Sistemas Distribuídos e Tolerância a Falhas

Jorge Costa | m4003 - Bruno Santos | e7237

O que é?

1. **jQuery é um novo tipo de biblioteca JavaScript;**
2. jQuery é uma biblioteca JavaScript rápida e concisa que simplifica a manipulação de HTML, CSS, eventos, animações e interacções Ajax para um rápido desenvolvimento;
3. jQuery é feito para mudar a maneira de como escrever JavaScript.

Today's World Wide Web is a dynamic environment, and its users set a high bar for both style and function of sites. To build interesting, interactive sites, developers are turning to JavaScript libraries such as jQuery to automate common tasks and simplify complicated ones. One reason the jQuery library is a popular choice is its ability to assist in a wide range of tasks.

It can seem challenging to know where to begin because jQuery performs so many different functions. Yet, there is a coherence and symmetry to the design of the library; most of its concepts are borrowed from the structure of **HTML and Cascading Style Sheets (CSS). The library's design lends itself to a quick start for designers with little programming experience since many web developers have more experience with these technologies than they do with JavaScript. In fact, in this opening chapter we'll write a functioning jQuery program in just three lines of code.**

O que faz?

1. Acede a elementos HTML
2. Modifica a aparência da pagina Web
3. Altera o conteúdo de um documento
4. Interacção com o utilizador
5. Animações
6. Processamento de informações proveniente de um servidor, sem ser necessário fazer refresh da pagina (AJAX)
7. Simplificação de tarefas mais rotineiras de JavaScript

What jQuery does
The jQuery library provides a general-purpose abstraction layer for common web scripting, and is therefore useful in almost every scripting situation. Its extensible nature means that we could never cover all possible uses and functions in a single book, as plugins are constantly being developed to add new abilities. The core features, though, address the following needs:
Access elements in a document. Without a JavaScript library, many lines of code must be written to traverse the Document Object Model (DOM) tree, and locate specific portions of an HTML document's structure. A robust and efficient selector mechanism is offered in jQuery for retrieving the exact piece of the document that is to be inspected or manipulated.
• Modify the appearance of a web page. CSS offers a powerful method of influencing the way a document is rendered, but it falls short when web browsers do not all support the same standards. With jQuery, developers can bridge this gap, relying on the same standards support across all browsers. In addition, jQuery can change the classes or individual style properties applied to a portion of the document even after the page has been rendered.
Alter the content of a document. Not limited to mere cosmetic changes, jQuery can modify the content of a document itself with a few keystrokes. Text can be changed, images can be inserted or swapped, lists can be reordered, or the entire structure of the HTML can be rewritten and extended—all with a single easy-to-use Application Programming Interface (API).
Respond to a user's interaction. Even the most elaborate and powerful behaviors are not useful if we can't control when they take place. The jQuery library offers an elegant way to intercept a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers. At the same time, its event-handling API removes browser inconsistencies that often plague web developers.
Animate changes being made to a document. To effectively implement such interactive behaviors, a designer must also provide visual feedback to the user. The jQuery library facilitates this by providing an array of effects such as fades and wipes, as well as a toolkit for crafting new ones.
Retrieve information from a server without refreshing a page. This code pattern has become known as Asynchronous JavaScript And XML (AJAX), and assists web developers in crafting a responsive, feature-rich site. The jQuery library removes the browser-specific complexity from this process, allowing developers to focus on the server-end functionality.
Simplify common JavaScript tasks. In addition to all of the document-specific features of jQuery, the library provides enhancements to basic JavaScript constructs such as iteration and array manipulation.

Porque o jQuery funciona bem?

1. É compatível com os diferentes browser's
2. Conhecimento da sintaxe do CSS
3. Interacção implícita
4. Várias acções numa única linha

With the recent resurgence of interest in dynamic HTML comes a proliferation of JavaScript frameworks. Some are specialized, focusing on just one or two of the above tasks. Others attempt to catalog every possible behavior and animation, and serve these all up pre-packaged. To maintain the wide range of features outlined above while remaining compact, jQuery employs several strategies:

Leverage knowledge of CSS. By basing the mechanism for locating page elements on CSS selectors, jQuery inherits a terse yet legible way of expressing a document's structure. The jQuery library becomes an entry point for designers who want to add behaviors to their pages because a prerequisite for doing professional web development is knowledge of CSS syntax.

Support extensions. In order to avoid "feature creep", jQuery relegates special-case uses to plugins. The method for creating new plugins is simple and well-documented, which has spurred the development of a wide variety of inventive and useful modules. Even most of the features in the basic jQuery download are internally realized through the plugin architecture, and can be removed if desired, yielding an even smaller library.

Abstract away browser quirks. An unfortunate reality of web development is that each browser has its own set of deviations from published standards. A significant portion of any web application can be relegated to handling features differently on each platform. While the ever-evolving browser landscape makes a perfectly browser-neutral code base impossible for some advanced features, jQuery adds an abstraction layer that normalizes the common tasks, reducing the size of code, and tremendously simplifying it.

Always work with sets. When we instruct jQuery, *Find all elements with the class collapsible and hide them, there is no need to loop through each returned element. Instead, methods such as .hide() are designed to automatically work on sets of objects instead of individual ones. This technique, called implicit iteration, means that many looping constructs become unnecessary, shortening code considerably.*

Allow multiple actions in one line. To avoid overuse of temporary variables or wasteful repetition, jQuery employs a programming pattern called chaining for the majority of its methods. This means that the result of most operations on an object is the object itself, ready for the next action to be applied to it.

Download

O site oficial do jQuery é **http://jquery.com/**

- **Não necessita de instalação;**
- **Não necessita de ser compilado;**
- **Apenas referenciar na nossa página Web;**

É possível fazer o download do documento jQuery .js que contem a biblioteca completa do jquery

Também é possível aceder a esta biblioteca remotamente

Muitas funções jQUERY como o .dialog() necessitam de uma interface já gerada, por isso o jQUERY necessita de uma biblioteca .CSS que pode ser descarregada em http://jquery.com/

Como referenciar a biblioteca?

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

Operação fundamental

- Uma das operações fundamentais no jQuery é seleccionar elementos ou partes do documento especificas, utilizando o construtor **$(), por exemplo:**

1. $(“.class”)
2. $(“#id”)
3. $(elemento)

Função de jquery .addclass(“highlight”);

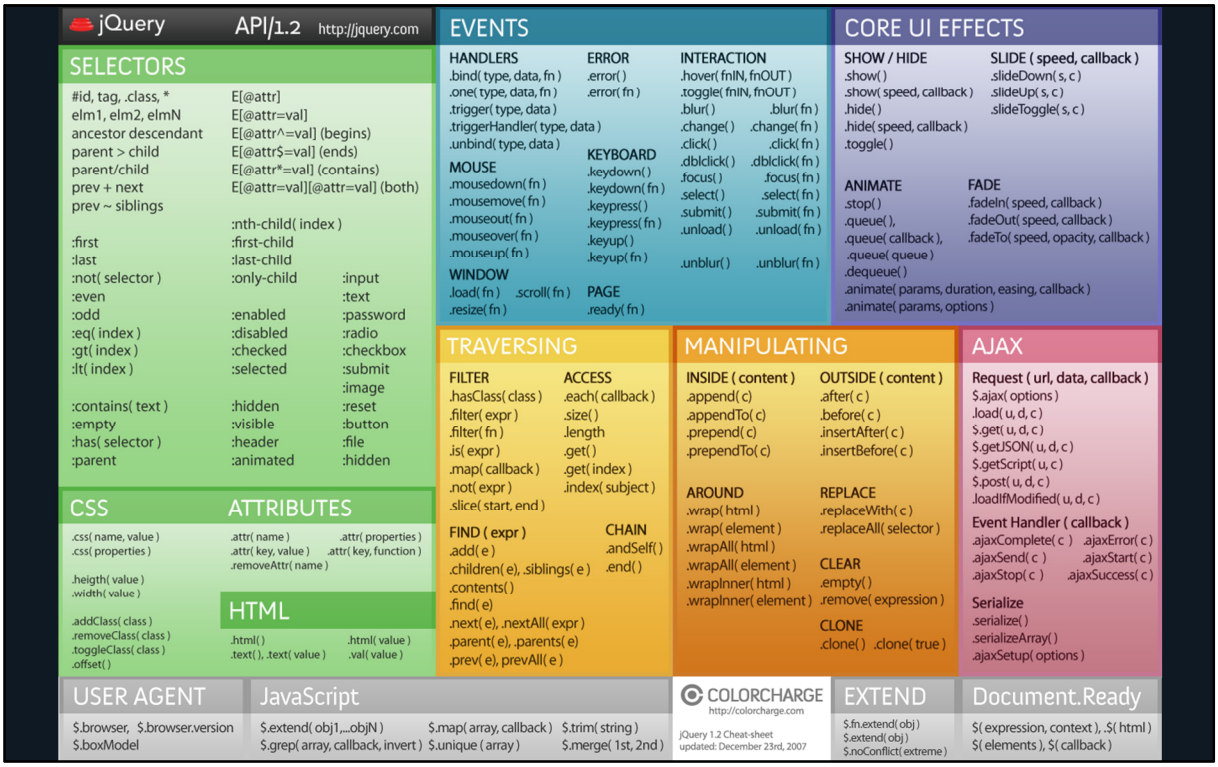
Uma das operações fundamentais no jQuery é seleccionar elementos ou partes do documentos especificas:

$(“.class”) ou $(“#id”) seguindo a função a executar .function();

$(“#div_text”).animate();

Primeiro exemplo

```
$(document).ready(function() {
    $('.poem-stanza').addClass('highlight');
});
```


jQuery API/1.2 http://jquery.com
SELECTORS
#id, tag, .class, *
elm1, elm2, elmN
ancestor descendant
parent > child
parent/child
prev + next
prev ~ siblings
:first
:last
:not(selector)
:even
:odd
:eq(index)
:gt(index)
:lt(index)
:contains(text)
:empty
:has(selector)
:parent
E[@attr]
E[@attr=val]
E[@attr^=val] (begins)
E[@attr$=val] (ends)
E[@attr*=val] (contains)
E[@attr=val][@attr=val] (both)
:nth-child(index)
:first-child
:last-child
:only-child
:enabled
:disabled
:checked
:selected
:hidden
:visible
:header
:animated
:input
:text
:password
:radio
:checkbox
:submit
:image
:reset
:button
:file
:hidden
CSS
.css(name, value)
.css(properties)
.heigth(value)
.width(value)
.addClass(class)
.removeClass(class)
.toggleClass(class)
.offset( )
ATTRIBUTES
.attr(name)
.attr(properties)
.attr(key, value)
.attr(key, function)
.removeAttr(name)
HTML
.html( )
.text( ), .text(value)
.html(value)
.val(value)
EVENTS
HANDLERS
.bind(type, data, fn)
.one(type, data, fn)
.trigger(type, data)
.triggerHandler(type, data)
.unbind(type, data)
MOUSE
.mousedown(fn)
.mousemove(fn)
.mouseout(fn)
.mouseover(fn)
.mouseup(fn)
WINDOW
.load(fn)
.scroll(fn)
.resize(fn)
ERROR
.error( )
.error(fn)
KEYBOARD
.keydown( )
.keydown(fn)
.keypress( )
.keypress(fn)
.keyup( )
.keyup(fn)
PAGE
.ready(fn)
INTERACTION
.hover(fnIN, fnOUT)
.toggle(fnIN, fnOUT)
.blur( )
.blur(fn)
.change( )
.change(fn)
.click( )
.click(fn)
.dblclick( )
.dblclick(fn)
.focus( )
.focus(fn)
.select( )
.select(fn)
.submit( )
.submit(fn)
.unload( )
.unload(fn)
.unblur( )
.unblur(fn)
CORE UI EFFECTS
SHOW / HIDE
.show( )
.show(speed, callback)
.hide( )
.hide(speed, callback)
.toggle( )
SLIDE (speed, callback)
.slideDown(s, c)
.slideUp(s, c)
.slideToggle(s, c)
ANIMATE
.stop( )
.queue( ),
.queue(callback),
.queue(queue)
.dequeue( )
.animate(params, duration, easing, callback)
.animate(params, options)
FADE
.fadeIn(speed, callback)
.fadeOut(speed, callback)
.fadeTo(speed, opacity, callback)
TRAVERSING
FILTER
.hasClass(class)
.filter(expr)
.filter(fn)
.is(expr)
.map(callback)
.not(expr)
.slice(start, end)
FIND (expr)
.add(e)
.children(e), .siblings(e)
.contents( )
.find(e)
.next(e), .nextAll(expr)
.parent(e), .parents(e)
.prev(e), prevAll(e)
ACCESS
.each(callback)
.size( )
.length
.get( )
.get(index)
.index(subject)
CHAIN
.andSelf( )
.end( )
MANIPULATING
INSIDE (content)
.append(c)
.appendTo(c)
.prepend(c)
.prependTo(c)
AROUND
.wrap(html)
.wrap(element)
.wrapAll(html)
.wrapAll(element)
.wrapInner(html)
.wrapInner(element)
OUTSIDE (content)
.after(c)
.before(c)
.insertAfter(c)
.insertBefore(c)
REPLACE
.replaceWith(c)
.replaceAll(selector)
CLEAR
.empty( )
.remove(expression)
CLONE
.clone( ) .clone(true)
AJAX
Request (url, data, callback)
$.ajax(options)
.load(u, d, c)
$.get(u, d, c)
$.getJSON(u, d, c)
$.getScript(u, c)
$.post(u, d, c)
.loadIfModified(u, d, c)
Event Handler (callback)
.ajaxComplete(c) .ajaxError(c)
.ajaxSend(c) .ajaxStart(c)
.ajaxStop(c) .ajaxSuccess(c)
Serialize
.serialize( )
.serializeArray( )
.ajaxSetup(options)
USER AGENT
$.browser, $.browser.version
$.boxModel
JavaScript
$.extend(obj1,...objN)
$.grep(array, callback, invert)
$.map(array, callback)
$.unique (array)
$.trim(string)
$.merge(1st, 2nd)
COLORCHARGE
http://colorcharge.com
jQuery 1.2 Cheat-sheet
updated: December 23rd, 2007
EXTEND
$.fn.extend(obj)
$.extend(obj)
$.noConflict(extreme)
Document.Ready
$(expression, context), .$(html)
$(elements), $(callback)

Referencias

- Learning jQuery 1.3, 2009, Jonathan Chaffer e Karl Swedberg