 2 – Introdução ao PHP e será aprofundado no próximo semestre como o PHP, e nesta aula iremos criar formulários que serão utilizados no próximo semestre com diversas funcionalidades.

4.1. Formulário de cadastro de clientes.

Nesta aula vamos criar um formulário de cadastro e enviar os dados via método post, abaixo imagem de como ficara o nosso formulário.

The screenshot shows a web-based client registration form. At the top, a header bar reads 'Cadastro de Cliente'. Below it, a title 'Formulário de Cadastro de Clientes' is displayed. The form contains five input fields: 'Nome' (Name), 'Endereço' (Address), 'E-mail' (Email), and 'Telefone' (Phone). A 'Salvar' (Save) button is located at the bottom right of the form area.

Para este formulário criaremos um arquivo **form_cli** com a extensão *.php, quando o usuário preencher os campos neste formulário e salvar será chamado um outro arquivo, **dados_cli.php**, será neste arquivo onde enviaremos os dados do **form_cli** para serem exibidos.

➤ Folha de estilo

```

1  body {
2      width:600px;
3      font: 80%/1.2 Arial, Helvetica, sans-serif;
4      margin:30px auto;
5      color:#666;
6      padding:0;
7  }
8  label {
9      width:190px;
10     display:block;
11     float:left;
12  }
13  fieldset{border:1px solid #ccc;padding:10px;}
14  legend {
15      font-weight:bold;
16      border:1px solid #ccc;
17      padding:1px 4px;
18  }
19  label.cab {
20      font-weight:bold;
21      margin:0;
22  }
23  form div {margin:15px 0 0 0;}
24  .form_cli, .cab {
25      width:100%;
26      overflow:auto;
27      font-weight:bold;
28  }
29  .form_cli{background:#d6e2e5;}
30  .cab{background:#add6ef;padding:10px 0;}

```

Nesta etapa do curso já sabemos manipular bem a folha de estilo css, então criaremos o arquivo css separado e faremos o link dentro da tag <head> para que quando a pagina for carregada seja carregada a formatação do formulário.

Desta forma iremos chamar a nossa folha de estilo dentro da **<head>**.

```
<link href="../css/estilo_form.css" rel="stylesheet">
```

➤ Estrutura da página

Como já temos conhecimento sobre as tag HTML, vamos abordar apenas alguns pontos importantes da estrutura da página.

```
14  <body>
15      <form action="dados_cli.php" method="post" id="formulario">
16          <fieldset>
17              <legend>Cadastro de Cliente</legend>
18              <span class="form_cli">
19
20                  </span>
21              <div class="form_cli">
22
23                  <div class="cab" align="center">Formulário de Cadastro de Clientes</div>
24                  </label>
25                  <table width="316" border="0" align="center">
26                      <tr>
27                          <td width="310">
28                              <label>
29                                  Nome
30                                  <input type="text" size="40" name="nome" />
31                              </label>
32                          </td>
33                      </tr>
34                      <tr>
35                          <td>
36                              <label>Endereço
37                                  <input type="text" size="40" name="end" />
38                              </label>
39                          </td>
40                      </tr>

```

Linha 15: Na tag **<form>** estamos definindo qual será o **action** do formulário, o arquivo **dados_cli.php** será responsável por exibir em uma nova página os dados digitados no formulário. Depois definimos o método **post**, responsável por enviar os dados. Como utilizamos o **post** os dados não serão mostrados na URL.

E por último definimos o **id** do formulário que é a identificação do mesmo.

Linha 18: Na tag **** adicionamos a propriedade **form_cli** que foi criada em nossa folha de estilo.

Linha 23: Na tag **<div>** fazemos a formatação do cabeçalho do formulário que também vem da nossa folha de estilo criado anteriormente.

Arquivo dados_cli.php

```

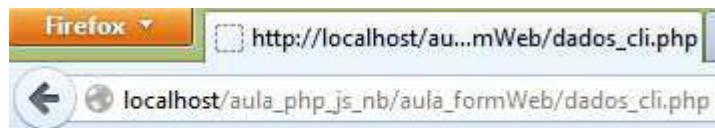
1 <?php
2
3 echo '<b>Nome: </b>', $nome = $_POST['nome'];
4 echo '<br><b>Endereço: </b>', $end = $_POST["end"], '</b><br>';
5 echo '<br><b>Email: </b>', $email = $_POST["e_mail"], '</b><br>';
6 echo '<b>Telefone: </b>', $tel = $_POST["tel"], '</b>';
7
8 ?>

```

Este trecho de código do arquivo **dados_cli.php**, será responsável por pegar o valor dos inputs do formulário pelo método **post**. Neste arquivo a única coisa que foi digitada nela esta apresentada na imagem acima.

Na linha 3: Temos a instrução **echo** que gera a escrita na página com o conteúdo que esta sendo passado, dentro das aspas simples passo tags HTML para estruturar a página e apresentar o resultado como na imagem abaixo.

Definimos a variável **\$nome** que vai receber **\$_POST['nome']**, onde **\$_POST** representa obviamente o método que esta sendo passado, o qual foi definido no formulário, o valor '**nome**' representa o nome do input que esta no formulário, logo se quisermos passar valores por meio dos métodos **get** ou **post** utilizo **\$_POST['nome do objeto do formulário']** ou **\$_GET['nome do objeto do formulário']**.



Nome: João Paulo de Oliveira Lima

Endereço: rua monte-mor n 483

Email: jplima@gmail.com

Telefone: (85)8859-3126

Este será o resultado final do nosso formulário.



Exercício Prático

- 1) Criar um efeito zebrado em uma tabela, com os dados de modelo do carro e suas características, usando a biblioteca Jquery.
- 2) Criar um formulário de cadastro de notas, para ler 3 notas e apresentar a média.

4.2. Formulário de Contato.

Nesta aula iremos desenvolver um formulário de contato, usado para o internauta entrar em contato com o administrador do site, iremos utilizar PHP para validação do email, JavaScript e folha de estilos.

Para isso iremos construir este formulário com algumas funções são estas;

- Campo formatado, adicionado uma máscara no campo de telefone.
- Validando o campo email para que o usuário informe o email de forma correta.
- Incluir folha de estilo CSS.

O resultado final desta prática será como na imagem abaixo;

The screenshot shows a web-based contact form. At the top, a header bar reads "Informe seus dados para Contato". Below it, the main title is "Formulário de Contatos". The form contains three input fields: "Nome para Contato" (Name for Contact), "E-mail" (Email), and "Telefone" (Phone). The phone field includes a placeholder "()". At the bottom right of the form is a blue "Enviar" (Send) button.

Figura 19 - Formulário de Contatos

A folha de estilo será a mesma utilizada na aula anterior, nesta aula o principal objetivo será trabalhar com validação de campos e máscara de formato, por exemplo, o formato do telefone.

Iremos iniciar com o script em JavaScript criado na tag <head>, observe que usamos uma estrutura de decisão em JavaScript que aprendemos em aulas anteriores, temos uma função

chamada `Mascara` que recebe um objeto, depois testa a igualdade do valor do objeto, este objeto será o campo que receberá o formato.

```

8      <script>
9      function Mascara(objeto) {
10         //verifica se o valor do objeto
11         //é igual a 0, para inserir o 1º caractere
12         if(objeto.value.length == 0)
13             objeto.value = '(' + objeto.value;
14
15         if(objeto.value.length == 3)
16             objeto.value = objeto.value + ')';
17
18         if(objeto.value.length == 8)
19             objeto.value = objeto.value + '-';
20     }
21
22     </script>

```

Abaixo na estrutura html iremos aplicar o nosso script, no evento **onKeypress** faço a chamada da função **Mascara(this)**, o **this** indica que será aplicado neste objeto no caso o input.

```

51     <td><label>
52         Telefone
53         <input type="text" name="tel" value="" size="40" onkeypress="Mascara(this);"/>
54     </label>
55 </td>
```

Agora vamos fazer a validação do email, para evitar que o usuário informe um email errado.

Na tag **<form>** definimos o action o arquivo **valida_email.php** que será chamado na ação deste formulário, agora no method usamos o método get, e a identificação do nosso formulário na propriedade id.

```

27 <form action="valida_email.php" method="get" id="formulario">
28     <fieldset>
29         <legend>Informe seus dados para Contato</legend>
30         <span class="form_cont">
```

Escrevendo o **valida_email.php**

Neste exemplo usamos uma função nativa do php a **filter_var**, que pode ser usada para validar diversas coisas, nela adicionamos o filtro de email através da propriedade **FILTER_VALIDATE_EMAIL**.

```

1 <?php
2
3 $email = $_GET['e_mail'];
4 if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
5     require("form_contato.php");
6     echo '<h1>E-mail válido</h1>';
7 } else {
8     require("form_contato.php");
9     echo '<h1>E-mail invalido</h1>';
10 }
11 ?>
```

Nas linhas 5 e 8 usamos função require que fará uma requisição de página, em nosso exemplo chamará o **form_contato.php** que é o formulario de onde vieram o email para validar, fazendo com que

seja exibido a mensagem de email valido ou invalido abaixo do formulario, como é mostrado na imagem seguinte.

E-mail válido

E-mail invalido



Exercício Prático

- 1) No formulário de contato desenvolvido em sala de aula adicione as legendas tooltips com jquery, para os campos.
- 2) Desenvolver uma página de contatos com JqueryMobile.
- 3) Crie outra página para enviar os dados cadastrados no formulário de contatos.
- 4) Crie um formulário em PHP que receba a altura e o sexo de uma pessoa, calcule e imprima o seu peso ideal, utilizando as seguintes fórmulas:
 - para homens: $(72.7 * H) - 58$
 - para mulheres: $(62.1 * H) - 44.7$

5. Estudos de Caso



Caro Aluno (a),

Nesta aula iremos praticar usando os conhecimentos aprendidos, o objetivo de trabalhar com estudos de caso, é trazer aos alunos vivência de uma situação real no desenvolvimento web.

Os dois estudos apresentados serão continuados na disciplina de PHP/MySQL no próximo semestre, nesta etapa iremos trabalhar a leitura e a discussão de como resolver uma situação problema.

Serão desenvolvidos nesta fase projetos gráficos com algumas interações, para que na próxima disciplina iremos utilizar estes exemplos para realizar a integração com o banco de dados e outras práticas.

5.1. Carrinho de compras.

Um empresário observando o crescimento da internet e a possibilidade de fechar negócios convocou uma equipe para desenvolver um protótipo de uma loja virtual, após a aprovação do empresário a equipe terá acesso ao servidor da empresa para implantar o banco de dados e hospedar o site.

Em uma reunião foram discutidos alguns pontos:

A loja virtual em sua página inicial deverá apresentar uma vitrine com as principais ofertas, de onde o usuário poderia visualizar e ao selecionar algum produto mostrar o produto e outras ofertas.

Os produtos desta empresa são divididos por categoria, quando o cliente selecionar alguma categoria será exibido todos os produtos referentes àquela categoria.

Quando o cliente visualizar na página um produto o mesmo poderá comprá-lo ou adicioná-lo em um carrinho de compras para depois escolher o que irá comprar ou descartar, ao adicionar no seu carrinho de compras o produto não será dada baixa no estoque, somente quando a compra for efetivada é que será dada baixa no estoque.

Para melhor conhecer o cliente e melhor controle o cliente deve se cadastrar na loja virtual, para o mesmo visualizar os seus pedidos e ofertas especiais para o cliente.

A loja virtual oferece vários meios de pagamento para o cliente.



O tipo de produto que será vendido na loja virtual depende da equipe, o empresário irá investir na melhor proposta.

5.2. Chat de atendimento.

Um empresário sentindo a necessidade de uma melhor comunicação com seus clientes para tirar dúvidas.

Requisitou a uma equipe de desenvolvedores a criação de um protótipo de um chat de atendimento a clientes dentro de um site, para o próprio cliente tirar duvidas com relação aos produtos ou serviços oferecidos pela empresa.

Para este contato com um atendente o cliente é necessário um cadastro para ser direcionado ao ambiente do chat.

Com relação ao horário de funcionamento do atendimento ficou estabelecido que fosse de 7:00 horas da manhã até as 12:00h e de 13:00h as 18:00h, quando o chat estiver fora do horário de atendimento deverá ser exibido um ícone que o mesmo está off-line.

O empresário pediu que fosse apresentada também uma versão para dispositivos móveis do site para uma aprovação do site em uma nova plataforma.

Referências Bibliográficas

Niederauer, J. *Web Interativa com Ajax e PHP*. Novatec.

Silva, M. S. *JQuery A Biblioteca*

Almeida, R. S. *Php Para Iniciantes*. CIENCIA MODERNA.

Bibeault, B., & Katz, Y. *Jquery em Ação*. Alta Books .

Dall'oglio, P. *PHP - Programando com Orientação a Objetos - 2ª Ed. 2009*. NOVATEC.

<http://jquerymobile.com/demos/1.0rc1/docs/api/events.html>. (s.d.). Acesso em 13 de 11 de 2012, disponível em jquerymobile: <http://jquerymobile.com/demos/1.0rc1/docs/api/events.html>.

<http://jquerymobile.com/demos/1.2.0/docs/pages/index.html>. (s.d.). Acesso em 20 de 11 de 2012, disponível em jquerymobile: <http://jquerymobile.com>

Legnstorff, J. *Pro PHP e Jquery. Php Para Iniciantes*.

Morrison, M. *Use a Cabeça Javascript*. Alta Books .

Powers, S. *Aprendendo Javascript*. Novatec.

Rutter, J. *Smashing Jquery - Interatividade Avançada Com Javascript Simples*. Bookman.

Silva, M. S. *Javascript - Guia do Programador*. Novatec.

Silva, M. S. *jQuery Mobile Desenvolva aplicações web para dispositivos móveis*. Novatec.

Silva, M. S. *Jquery Ui - Componentes de Interface Rica Para Suas Aplicações Web*. Novatec .

do Programador JavaScript. Novatec.

Índice de Ilustrações

Figura 1 - Estrutura JavaScript	10
Figura 2 - Declaração de variáveis	11
Figura 3 - Visualização da página para calcular IMC.....	23
Figura 4 - Script em JavaScript	27
Figura 5 - Estrutura HTML, chamando script.	27
Figura 6 - Tag inicial do PHP	29
Figura 7 - Tag PHP dentro do HTML	29
Figura 8 - Enviando dados ao servidor	31
Figura 9 - Declaração de Variáveis.....	35
Figura 10 - Exemplo de uma constante	36
Figura 11 - Sintaxe de uma função.....	45
Figura 12 - resultado gerado pela função var_dump	49
Figura 13 - Site Jquery.....	51
Figura 14 - Estrutura básica de uma página no JqueryMobile	66
Figura 15 - Ferramenta ThemeRoller	69
Figura 16 - Widget button	74
Figura 17 - ListView com data-inset=true.....	76
Figura 18 - ListView com data-inset=false	76
Figura 19 - Formulário de Contatos	88

Hino Nacional

Ouviram do Ipiranga as margens plácidas
De um povo heróico o brado retumbante,
E o sol da liberdade, em raios fúlgidos,
Brilhou no céu da pátria nesse instante.

Se o penhor dessa igualdade
Conseguimos conquistar com braço forte,
Em teu seio, ó liberdade,
Desafia o nosso peito a própria morte!

Ó Pátria amada,
Idolatrada,
Salve! Salve!

Brasil, um sonho intenso, um raio vívido
De amor e de esperança à terra desce,
Se em teu formoso céu, risonho e límpido,
A imagem do Cruzeiro resplandece.

Gigante pela própria natureza,
És belo, és forte, impávido colosso,
E o teu futuro espelha essa grandeza.

Terra adorada,
Entre outras mil,
És tu, Brasil,
Ó Pátria amada!
Dos filhos deste solo és mãe gentil,
Pátria amada, Brasil!

Deitado eternamente em berço esplêndido,
Ao som do mar e à luz do céu profundo,
Fulguras, ó Brasil, florão da América,
Iluminado ao sol do Novo Mundo!

Do que a terra, mais garrida,
Teus risonhos, lindos campos têm mais flores;
"Nossos bosques têm mais vida",
"Nossa vida" no teu seio "mais amores."

Ó Pátria amada,
Idolatrada,
Salve! Salve!

Brasil, de amor eterno seja símbolo
O lábaro que ostentas estrelado,
E diga o verde-louro dessa flâmula
- "Paz no futuro e glória no passado."

Mas, se ergues da justiça a clava forte,
Verás que um filho teu não foge à luta,
Nem teme, quem te adora, a própria morte.

Terra adorada,
Entre outras mil,
És tu, Brasil,
Ó Pátria amada!
Dos filhos deste solo és mãe gentil,
Pátria amada, Brasil!

Hino do Estado do Ceará

Poesia de Thomaz Lopes
Música de Alberto Nepomuceno
Terra do sol, do amor, terra da luz!
Soa o clarim que tua glória conta!
Terra, o teu nome a fama aos céus remonta
Em clarão que seduz!
Nome que brilha esplêndido luzeiro!
Nos fulvos braços de ouro do cruzeiro!

Mudem-se em flor as pedras dos caminhos!
Chuvas de prata rolem das estrelas...
E despertando, deslumbrada, ao vê-las
Ressoa a voz dos ninhos...
Há de florar nas rosas e nos cravos
Rubros o sangue ardente dos escravos.
Seja teu verbo a voz do coração,
Verbo de paz e amor do Sul ao Norte!
Ruja teu peito em luta contra a morte,
Acordando a amplidão.
Peito que deu alívio a quem sofria
E foi o sol iluminando o dia!

Tua jangada afoita enfune o pano!
Vento feliz conduza a vela ousada!
Que importa que no seu barco seja um nada
Na vastidão do oceano,
Se à proa vão heróis e marinheiros
E vão no peito corações guerreiros?

Se, nós te amamos, em aventuras e mágoas!
Porque esse chão que embebe a água dos rios
Há de florar em meses, nos estios
E bosques, pelas águas!
Selvas e rios, serras e florestas
Brotam no solo em rumorosas festas!
Abra-se ao vento o teu pendão natal
Sobre as revoltas águas dos teus mares!
E desfraldado diga aos céus e aos mares
A vitória imortal!
Que foi de sangue, em guerras leais e francas,
E foi na paz da cor das hóstias brancas!



GOVERNO DO ESTADO DO CEARÁ

Secretaria da Educação