**23/07 – jQuery**

O jQuery é uma biblioteca JavaScript muito popular utilizada para facilitar a interação com o usuário e a manipulação dos elementos do DOM, bem como a conexão com servidores por meio de requisições assíncronas (AJAX). Só para se ter uma ideia, atualmente o jQuery é usado por 74% dos sites na web.

Essa biblioteca foi fundada em 2005 por John Resig, com o objetivo de simplificar tarefas muito comuns no JavaScript, como manipulação de eventos, animações, efeitos visuais, tratamento de requisições assíncronas, etc. Portanto, seus principais objetivos são:

* Fazer trabalhos repetitivos ficarem mais simples
* Facilitar a manipulação de uma página web (seletores CSS)
* Melhorar a interação do usuário (eventos e animações)
* Diminuir o conflito entre navegadores

O jQuery é suportado por vários navegadores e oferece uma abordagem de fácil utilização, prezando por uma aplicação bem interativa. Muitos recursos atuais do JavaScript lembram do jQuery, como por exemplo, os métodos **querySelector() e querySelectorAll()**, baseados em CSS3, ou o **classList** para gerenciamento de classes.

**Download do jQuery**

**1º método: CDN –** CDN significa Content Delivery Network (Rede de Distribuição de Conteúdo). Representa uma rede de servidores distribuída pelo mundo inteiro que possui vários tipos de conteúdo, inclusive o arquivo de texto trazendo o conteúdo do jQuery.

Para usar o jQuery via CDN, basta colar um link na tag <head> do document HTML:

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>

**2º método: Arquivo local** – basta ir ao site do jQuery, baixar o arquivo JavaScript que possui o conteúdo do jQuery e referenciar ele no documento HTML.

**3º método: npm** – tendo o Node instalado, basta usar o comando **npm i jquery**.

**DIFERENÇA ENTRE DOCUMENT READY E WINDOW ONLOAD**

A principal diferença entre essas duas funções é que o document.ready() é executado quando o **documento HTML** é carregado. Já o window.onload() é executado quando **todo o conteúdo** é carregado (imagens, vídeos, etc).

Além disso, podemos executar mais de uma função no document.ready(), o que não ocorre no window.onload()

Quando vamos usar o jQuery, usamos o símbolo de dólar ($) para nos referir a ele, observe:

<div>1</div>

    <div>2</div>

    <script>

        $('div').hide(5000).show(5000).fadeOut(5000)

        $.fn.fundoVerde = function(){

*this*.css('background-color', 'green')

            return *this*

        }

        $('body').fundoVerde().fadeOut(1000)

    </script>

Na primeira linha do código, usamos $(‘div’) para selecionar todas as divs existentes no arquivo HTML. Depois usamos funções encadeadas hide(), show() e fadeOut(), para esconder a div, mostra-la novamente e fazer um efeito de desaparecimento lento, respectivamente.

Já na terceira linha, criamos nossa própria função (ou plugin) para dar fundo verde a um elemento usamos a função css(). Depois disso, aplicamos essa função ao body na linha seguinte.

**Principais funções do jQuery:**

**Selecionando elementos (navegando pela árvore do DOM)**

**parent()**: seleciona o elemento pai de determinado elemento.

**parents():** seleciona os elementos pais, avôs (e assim por diante) de um determinado elemento.

**parentsUntil():** seleciona todos os elementos pais, avôs (e assim por diante) de um determinado elemento, até chegar em um certo elemento.

**children():** seleciona todos os filhos de um elemento,

**each():** recebe uma função call-back, onde para cada elemento referido, essa função é aplicada a ele.

**find():** seleciona um determinado elemento que se encontra dentro de outro.

**next():** seleciona o irmão a frente do elemento.

**prev():** seleciona o irmão anterior do elemento.

**Adicionando elementos**

**append():** anexa determinado elemento dentro de outro.

**prepend():** anexa determinado elemento dentro de outro (no começo).

**appendTo():** anexa determinado elemento dentro de outro.

\* A diferença entre append() e appendTo() é que na sintaxe de append(), antes vem o seletor e depois vem o conteúdo. Já em appendTo(), antes vem o conteúdo a ser anexado, e só depois vem o seletor.

**prependTo():** anexa determinado elemento dentro de outro (no começo).

\* A diferença entre prepend() e prependTo() é que na sintaxe de prepend(), antes vem o seletor e depois vem o conteúdo. Já em prependTo(), antes vem o conteúdo a ser anexado, e só depois vem o seletor.

$('#div1').append(', 2')

$('#div1').prepend('0, ')

$('<span> 3</span>').appendTo($('#div1'))

$('<span>-1, </span>').prependTo($('#div1'))

**before():** insere algum conteúdo ANTES de determinado elemento.

**after():** insere algum conteúdo DEPOIS de determinado elemento,

**insertBefore():** insere algum conteúdo ANTES de determinado elemento.

\* A diferença entre before() e insertBefore() é que em before(), antes vem o elemento e depois vem o conteúdo. Já em insertBefore(), primeiro vem o conteúdo e depois vem o elemento.

**insertAfter():** insere algum conteúdo DEPOIS de determinado elemento,

\* A diferença entre after() e insertAfter() é que, em after(), antes vem o elemento e depois vem o conteúdo. Já em insertAfter(), primeiro vem o conteúdo e depois vem o elemento.

$('#div2').before('<p>Antes do 2</p>')

$('#div2').after('<p>Depois do 2</p>')

$('<p>Antes do 3</p>').insertBefore($('#div3'))

$('<p>Depois do 3</p>').insertAfter($('#div3'))

**wrap():** envolve um elemento para dentro de outro.

**wrapAll():** envolve todos os elementos para dentro de um único elemento.

**wrapInner():** envolve um elemento para dentro de outro, mas apenas em volta de seu conteúdo.

**Removendo elementos**

**remove():** remove um elemento.

**empty():** torna determinado elemento vazio, sem conteúdo.

**detach():** remove um elemento, mas mantém seu evento associado a ele.

**unwrap():** remove o que está em volta de um elemento.

**Substituindo elementos**

**replaceAll():** substitui o conteúdo de um elemento.

**replaceWith():** substitui o conteúdo de um elemento.

A diferença entre **replaceAll()** e **replaceWith()** está na sintaxe. Em replaceAll(), primeiro vem o conteúdo a ser substituído e depois vem o elemento. Já em replaceWith(), primeiro vem o elemento e depois o conteúdo a ser substituído.

$('<div>Substituto</div>').replaceAll($('body > p'))

        $('body > p').replaceWith('<div>Substituto</div>')

**Inserindo texto nos elementos**

**html()**: retorna ou define o conteúdo HTML dentro de um elemento.

**text():** retorna ou define conteúdo textual dentro de um elemento.

**Clonando elementos**

**clone():** clona um elemento HTML. Recebe um parâmetro booleano que determina se os eventos do elemento também serão clonados.

**Animações com jQuery:**

**\* Animações são estilos CSS que ocorrem em um determinado intervalo de tempo.**

**\*\* As animações em geral possuem dois parâmetros, que são opcionais: o primeiro determina a duração da animação (em milissegundos), o segundo é uma função call-back determinando o que vai ocorrer quando a animação terminar.**

**animate():** cria animações de forma personalizada. Recebe três parâmetros: um com a mudança do css (dentro de um objeto), o segundo recebe a duração da animação, e o terceiro recebe uma função de call-back (opcional).

**hide():** esconde o elemento.

**show():** mostra o elemento.

**toggle():** faz alternância entre os métodos hide() e show().

**fadeOut()** e **fadeIn():** cria o efeito de desaparecimento do elemento aos poucos (fadeOut) e o aparecimento do elemento aos poucos (fadeIn).

**fadeTo():** altera a opacidade de um elemento. **(parâmetro obrigatório)**

**fadeToggle():** faz alternância entre os métodos fadeIn() e fadeOut().

**slideUp()** e **slideDown():** encolhe um elemento para cima (slideUp) ou expande ele para baixo (slideDown), usando efeito de deslizar (sliding).

**slideToggle():** faz alternância entre os métodos slideUp() e slideDown().

**jQuery.fx.off = true:** desabilita animações.

**queue():** cria uma fila de animações para que elas sejam realizadas em sequência.

**clearQueue():** limpa a fila de animações.

**dequeue():** executa a próxima animação na fila.

**delay():** atrasa a execução de uma animação no contexto de fila.

**stop():** interrompe uma animação. Recebe um parâmetro booleano se apenas a animação corrente deve parar (false) ou se todas as animações da fila devem parar (true).

**finish():** interrompe uma animação, remove todas as animações da fila e completa todas as animações restantes.

**Manipulando dataset:**

**data():** lê ou altera o valor do atributo data de um elemento.

Caso possua apenas um parâmetro, irá ler o valor do atributo data. Caso possua dois parâmetros, irá alterar o valor do atributo data

**removeData():** remove o atributo data de um elemento.

**Manipulando CSS:**

**addClass()** e **removeClass():** adiciona ou remove uma classe de determinado elemento.

**toggleClass():** realiza alternância de classes. Se já existe, ele remove. Se ainda não tem, ele coloca.

**css():** aplica estilo CSS ao elemento, sendo o primeiro parâmetro a propriedade, e como segundo parâmetro, o valor dessa propriedade.

**Manipulando dimensões:**

**width():** retorna ou define a largura de um elemento, sem levar em conta o padding.

Caso não possua parâmetro, retorna o valor da largura do elemento. Caso possua um parâmetro, define uma nova largura para o elemento.

**innerWidth():** retorna a largura de um elemento, levando em conta o padding.

**height():** retorna ou define a altura de um elemento, sem levar em conta o padding.

Caso não possua parâmetro, retorna o valor da altura do elemento. Caso possua um parâmetro, define uma nova altura para o elemento.

**innerHeight():** retorna a altura de um elemento, levando em conta o padding.

**Manipulando posições:**

**offset():** retorna ou define a posição de um elemento em relação ao documento HTML.

Caso não possua um parâmetro, retorna a posição do elemento. Caso possua um parâmetro, define a posição do elemento com base nele.

**position():** retorna a posição de um elemento em relação ao elemento pai.

**scrollTop():** retorna ou define a posição do scroll vertical.

Caso não possua parâmetro, retorna a posição do scroll vertical. Caso possua um parâmetro, define a nova posição do scroll vertical.

**scrollLeft():** retorna ou define a posição do scroll horizontal.

Caso não possua parâmetro, retorna a posição do scroll horizontal. Caso possua dois parâmetros, define a nova posição do scroll horizontal.

**Manipulando valores:**

**val():** retorna ou define um valor para um elemento (semelhante á propriedade .value).

Caso não possua parâmetro, retorna o valor de um elemento. Caso possua um parâmetro, define um novo valor para o elemento.

**Manipulando propriedades:**

**\***Atributos e propriedades são bem semelhantes, mas tem suas diferenças. Ambos definem informações adicionais para um elemento, a fim de que ele funcione da maneira correta. Mas um atributo está diretamente no arquivo HTML. Já as propriedades estão no DOM, de maneira virtual.

**prop():** retorna ou define o valor de uma propriedade em um elemento.

Caso possua apenas um parâmetro, retorna o valor da propriedade. Caso possua dois parâmetros, define um novo valor para a propriedade.

**removeProp():** remove uma propriedade de um elemento. Válido apenas para propriedades criadas pelo jQuery, e não para as nativas do DOM.

**Filtros do jQuery:**

**$(‘div:first’):** seleciona apenas a primeira div existente.

**$(‘div:last’):** seleciona apenas a última div existente.

**$(‘div:odd’):** seleciona apenas as divs que são pares.

**$(‘div:even’):** seleciona apenas as divs que são ímpares.

**$(‘div:eq(x)’):** seleciona uma div por sua posição específica (x).

**$(‘div:gt(x)’):** seleciona apenas as divs que possuem índice maior do que x.

**$(‘div:lt(x)’):** seleciona apenas as divs que possuem índice menor do que x.

**$(‘div:not(:last)’):** seleciona todas as divs, exceto a última.

**Filtros de conteúdo**

**$(‘div:contains(“x”)’):** seleciona apenas as divs que possuem “x” em seu conteúdo.

**$(‘div:empty’):** seleciona apenas divs vazias, sem conteúdo.

**$(‘div:parent’):** seleciona apenas divs que possuem algum conteúdo, seja ele qual for.

**$(‘div:has(tag)’):** seleciona apenas divs que possuem determinada tag dentro delas.

**Filtros de visibilidade**

**$(‘div:hidden’):** seleciona apenas as divs que estão invisíveis.

**$(‘div:visible’):** seleciona apenas as divs que estão visíveis

**Filtros de formulário:**

**$(‘:button’):** seleciona todas as tags <button> ou inputs do tipo ‘button’ que se encontram dentro da div.

**$(‘:checkbox’):** seleciona todos os inputs do tipo checkbox.

**$(‘:disabled’):** seleciona todos os inputs desabilitados.

**$(‘:submit’):** seleciona o botão ou input do tipo ‘submit’.

**$(‘:focus’):** seleciona o input que está com foco.

**Filtros de herança**

**$('div h1'):** seleciona o elemento h1 que é filho de div.

**$('div > .title'):** seleciona o elemento com a classe title que é filho direto de div.

**$('.paragrafo + p'):** seleciona a tag <p> que é irmã adjacente (mais próxima) do elemento com a classe parágrafo.

**$('.paragrafo ~ p'):** seleciona as tags <p> que são irmãos do elemento com a classe parágrafo.

Lembrando que a div foi usada apenas como exemplo, mas pode ser qualquer elemento.

**Manipulação de atributos:**

**attr():** lê ou altera o valor de um atributo.

caso possua apenas um parâmetro, lê o valor de um atributo. Caso possua dois parâmetros, altera o valor de um atributo.

**removeAttr():** remove o atributo de um elemento.

**Trabalhando com eventos:**

**on(event, fn):** aplica um ou mais eventos a determinado elemento. (em JavaScript, corresponde ao new Event().)

**off(event. fn):** desliga um evento de um elemento.

\* Parâmetro event: nome do evento

\* Parâmetro fn: função aplicada quando o evento é disparado.

**one(event, fn):** aplica um evento que é válido apenas uma vez. Após o evento ser disparado pela primeira vez, ele é descartado.

**trigger(event):** dispara um evento manualmente, sem precisarmos esperar que ele aconteça. (em JavaScript, corresponde ao dispatchEvent().)

**event():** aplica um evento personalizado. Ao invés de event, você pode inserir diretamento o nome do evento (click, mouseover, hover, dblclick, select, resize, mouseleave, etc).

**e.preventDefault():** remove o comportamento padrão do navegador.

**e.stopPropagation():** impede a propagação de um evento para outros elementos.

**Propriedades do evento:**

**e.target:** retorna o elemento no qual o evento foi definido.

**e.currentTarget:** retorna o elemento no qual o evento foi disparado. A diferença entre eles é que o currentTarget retorna informações exatamente sobre o elemento no qual o evento foi disparado, e não sobre o elemento no qual o evento foi definido.

Exemplo: <p onClick=”console.log(‘oi’)”><span>cliquei</span></p>

Ao clicar em cima de “cliquei”, o evento será disparado. O **target** apontará para **p**, enquanto **currentTarget** apontará para **span.**

**e.target.id:** retorna o id desse elemento.

**e.target.className:** retorna o nome da classe desse elemento

**e.type:** retorna o nome do evento.

**e.target.localName:** retorna o nome da tag na qual o elemento foi dispardo.

**e.keyCode:** retorna o código da tecla pressionada.

**e.which:** retorna o código da tecla pressionada (alternativa ao keyCode).

**e.key:** retorna o nome da tecla que foi pressionada (para eventos de teclado).

**e.ctrlKey:** retorna um true ou false se a tecla CTRL foi pressionada.

**e.metaKey:** retorna um true ou false se a tecla do Windows foi pressionada.

**e.pageX:** retorna a posição do mouse em relação ao lado esquerdo da página.

**e.pageY:** retorna a posição do mouse em relação ao topo da página.

**Delegação de eventos:** uma delegação de evento ocorre quando definimos um evento colocamos nele um parâmetro intermediário. Esse parâmetro se refere ao elemento pai do elemento em que vai ser disparado esse evento. Feita essa delegação, o elemento pai fica responsável de escutar esse evento que irá ocorrer no elemento filho.

**Fazendo requisições AJAX**

**$.ajax:** faz requisições AJAX em uma URL. Possui como parâmetro um objeto de configuração, onde podemos definir várias propriedades, como URL, method, success (call-back de successo) e error (call-back de erro).

**$.post:** faz requisições AJAX em uma URL utilizando o método POST. Possui três parâmetros: a URL em que será feita a requisição, os dados que serão enviados e uma função de retorno para caso dê certo ou errado.

**Plugins**

Os plugins são funções personalizadas que adicionamos ao jQuery. Esses pedaços de código estendem as funcionalidades da biblioteca. Os plugins possuem uma alta versatilidade, uma vez que podemos tornar uma função nossa que seja útil como plugin no jQuery, e assim poder usá-la em toda a nossa aplicação.