

11.0 Introdução ajQuery

O FrameWork jQuery nada mais é que uma um conjunto de bibliotecas escritas em JavaScript, contendo funções que tratam de efeitos interativos em processos Web. O jQuery veio para facilitar os efeitos do JavaScript, realizando em poucas linhas uma vastas possibilidades de efeitos. Para ter um bom aprendizado, será importante conhecer bem o JavaScript, especialmente modelo de Objeto de Documento ou seja (**DOM**) e **Eventos**. Pois podemos ter interações com todos os elementos da estrutura básica de um site. Nesse curso iremos ver **alguns** comandos simples do jQuery, em seguida, nos focaremos na biblioteca **jQueryUI**, utilizado para abstrair a programação complexa de baixo nível em interações WEB/usuário aumentando a produtividade, funções e beleza do site.



Um exemplo de sua funcionalidade:

Enquanto em Java Script escrevíamos: `document.getElementById("elemento")`

No jQuery só precisamos : `$("#elemento")`

11.1 Instalação e configuração

Para realizar a instalação e configuração, primeiramente devemos acessar o site oficial da jQuery: “<http://code.jquery.com/>” E acessar o link de download. Você deverá copiar o conteúdo de texto e salvar no seu computador como “**jquery.js**”. Como iremos trabalhar na versão: 2.1.4, o link direto que contém o a jquery é : “<http://code.jquery.com/jquery-2.1.4.min.js>”. Assim que o arquivo tiver salvo, adicione-o dentro do seu projeto, de preferência em uma pasta para separá-la dos demais elementos do seu site .

Pronto, falta apenas importar a biblioteca para sua página Web, para isso, basta importar o jQuery na sua

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

página WEB digitando o seguinte comando na estrutura <head>:

Perceba que fizemos esse mesmo procedimento em capítulos anteriores para importação dos nossos projetos em JavaScript. A diferença que copiamos um projeto já feito pelo grupo da jQuery que importamos no nosso site. Podemos também criar o nosso projeto e importá-lo abaixo, unindo a nossa biblioteca com a do jQuery. Veja no exemplo abaixo:

```
<script type="text/javascript" src="js/anima.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
```

Obs.: Se por acaso precisarmos migrar um site para um outro projeto que tiver recursos do jQuery implementados, precisamos exportar a biblioteca junto ou linkar o arquivo externamente alterando o caminho src para um endereço da página web onde encontra-se a biblioteca jQuery, no nosso caso: “code.jquery.com/jquery-2.1.4.min.js”.

11.2 Primeiros Passos

Para inicializar as funções do JavaScript e jQuery precisamos primeiro criar um evento de inicialização do documento DOM. Inicialmente vamos gerar um evento do tipo “ready”, ou seja, quando o documento raiz da nossa página estiver pronto e iniciado veja a imagem abaixo.

```
<script type="text/javascript">
$(document).ready(
  function () {
    alert("Função principal na inicialização da página");
  });
</script>
```

O termo “\$(document) ” indica que estamos acessando o documento DOM como um todo. O termo “ready” indica que quando o DOM estiver pronto, a página irá inicializar um evento. O evento está indicado por “function()”. Dentro dessa “função principal” podemos realizar nossas animações, criar outros eventos, etc.

```
<script>
$(document).ready(
  function(){
    $("a").click(
      function(){
        alert("o botão foi pressionado");
      }
    );
  });
</script>
```

No exemplo acima, geramos uma mensagem ao entrar na página, iremos agora gerar um, ao evento, ao clique de um botão obedecendo o mesmo padrão.

Usando o mesmo raciocínio, ativamos um evento de alerta, quando um botão com referência de link “a” for clicado. Assim podemos manipular qualquer elemento que esteja presente na nossa página WEB.

Estrutura

HTML para script acima:

```
<body>
    Clique no botão: <br />
    <a href=""> Primeiro Evento </a><br />
</body>
```

Exercício Rápido: Como seria ativar um botão ao clicar ou ao selecionar uma div?

Usando o método do exemplo acima, criamos um mesmo evento caso clicássemos em qualquer elemento de link “a”. Porém como poderíamos criar eventos isolados por referência de elemento? Para isso, podemos usar “#id” ou “.class” conteúdo esse visto no nosso curso de HTML/CSS. Iremos agora gerar um evento isoladamente para um botão, veja no exemplo abaixo.

```
<script>
$(document).ready(
    function(){
        $("#evento1").click(
            function(){
                alert("Evento 1 ativo");
            }
        );
        $("#evento2").click(
            function(){
                alert("Evento 2 ativo");
            }
        );
    });
</script>
```

Perceba que agora não iremos mais ativar um evento ao clicar em uma tag “a” qualquer, podemos direcionar um evento para um elemento específico do nosso HTML usando “#id”. Sabemos que um ID só poderá ser usado em um elemento HTML por página, então podemos usar esse parâmetro para gerar eventos em elementos individuais.

Estrutura HTML para script acima:

```
<body>
    Clique no botão: <br />
    <a id="evento1" href=""> Primeiro Evento </a><br />
    <a id="evento2" href=""> Segundo Evento </a>
</body>
</html>
```

Podemos também criar funções separados do HTML, da mesma forma que fizermos no curso de JavaScript, faça um teste, adapte o código acima com funções em um arquivo chamado “anima.js” e chame-o dentro do seu código.

```
<script>
$(document).ready(
    function(){
        $("#content a").click(
            function(){
                alert("Evento acionado");
            }
        );
    });
</script>
</head>
<body>

<div id="content">
    <a href=""> Pronto Para Evento JS</a>
</div>
    <a href=""> Sem evento JS</a>
</body>
```

Se você quiser acessar um elemento que está dentro de um outro elemento identificado por algum “id”, “class” ou “tag”, podemos colocar o caminho completo do elemento que queremos acessar da seguinte forma:

Perceba que um dos links “<a>” está dentro da div “content”. Então geramos um evento para ser acionado apenas no link que está dentro dessa div adicionando o caminho :

`$("#content a").evento`

Veremos no próximo tópico outras formas de acesso a elementos HTML pelo jQuery:

Exercício Rápido: Como seria ativar esse mesmo evento levando em consideração que a camada “#content” estaria dentro de uma outra camada chamada “fundo”?

11.2.1 Métodos de acesso a Elementos HTML

Mostraremos algumas formas de selecionar um elemento dentro da sua estrutura HTML. Conhecer os exemplos abaixo é bastante importante, pois em nossos algoritmos estaremos sempre acessando e alterando valores das nossas páginas WEB. Estaremos usando alguns desses seletores no decorrer do nosso conteúdo. Seletores:

| Seletor | Descrição |
|-------------------------|---|
| <code>\$(this)</code> | Seleciona o elemento referendado por ela mesma. |

| | |
|----------------------------------|--|
| <code>\$('*')</code> | Seleciona todos os elementos do documento HTML |
| <code>\$(' div ')</code> | Seleciona todos os elementos <DIV> poderá ser usado para outros elementos. Ex: “img”, “a”, “table”. |
| <code>\$(' .curso ')</code> | Seleciona todos os elementos cuja a classe seja “curso”. |
| <code>\$(' #camada ')</code> | Seleciona um único elemento de id = “camada”. |
| <code>\$(' #camada img ')</code> | Seleciona todas as imagens que estiver dentro da camada (exemplo). |

Pseudo-Seletores:

| Seletor | Descrição | Exemplo |
|------------------------|--|-------------------------------------|
| <code>:first</code> | Seleciona por um primeiro item | <code>\$(“ul li:first”)</code> |
| <code>:last</code> | Seleciona por um ultimo item | <code>\$(“ul li:last”)</code> |
| <code>:not</code> | Ignora um item selecionado | <code>\$(“ul li:not(:last)”)</code> |
| <code>:contains</code> | Seleciona um item que contenha texto indicado. | <code>\$(“p:contains(fab)”)</code> |
| <code>:empty</code> | Seleciona itens vazios | <code>\$(“p:empty”)</code> |
| <code>:visible</code> | Seleciona itens visiveis | <code>\$(“div:visible”)</code> |

Pseudo-Seletores de formulários:

| Seletor | Descrição | Exemplo |
|------------------------|--|------------------------------|
| <code>:text</code> | Seleciona campos do tipo text | <code>\$(“:text”)</code> |
| <code>:hidden</code> | Seleciona campos invisíveis | <code>\$(“:hidden”)</code> |
| <code>:file</code> | Seleciona arquivos de um formulário | <code>\$(“:file”)</code> |
| <code>:button</code> | Seleciona botões do tipo “button” em um formulário | <code>\$(“:button”)</code> |
| <code>:checkbox</code> | Seleciona um checkbox de formulário | <code>\$(“:checkbox”)</code> |
| <code>:image</code> | Seleciona campo/Imagem de um formulário | <code>\$(“:image”)</code> |
| <code>:password</code> | Seleciona um campo de texto do tipo Senha de um formulário | <code>\$(“:password”)</code> |
| <code>:radio</code> | Seleciona uma caixa de texto do tipo Radio | <code>\$(“:radio”)</code> |
| <code>:submit</code> | Seleciona um botão do tipo submit de um formulário. | <code>\$(“:submit”)</code> |

| | | |
|---|---|---------------------------|
| <code>:reset</code> | Seleciona um botão do tipo reset de um formulário. | <code>\$(“:reset”)</code> |
| Seletores de Formulários: | | |
| Seletor | Descrição | |
| <code>\$('input [name = “bot”]')</code> | Seleciona todos os elementos de formulário com o nome “bot”. | |
| <code>\$('input [name != “curso”]')</code> | Seleciona todos os elementos de formulário que contenham um valor diferente de “curso”. | |

12.0 Biblioteca jQuery

Neste capítulo estudaremos algumas funções da biblioteca jQuery, não estudaremos toda a biblioteca, pois são muitas funções, o objetivo do nosso curso é mostrar através de algumas funções o uso prático dessa ferramenta, se você pretende se especializar nesse framework indicamos esse link: "<http://api.jquery.com/>". Este link contém uma documentação completa de todas as funções do jQuery com exemplos práticos do jQuery para as mais diferentes situações.

12.1 Funções de Estilo

Com a biblioteca jQuery, Podemos alterar dinamicamente os estilos de elementos HTML de forma simples, podendo adicionar ou alterar estilos através dos comandos “css()”, “add()” , “addClass()” , “removeClass()” e “toggleClass()”. Veremos em exemplos suas diferenças, como podemos utilizar esses recursos para cada caso utilizando também recursos aprendidos no módulo anterior de JavaScript.

12.2 Função .css()

Com esse comando podemos adicionar um estilo CSS mesmo que o elemento html não tenha estilo pronto para ela, vejamos como podemos aplicar isso.

```
$(“elemento”).css(“Atributo CSS”, “valor”)
```

```
<script>
    $(document).ready(
        function(){
            $("#bot").click(
                function(){
                    $("#texto").css("color", "red");
                });
        });
    </script>
</head>
<body>
    <p id="texto"> Texto a ser modificado uma cor </p>
    <button id="bot"> Alterar estilo </button>
</body>
```

No exemplo abaixo, temos um botão que aciona um evento responsável por criar um estilo ao texto Html veja:

Este código mostra como alterar a cor de um texto usando o jQuery. Criamos um evento de clique em um botão chamado “bot” que será responsável por alterar a cor do texto, perceba a cor do texto que será

alterada está em um parágrafo que tem como seu id = “texto”. Alteramos o conteúdo do texto para cor vermelha juntamente no na função .css(“color”, “red”); . Podemos alterar qualquer CSS de qualquer elemento usando essa função.

Podemos também guardar em uma variável JS, informações de algum atributo CSS já criado usando:

```
var corAtual = $("elemento").css("Atributo CSS");
```

Exercício Rápido: Como seria alterar a cor de plano de fundo de um campo de texto de formulário ao selecionar o clique?

12.3 Função .add()

Com este comando podemos adicionar estilo a vários elementos distintos com um mesmo comando, podemos também definir os estilos mais genéricos, e específicos entre os elementos envolvidos no relacionamento.

No exemplo abaixo, temos um exemplo de uma tabela editada com recursos de estilo “add()”. Iremos posteriormente analisar o código.

```

<script>
    $(document).ready(
        function(){
            $("th").css("color", "navy")
                .add("#tabelaEdit td")
                .css("background-color", "#DDDDDD");
        });
    </script>
</head>
<body>
    <table id="tabelaEdit" border="1">
        <thead>
            <tr>
                <th>Nome</th>
                <th>e-mail</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Fabricio</td>
                <td>emailFabricio@gmail.com</td>
            </tr>
            <tr>
                <td>Rosal</td>
                <td>emailRosal@gmail.com</td>
            </tr>
        </tbody>
    </table>
</body>

```

Se olharmos o código JS de trás para frente, podemos perceber que incluímos em todas as nossas células “td” que estão dentro da tabela com id = “tabelaEdit” um plano de fundo de cor cinza. Continuando a leitura, as tags “th” recebem um estilo de cor de texto com nome “navy”, porém as tags “th” também irão “herdar” as características de estilos que foram atribuídas à tag “td”. Portanto o plano de fundo da tag “th” também será preenchido como cinza em seu plano de fundo.

Veja como ficou a nossa tabela.

| Nome | e-mail |
|----------|-------------------------|
| Fabricio | emailFabricio@gmail.com |
| Rosal | emailRosal@gmail.com |

Exercício Rápido: Como seria aplicar esse mesmo conceito em uma lista HTML?

13.1.1. Função `.addClass()` e `removeClass()`.

Estas funções são responsáveis por alterar um estilo estruturado de CSS de um determinado elemento html substituindo um estilo por outro. Veja sua Sintaxe::

```
$(“elemento”).addClass(“.classNova”);
```

E para remover o estilo da classe adicionada temos:

```
$(“elemento”).removeClass(“.classNova”);
```

Iremos criar um algoritmo para alterar o estilo de uma DIV utilizando o evento hover, ou seja, quando o mouse estiver selecionado sobre a camada. Para isso iremos inicialmente criar 2 classes de estilo para uma

```

.estilo1{
    width: 200px;
    height: 100px;
    background-color: #999900;
    margin: 10px;
    color: yellow;
}

.estilo2{
    width: 200px;
    height: 100px;
    background-color: blue;
    margin: 10px;
    color: white;
}

```

```

<script>
    $(document).ready(
        function(){
            $(".estilo1").hover(
                function(){
                    $(this).addClass("estilo2");
                },function(){
                    $(this).removeClass("estilo2");
                });
        });
    </script>
</head>
<body>
    <div class="estilo1">
        Texto Teste.
    </div>

```

camada e usar os comandos da seguinte forma:

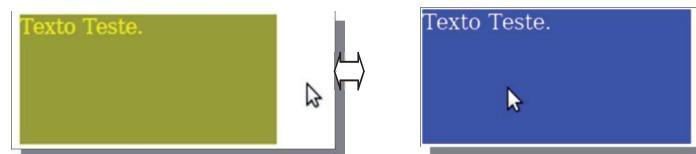
Vamos analisar cada parte do algoritmo:

\$(“.estilo1”).hover()- Com esse comando, iremos chamar um evento ao selecionar o mouser por cima do elemento. No nosso caso na DIV que estiver com estilo “.estilo1” associado.

\$(this).addClass(“estilo2”) - Com esse comando estamos querendo alterar a própria camada que foi selecionada, indicado pelo termo “this”, para o “.estilo2”.

\$(this).removeClass(“estilo2”) - Quando o mouser for retirado da camada, o queremos retirar o estilo que foi incluído anteriormente na camada. Voltando ao seu estado original.

Veja como ficaria essa operação no browser:



Exercício Rápido: Crie 2 camadas, onde cada camada seja responsável por alterar temporariamente o estilo da outra camada.(camada 1 altera estilo da camada 2 e vice-versa)

Podemos alterar não só as Div, mas também qualquer elemento HTML

12.4 Função `toggleClass()`

Com essa função podemos montar de forma mais simples a animação do tópico passado. podemos modificar um estilo de um elemento e quando esse elemento for ativado novamente, a função irá voltar para o seu estado original. Veja sua Sintaxe:

```
$(“elemento”).toggleClass(“.classNova”);
```

Como será a modificação do nosso código.

```
<script>
$(document).ready(
    function(){
        $(".estilo1").hover(
            function(){
                $(this).toggleClass("estilo2");
            });
    });
</script>
```

12.5 Funções de visibilidade

Com a biblioteca jQuery, podemos alterar dinamicamente a visibilidade de elementos HTML de forma simples, podendo mostrar ou ocultar os elementos através dos comandos “show()”, “hide()” , “toggle()”.Estas funções da Biblioteca jQuery são referentes ao modo de visibilidade, sendo possível alterar a visão de qualquer elemento da nossa estrutura HTML dinamicamente. Iremos ver nos próximos tópicos como utilizá-los.

12.6 Funções `show()` e `hide()`

Podemos mostrar um determinado elemento HTML através do comando “show()” ou esconder-los usando o comando “hide()” de acordo com as nossas necessidades. Veremos a sintaxe:

Para ocultar um elemento dinamicamente por um evento:

```
$(“elemento”).hide()
```

Para mostrar um elemento dinamicamente por um evento:

```
$(“elemento”).show()
```

As 2 funções podem receber como parâmetro uma String com 2 tipos de valores.

Fast - Tornará oculto / visível um elemento com o efeito de transição de forma rápida. Slow - Tornará oculto / visível um Elemento com o efeito de transição de forma lenta. Sintaxe com Parâmetros:

```
$(“elemento”).show(“fast”) / $(“elemento”).show(“slow”)
$(“elemento”).hide(“fast”) / $(“elemento”).hide(“slow”)
```

Se quisermos editar o tempo de duração do evento, podemos também atribuir um valor em milisegundos referente ao tempo da animação:

```
$(“elemento”).show(2000);
```

Para esse exemplo iremos mostrar uma camada em um tempo de 2000 milissegundos, ou seja,2 Segundos.

Obs.: existe um estilo CSS chamado “display” Quando ele está atribuído inicialmente como “none”, queremos dizer que inicialmente o elemento que tiver esse atributo, inicialmente estará oculto, sendo assim podemos torná-lo visível através do show() e tornar oculto novamente através do hide();

Vamos analisar um exemplo prático na prima página, mostrando como mostrar ou ocultar uma pequena

```

<script>
$(document).ready(
    function(){
        $("#bot").click(
            function(){
                $("p").hide();
                $("#divFormulario").show();
            }
        );
        $("#voltar").click(
            function(){
                $("p").show();
                $("#divFormulario").hide();
            }
        );
    });
</script>
</head>
<body>
    Deseja Responder nossa enquete?  

</p>
    <button id="bot">Responder</button>

    <div id="divFormulario">
        Quantas horas de estudo você realiza por dia ?
        <form name="form1" action="#" method="GET">
            <input type="radio" name="escHoras" value="hora1" />
            Pelo menos 1 hora<br />
            <input type="radio" name="escHoras" value="hora2" />
            Pelo menos 2 horas<br />
            <input type="radio" name="escHoras" value="hora3" />
            Pelo menos 3 horas<br />
            <input type="radio" name="escHoras" value="horaMais" />
            Mais de 3 horas<br />
            <input type="submit" value="Enviar" name="botao"/>
            <button id="voltar"> Voltar </button>
        </form>
    </div>
</body>

```

enquete que poderá está dentro do seu site:

Exercício Rápido: Crie 2 formulários, onde o segundo só aparecerá quando o usuário terminar de preencher o primeiro formulário.

A camada “idFormulario” estará inicialmente com “display : none;” em seu CSS. Assim ela estará oculta inicialmente. O Botão chamado “#bot” realizará um evento ao clique do botão que será responsável por esconder o paragrafo de texto e mostrar a camada “idFuncionário” que é justamente a camada que contém a enquete. Dentro da camada enquete podemos perceber que temos um botão chamado “voltar” ele fará o inverso, irá ocultar a enquete e tornará visível novamente o texto inicial que está dentro do parágrafo.

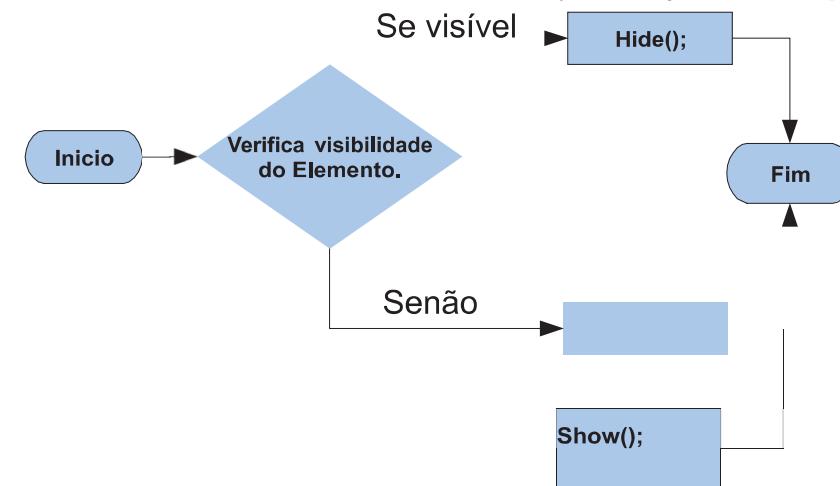
Perceba que não colocamos nenhum parâmetros na função: show() ou hide(). Veja o que acontecerá se colocarmos os parâmetros aprendidos nesse tópico.

12.6.1 Função Toggle()

Esta função, realiza o mesmo resultado das funções show() e hide() estudados no tópico anterior.

Porém ele tem uma pequena implementação de algoritmo que permite analisar se o elemento está visível ou

não, alterando dinamicamente o seu estado de visibilidade. Veja no Fluxograma abaixo seu processo.



Veja sua Sintaxe:

`$(“elemento”).toggle();`

Obs.: Esta função permite os mesmos parâmetros aprendidos nas funções “show()” e “hide()” estudados do tópico anterior.

Vejamos um exemplo abaixo usando o “toggle();” levando em consideração a mesma estrutura HTML do tópico anterior.

```
<script>
    ocult = true;
    $(document).ready(
        function(){
            $("#bot").click(
                function(){
                    $("p").toggle("slow");
                    $("#divFormulario").toggle("slow");
                    if(ocult){
                        document.getElementById('bot').innerHTML = 'Ocultar';
                        ocult = false;
                    }else{
                        document.getElementById('bot').innerHTML = 'Responder';
                        ocult = true;
                    }
                });
        });
    </script>
```

Perceba que com apenas um botão botemos mostrar e ocultar vários conteúdos simultaneamente

12.6.2 Função delay()

Essa função poderá ser combinada com outras funções jQuery, ela irá programar cada evento com um tempo de espera estimado em milissegundos como parâmetro. Iremos mostrar a seguir a sua sintaxe e um exemplo de sua aplicação:

```
$(“elemento”).delay(tempo).outrasFunções();
```

Iremos mostrar 2 eventos que poderá ser disparado através do clique, um deles terá um elemento com delay entre o título e o conteúdo de texto, o outro não terá, vejamos a diferença:

```
<script>
    $(document).ready(
        function(){
            $("#b1").click(
                function(){
                    $('div').add('p').hide();
                    $('#conteudo1').show('slow');
                    $('#conteudo1 p').show('slow');
                });
            $("#b2").click(
                function(){
                    $('div').add('p').hide();
                    $('#conteudo2').show('slow');
                    $('#conteudo2 p').delay(1500).show('slow');
                });
        });
    </script>
</head>
<body>
    <button id="b1">Conteúdo 1</button>
    <button id="b2">Conteúdo 2</button>

    <div id="conteudo1" style="display: none">
        <h2>Título1</h2>
        <p>Texto para conteúdo 1</p>
    </div>

    <div id="conteudo2" style="display: none">
        <h2>Título2</h2>
        <p>Texto para conteúdo 2</p>
    </div>
</body>
```

Perceba que todos os eventos irão inicializar ao acionar o botão “b2”. Porém o conteúdo do parágrafo da camada div “conteudo2” irá esperar 1.5 segundos até começar seu efeito de “fadeIn()”;

12.7 Funções de Opacidade

Com a biblioteca jQuery, podemos controlar dinamicamente a opacidade de elementos HTML de forma simples, podendo tornar completamente opaco ou invisível, também podemos controlar o nível de opacidade de um elemento através dos comandos “fadeIn()”, “fadeOut()”, “fadeTo()”, e “fadeToggle()”. Veremos nos próximos tópicos como essas funções funcionam

12.7.1 Funções FadeIn() e Fadeout()

A função fadeIn() faz tornar visível um elemento HTML, enquanto fadeOut() torna invisível. Veja sua sintaxe abaixo:

Para tornar visível:

```
$(“elemento”).fadeIn();
```

Para tornar invisível;

```
$(“elemento”).fadeOut();
```

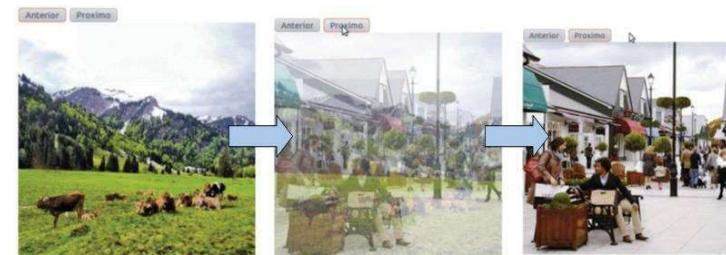
Podemos também controlar a velocidade que esses elementos poderão aparecer ou sumir desta forma:

Obs.: Podemos também controlar o tempo dos efeitos de opacidade colocando um parâmetro “slow” ou “fast” ou atribuir valores de parâmetro em milissegundos ao invés de um texto, da mesma forma que fazemos para os comandos **show()** e **hide()** estudado nos tópicos anteriores.

Veja um exemplo abaixo usando as funções aprendidas:

```
<script>
$(document).ready(
    function(){
        $('input[name="bot1"]').click(
            function(){
                $('img').fadeOut('slow');
                $("#img1").fadeIn("slow");
            });
        $('input[name="bot2"]').click(
            function(){
                $('img').fadeOut('slow');
                $("#img2").fadeIn("slow");
            });
    });
</script>
</head>
<body>
    <input type="button" name="bot1" value="Anterior" />
    <input type="button" name="bot2" value="Proximo" /><br />
    
    
</body>
```

Neste exemplo temos 2 imagens que foram colocadas no nosso site. Existem 2 botões que são responsáveis por fazer um efeito de transição entre as imagens inseridas, perceba que para cada botão,



enquanto uma imagem desaparece (fadeOut) uma outra vai aparecendo (fadeIn). Veja:

12.7.2 FadeTo();

Esta função é responsável por controlar o nível de opacidade de um elemento HTML, veja sua sintaxe.

```
$(“elemento”).fadeTo(velocidade, opacidade);
```

onde:

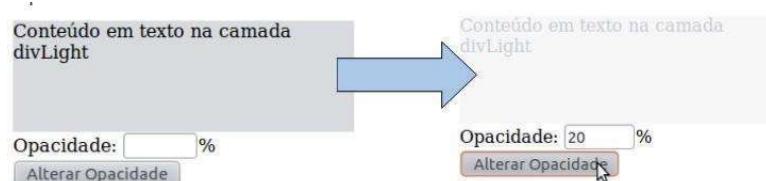
Velocidade = Velocidade do efeito de transição da opacidade antiga com a nova valor de opacidade poderá ser numérico em milissegundos ou uma String “slow” e “fast”.

Opacidade = nível de opacidade do elemento, poderá variar de 0(zero) até 1(100%) onde os valores intermediários seriam o nível da opacidade. Exemplo: 0.5 = (50% de opacidade).

```
<script>
$(document).ready(
    function(){
        $('input[name="bot"]').click(
            function(){
                var valor = $('input[name="opacidade"]').val();
                valor = valor/100;
                $(".divLight").fadeTo("slow",valor);
            });
    });
</script>
</head>
<body>
    <div class="divLight">
        Conteúdo em texto na camada divLight
    </div>
    Opacidade:
    <input type="text" name="opacidade" size="5"/>%<br />
    <input type="button" name="bot" value="Alterar Opacidade" />
</body>
```

Abaixo temos um exemplo que muda a opacidade de uma camada ativado por um botão que verifica o valor de uma caixa de texto e realiza a animação de mudança de opacidade.

Usamos também um novo Método para receber valores em um formulário usando jQuery, através do comando: (“elemento input”).val(). Em JavaScript, usávamos outros métodos, não fará diferença para esse exemplo, nos 2 casos podemos receber valores de campos HTML. Temos a seguir o seguinte resultado:



12.7.3 FadeToggle()

Essa função é responsável por alterar o modo de visibilidade de um elemento HTML voltando ao seu estado inicial quando ativado novamente, ele segue o mesmo raciocínio das funções que contém “Toggle” nos tópicos anteriores, alterando apenas a maneira como é feita a animação. Veja sua sintaxe:

```
$(“elemento”).fadeToggle(velocidade);
```

Velocidade = Velocidade do efeito de transição da opacidade antiga com a novo valor de opacidade poderá ser numérico em milissegundos ou uma String “slow” e “fast”. Esse parâmetro é opcional. Veja um Exemplo:

```
<script>
    $(document).ready(
        function(){
            $("a").hover(
                function(){
                    $("#resposta").fadeToggle();
                });
        });
    </script>
</head>
<body>

<p> Qual a função básica do Jquery <a href="">Ver resposta</a> </p>
<p id="resposta" style="display: none">
    jQuery foi feito para tornar mais simples a navegação<br />
    do documento HTML, a seleção de elementos DOM e <br />
    criar animações </p>
</body>
```

Resultado:

Qual a função básica do Jquery [Ver resposta](#)

Qual a função básica do Jquery [Ver resposta](#)

jQuery foi feito para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM e criar animações



Qual a função básica do Jquery [Ver resposta](#)

jQuery foi feito para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM e criar animações

Exercício rápido: Como seria por uma legenda com uma camada semitransparente usando fade por cima das imagens em efeito?

13.0 Biblioteca jQuery UI

A jQuery UI proporciona abstrações de baixo nível de interação, animação e avançados efeitos especiais de alto nível, construída em cima do jQuery JavaScript Library, que pode ser utilizada para construir aplicações web altamente interativas.

13.1 Instalação e configuração

A instalação e configuração é feita de forma semelhante a que fizemos com jQuery, vamos para o site Oficial do projeto jQuery UI : <http://jqueryui.com/>. Onde iremos baixar um pacote para a instalação. Iremos trabalhar com a versão **jquery-ui-1.11.4.custom**.

O jQuery UI nos disponibiliza um pacote contendo um jQuery e jQuery UI, junto com uma pasta chamada CSS. Essa página também deverá ser importada no seu projeto, pois alguns recursos utilizando de estilos pré-definidos. Acesse o site da jQuery UI e você poderá receber outros pacotes com outros efeitos, por padrão usaremos o estilo “**ui-lightness**” pacote esse que você escolhe no momento do download do pacote. Você poderá acessar o CSS modificando configurações existentes ou criando um próprio estilo para você interagindo com seu jQuery.

O link de importação deverá ser da seguinte forma:

```
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/jquery-ui.min.js"></script>
<script type="text/javascript" src="js/jquery-ui.js"></script>
<link rel="stylesheet" type="text/css" href="css/jquery-ui.min.css">
```

Pronto, agora além de melhorarmos algumas funções existentes do pacote jQuery já estudados, estamos também implementando novas funções da biblioteca jQuery UI. Estudaremos nesse curso algumas funções dessa biblioteca, recomendamos que veja a documentação do site oficial da jQuery UI para estudos complementares.

13.2 Datepicker.

Essa função é responsável por mostrar um calendário para o usuário, as informações do dia selecionado pelo usuário, poderá diretamente ser alterado em uma caixa de texto, oferecendo estilo e poupar o usuário de ter que digitar uma data manualmente.

Sua Sintaxe:

```
$(“elemento”).datepicker();
```

Elemento – Elemento HTML poderá ser responsável por representar o retorno da data escolhida. Veja um exemplo:

```
<html>
  <head>
    <title></title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="js/jquery.js"></script>
    <script type="text/javascript" src="js/jquery-ui.min.js"></script>
    <script type="text/javascript" src="js/jquery-ui.js"></script>
    <link rel="stylesheet" type="text/css" href="css/jquery-ui.min.css">
    <script type="text/javascript">
      $(document).ready(
        function () {
          $("#data").datepicker();
        });
    </script>
  </head>
  <body>
    Faça seu agendamento: <br>
    <input type="text" id="data" size="10">
    <button>Enviar</button>
  </body>
</html>
```



Temos então o resultado a seguir no navegador:

13.3 Accordion

Essa função é responsável por segmentar conteúdos de forma mais elegante, podemos mostrar um conteúdo selecionado ocultando os outros conteúdos que tiverem dentro dessa camada. Sua Sintaxe é descrita da seguinte forma:

```
$(“elemento”).accordion();
```

Onde esse elemento descrito na sintaxe poderá ser uma ID de uma camada. Para organizar o conteúdo interno, devemos estruturar da seguinte forma:

<H3>: Dentro da camada indicada com “**accordion()**” irá representar o título de cada camada de conteúdo.

<div>: as Camadas internas à camada indicada pelo “**accordion()**” irá representar o conteúdo interno que será visível ou não dependendo de sua seleção.

Todos os efeitos já serão gerados automaticamente se organizados os títulos e conteúdo dessa forma.

Vejamos um exemplo a seguir. Resultado no navegador:

Código:



```

<html>
  <head>
    <title></title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="js/jquery.js"></script>
    <script type="text/javascript" src="js/jquery-ui.min.js"></script>
    <script type="text/javascript" src="js/jquery-ui.js"></script>
    <link rel="stylesheet" type="text/css" href="css/jquery-ui.min.css">
    <script type="text/javascript">
      $(document).ready(
        function () {
          $("#conteudo").accordion();
        });
    </script>
  </head>
  <body>
    <div id="conteudo" style="width=500px">
      <h3> Artigo I </h3>
      <div>
        <p> Blockchain talvez seja o recurso mais interessante da criptomoeda, com sua capacidade de registrar transações com segurança e publicamente. Considerando que o mercado de ações é uma confusão de troca de negociações, usar o blockchain para essas transações pode ser uma ideia surpreendentemente boa. </p>
      </div>
      <h3> Artigo II</h3>
      <div>
        <p> O Overstock.com é uma loja online que abraçou o Bitcoin e o blockchain, e agora está levando o amor à criptografia a um novo nível: com aprovação da Comissão de Valores Mobiliários (SEC, na sigla em inglês) dos EUA, o Overstock planeja emitir títulos públicos. É um grande passo para o blockchain como tecnologia, considerando que até hoje se manteve limitado ao Bitcoin. </p>
      </div>
      <h3> Artigo III</h3>
      <div>
        <p> O blockchain é uma plataforma para tornar registros de transações publicamente disponíveis e que não pode ser modificada. Usuários podem escrever transações na cadeia (e elas são verificadas por uma vasta rede de computadores), mas não podem, posteriormente, eliminar essas entradas. Na ausência de um banco central, é isso o que mantém o Bitcoin funcionando. </p>
      </div>
    </div>
  </body>
</html>

```

Além de texto, podemos colocar outros elementos HTML, como Imagens, links, tabelas, listas etc.

Podemos usar esses recursos para uma área de notícias ou artigos por exemplo, pesquise em outros sites novos contextos onde podemos utilizar **accordion** para facilitar o acesso à conteúdos WEB.

13.4 Tabs

Essa função é uma outra forma elegante de organizar o conteúdo de uma página WEB através de abas por link para visualização de conteúdo de forma dinâmicas. Vejamos sua sintaxe e como aplica-las.

\$("elemento").tabs();

Onde esse elemento descrito na sintaxe poderá ser uma ID de uma camada. Para organizar o conteúdo interno, devemos estruturar da seguinte forma:

<a>: Inicialmente criamos uma lista de links para representar as abas de conteúdo. Os índices deverão linkar um ID será responsável por alterar o conteúdo.

<div>: Cada camada deverá ter uma indicação “**ID**” que irá representar um conteúdo.

Todos os efeitos já serão gerados automaticamente se organizados os títulos e conteúdo dessa forma:

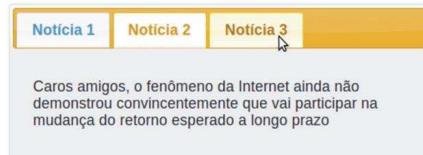
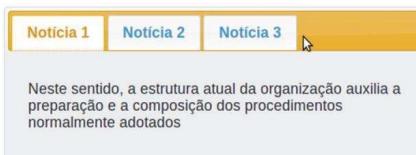
Vejamos um exemplo a seguir:

```

<script>
  $(document).ready(
    function(){
      $("#noticias").tabs();
    });
  </script>
</head>
<body>
  <div id="noticias" style="width: 500px">
    <ul>
      <li><a href="#not-1">Notícia 1</a></li>
      <li><a href="#not-2">Notícia 2</a></li>
      <li><a href="#not-3">Notícia 3</a></li>
    </ul>
    <div id="not-1">
      <p> Neste sentido, a estrutura atual da organização auxilia a preparação e a composição dos procedimentos normalmente adotados </p>
    </div>
    <div id="not-2">
      <p>Caros amigos, o fenômeno da Internet ainda não demonstrou convincentemente que vai participar na mudança do retorno esperado a longo prazo</p>
    </div>
    <div id="not-3">
      <p> Assim mesmo, a determinação clara de objetivos agrupa valor ao estabelecimento de todos os recursos funcionais envolvidos</p>
    </div>
  </div>
</body>

```

Temos então o resultado a seguir no navegador:



13.5 Menu

Esta função é responsável por montar um Menu DropDown de forma mais simples e elegante usando apenas listas e links como referência. Vejamos sua sintaxe:

```
$(“elemento”).menu();
```

Onde esse elemento descrito na sintaxe poderá ser uma ID de uma camada. Para organizar o conteúdo interno, devemos estruturar da seguinte forma:

<a>: monta um novo botão para link no menu. Para criar um menu DropDown basta colocar uma lista dentro de outra lista, como fizemos no link “Opção 2” no link abaixo, veja o exemplo:

```
<script>
  $(document).ready(
    function(){
      $("#menu").menu();
    });
</script>
</head>
<body>
  <ul id="menu" style="width: 100px">
    <li>
      <a href="">Opção 1</a>
    </li>
    <li>
      <a href="">Opção 2</a>
      <ul>
        <li>
          <a href="">Opção 2.1</a>
        </li>
        <li>
          <a href="">Opção 2.2</a>
        </li>
      </ul>
    </li>
    <li>
      <a href="">Opção 3</a>
    </li>
  </ul>
</body>
```



Temos então o resultado a seguir no navegador:

13.6 Tooltip

Essa função é responsável por indicar aos usuários alguma informação relevante para o preenchimento de algum dado do formulário. Esse método já existe usando apenas o HTML, porém o jQuery pode tornar esse recurso mais elegante. Vejamos sua sintaxe:

```
$(“elemento”).tooltip();
```

Você poderá utilizar “document” como parâmetro do “elemento”. Assim todos os campos de um formulário que tiver “title” serão alterados com o novo design baseado no tema escolhido do jQuery.

```
<script>
    $(document).ready(
        function(){
            $(document).tooltip();
        });
    </script>
</head>
<body>
    <form name="formulario" action="#" method="POST">
        <p>Nome:<br/>
            <input id="nome" size="20"/></p>
        <p>CPF :<br/>
            <input id="cpf" title="Digite seu CPF sem pontos e hifen"/></p>
            <input type="submit" value="enviar" name="enviar" />
            <input type="reset" value="limpar" name="limpar" />
    </form>
</body>
```

Temos então o resultado a seguir no navegador:

Nome:

CPF : 

Digite seu CPF sem pontos e hifen