

Marakana jQuery Training

Copyright © 2011 Marakana, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of Marakana Inc.

We took every precaution in preparation of this material. However, we assume no responsibility for errors or omissions, or for damages that may result from the use of information, including software code, contained herein.

JavaScript is trademark of Oracle. All other names are used for identification purposes only and are trademarks of their respective owners.

Marakana offers a whole range of training courses, both on public and private. For list of upcoming courses, visit <http://marakana.com>

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Introduction To JQuery	1
1.1	Useful Learning Resources	1
1.2	History Of JQuery	1
1.3	What is JQuery?	2
1.4	JQuery vs Custom JavaScript	3
1.5	JQuery vs Other JavaScript Libraries	4
1.6	Where Do I Start?	4
1.7	Download the Library	4
1.8	Using Google Content Deliver Network (CDN)	5
1.9	The jQuery Function Object	6
1.10	Is The Document Ready?	6
1.11	Where to Run Your Code?	7
1.12	Execute Some JQuery Code	7
1.13	Introduction To JQuery: Exercises	8
1.13.1	Questions	8
1.13.2	Labs	8
2	Selection and DOM Traversal	9
2.1	Selection	9
2.2	Basic Selectors	9
2.3	Hierarchy Selectors	10
2.4	Selection By Attribute	10
2.5	Form Selectors	11
2.6	Position Filters	12
2.7	Other Filters	13
2.8	jQuery Method Chaining	13
2.9	DOM Traversal	14
2.10	Filter Methods	14
2.11	Advanced Method Chaining	15
2.12	Selection and DOM Traversal: Exercises	15

3	DOM And Attributes Manipulation	16
3.1	DOM Manipulation	16
3.2	Creating Elements	16
3.3	Inserting Elements	17
3.4	Inserting an Element Before or After Another Element	17
3.4.1	Before an Existing Element	17
3.4.2	After an Existing Element	18
3.5	Inserting an Element as the First or Last Child of a Parent	18
3.5.1	As the First Child	18
3.5.2	As the Last Child	18
3.6	Mass Insertion	18
3.7	Moving Elements	19
3.8	Cloning (Copying) Elements	19
3.9	Removing Elements	19
3.10	Replacing Elements	20
3.11	Element Content: HTML vs Text	20
3.12	Element Attributes And DOM properties	20
3.13	DOM And Attributes Manipulation: Labs	21
4	CSS Styling and JQuery	22
4.1	Reading And Modifying CSS Properties	22
4.2	Reading And Modifying CSS Properties	22
4.3	Removing CSS Properties	23
4.4	CSS Classes	23
4.5	Element Dimensions	24
4.6	Element Position	25
4.7	CSS Styling and JQuery: Labs	25
5	Events	26
5.1	Events Overview	26
5.2	Binding An Event Handler	26
5.3	Binding Shortcut Methods	27
5.4	Unbinding Handlers and “One-Shot” Handlers	27
5.5	The Event Object	27
5.6	Event Delegation	28
5.7	Event Delegation And JQuery	29
5.8	<code>.live()</code> vs <code>.delegate()</code> vs <code>.on()</code>	29
5.9	Events: Labs	30

6	Writing Your Own Plugins	31
6.1	Creating Your Own Plugin: Why?	31
6.2	Creating Your Own Plugin: How?	31
6.3	Adding configurability	32
6.4	Creating Your Own Plugin: Best Practices	32
6.5	Plugins: Labs	33
7	Effects and Custom Animations	34
7.1	JQuery Built-in Effects	34
7.2	Showing or Hiding Elements	34
7.3	Fading Effects	35
7.4	Sliding Effects	35
7.5	Creating Your Own Animations	35
7.6	Animation Notes	36
7.7	Animation Queues	36
7.8	Stopping Animations	37
7.9	Additional Animation controls	37
7.10	Effects and Custom Animations: Labs	37
8	Ajax	39
8.1	What Is Ajax?	39
8.2	Limitations Of Ajax	40
8.3	Load HTML Content: The <code>.load()</code> Method	40
8.4	Hijax With JQuery	41
8.5	Select In Detail What To Load	41
8.6	Advanced Loading Using The <code>.load()</code> Method	42
8.7	JSON and JSONP	42
8.8	Load JSON Data: The <code>.getJSON()</code> Method	43
8.9	The Main Ajax Method: <code>.ajax()</code>	43
8.10	Ajax Global Settings	44
8.11	Error Handling	44
8.12	Ajax: Labs	45
9	Plugins	46
9.1	Popular Plugins	46
9.2	Easing Plugin	46
9.3	Slideshow Plugins	47
9.4	The ColorBox Plugin	48
9.5	The Cycle Plugin	49
9.6	Jcrop Plugin	51
9.7	The Validation Plugin	53
9.8	DataTable Plugin	54
9.9	Plugins: Labs	54

10 jQuery UI	55
10.1 What Is JQuery UI?	55
10.2 Getting Started	55
10.3 JQuery Widgets	56
10.4 The Accordion Widget	56
10.5 The Tabs Widget	57
10.6 Dialogs	58
10.7 JQuery UI Controls	60
10.8 The Datepicker	60
10.9 The Slider	61
10.10The Progress Bar	62
10.11Jquery UI Interactions	62
10.12Drag & Drop	63
10.13Drag & Drop: Example	63
10.14Sortable elements	64
10.15Selectable Elements	65
10.16Resizable Elements	65
10.17jQuery UI animation	66
10.18jQuery UI: Labs	66
11 Advanced Concepts	68
11.1 Best Practices: Loops	68
11.2 Best Practices: Avoid Anonymous Functions	68
11.3 Best Practices: Optimizing Selectors	69
11.4 JQuery Utility Methods	69
11.5 Avoiding Conflict With Other JavaScript Libraries	70
11.6 Queuing And Dequeuing Animations	71

Chapter 1

Introduction To JQuery

In this chapter, we are going to discuss why learning about jQuery is a good investment, what are its strength, competitive advantages to other JavaScript libraries out there and what are the advantages of using jQuery rather than pure JavaScript

We are also going to discuss about how to start with jQuery (discussing about where to get the library, how to be "ready" to use it in your web pages and where to look for useful complementary learning resources)

1.1 Useful Learning Resources

- Along with this class and courseware, you will find a lot of great learning resources out there:
 - The jQuery documentation: <http://api.jquery.com>
 - The jQuery forum: <http://forum.jquery.com/>
 - The jQuery mailing list archives: <http://docs.jquery.com/Discussion#Archives>
 - The jQuery IRC channel: http://docs.jquery.com/Discussion#Chat_%2F_IRC_Channel
 - Stackoverflow.com: <http://stackoverflow.com/questions/tagged/jquery> (John Resig hangs out there sometimes)

1.2 History Of JQuery

Aug. 22, 2005

Everything started here where John Resig, creator of jQuery, showed us a few ideas of what would become jQuery(<http://ejohn.org/blog/selectors-in-javascript/>)

Jan. 14, 2006

jQuery announced

Jan. 24, 2006

Creation of the jQuery Blog

Jan. 27, 2006

Creation of the jQuery mailing list

Aug. 26, 2006

First stable version of jQuery, **v1.0** released

Jan. 14, 2007

Anniversary of jQuery. **v1.1** released: significant performance improvements; reduced, simplified API

Sep. 10, 2007

v1.2 released: additional DOM traversal and manipulation methods; improved animation control; JSONP support; XPath selectors removed

Sep. 17, 2007

First version of jQuery UI released: fully themed interaction and widget library built on top of jQuery

Jan. 14, 2009

Third anniversary of jQuery. **v1.3** released: CSS selector engine, Sizzle, available as a standalone component; "live event" binding; improved cross-browser event abstraction; significant performance improvements

Mar. 6, 2009

jQuery UI 1.7 released: ThemeRoller theme generation; new project hosting domain

Jan. 14, 2010

Fourth anniversary of jQuery. **v1.4** released: more performance improvements; more DOM traversal and manipulation methods; more animation control; more moreness

Jan. 21, 2010

[jQuery.org](http://jquery.org) goes live (site containing resources for jQuery and related projects)

Mar. 23, 2010

jQuery UI 1.8 released: new utilities; new widgets; upgraded widget factory; more modular core

Jan. 31, 2011

v1.5 released: deferred objects; improved, extensible Ajax support

May. 3, 2011

v1.6 released: major rewriting of the attribute module; performance improvements

Nov. 3, 2011

1.7 released: `.on` and `.off` become preferred event binding mechanisms

Aug. 9, 2011

1.8 released: smaller and faster code size - Sizzle selector engine rewritten, animation support improved and cleaned up, css prefixing supported automatically.

2013?

1.9 and 2.0 released. 1.9 will be the last version that supports IE 6/7/8

Note

At the time of this writing, the latest version of jQuery is v1.6.4 which has been released in September 12th, 2011.

1.3 What is JQuery?

- jQuery is a JavaScript library that simplifies:
 - HTML element selection and document object model (DOM) traversal
 - Element creation, deletion, and modification
 - Event handling
 - Animation
 - Ajax interactions
 - Custom widget integration (date picker, slider, dialogs, tabs, etc. . .) with jQuery UI

- jQuery is:
 - Free!
 - Open-source and available under both MIT and GPL licences
 - Only 31Kb (minified and gzipped, ready for production) ⇒ lightweight footprint
 - Cross-browser compatible
 - Extensible! You can write your own plugins or pick the one that interests you among a large list of existing ones.

1.4 JQuery vs Custom JavaScript

- jQuery is written entirely in JavaScript, so you could replicate any of its functionalities yourself directly in JavaScript.
 - Why reinvent the wheel?
 - jQuery is tested and used by thousands of web sites.
 - jQuery is optimized for performance by JavaScript experts.
- jQuery is also designed to circumvent cross-browser compatibility problems, such as:
 - Events not fired in some browsers: <http://quirksmode.org/dom/events/index.html>
 - Adding event listeners (`addEventListener` standard way vs `attachEvent`)
 - Getting the cursor position (`e.pageX` and `e.pageY` vs `e.clientX` and `e.clientY`)
 - Changing the CSS float (`element.style.styleFloat` vs `element.style.cssFloat`)
 - So much to remember...
- The jQuery team works to provide a consistent interface for maximal functionality across different browsers.
- jQuery supports the following browsers:
 - Firefox 2.0+
 - Internet Explorer 6+
 - Safari 3+
 - Opera 10.6+
 - Chrome 8+

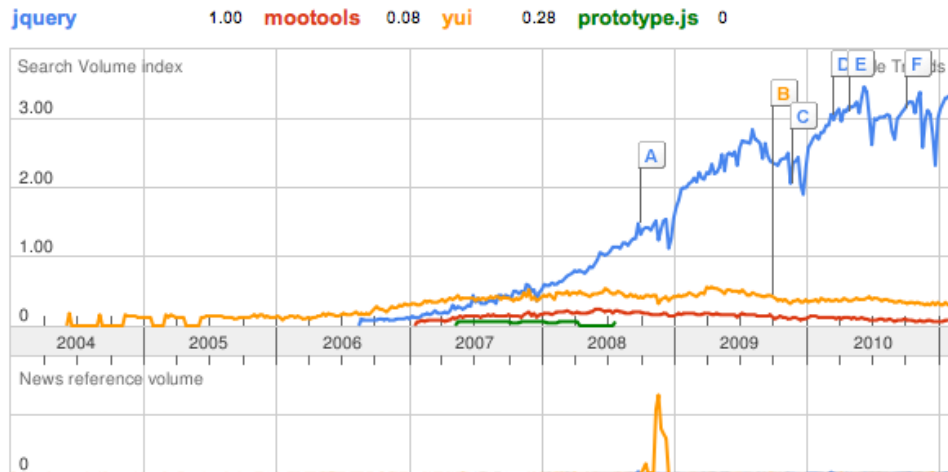
Note

You can see a list of known jQuery browser issues at URL: http://docs.jquery.com/Known_Issues

John Resig, creator of jQuery, gave an interesting talk at Yahoo addressing browser inconsistencies: "The DOM is a mess" (<http://ejohn.org/blog/the-dom-is-a-mess/>)

1.5 JQuery vs Other JavaScript Libraries

- With so many existing JavaScript libraries out there, why should you consider using jQuery?
- Some simple research on Google Trends comparing jQuery to other JavaScript libraries shows the following:



- The job market factor is also important: There are more job offers in JavaScript related to jQuery.
- jQuery is used by big companies such as: Google, Microsoft, Nokia, Dell, Bank of America, Major League Baseball, Digg, NBC, CBS, Netflix, Technorati, Mozilla.org, Wordpress, Drupal and so on. . .

Note

Microsoft now includes jQuery with its Visual Studio and ASP.NET MVC framework.

- Large community and developer pool ⇒ support is huge!
 - Blog posts
 - Tutorials

1.6 Where Do I Start?

- You can start with jQuery in three easy steps:
 - Access the jQuery library and include it in your page.
 - Make sure the page is ready.
 - Execute some jQuery code!

1.7 Download the Library

- From the <http://jquery.com/> web site, you can download two versions of the code:

Production

Minified JavaScript (illegible) for use in production, lightweight footprint (29Kb) ⇒ Saves bandwidth for you and speeds up page requests for your users.

Development

Uncompressed JavaScript for use in development ⇒ Very useful if you need to look at how jQuery is implemented.

- To include the jQuery library in your page:

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript" src="jquery-1.6.4-min.js"></script> ❶
  <script type="text/javascript" src="myscript.js"></script> ❷
</head>
```

- ❶ This is your local copy of the jQuery library
- ❷ This is where you would write your own code using the jQuery library

Warning

Some browsers do not handle "self-closing" `<script>` tags correctly. For example, some browsers fail to load the script if you use the following syntax:



```
<script type="text/javascript" src="jquery-1.6.4-min.js" />
```

To ensure proper loading of your script, always use an explicit close tag, like this:

```
<script type="text/javascript" src="jquery-1.6.4-min.js"></script>
```

1.8 Using Google Content Deliver Network (CDN)

- As an alternative to serving the jQuery library from your own server, you can choose to use the hosted jQuery library from Google Content Delivery Network (CDN).
 - A single address to access - chances are that users already have this library in their cache if they have visited a site using the jQuery library (hosted by Google CDN) ⇒ Your page loads faster and you spend less on the bandwidth

```
<head>
  <title>My jQuery page</title>
  <script type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script> ❶ ←
  <script type="text/javascript"
    src="myscript.js"></script> ❷
</head>
```

Tip

When loading jQuery from CDN, your `src` attribute can omit the patch level in the URL, in which case you automatically load the most recent version of that minor release of jQuery. For example:

```
<script type="text/javascript"
  src="https://ajax.googleapis.com/ajax/libs/jquery/1.6/jquery.min.js"></script>
```

1.9 The jQuery Function Object

- Loading the jQuery library creates a single *function object* named `jQuery`.
 - All jQuery functionality is provided through the methods and properties of the `jQuery` function object. For example:

```
jQuery('li').css('background-color', 'yellow');
```

- As a convenience, jQuery also provides an *alias* for the `jQuery` function object named `$`.
 - Anywhere you might use `jQuery`, you can use `$`. For example:

```
$('.li').css('background-color', 'yellow');
```

Many other JavaScript libraries, such as Prototype, also use `$` as a function or variable name. If you need to use one of the libraries in conjunction with jQuery, you can have jQuery relinquish the `$` alias by executing:

```
jQuery.noConflict();
```

1.10 Is The Document Ready?

- Before jQuery can manipulate the elements in your document, they must be created.
 - By placing the jQuery code in the `<script>` element inside the `<head>` of your document, the jQuery code gets a chance to execute before the rest of the document is loaded and rendered by the browser.
- A common pure JavaScript approach is to register an `onLoad` handler on the `<body>` element to execute your JavaScript code, which triggers when the page has loaded and rendered all objects on the page.
 - A disadvantage of the `onLoad` event is that it isn't triggered until all elements have *loaded*. For example, it isn't triggered until all `` data has been retrieved from the server.
 - This keeps us from accessing elements that haven't been created yet, but it also blocks us — and the user — from manipulating the page until all data has been retrieved from the server.
- jQuery provides a mechanism that allows JavaScript code to begin execution faster.
 - The `ready` event occurs when the page DOM is complete, but some elements — such as `` elements — might still be awaiting data from the server to render.
 - You can register a *callback* function (also known as a *handler*) to execute when the document DOM is ready:

```
$(document).ready(function() {  
    //Perform here all the jQuery magic  
});
```

jQuery also provides a shortcut, in which you provide your callback function as the sole argument to `$`:

```
$(function() { /* jQuery code */ });
```

Note

Observe that we are actually providing a reference to an *anonymous function* as the argument to the `ready()` method.

1.11 Where to Run Your Code?

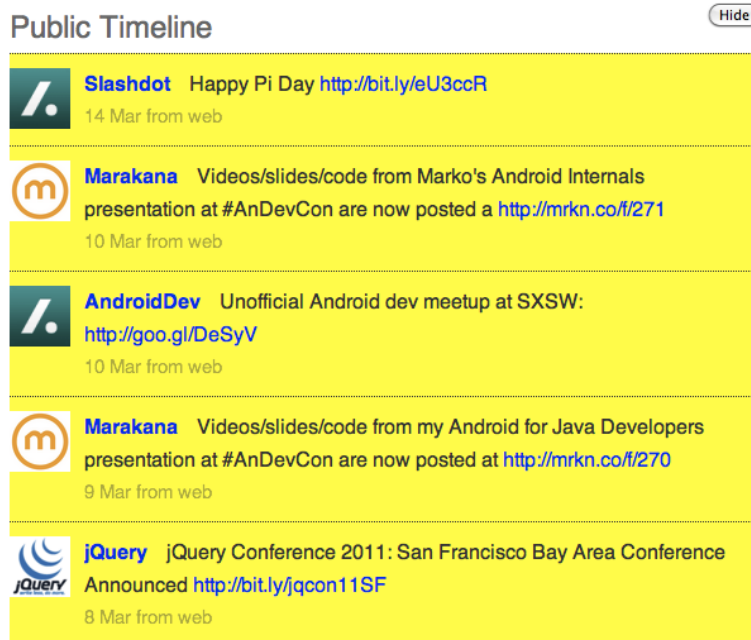
- It's common to place the `<script>` elements that load external JavaScript files in the `<head>` of a document.
 - Structurally, this is a good location, as it separates the *behavior* (the JavaScript code) from the *content* (the HTML elements and their content).
 - Functionally, this can be a problem. The browser *blocks* (pauses) to load the script before it continues loading and rendering the rest of the document.
- To increase page responsiveness, consider placing your `<script>` element(s) just before the `</body>` close tag.
 - This causes the browser to load and render all other page content before loading and executing your JavaScript code.
 - If you follow the design pattern of *progressive enhancement*, this gives users a functional web page they can browse before the interactive components implemented in JavaScript are fully loaded and initialized.
 - The page feels “more responsive” to users.

1.12 Execute Some JQuery Code

- Now that you have the library and that you know where to perform jQuery operations, let us try this out:
 1. In your `labs` folder, open `yamba1.html` in a web browser and note the page appearance.
 2. Create a JavaScript file containing these operations:

```
$(document).ready(function() {  
    $('.status').css('backgroundColor', 'yellow');  
});
```

3. Edit the HTML page to load jQuery and your JavaScript file.
4. Now reload the page, and observe the difference:



The screenshot shows a 'Public Timeline' with a 'Hide' button in the top right corner. It contains five items:

- Slashdot**: Happy Pi Day <http://bit.ly/eU3ccR>, 14 Mar from web.
- Marakana**: Videos/slides/code from Marko's Android Internals presentation at #AnDevCon are now posted a <http://mrkn.co/1/271>, 10 Mar from web.
- AndroidDev**: Unofficial Android dev meetup at SXSW: <http://goo.gl/DeSyV>, 10 Mar from web.
- Marakana**: Videos/slides/code from my Android for Java Developers presentation at #AnDevCon are now posted at <http://mrkn.co/1/270>, 9 Mar from web.
- jQuery**: jQuery Conference 2011: San Francisco Bay Area Conference Announced <http://bit.ly/jqcon11SF>, 8 Mar from web.

Note

After you have observed the effect of this jQuery script, you can revert `yambal.html` to its original state. The highlight effect shown is not used in subsequent exercises.

Of course, the result of this simple script could be achieved more easily just by editing your document's CSS, rather than running jQuery code. The advantage of jQuery will become more apparent as we create more complex and dynamic effects.

1.13 Introduction To JQuery: Exercises

1.13.1 Questions

1. Are these statements the same:

```
<script src="libs/jquery-1.6.4.js"></script>
<script src="libs/jquery-1.6.4.js" />
```

2. What are pros and cons of including jQuery at top of the page versus bottom?

1.13.2 Labs

Download the lab files at http://marakana.com/static/student-files/jquery_labs.zip if you have not done it yet

For these labs, start with the file `labs/yambal.html`. It includes a `<script>` element to load a file named `introduction.js`, which you should edit in the `labs` directory.

1. Write code that will check if the document is ready, and then alert the user. Do this using the “long” syntax.
 2. Update the code you wrote to check if the document was ready with the “short” syntax.
-

Chapter 2

Selection and DOM Traversal

Before you start doing any interactions with your page you need to select one or more elements in your page.

This chapter will illustrate different ways to select elements on your page.

You will also learn jQuery techniques for traversing the document object model (DOM)—that is, finding parent elements, siblings, etc. from an original selection.

2.1 Selection

- You must first select elements before you can manipulate them in any way.
 - jQuery does an excellent job at selecting elements (that was the original purpose of jQuery).
 - Selection is based on the open source *Sizzle* selector engine.
 - It provides support for most CSS1-3 selectors, plus its own custom extensions.
- jQuery selection returns a *jQuery object*.
 - The jQuery object represents **0** or more elements in your document.
 - The jQuery object is derived from the “global” `jQuery` function object, so it has access to all methods and properties defined in the `jQuery` function object.



Warning

If no elements match the selection, the result is a jQuery object representing **0** elements. You can use the object's `.length` property to determine the number of elements it represents. For example, `$('.div').length` returns the number of `<div>` elements in the page

- We are going to discuss different categories of selectors, but not all selectors are covered here. See <http://api.jquery.com/category/selectors/> for a complete list and description of supported jQuery selectors.

2.2 Basic Selectors

Basic jQuery selectors work the same as CSS selectors:

```
$('p');
```

Selects all the `<p>` elements in your page

```
$( '*' );
```

Selects all elements on the page

```
$( '.status' );
```

Selects all elements with the class `status`

```
$( '#header' );
```

Selects the element with the id `header`

Note

You should not use the same id for more than one element. If you do so and if you perform a selection by id, only the first matched element in the DOM will be returned

```
$( 'p, li, .status, #nav' );
```

Selects all `<p>` and `` elements, all elements with the `status` class, and the element with the id `nav`.

2.3 Hierarchy Selectors

jQuery hierarchical selectors also work the same as CSS selectors:

```
$( '#footer span' );
```

Selects the `` elements that are descendants of the element with the id `footer`.

```
$( 'ul > li' );
```

Selects the `` elements that are immediate children of `` elements.

```
$( 'h2 + p' );
```

Selects the `<p>` elements that are **immediately** preceded by an `<h2>` element.

```
$( 'h2 ~ p' );
```

Selects **all** `<p>` elements following an `<h2>` element that have the same parent as the `<h2>` element.

2.4 Selection By Attribute

Another powerful selection type is selection by attribute:

```
$( 'a[href]' );
```

Selects all `<a>` elements that have an `href` attribute.

```
$( 'a[href="http://mrkn.co/f/271"]' );
```

Selects all `<a>` elements whose `href` attribute exactly matches `"http://mrkn.co/f/271"`.

```
$( 'a[href*="goo.gl"]' );
```

Selects all `<a>` elements whose `href` attribute contains `"goo.gl"`. For example, this would match:

```
<a href="http://goo.gl/DeSyV" class="url">Interesting link</a>
```

```
$( 'a[href^="http://bit.ly"]' );
```

Selects all `<a>` elements whose `href` attribute starts with `"http://bit.ly"`. For example, this would match both of these elements:

```
<a href="http://bit.ly/eU3ccR" class="url">One link</a>  
<a href="http://bit.ly/jqcon11SF" class="url">Another link</a>
```

```
$( ' a [ href $ = " . pdf " ] ' ) ;
```

Selects all <a> elements whose href attribute ends with ".pdf". For example:

```
<a href="http://mrkn.co/docs/info.pdf" class="url">Download PDF version</a>
```

```
$( ' p [ lang | = " en " ] ' ) ;
```

Selects all <p> elements whose lang attribute matches either "en" or "en-*", where * can be anything.

Note

A detailed explanation on selection by attribute can be found at the following URL: <http://api.jquery.com/category/selectors/attribute-selectors/>

2.5 Form Selectors

jQuery also comes with useful selectors related to forms.

```
$( ' : button ' ) ; , $( ' : checkbox ' ) ; , $( ' : file ' ) ; , $( ' : image ' ) ; , $( ' : password ' ) ;
$( ' : radio ' ) ;
```

```
$( ' : reset ' ) ; , $( ' : submit ' ) ; , $( ' : text ' ) ;
```

Selects all <input> elements of the specified type.

```
$( ' : input ' ) ;
```

Selects all form elements, including <textarea> and <select> elements.

```
$( ' : checked ' ) ;
```

Selects all checkbox and radio elements that are checked.

```
$( ' : selected ' ) ;
```

For <option> elements, returns the selected option(s).

```
$( ' : disabled ' ) ; , $( ' : enabled ' ) ;
```

Selects all disabled/enabled form elements.

Note

It's recommended to precede these selectors with a tag name or some other selector; otherwise, the universal selector (*) is implied (e.g., \$(":checkbox") implies \$("*:checkbox")).

The following will serve as an example to demonstrate form selectors:

```
<form name="foodieform">
  <fieldset>
    <legend>Search a recipe:</legend>
    <label for="txt_recipe_name">Recipe name:</label>
    <input type="text" name="txt_recipe_name" id="txt_recipe_name" />
    <label for="meal_type">Type of meal:</label>
    <select id="meal_type" name="meal_type">
      <option value="Apetizer" selected>Apetizer</option>
      <option value="Entree">Entree</option>
      <option value="Drinks">Drinks</option>
      <option value="Desert">Desert</option>
    </select>
  </fieldset>
</form>
```

```

</fieldset>
<fieldset>
  <legend>Choose a type of food</legend>
  <input type="checkbox" name="food_type" value="French" checked>French</input>
  <input type="checkbox" name="food_type" value="Italian">Italian</input>
  <input type="checkbox" name="food_type" value="Mexican">Mexican</input>
  <input type="checkbox" name="food_type" value="Chinese">Chinese</input>
</fieldset>
<fieldset>
  <legend>Skills required</legend>
  <input type="radio" name="food_skills" value="1"/>Easy
  <input type="radio" name="food_skills" value="2"/>Medium
  <input type="radio" name="food_skills" value="3" checked/>Hard<br/>
</fieldset>
<input type="button" name="form-button" id="form-button" value="search"/>
<input type="reset" name="form-reset" id="form-reset"/>
</form>

```

- `$('. :checked')`; would select:

```

<input type="checkbox" name="food_type" value="French" checked>French</input>
<input type="radio" name="food_skills" value="3" checked/>Hard<br/>

```

- `$('. :radio')`; would select:

```

<input type="radio" name="food_skills" value="1">
<input type="radio" name="food_skills" value="2">
<input type="radio" name="food_skills" value="3" checked>

```

- `$('. :reset')`; :: Selects reset elements. In this case:

```

<input type="reset" name="form-reset" id="form-reset">

```

- `$('. :selected')`; would select:

```

<option value="Appetizer" selected>Appetizer</option>

```

- `$('. :text')`; would select:

```

<input type="text" name="txt_recipe_name" id="txt_recipe_name">

```

Note

More information on form selectors is available here ⇒ <http://api.jquery.com/category/selectors/form-selectors/>

2.6 Position Filters

jQuery allows you to filter the elements selected based on their position in the collection. For example:

```
$('.article:eq(2)');
```

Selects the **third** element with the class `article`. Elements in a collection are numbered starting with 0.

```
$('.article:gt(1)');
```

From the collection of all elements with the class `article`, select all elements following the second one.

```
$('.article:lt(3)');
```

From the collection of all elements with the class `article`, select up to the first three.

```
$('.article:first');
```

Selects the first element with the class `article`

```
$('.article:last');
```

Selects the last element with the class `article`

```
$('.article:odd');
```

Selects the odd elements out of all the elements with the class `article` (zero-based index: 1,3,5,etc)

```
$('.article:even');
```

Selects the even elements out of all the elements with the class `article` (zero-based index: 0,2,4,etc)

Note

The order of elements in a collection is determined by the order in which they appear in the document.

2.7 Other Filters

Other useful jQuery filters include:

```
$(p:contains('Marakana'));
```

Selects all `<p>` elements that contain the string "Marakana", either directly or in any of the child elements. The text matching is case sensitive.

```
$(div:has(h2));
```

Selects all `<div>` elements that have at least one `<h2>` element as a descendant.

```
$(option:not(:selected));
```

Selects all `<option>` elements that are **not** selected.

```
$(p:hidden);, $(p:visible);
```

Selects all `<p>` elements that are hidden/visible. An element is considered hidden if:

- It has a CSS `display` value of `none`
- It is a form element with `type="hidden"`
- Its width and height are explicitly set to 0
- An ancestor element is hidden, so the element is not shown on the page

2.8 jQuery Method Chaining

- The value returned from a jQuery selection is an instance of a jQuery object.
 - You can then invoke methods on that object to manipulate the HTML elements it represents. For example, `css()` to query and set CSS properties, `attr()` to query and set HTML attributes, `addClass()` to add one or more classes, etc.
- The value returned by most of these manipulator methods is the same jQuery object.
 - That means that you can *chain* method calls on the object. Here is an example:

```
$('.status')  
  .css('backgroundColor', 'yellow')  
  .attr('title', 'Training by Marakana');
```

This selects the elements with the class `status`, sets the background color of the selected elements to yellow, and then sets the `title` attribute of the selected elements.

2.9 DOM Traversal

- Starting with a jQuery object representing the results of a selection, you have the ability to traverse the DOM.
 - jQuery provides methods for selecting the children, the parents, and the siblings of each element in the wrapped set.
 - Most of these methods accept an option jQuery selector argument, which filters out traversed elements that don't match the filter.
 - The value returned by these methods is a jQuery object representing the newly selected elements.
 - There are a large number of traversal methods, including:

```
$('.article').children('p');
```

Selects all <p> elements that are direct first-level children of the elements with the class `article`.

```
$('.article').find('p');
```

Selects all descendant <p> elements contained in elements with the class `article`

Note

The difference between `.children()` and `.find()` is that the `.children()` method only travels a single level down in the DOM.

```
$('#h2').prev();
```

Selects the set of elements that immediately precede <h2> elements.

```
$('#h2').next('p');
```

Selects the set of <p> elements that immediately follow <h2> elements.

Note

In this example, if the element immediately following an <h2> is not a <p>, it is not included in the set of elements returned.

```
$('#form-button').siblings('fieldset');
```

Gets all the siblings that are <fieldset>

```
$('.menu-li2').parent();
```

Gets the parent elements of elements having the class `menu-li2`

- For more information on DOM traversal, see: <http://api.jquery.com/category/traversing/>

2.10 Filter Methods

- Starting with a jQuery object representing a set of elements, you also have the ability to apply a *filter method* to obtain a subset of elements.
 - The value returned by these methods is a jQuery object representing the subset of elements.
 - Filter methods include:

```
$('#content h2').first();, $('#content h2').last();
```

Selects the first/last of the <h2> elements contained within the element whose id is `content`.

```
$('#menu > li').eq(1);
```

Selects the second element within the element whose id is `menu`.

```
$('#li').filter(':even');
```

Selects the even-numbered elements in the set of elements.

```
$('#div').not('.article');
```

Selects the <div> elements that do not have the `article` class.

```
$('#li').has('ul');
```

Selects the elements that have a descendant.

- For more information on filter methods, see: <http://api.jquery.com/category/traversing/filtering/>
-

2.11 Advanced Method Chaining

- Combining method chaining with traversal methods can mix several related actions into one statement. For example:

```
$('#selected')
  .css('backgroundColor', 'yellow')
  .parent()
  .addClass('current');
```

- An important method to know when doing traversal and chaining is the `.end()` method.
 - The `.end()` method ends the most recent filtering/traversal operation in the current chain and returns the set of matched elements to its previous state. Here is an example:

```
$('.article') // ❶
  .find('p').css('color', 'green') // ❷
  .end() // ❸
  .find('h2').css('color', 'red'); // ❹
```

- ❶, ❶ Select elements with the class `article`.
- ❷, ❷ Select all `<p>` descendants and set their text color to green.
- ❸ Return to the previous selection (elements with the `article` class).
- ❹ Select all `<h2>` descendants and set their text color to red.

2.12 Selection and DOM Traversal: Exercises

For these labs, start with the file `labs/yamba2.html`. It includes a `<script>` element to load a file named `selections_and_dom_traversing.js`, which you should edit in the `labs` directory.

Note

To get a visual result for each selection we made, we are going to use jQuery CSS styling. To apply a background color to a selection, execute:

```
<your selection>.css('backgroundColor', 'color');
```

1. Select all but the first tweet and change the background color of those tweets to be yellow. See if you can come up with two different ways to accomplish this.
2. Select the button element using form selectors and change the background color of this button to be yellow.
3. Select all "bit.ly" links in tweets using attribute selectors and change their background color to be red. See if you can come up with two different ways to accomplish this.
4. Post an alert reporting the number of `<div>` elements on the page.
5. Add a dashed, black border of 1px to the profile image in each tweet.
 - You will have to apply the following CSS operations.

```
<your selection>.css('borderWidth', '1px');
<your selection>.css('borderColor', 'black');
<your selection>.css('borderStyle', 'dashed');
```

Chapter 3

DOM And Attributes Manipulation

- To provide interaction on a page you want to be able to manipulate elements dynamically.
- In this chapter, you are going to learn how to dynamically:
 - Create elements on the page
 - Insert existing or freshly created elements on the page
 - Move elements to a different location on the page
 - Clone elements
 - Remove elements
 - Replace elements
- From a selection of elements, you will be able to get and/or set their attribute values.

3.1 DOM Manipulation

- Manipulating the DOM using jQuery is a common operation you will have to do most of time.
- First we'll focus on the different methods allowing you to create, insert, move, clone, remove, and replace elements in the DOM.
 - Later we'll discuss modifying the content of an element and its attributes.

3.2 Creating Elements

There are two ways to create elements in jQuery:

- Often, the easiest way is to provide a string with HTML markup of the element as the sole argument to the `jQuery` function:

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
```
- Alternatively, you can provide a string with the basic HTML markup as the first argument, and a JavaScript object as the second argument.
 - The object's properties can specify `text`, event callbacks (discussed later), and other HTML attributes.
 - This is often more convenient when using calculated values for attributes.

```
var $menuLi1 = $('<li/>', {  
  'class': 'menu-li1',  
  'text': 'Italian Food'  
});
```

Note

It is considered a best practice to put quotes around object property names because some words such as `class` are reserved keywords in JavaScript.

- Using either method, the result is a jQuery object representing the new element.
-

Some jQuery programmers follow a convention of using `$` as the first character for variable names when those variables contain jQuery objects. Although this naming convention is not universal, we will follow it in this course.

3.3 Inserting Elements

- We have just seen how to create elements. However, these elements have not been inserted into the page yet.
- There are many ways to insert elements into a page. Elements can be inserted:
 - Before an existing element
 - After an existing element
 - Inside a container element, at its beginning
 - Inside a container element, at its end
- Moreover, there are two approaches to each method of inserting elements into a page:
 - Place the selected element(s) relative to another element
 - Place an element relative to the selected element(s)

3.4 Inserting an Element Before or After Another Element

3.4.1 Before an Existing Element

- `.insertBefore()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
$menuLi1.insertBefore('.menu-li1:first');
```

- `.before()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');  
$('.menu-li1:first').before($menuLi1);  
// Or...  
$('.menu-li1:first').before('<li class="menu-li1">Italian Food</li>');
```

3.4.2 After an Existing Element

- `.insertAfter()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$menuLi1.insertAfter('.menu-ul2:last');
```

- `.after()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$('.menu-ul2:last').after($menuLi1);
// Or...
$('.menu-ul2:last').after('<li class="menu-li1">Italian Food</li>');
```

3.5 Inserting an Element as the First or Last Child of a Parent

3.5.1 As the First Child

- `.prepend()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$('.menu-ul1').prepend($menuLi1);
// Or...
$('.menu-ul1').prepend('<li class="menu-li1">Italian Food</li>');
```

- `.prependTo()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$($menuLi1).prependTo('.menu-ul1');
```

3.5.2 As the Last Child

- `.append()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$('.menu-ul1').append($menuLi1);
// Or...
$('.menu-ul1').append('<li class="menu-li1">Italian Food</li>');
```

- `.appendTo()`

```
var $menuLi1 = $('<li class="menu-li1">Italian Food</li>');
$($menuLi1).appendTo('.menu-ul1');
```

3.6 Mass Insertion

- The various insertion methods can result in multiple insertions if the target selection contains more than one element.
 - For example, the following code results in inserting a new paragraph as the first child of each `<div>` in the document:

```
$('.div').prepend('<p>A new first paragraph</p>');
```

3.7 Moving Elements

- The same techniques that we have just discussed for inserting elements into a page, can also be used to move elements.
 - The only difference is that the elements to be inserted are the result of a previous selection.
- Here are two examples of moving the first element with the class `article` so that it is the last child of the element whose `id` is `section`.
 - Functionally, the only difference is the jQuery object returned by the method. This would be important if this code appeared within chained methods.
 - In this example, the jQuery object returned is the `section` element.

```
$('#section').append($('.article:first'));
```

- In this example, the jQuery object returned is the `article` element.

```
$('.article:first').appendTo($('#section'));
```

Note

If the target selection contains more than one element, the elements being moved are duplicated and inserted into each target.

3.8 Cloning (Copying) Elements

- We have just seen how to move elements, but what if we want a copy of those elements?
- The `.clone()` method creates a “deep copy” of an existing selection.
 - All descendant elements and text are duplicated.
 - You can optionally request replication of event handlers and additional jQuery metadata associated with the original selection.
 - `.clone()` returns a jQuery object of the duplicate, leaving the original selection in place.
 - For example:

```
var $article=$( '.article:first' ).clone(); // ❶  
$('#section').append($article); // ❷
```

- ❶ Here we do not get a reference to the jQuery selection object but a copy of the object.
- ❷ Now we insert the copy of the elements; the original one has not moved.

3.9 Removing Elements

- Once you have a selection of elements in the page, you can choose to remove this selection.
- There are two methods in jQuery that allow you to remove a selection: `.detach()` and `.remove()`.
 - Those two methods are similar because they both return the removed selection in case you want to restore it later.
 - The major difference is that when using `.remove()`, the selection loses its associated jQuery data and its attached events. This is not the case when using `.detach()`.

Note

Generally, use `.detach()` when you want to “reattach” the selection to the page again in the future; use `.remove()` if you intend to discard the selection.

3.10 Replacing Elements

- To replace a selection of elements on the page, we can use two methods: `.replaceWith()` and `.replaceAll()`.
 - The `.replaceWith()` method removes content from the DOM and inserts new content in its place with a single call.

```
$('.menu-li1').each(function() {  
    $(this).replaceWith('<h3>'+$(this).text()+'</h3>');  
});
```

Note

`.replaceWith()` returns the *original* jQuery object (that is, the one just removed from the DOM).

- The `.replaceAll()` method is corollary to `replaceWith()`, but with the source and target reversed.

```
$('.menu-li1').each(function() {  
    $('<h3>'+$(this).text()+'</h3>').replaceAll($(this));  
});
```

3.11 Element Content: HTML vs Text

- There are two methods in jQuery that can be used to get the content of a selection:
 - The `.html()` method that returns the inner html of the first element in a set of matched elements:

```
$('.article').html();
```

- The `.text()` method that returns the combined text contents of each element in the selection including their descendants:

```
$('.article').text();
```

- Using the same methods, you can set the html or text content of a selection:
 - The `.html(content_to_insert)` treats the content to be inserted as html:

```
$('.header').html("<b>cool</b>");
```

- The `.text(content_to_insert)` treats the content to be inserted as text:

```
$('.header').text("<b>cool</b>");
```

Note

Do not use the `.text()` method to get or set value for `<input>` elements. Use the `.val()` method instead

3.12 Element Attributes And DOM properties

- To get or set other attributes of an HTML element, use the `.attr()` method:
 - When used as a getter, the `.attr(attribute_name)` method only applies to the first element of the selection.
 - When used as a setter, the `.attr(attribute_name, attribute_value)` method applies to all elements of the selection.
-

- Prior to jQuery 1.6, the `.attr()` method could be used to set/get attributes or DOM properties. However, that could be also confusing in some places where the name is the same for both the attribute and the DOM property of the HTML element (for instance: `href`).

– For instance, consider the following HTML code:

```
...
<a href="about.html">About</a>
...
```

Does this return the element's attribute or the DOM property? (and both of those values are different)

```
$('#a').attr('href');
```

- As a solution to this problem, jQuery 1.6 has introduced the `.prop()` method to set or get DOM properties only:
 - When used as a getter, the `.prop(property_name)` method only applies to the first element of the selection.
 - When used as a setter, the `.prop(property_name, property_value)` method applies to all elements of the selection.

3.13 DOM And Attributes Manipulation: Labs

For these labs, start with the file `labs/yamba3.html`. It includes a `<script>` element to load a file named `dom_manipulation.js`, which you should edit in the `labs` directory.

Note

Here we are going to use the click event. Since we have not discussed event handling in detail yet, keep the following in mind for this lab:

- You use the `.click()` method to register a function to be executed when the user clicks on an element:

```
<your selection>.click(function(){ ... });
```

- Within the handler function, use `$(this)` to refer to the object the user just clicked on.

-
1. Let the user type in a new tweet. When the user clicks on the "Tweet" button, add that tweet to the top of the timeline by creating a new status line item.
 2. Repeat the previous lab, but this time try cloning an existing tweet and updating it with information for the new tweet.
 3. Let the user remove tweets by clicking on the X remove icon.



Important

You will notice that tweets that are added dynamically will not be removed when the user clicks on their remove icon. This is absolutely normal and this is a problem that we are going to fix in the labs for the events module.

Chapter 4

CSS Styling and JQuery

- In this chapter, you are going to learn how to read and/or modify the CSS properties of element(s).

4.1 Reading And Modifying CSS Properties

- Accessing CSS properties in plain old JavaScript is really a pain because of browser inconsistencies:
 - `getComputedStyle()` method in standards-based browsers
 - `currentStyle` and `runtimeStyle` properties in Internet Explorer
- As mentioned in the introduction chapter, some CSS properties such as `float` are named differently in the DOM depending on the browser (`styleFloat` for IE and `cssFloat` for W3C standards-compliant browsers).

Note

jQuery handles these browser inconsistencies. For example, it recognizes `float`, `styleFloat`, and `cssFloat` as synonyms for the same property.

4.2 Reading And Modifying CSS Properties

- jQuery provides the `.css()` method to allow you to get or set the CSS properties of element(s)
 - You can access CSS properties using their CSS names (e.g., `background-color`) or DOM name (e.g., `backgroundColor`).
- `.css(propertyName)` returns the value of the CSS property for the first element of the selection:

```
$('.div').css('backgroundColor');
```

Note

You cannot retrieve shorthand CSS properties such as `margin`, `border`, and `background` using the `.css()` method. You must request individual properties, such as `margin-left`, `border-left-width`, and `background-color`.

- `.css('propertyName', 'propertyValue')` sets the CSS property to the given value for all elements in the selection.

```
$('.header').css('backgroundColor', 'yellow');
```

- If you need to set more than one property at a time on a selection, you can pass the `.css()` method an object literal containing CSS properties along with their values:

```
$('.article').css({
  'backgroundColor': '#C2F5BF',
  'borderColor': 'yellow',
  'marginBottom': '10px'
});
```

When you use `.css()` to change a property, jQuery modifies the element's `style` property. For example:

```
$('#mydiv').css('color', 'green');
```

is equivalent to:

```
document.getElementById('mydiv').style.color = 'green';
```

Note

You can use shorthand CSS properties (e.g., `margin`, `background`, `border`) only when *setting* properties. For example, jQuery supports:

```
$selection.css('border-left', '5px solid #ccc');
```

For retrieving property values, you must specify individual property names. For example, if you want to retrieve the rendered margin, use: `$selection.css('marginTop')`, `$selection.css('marginRight')`, and so on.

4.3 Removing CSS Properties

- Setting the value of a style property to an empty string removes that property from an element if it has already been directly applied:
 - In the HTML `style` attribute
 - Through jQuery's `.css()` method
 - Through direct DOM manipulation of the `style` property
- This technique does **not** remove a style that has been applied with a CSS rule in a stylesheet or `<style>` element.
 - jQuery does not provide methods for directly modifying CSS rules.

4.4 CSS Classes

- We have just discussed how you can use the `.css()` method to quickly change CSS properties, however this one presents one main drawback:
 - When modifying CSS properties, we modify the `style` attribute of each element in the selection.
 - This mixes content and presentation, which is poor design.
 - To keep your code clean and maintainable, externalize the presentation in a CSS class that you can then apply to or remove from your selection of elements.
-

- jQuery provides you with the ability to apply or remove classes to a selection of elements using three main methods:

.addClass (className)

Dynamically adds one or more classes to each of the elements in the selection.

.removeClass (className)

Dynamically removes one or more classes to each of the elements in the selection.

.toggleClass (className)

Dynamically toggles the presence of one or more classes to each of the elements in the selection.

Note

For each of these methods, you may provide a space-separated list of class names.

- You can also use `.hasClass (className)` to test for the presence of a class on **any** of the elements in the selection.

4.5 Element Dimensions

- jQuery provides several methods for querying the size of an element.
 - The value returned is the *computed* size of the element, not necessarily the size specified in CSS.
 - The value returned is numeric, a unit-less pixel value (e.g., 400, not "400px").
 - When applied to a set of elements, the size of only the first element is returned.

<code>.height ()</code>	The height of the element itself
<code>.innerHeight ()</code>	The height including padding
<code>.outerHeight ()</code>	The height including padding and border, optionally including margin if you also provide a <code>boolean true</code> argument
<code>.width ()</code>	The width of the element itself
<code>.innerWidth ()</code>	The width including padding
<code>.outerWidth ()</code>	The width including padding and border, optionally including margin if you also provide a <code>boolean true</code> argument

- The `height ()` and `width ()` methods can also act as setters, changing the element size, if you provide a value as an argument.
 - If you provide only a numeric value, jQuery interprets it as pixel units.
 - If you provide a string value, you may use any valid CSS measurement (such as "100px", "50%", or "auto").
 - When applied to a set of elements, it changes the size of all elements in the selection.

In contrast, the `.css ('height')` and `.css ('width')` methods return values with units intact (for example, "400px").

jQuery 1.4.1 and later allows you to provide a function as an argument to `.height ()` and `.width ()`. The function is invoked for each element in the selection, and it receives two arguments: the index position of the element in the set and the current height/width of that element. The function should return the new height/width for that element.

4.6 Element Position

- The `.position()` method returns the position object of the first element in the selection relative to its first positioned ancestor (the *offset parent*).
 - It returns an object containing the properties `top` and `left`, with numeric values representing pixel offsets.
 - This method can be used only as a getter.
- The `.offset()` method returns the position object of the first element in the selection *relative to the document*.
 - It returns an object containing the properties `top` and `left`, with numeric values representing pixel offsets.
- You can also use `.offset()` to *change* the position of the selection.
 - Provide an object containing the properties `top` and `left`, which are integers indicating the new top and left coordinates for the elements.
 - If the element's `position` style property is currently `static`, the `.offset()` method changes it to `relative` to allow for this repositioning.

4.7 CSS Styling and JQuery: Labs

For these labs, start with the file `labs/yamba4.html`. It includes a `<script>` element to load a file named `css_styling.js`, which you should edit in the `labs` directory.

1. Select the second line of every status update (where it says the date it was posted) and make it smaller by 20%.
2. Change the color of every other row to a slightly gray color (`#efefef`). Do this by modifying the CSS property of each matched status item.
3. Change the color of every other row, but this time do it by adding the CSS class `zebra` to all the rows you want to be a different color.

Note

The `zebra` class is already defined in the external CSS file, `yamba.css`.

4. Set the class of the current hovered tweet to `zebraOver`. Remove that class when the mouse is not over the tweet. Hint: You can use the register handlers for the hover event like this:

```
<your selection>.hover(  
    function () { //On mouse in },  
    function () { //On mouse out }  
);
```

Note

The `zebraOver` class is already defined in the external CSS file, `yamba.css`.

Chapter 5

Events

5.1 Events Overview

- When a user is interacting with a web page, a lot of events are fired, such as:
 - Clicking on an element
 - Typing in a text box
 - Scrolling on the page
 - Hovering on an element
 - etc...
- JavaScript allows you to detect an event and execute code in response to it.
 - The code is known as an *event handler* or a *callback*.
- jQuery provides cross-browser event handling support so you do not have to worry about the differences in how events are handled by various browsers.

5.2 Binding An Event Handler

- To handle an event in your script, you *bind* an event handler function to one or more elements for a specific event type.
- The `.on()` method binds your event handler to a selection of elements. (Formerly jQuery used the `.bind` method. As of 1.8 `.bind` is still supported but deprecated.)
 - The first argument is a string specifying the event type.
 - The second argument is your handler function. The `.on` method accepts other arguments but we'll discuss those later.
- When your handler is triggered, jQuery automatically sets the local variable `this` to refer to the *DOM* element to which the handler is bound.
 - You can make use of it as a jQuery object by passing it to the jQuery function: `$(this)`
 - For example:

```
$('.header').on('click', function () {  
    $(this).css('background-color', 'yellow');  
});
```

Now clicking on an element having the `header` class changes its background to yellow.

You can bind multiple events to the same handler by providing a space-separated set of event types, for example:

```
$('.header').on('click mouseleave', function () {
  $(this).css('background-color', 'yellow');
});
```

5.3 Binding Shortcut Methods

- jQuery has convenient methods for binding common event types.
 - You provide only your handler function as an argument, as in:

```
$('.header').click(function () {
  $(this).css('background-color', 'yellow');
});
```

blur	change	click	dblclick	error	focus
focusin	focusout	keydown	keypress	keyup	load
mousedown	mouseenter	mouseleave	mousemove	mouseout	mouseover
mouseup	resize	scroll	select	submit	unload

5.4 Unbinding Handlers and “One-Shot” Handlers

- You can remove event handlers using the `.off()` method.
 - With no argument, `.unbind()` handlers for all event types on the selection.
 - Providing the name of an event type as an argument removes all handlers for that event type.
 - Providing a reference to the handler function as a second argument removes only that handler for the event type.

Note

To remove a specific handler function, you would need to store a reference to the handler function in a variable for later access.

Tip

If you plan to unbind handlers, it's best to do so by specifying the particular handler you want to unbind.

- If you want to have an event handler execute, at most, one time for a particular element, you can register the handler with `.one()` instead of `.on()`.
 - The syntax of `.one()` is the same as the simple usage of `.on()`.
 - jQuery automatically unbinds the handler from the element after the first time it executes.

5.5 The Event Object

- jQuery automatically provides your handler function with an event object as an argument.
-

- The event object provides detailed information about the event.
- jQuery provides a custom event object normalized according to W3C standards for cross-browser standardization.
- If you don't need the event object for your function's operation, you don't need to include it in your function's declaration.
- Here are some of the properties and methods of the jQuery event object:

pageX, pageY

The coordinate of the mouse position when the event was fired

target

The DOM element that initiated the event

timestamp

The time the event occurred, expressed in milliseconds since midnight UTC, January 1, 1970

type

The type of event that has been fired (such as `click`, `mousedown`, etc...)

which

The button or key that was pressed

preventDefault ()

Invoke to prevent the default action of an event (e.g., to suppress following a link when clicked)

stopPropagation ()

Invoke to prevent the event from bubbling up to parents elements

Note

If your handler function returns `false`, it has the same effect as invoking `preventDefault ()` and `stopPropagation ()` on the event.

5.6 Event Delegation

- The techniques shown so far establish event bindings for only the elements that exist in the selection at that time.
- This can be a problem if your script dynamically adds new similar elements to the page. For example:

```
$('.article .header').on(function () { // ❶
    $(this).css('background', 'yellow');
});

var $article=$('#section .article:first').clone();
$('#section').append($article); // ❷
```

- ❶ Binding an event handler to each element with the class `header` in elements with the class `article`
- ❷ Adding a similar element at the end of the element with the id `section`

- If you click on the new element with the `header` class, you can see that the event is not bound to any event handler.
- So, should you do this binding operation again for these new elements? In a word: No!
 - Not convenient at all
 - The time taken to bind event handlers affects performance
- The solution to this problem is to use *event delegation*.
 - Event delegation lets an event “bubble up” the DOM tree to a parent element where it is handled.
 - In the handler, `this` still refers to the DOM element on which the event occurred, not the parent element where it is handled.

5.7 Event Delegation And JQuery

- jQuery formerly provided two methods: `.live()` and `.delegate()`. As of jQuery 1.7 these methods are deprecated in favor of `.on()`. We'll explain all three methods.
- The syntax for `.live()` is the same as for `.bind()` or simple usage of `.on()`.

- `.live()` delegates to the document root.
- For example, we could solve the problem of the previous slide with:

```
$('.article .header').live('click', function () {  
    $(this).css('background', 'yellow');  
});
```

- The `.delegate()` method—introduced in jQuery 1.4.2 has a different syntax.
 - The jQuery object on which it is invoked serves as the delegate.
 - The first argument is a selector to filter the elements that trigger the event.
 - The second and third arguments are the event type(s) and handler function.
 - The following example solves the problem encountered in the last section:

```
$('.article ').delegate('.header', 'click', function () {  
    $(this).css('background', 'yellow');  
});
```

- The `.on()` method—introduced in jQuery 1.7 works like `.delegate()`
 - The jQuery object on which it is invoked serves as the delegate if an optional filter expression is passed
 - The first argument is still the event name passed as a string
 - The second argument is an optional selector string. When passed it is used to filter the objects that trigger the event
 - The following example solves the problem encountered in the last section:

```
$('.article ').on('click', '.header', function () {  
    $(this).css('background', 'yellow');  
});
```

5.8 .live() vs .delegate() vs .on()

- The `.delegate()` method was preferred method to use instead of `.live()`.
- The `.live()` method fails to work when chaining DOM traversal methods with it. Example:

```
// This will not work  
$('.article').children('.header').live('click', function() {  
    $(this).css('background-color', 'yellow');  
});
```

- The `.live()` method binds event handlers to the document. When using `.delegate()`, you can provide a context for your events
 - The performance is improved when using `.delegate()`
- The `.live()` method cannot use the event object's `.stopPropagation()` method.
 - You can only return `false` from the handler function to stop event propagation, which also suppresses the default action of the event.

- However both `.live()` and `.delegate()` are deprecated now: `.on()` has all the performance benefits of `.delegate()` without changing the order or arguments. All new code written against jQuery 1.7 or greater should use `.on()`.

As of jQuery 1.4, `.live()` can delegate to an element other than the document root. However, the syntax for doing so is cryptic.

5.9 Events: Labs

For these labs, start with the file `labs/yamba5.html`. It includes a `<script>` element to load a file named `events.js`, which you should edit in the `labs` directory.

1. Remember that during the lab on DOM manipulation, we saw that our new dynamically-added tweets were not removed when the user clicked on their little remove icon. Fix this problem with the new techniques you have just learned.
2. Keep the status text area collapsed (define its height with jQuery to be 37px) and hide the "Tweet" button and counter (using the `<your selection>.hide()` method) until the user clicks on it for the first time. Set the text area height to be 56px and show the Tweet button and counter using the `<your selection>.show()` method.
3. We have a character counter that displays 140 but currently it doesn't update as we type our status. Implement this functionality into the character counter. (Hint: looking at this page might help: <http://api.jquery.com/category/events/-keyboard-events/>)
4. Modify the counter so that when the user has 10 or fewer characters remaining, it turns red. If the counter goes negative, disable the Tweet button.
5. Change CSS width properties when the size of the browser window changes. Below are the conditions:
 - When the body width is less than 500px: `min-width:300px` for `<div id="wrap">` and `width:85%` for `<div id="content">`
 - If the body width is larger than 500px: `min-width:500px` for `<div id="wrap">` and `width:66%` for `<div id="content">`
6. We already have hint text in the textarea, but it does not disappear when the user clicks on the textarea. Implement this hint functionality using the `blur()` and `focusin()` methods:
 - When the user gives focus to the text area, empty the default message or keep the message entered by the user.
 - When the text area does not have the focus anymore, check if the textarea is empty, and if so, display the hint.
7. Change the background color of the tweets as the user hovers the mouse over them (just like we did in the CSS styling labs), except now you also want to support the same behavior for dynamically-added tweets.

Chapter 6

Writing Your Own Plugins

The plugin pattern is a powerful mechanism for promoting good code organization and reuse. In this lesson we will learn to structure our code to create jquery plugins.

6.1 Creating Your Own Plugin: Why?

- Encapsulate your code for reuse purposes
- Public distribution
- Maintain chainability
- Public distribution
- Prevent namespace clashing
- Plugins in jQuery are easy to write!

6.2 Creating Your Own Plugin: How?

- Plugin creation follows three main steps:
 1. The method you create is passed a DOM selection (a jQuery object)
 2. The method performs some DOM manipulation, event binding, etc...
 3. The method returns the original selection (a jQuery object) so it maintains chainability
- To start creating a plugin you would add your method to the jQuery prototype.
- The jQuery prototype is accessed with `$.fn` or `jQuery.fn`:

```
$.fn.pluginName = function(options) {  
  //Implement your plugin here  
}
```

- However, this is poor design. The `$` syntax could conflict with other JavaScript libraries you might be using.
- You could use a self-invoking anonymous function like this (recommended design for avoiding conflicts):

```
(function($){  
  $.fn.pluginName = function() {  
    //Implement your plugin here  
  };  
})(jQuery);
```

- Inside the function your selection (jQuery object on which you called the plugin) is accessible using the `this` keyword. There is no need to wrap it like `$(this)` since it is already a jQuery object.
- Once the plugin is done, you would save it into a JavaScript file (generally `jquery.pluginname.js`) and include it in your HTML file along with the jQuery library.
- Finally to use your plugin you would call it on a selection.

```
$('.selection').pluginName();
```

6.3 Adding configurability

It's ok to accept arguments to the plugin function itself. If you have multiple arguments it is common practice to use an object and jQuery's `$.extend` function to accept multiple configuration arguments but provide default values.

```
(function($){
  $.fn.pluginName = function(options) {
    var defaults = {'color': '#FF0000', 'length': 1000};
    var args = $.extend(defaults, options);
    //Implement your plugin here, using the args
  };
})(jQuery);
```

6.4 Creating Your Own Plugin: Best Practices

- As we have seen, there are two structures for creating a plugin.

– The first one (not recommended at all):

```
$.fn.pluginName = function(options) {
  //Implement your plugin here
}
```

– And the second one (avoid conflicts when using other libraries ⇒ always use this one):

```
(function($){
  $.fn.pluginName = function() {
    //Implement your plugin here
  };
})(jQuery);
```

- Do not wrap `this` in the immediate scope of the plugin's method since `this` refers to the jQuery object on which we call the plugin.
- For chainability purposes, it is highly recommended to return the jQuery object (`this`) on which we call the plugin.
- In case you need to pass the plugin some settings, consider using objects over a list of arguments.
- Only use a method for your plugin in order to avoid cluttering the namespace.

Note

With so many plugins out there, you should analyze the plugins you are considering using as if you were writing them with the previously mentioned best practices in mind.

6.5 Plugins: Labs

For these labs, start with your previous completion of `labs/yamba5.html`. You finished the events lab with code in a file named `plugins.js`. Reorganize your code into plugins:

1. Create a new file called `jquery.hint.js` and create a plugin that encapsulates the "hint" functionality. Using the hint plugin should look like:

```
$('#mytextarea').hint();
```

Be sure that your plugin would work for multiple controls on the same page!

2. Create a new file called `jquery.counter.js` and create a `counter` plugin which counts the number of characters in a form input control and displays the number of remaining characters. The plugin should accept options to specify the max number of chars, the selector rule to display the count in, and the cutoff point at which the count color changes to red. It should also accept callbacks to be called when the number of characters remaining goes below or above 0. Using the plugin should look something like:

```
$('#mytextarea').counter({'max': 140, 'warning': 10, 'valid': show_button, 'invalid': ↵  
hide_button, 'showin': '#counter'});
```


Chapter 7

Effects and Custom Animations

In this chapter, we are going to discuss how to:

- Use jQuery built-in effects
- Do your own custom animations

7.1 JQuery Built-in Effects

- jQuery comes with a lot of convenient and common built-in effects that are seen on the web:
 - Hiding or showing elements
 - Fading in or fading out elements
 - Sliding up or sliding down elements
- The jQuery effect methods accept two optional arguments:
 - An animation speed (`slow` for 600ms, `fast` for 200ms, or a specified integer number of milliseconds)

Note

For methods other than `.show()` and `.hide()`, the default animation speed is 400ms.

- A callback function, which is invoked after the animation is complete

7.2 Showing or Hiding Elements

- The `.hide()` method hides a selection of elements on the page.
 - It sets the elements' `width`, `height` and `opacity` to 0, using an animation effect if you provide a speed argument, and then sets the elements' `display` to `none`.
 - The `.show()` method displays a selection of elements on the page.
 - It restores the elements' `display` to their initial values, then sets the elements' `width`, `height` and `opacity` to their initial values using an animation effect if you provide a speed argument.
-

```
$('.menu-ull').hide();

$('.menu-lil').toggle(function() {
    $(this).next('.menu-ull').show();
}, function() {
    $(this).next('.menu-ull').hide();
});
```

Note

If you don't provide a speed argument, the `.show()` and `.hide()` methods take effect immediately without an animation.

7.3 Fading Effects

- The `.fadeOut()` method progressively sets the opacity of the selected elements to be transparent (0), and then sets the elements' display to none.
- The `.fadeIn()` method restores the elements' display to their initial values, and then progressively sets the opacity of elements to be fully opaque (1).

```
$('.menu-ull').hide();

$('.menu-lil').toggle(function() {
    $(this).next('.menu-ull').fadeIn();
}, function() {
    $(this).next('.menu-ull').fadeOut();
});
```

7.4 Sliding Effects

- The `.slideUp()` method progressively sets the height property of the selected elements to 0, and then sets the elements' display to none.
- The `.slideDown()` method restores the elements' display to their initial values, and then animates their height property to their original values.

```
$('.menu-ull').hide();

$('.menu-lil').toggle(function() {
    $(this).next('.menu-ull').slideDown();
}, function() {
    $(this).next('.menu-ull').slideUp();
});
```

7.5 Creating Your Own Animations

- The previous methods are convenient and already coded for you, but what if this does not fit your needs and you want to provide your own animation?
 - jQuery comes with the `.animate()` method, which allows you to do custom animations of a set of CSS properties.
 - The only mandatory argument you have to pass to this function is the map of CSS properties.
-

- The three other optional arguments are:
 - The duration of the animation (`fast` for 200ms or `slow` for 600ms or any integer value)
 - The easing parameter defines the speed of the animation at different "stages" of the animation. For instance, `slow` at the beginning and end of the animation, and `fast` between the beginning and the end. jQuery has two built-in types of easing available: `linear` and `swing` that provides a more natural animation and is used by default
 - A callback function to execute after completion of the animation

```
$('.menu-li2').click(function(){
    $(this).animate({
        borderLeftWidth: "0px",
        borderRightWidth: "12px"
    }, 600);
});
```

Note

CSS shortcut properties such as `border: '1px solid black'` are not supported. You would have to be more explicit:

```
$(selection).animate({
    borderWidth: "1px",
    borderStyle: "solid",
    borderColor: "black"
}, 600);
```

7.6 Animation Notes

- There are some things to keep in mind when using `.animate()`:
 - Color animation is not supported (unless you either use the Color Animation plugin <http://plugins.jquery.com/project/color> or jQuery UI that includes this plugin)
 - If you wish to add easing functionality beyond what is built-in to the jQuery library, you can either use the easing plugin (<http://plugins.jquery.com/project/Easing>), or use jQuery UI which has the easing plug-in built into it.

7.7 Animation Queues

- If you invoke an effect method on an element that is already running an animation effect, the new effect is added to the element's *animation queue*.
 - The effects in the animation queue are executed in sequence.
 - Each element has its own animation queue. Animation effects on separate elements execute simultaneously.
 - For example, slowly fade out then fade in a selection:

```
$('#icon')
    .fadeOut('slow')
    .fadeIn('slow');
```

- The `.delay()` method introduces a pause before the next animation effect. Specify the delay in milliseconds. For example, to fade in slowly, pause 1 second, and then fade out slowly:

```
$('#icon')
  .fadeIn('slow')
  .delay(1000)
  .fadeOut('slow');
```

7.8 Stopping Animations

- The `.stop()` method immediately stops any current animation on the selection. It accepts two optional boolean arguments.
- By default, the next effect in the animation queue begins execution.
 - By specifying `true` for the first argument, the animation queue for the element is completely cleared.
- By default, when the animation is stopped, the element is left in any intermediate state it was in at the time.
 - By specifying `true` for the second argument, the element immediately goes to the end state of the terminated animation effect.

7.9 Additional Animation controls

- There are a couple of “global” properties to control your animations:
 - You can choose to disable all animations by doing: `jQuery.fx.off=true;`
 - We have also seen that jQuery has preset speeds for animations: `slow` (600ms), `fast` (200ms), and a default speed of 400ms. If you want to set your own custom speed for these presets, instead of hard coding them in every instance you use them, jQuery provides an object containing those speeds `jQuery.fx.speeds`:

```
jQuery.fx.speeds.slow = 10000;
```

7.10 Effects and Custom Animations: Labs

For these labs, start with the file `labs/yamba6.html`. It includes a `<script>` element to load a file named `effects_and_custom` which you should edit in the `labs` directory.

1. We want to slowly animate the whole page to move to the top when the user clicks on the "Go to top" button. Hint: use the `scrollTop` property on the `<html>` and `<body>` elements.
2. When the user clicks on the "Hide" button, we want to hide the header and footer leaving only the content on the page
3. Modify the "Hide" button from the previous lab so that it toggles between hide and show.
4. Modify the previous example to make a slow toggle.
5. When the user clicks on the "Tweet" button, we want to add this tweet to the top of the list of statuses. When the new status is added, slide it down slowly, fade it out fast, and then fade it in fast.
6. Make tweets fade out when deleted and make sure that you handle dynamically-added tweets.
7. Make the new status box slide down slowly when the user clicks on it for the first time.
8. Make the side menu (`<div id="sidemenu">`) scroll when the user scrolls the page:

- Listen for the `scroll` event on the window object.
- Then animate the `top` property of the side menu.

9. Create an accordion (without jQuery UI, which we will see later) using the side menu:

- First, hide all the elements but the first element.
 - Remove the borders for the last `` of each `.menu_ul2`.
 - When clicking on an `.menu_li1`, get its `` child.
 - Check if this same `` is hidden.
 - If this `` is hidden, find others that are visible and hide them.
 - Show the previously selected child.
-

Chapter 8

Ajax

- Ajax is the combination of technologies that has made the web 2.0 ⇒ Web Applications.
- In this chapter, you are going to use the power of jQuery to easily use Ajax without worrying about web browser inconsistencies.
- We are going to discuss:
 - The Ajax technology itself
 - The limitations of Ajax
 - The shorthand methods provided by jQuery to perform Ajax calls
 - The core method for performing Ajax calls
 - How to setup Ajax with jQuery
 - How to handle Ajax errors with jQuery

8.1 What Is Ajax?

- The term Ajax was coined in 2005 and stands for Asynchronous Javascript and XML.
- Before Ajax, the user experience flow when using an application was the following:
 - The user clicked on a link to access a new page or sent a form to search information on a website.
 - The server received the request and processed it.
 - During this time, the user had to wait for his/her browser to load the new page.
- Ajax is a combination of technologies that enables the web developer to make **asynchronous** calls to the server:
 - XML for interchanging data.

Note

Other formats can be used for interchanging data such as HTML, JSON or even plain text.

- JavaScript: The XMLHttpRequest object to make asynchronous calls + event binding to process the data once it is there.
 - Using Ajax, you provide end-users a user experience that feels more like using a desktop application.
 - Parts of the page are "refreshed".
 - User can still use the web application without having to wait for a new page to load.
-

8.2 Limitations Of Ajax

- As you might already think about creating big mashups, there is a big limitation to Ajax to take into consideration: The same-origin policy.
- The same origin policy has been implemented into major browsers since Netscape Navigator 2.0.
- It prevents a document or script loaded from one site of origin A from manipulating properties of or communicating with a document loaded from another site of origin B.
- It prevents attacks like hijacking an existing user session or phishing.
- For example, with the URL: <http://www.company.com/page.html>

URL	Same origin?	Reason
http://www.company.com/page2.html	YES	
http://www.company.com/dir/-page.html	YES	
http://blog.company.com/page.html	NO	Different host
https://www.company.com/-page2.html	NO	Different protocol
http://www.company.com:8080/-page2.html	NO	Different port

- Browsers check access against the same origin policy when we, for instance:
 - Manipulate browser windows
 - Request URL via the XMLHttpRequest
 - Manipulate frames (including inline frames)
 - Manipulate documents (included using the object tag)
 - Manipulate cookies

8.3 Load HTML Content: The .load() Method

- There are a lot of shorthand, convenient methods in jQuery to perform Ajax calls and the first one we will discuss here is the `.load()` method.
- The `.load()` method allows you to load returned HTML content from the server into a specified part of the page.

Note

You can directly load a static HTML file or a dynamic page that returns HTML content

- For example, let us say that you want to load the content of the first `<div class="article">` element from an external HTML static file:

```
$('div.article').eq(0).load('article_content.html'); // ❶
```

- ❶ Dynamically insert the content of the static file (content inside the `<body>` tag) in the specified selector
-

8.4 Hijax With JQuery

- In the case that the user has disabled JavaScript, or it is not present (as is the case with text-based browsers), we need to provide progressive enhancement using Ajax. We are going to use a technique called Hijax here:
 - Design your application in a way that users without JavaScript can click on links and load the information the old fashioned way (synchronous loading).
 - For users with JavaScript, we would intercept the links and dynamically load the content in a certain area of the page.
- Say we have a main page with a few hyperlinks in the footer pointing to an about page and a contact page:

```
<div id="footer">
  <ul>
    <li><a href="about.html">About</a></li> ❶
    <li><a href="contact.html">Contact</a></li> ❷
  </ul>
</div>
```

❶, ❷ The way hyperlinks are written makes them still usable by users that do not have JavaScript available

- Then let us do some hijax with jQuery:

```
$(document).ready(function() {
  $('#footer a').click(function(e) {
    var url=$(this).attr('href'); // ❶
    $('#section').load(url); // ❷
    e.preventDefault(); // ❸
  });
});
```

- ❶ Get the url to load
- ❷ Call the load method with the right URL
- ❸ Calling `preventDefault()` prevents the user from following the hyperlink

8.5 Select In Detail What To Load

- We have just talked about hijax techniques allowing both support for clients with and without JavaScript.
- However, we still have a big problem: We load the whole external page into a selection.
- How do we select just a part of the external page, so we can control what to load?
- Using the same `.load()` method, there is a way to specify which part of the page you want to load using selectors:

```
$(document).ready(function() {
  $('#footer a[href="about.html"]').click(function(e) {
    var url=$(this).attr('href');
    $('#section').load(url) + " #about"; // ❶
    e.preventDefault();
  });
  $('#footer a[href="contact.html"]').click(function(e) {
    var url=$(this).attr('href') + " #contact"; // ❷
    $('#section').load(url);
    e.preventDefault();
  });
});
```


- 1, 2 The syntax to provide here is a combination of the file to load plus the selector to specify which part of the page to load

Note

You might think that this reduces the bandwidth since we are precise about which part of the page to load using a selector. But this is not the case. The entire page is still loaded and then the selector is run against it.

8.6 Advanced Loading Using The `.load()` Method

- In the previous section on the `.load()` method, we discussed how to load an external static HTML file.
- Using the `.load()` method, you can also load a dynamic page that returns HTML content.
- In a real web application, the data you need to load is generated by server-side scripts operating on a database.
 - You can pass arguments to those scripts in order to get the data you want.
- The `.load()` method accepts a second parameter allowing you to specify query data either as a query string or a JavaScript property object.
 - The two examples below are equivalent:

```
$('#section').load('recipes.php', 'type=french&name=quiche');

$('#section').load('recipes.php', {
  type: 'french',
  name: 'quiche'
});
```

8.7 JSON and JSONP

- A few years ago, websites used to share XML data with other websites using APIs to create mashups.
 - However XML data is difficult to parse on the client side.
 - Main alternative to XML ⇒ JSON
 - Stands for JavaScript Object Notation.
 - The data is structured as plain JavaScript objects.
 - There is no parsing required, and it is ready to be used by our scripts.
 - Under the same origin policy, a web page served from one origin cannot normally connect to or communicate with a server located at a different origin ⇒ An exception is the HTML `<script>` element.
 - Using JSONP (JSON with Padding), you can take advantage of the `<script>` element by specifying the source of the server-side script you want to use.
 - You specify a callback method of your own in the URL of the script.
 - The script returns a JavaScript function call (your function) with the JSON data as a parameter.
-

8.8 Load JSON Data: The .getJSON () Method

- Let us take the case of the twitter API, which provides support for JSONP. The URL to call would look like this:

```
http://search.twitter.com/search.json?callback=processTweet&q=food
```

- Gets all the tweets mentioning food in a JSON structure and passes this as a parameter to our function "processTweet"
- jQuery provides a shorthand method that allows you to perform Ajax calls using the JSON format: `jQuery.getJSON()`.
 - The first argument is the URL for the request.

Note

If the URL includes a parameter `callback=?` then `jQuery.getJSON()` uses JSONP instead of JSON. jQuery automatically generates the `<script>` element and callback function to accept the JSONP response.

- The next optional argument is the query data, either as a query string or a JavaScript property object.
- The final optional argument is a callback function executed if the request succeeds. When invoked, it receives the JSON object as an argument.
- Let us look an example of that:

```
// Get the term to search
var searchRecipe=$('#txt_recipe_name').val();
// Clean the div tweetsresult
$('#tweetsresult').children().remove();

$.getJSON( "http://search.twitter.com/search.json?callback=?",
  {q: searchRecipe},
  function(data) {
    $.each(data.results, function() {
      $('#<div></div>')
        .hide()
        .append('')
        .append('<span><a href="http://www.twitter.com/'
          + this.from_user + '>' + this.from_user
          + '</a>' + this.text + '</span>')
        .appendTo('#tweetsresult')
        .fadeIn();
    });
  });
```

8.9 The Main Ajax Method: .ajax ()

- All the previously mentioned shorthand methods (and other Ajax methods) internally call the `jQuery.ajax()` method.
- Those previously mentioned methods are wrappers around the `jQuery.ajax()` method for specific tasks.
- In case you need more control over your Ajax call, the `jQuery.ajax()` method is the one to use. It allows you to use:
 - GET or POST method
 - Error callback
 - Different format of data
 - etc...
- This configuration is passed as an argument to the `jQuery.ajax()` method as a JavaScript object.
- Below are some of the configuration options that you can use (For more details, visit <http://api.jquery.com/jQuery.ajax/>):

<code>async</code>	Boolean that defines if you want your request to be sent asynchronously or not (<code>true</code> by default)
<code>cache</code>	Do we want to keep result in the cache or not? <code>true</code> by default excepts when the <code>dataType</code> is <code>json</code> or <code>script</code>
<code>complete</code>	This is the callback function to be called regardless of success or failure. This callback function is passed the raw <code>XMLHttpRequest</code> object and a string representing the status of the request as arguments
<code>context</code>	Defines the scope of the callback function (the value of <code>this</code>)
<code>data</code>	The data to send to the web server. This can be a query string or a JavaScript object
<code>dataType</code>	Specifies the expected data type returned by the server. When not specified, jQuery will look at the MIME type of the response
<code>error</code>	Defines the error callback function to be called in case of an error. This callback function is passed the raw <code>XMLHttpRequest</code> object and a string representing the status of the request as arguments
<code>jsonp</code>	Callback function name in the case of a JSONP request. This value will be used instead of <code>callback</code> in the <code>callback=?</code> part of the query string in the url
<code>success</code>	Callback function to be called if the request succeeded. This callback function is passed the data returned from the server, the status string of the request and finally the raw <code>XMLHttpRequest</code> object
<code>timeout</code>	Set the timeout for the request
<code>traditional</code>	Set this to <code>true</code> if you want to use the traditional style of param serialization ⇒ http://api.jquery.com/jquery.param
<code>type</code>	The HTTP method to use. Default is GET
<code>url</code>	The URL to request

8.10 Ajax Global Settings

- There are some settings that you might want to use in every Ajax call you make.
- However, putting those same settings in every Ajax call you make is not convenient.
 - Error prone
 - Difficult to maintain
- jQuery comes with the `jQuery.ajaxSetup()` method that allows you to specify global settings so you do not have to include them explicitly in every `jQuery.ajax()` call you make.

Note

You can override these global settings in any `jQuery.ajax()` call you make.

8.11 Error Handling

- We have seen that there is an `error` parameter where you specify the error callback function.
 - This error callback function might be the same for every Ajax call.
-

- The `.ajaxError()` method gives you the ability to configure error handling globally rather than on a per-call basis.
- These global events are attached to a DOM element where you would show an error message to the user.

```
$('#ajax_error').ajaxError(function(event, request, settings) {  
    $(this).html("<b>Error when requesting " + settings.url + "</b>");  
});
```

Note

There are also global versions for the success and complete events using `.ajaxSuccess()` and `.ajaxComplete()` methods

8.12 Ajax: Labs

For these labs, start with the file `labs/yamba7.html`. It includes a `<script>` element to load a file named `ajax.js`, which you should edit in the `labs` directory.

Note

`ajax.js` already implements some of the features we implemented in previous labs. Please do not remove them! It also provides a function, `statusHTML(status)`, that constructs the HTML code for a tweet. You will call that method when looping through the JSON result object.

1. Get the public timeline from `yamba.marakana.com` using JSONP:
 - Use the `.getJSON()` method
 - URL to query: http://yamba.marakana.com/api/statuses/public_timeline.json?callback=?

Note

For the rest of the labs, you will be provided a virtual machine hosting an open source micro-blogging web application (statusnet). The page and the front-end script will have the same origin as the backend script so you will not have any cross-origin issues.

2. Login and get the friends' timeline from the server using the `$.ajax()` method
 - URL to request: `/api/statuses/friends_timeline.json`
 - Data type: `json`
 - Username: `student`
 - Password: `password`
 - HTTP method: `GET`
 3. Post the new status to the server using the `$.ajax()` method
 - URL to request: `/api/statuses/update.json`
 - Data type: `json`
 - The status parameter contains the text to tweet
 - Username: `student`
 - Password: `password`
 - HTTP method: `POST`
-

Chapter 9

Plugins

- A plugin is a piece of code that adds more functionality to a software or a library.
- jQuery provides you the ability to use plugins (so you do not re-invent the wheel) and it also provides you the ability to write your own.
- In this chapter, we are going to discuss some of the popular Plugins available.

9.1 Popular Plugins

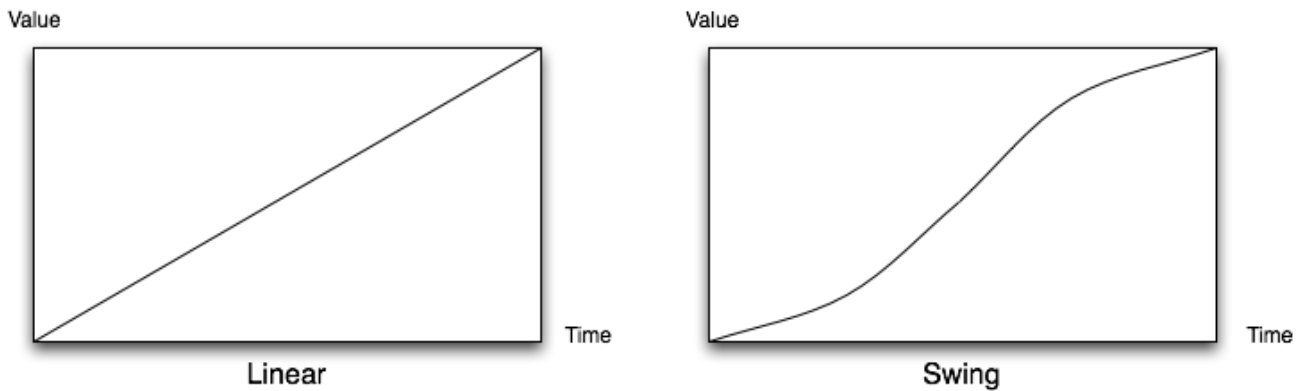
- Many times, you might consider adding more functionality to your applications, such as:
 - Slideshows
 - New animations
 - Form validation & autocompletion
 - Table enhancement (fixed headers, pagination, row editing and such)
 - Etc. . .
- Most of the time these functionalities have already been implemented by other developers in the form of a plugin.
 - There is no need to reinvent the wheel.

Note

Of course there are tons of plugins available out there on the web and obviously we cannot cover all of them.

9.2 Easing Plugin

- When we discussed effects and animations, we talked about easing which control the value of a CSS property over a certain time.
 - Acceleration & deceleration effect
 - Bouncing effect
 - Linear effect
 - And many others
 - jQuery has only two easing functions built-in: `linear` and `swing` (`swing` is the default easing function which is used if no other is specified).
-



- If you need more choice for easing functions you can use the easing plugin available at this location: <http://plugins.jquery.com/project/Easing>

Note

This plugin is also included in the jQuery UI project that we are going to discuss later

- To start using the easing plugin, download and include the JavaScript file in the HTML page:

```
<script type="text/javascript" src="jquery.easing.1.3.js"></script>
```

- Now, you will have access to more easing options. For instance:

```
$(document).ready(function() {
  $('.menu-li1').toggle(function() {
    $(this).next('.menu-ul1').slideDown('slow', 'easeOutBounce'); // ❶
  },
  function() {
    $(this).next('.menu-ul1').slideUp('slow', 'easeOutBounce'); // ❷
  });
});
```

- ❶, ❷ This will give a bouncy effect when sliding up or sliding down the side menu

Note

Of course, there are other easing functions available like `easeInCirc`, `easeInOutExpo`, `easeOutBack`, `easeOutElastic`, `easeOutBounce`, `easeInOutElastic`, etc...

9.3 Slideshow Plugins

- There are tons of plugins related to slideshow to facilitate image gallery creation.
- Of those plugins, we are going to discuss these two:
 - The ColorBox plugin (for lightbox purposes)
 - The Cycle plugin (to provide sliding of elements inside a container using special transitions)

9.4 The ColorBox Plugin

- There is a well known plugin for slideshows: ThickBox
 - Very good plugin but it is no longer maintained.
 - ColorBox is a serious challenger to ThickBox:
 - Appearance of the rendered lightbox is defined in external CSS
 - Light footprint: only 9Kb
 - MIT licensed
 - To start using ColorBox:
1. Download the archive from the website ⇒ <http://colorpowered.com/colorbox/?url=colorbox>. It contains:
 - Two versions of the plugin: a non-minified and a minified version of the library
 - Several examples and each of them contains a stylesheet along with several images that can serve as a basis for your own design (here we are just going to copy paste what we need)

2. Include the library in your HTML file:

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

3. Let us say you have a list of links like this one:

```
<ul>
  <li><a rel="foodpic" href="images/bday_cake.png">Birthday Cake</a></li>
  <li><a rel="foodpic" href="images/corn.png">Corn</a></li>
  <li><a rel="foodpic" href="images/egg.png">Egg</a></li>
  <li><a rel="foodpic" href="images/pear.png">Pear</a></li>
  <li><a rel="foodpic" href="images/thanksgiving.png">Thanksgiving Dessert</a></li>
</ul>
```

4. You can quickly start to use the ColorBox plugin like this:

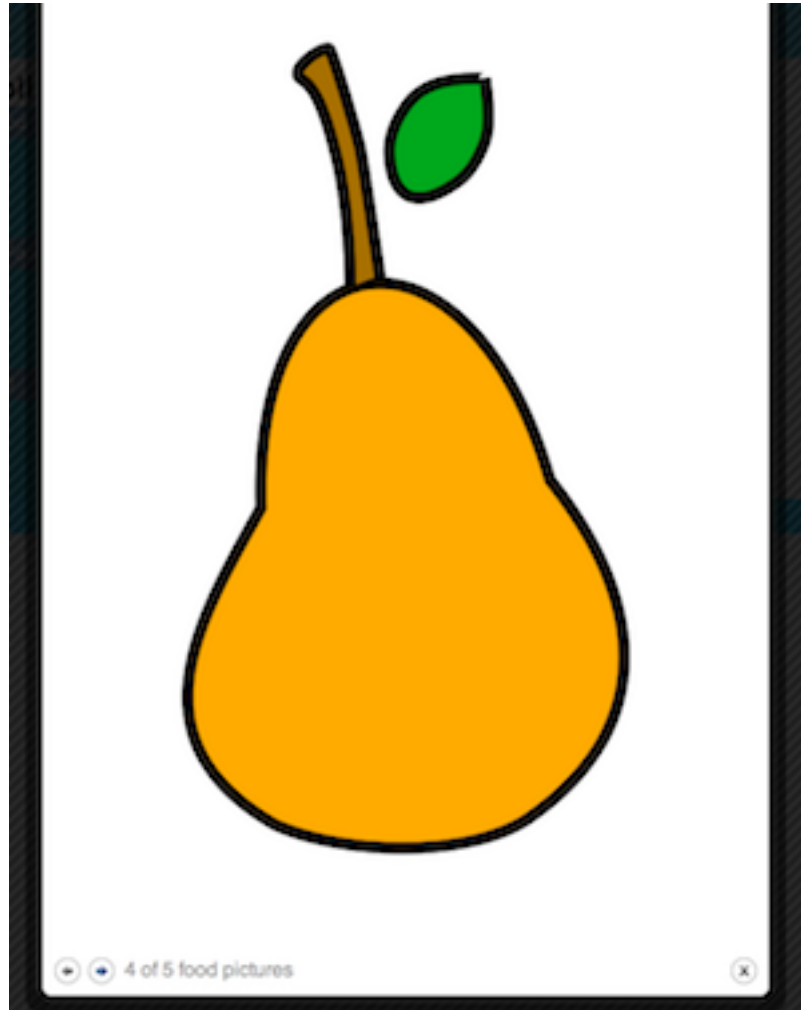
```
$( 'a[rel="foodpic"]' ).colorbox();
```

5. Of course, there are a lot of options you can specify when calling the `.colorbox()` method by passing it a JavaScript object containing those options. These options can be found at <http://colorpowered.com/colorbox/?url=colorbox>

```
$( 'a[rel="foodpic"]' ).colorbox({
  transition: 'fade', ❶
  speed: 500, ❷
  current: "{current} of {total} food pictures" ❸
});
```

- ❶ Effect to apply when changing picture
- ❷ Speed of the transition in milliseconds
- ❸ Text formatting that provides information to the user about the current picture number and the total number of pictures left

6. This produces the following result:



9.5 The Cycle Plugin

- The Cycle plugin is a plugin that can be used in conjunction with slideshow implementations to provide different types of transition effects.
- The method `.cycle()` provided by the plugin applies on a container element so each direct child of this same container becomes a "slide."
- To start using the cycle plugin, download the archive at the following address ⇒ <http://malsup.com/jquery/cycle/download.html>
- This archive contains three minified versions of the library:
 - `jquery.cycle.min.js`: basic version library containing only a slide transition
 - `jquery.cycle.all.min.js`: full featured version containing a lot of transitions (This is the one we are going to show here)
 - `jquery.cycle-lite.min.js`: stripped down version containing only the most the basic options

Note

If you need to see the source code of the library, there is also a `src` folder included in the archive

- Include the plugin in your HTML file:

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


- Here is an example of some HTML code:

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

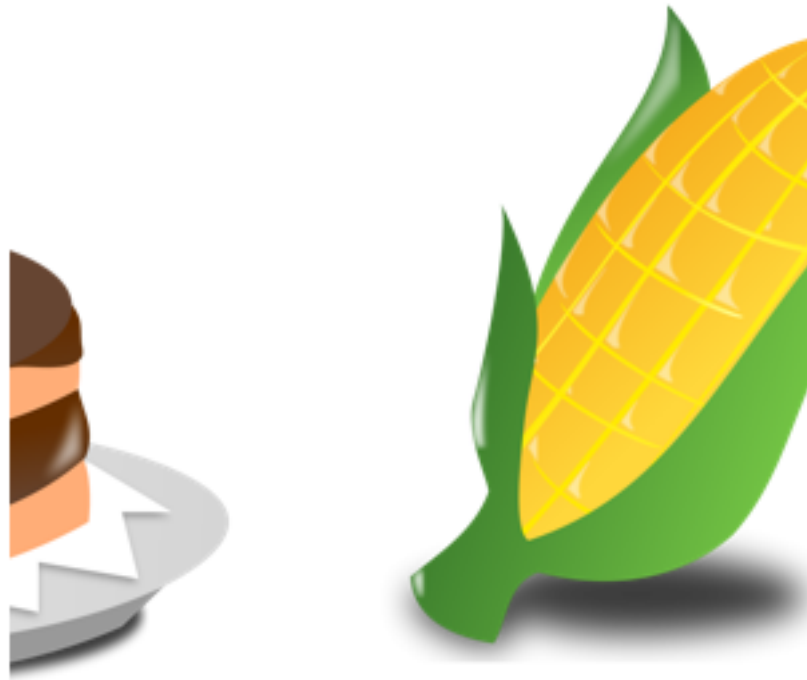
- To quickly use the cycle plugin, you could do the following:

```
$(document).ready(function() {
  $('#photos').cycle({ // ❶
    fx: 'fade'
  });
});
```

- ❶ The `.cycle()` method is called on a container element and is passed a JavaScript object containing multiple properties for our slideshow. In this case we just specify the effect to use (other effects can be seen here: <http://malsup.com/jquery/cycle/browser.html>)
- Of course, there is a list of options we can use to more precisely control the transition effects for the `.cycle()` method (<http://malsup.com/jquery/cycle/options.html>)

```
$('#photos').cycle({
  fx: 'scrollDown', ❶
  speedIn: 2500, ❷
  speedOut: 500, ❸
  timeout: 0, ❹
  next: '#photos' ❺
});
```

- ❶ This is the effect to use
 - ❷ Speed for the in transition in milliseconds (when the current slide gets in)
 - ❸ Speed for the out transition in milliseconds (when the current slide gets out)
 - ❹ Milliseconds between slide transition
 - ❺ Element to use as event trigger for next slide
- This would look like this:



9.6 Jcrop Plugin

- Another popular plugin related to images is the Jcrop plugin which allows the user to crop your images.

Note

Of course, the Jcrop plugin does not crop your image, the manipulation will still happen on the server-side. Jcrop just provides the user an intuitive way to define the bounding edges of the image

- This kind of feature is often used to crop a profile picture image.
- To start using Jcrop, download it at the following URL: http://deepliquid.com/content/Jcrop_Download.html
- The downloaded archive contains:
 - The JavaScript file containing the plugin
 - A CSS file for the plugin
 - An animated GIF that the plugin uses as a background
- Include the plugin in your HTML file:

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

- Copy/paste the CSS file along with the animated GIF in your project then include the CSS file in your HTML file

```
<link rel="stylesheet" type="text/css" href="jquery.Jcrop.css"/>
```

- We will use the following HTML code as an example:
-

```
<div id="cropsection">
   ❶
  <input type="button" value="crop"/> ❷
</div>
```

- ❶ Image where we will apply the Jcrop plugin
- ❷ This button will give us the detail of the selection

- This is how you apply the plugin to the image in the JavaScript part:

```
$(window).load(function() {
  var $image_to_crop=$('#cropsection > img').eq(0);
  var jcrop = $.Jcrop($image_to_crop, { // ❶
    setSelect: [10,10,100,100], // ❷
    minSize:[50,50] // ❸
  });
});
```

- ❶ Calling `$.Jcrop()` will return a Jcrop object so that we can later access its properties and methods to get the selected coordinates. This method has two arguments, the first one being the image on which we apply the plugin and the second one being the options for the plugin passed as a JavaScript object
- ❷, ❸ Options passed to the plugin. The detail of these options can be found here: http://deepliquid.com/content/Jcrop_Manual.html#S
- This will render the following:



- So far, if we click on the button nothing happens. We want to display the current selection as soon as the user clicks on this button

```
$('#cropsection :button').click(function() {
  var selection = jcrop.tellSelect();
  alert('selected size: ' + selection.w + 'x' + selection.h);
  alert('selected at coordinates: (' + selection.x + ', ' + selection.y + ')');
});
```

9.7 The Validation Plugin

- Form validation is very hard to do using simple JavaScript.
- Data like email addresses and urls are generally validated against regular expressions.
 - Hard to write
 - Hard to test
- The Validation plugin:
 - Was started in 2006
 - Written and maintained by Jörn Zaefferer
 - * Member of the jQuery team
 - * Lead developer on the jQuery UI team
- To start using the Validation plugin, go to <http://bassistance.de/jquery-plugins/jquery-plugin-validation/>
- Include the library in your HTML:

```
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery.validate/1.7/ ↵  
jquery.validate.js"></script> ❶
```

- ❶ In this case we let Microsoft host the plugin for us but you could also download it and host it along with your web site
- Here is an example of an HTML form:

```
$(document).ready(function() {  
    $('#signup form').validate({ // ❶  
        rules: {  
            name: {  
                required: true  
            },  
            email: {  
                required: true,  
                email: true  
            },  
            url: {  
                url: true  
            },  
            password: {  
                required: true,  
                minlength: 6  
            },  
            passconf: {  
                equalTo: "#password"  
            }  
        },  
        success: function(label) {  
            label.text('OK!').css('color', 'green');  
        }  
    });  
});
```

- ❶ The `.validate()` method is applied to a form. We need to provide an object to this method containing validation rules and handlers such as the `success` handler that will add a label element next to each input as soon as they are valid. Complete documentation for this plugin can be found here: <http://docs.jquery.com/Plugins/Validation>

9.8 DataTable Plugin

- Using only jQuery, we have seen that you have the ability to do a lot with a selection.
- Regarding tables, you could imagine the following functionalities:
 - Fixed tables headers
 - Repeated table headers
 - Pagination
 - Inline editing
 - Sort by columns
- All of this of course would take time to implement and this is not easy at all.
- As you might guess, someone might already have done this before: the DataTable plugin (<http://www.datatables.net/index>).
- The DataTable plugin:
 - Your HTML tables become fully functional data grids (pagination, column sorting, searching, ajax loading, etc)
 - Free and open source
 - Fully internationalisable
 - Supports jQuery UI ThemeRoller (will be discussed in jQuery UI)
 - Tested through 1400+ unit tests

9.9 Plugins: Labs

For these labs, start with the file `labs/yamba8.html`. It includes a `<script>` element to load a file named `plugins.js`, which you should edit in the `labs` directory.

1. Turn the list of tweets (`<ol id="statuses">`) into a sort of slideshow using the Cycle plugin.
2. Use the Validation plugin on our registration form using the following rules:
 - All fields are required
 - Email address must be a valid address
 - Password must be at least 6 characters
 - Password confirmation must be equal to the password

Chapter 10

jQuery UI

10.1 What Is JQuery UI?

- jQuery UI is a library built on top of jQuery.
- It provides advanced:
 - User-interface widgets (date picker, slider, progress bar, etc. . .)
 - Effects (color animation, shaking effect, explode effect, etc. . .)
 - Interactions (drag & drop, sorting, etc. . .)
- jQuery UI is managed by and is part of the "jQuery Project" ⇒ <http://jquery.org/>

10.2 Getting Started

- To start using jQuery UI, you need to download the library first.
- From the jQuery UI homepage, you have many choices:
 - Download the whole library and host it along with your web site.

Note

You can also use the version host by Google, accessible at this location: <https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.10/jquery-ui.min.js>

- Or you can pick which features you need from jQuery UI by clicking on "Build custom download".
- If you are using the Google hosted jQuery library, you need to download a theme (CSS file along with images) to apply on the components you are going to use soon.
- From this page <http://jqueryui.com/themeroller/>, you can:
 - Create your own theme using the ThemeRoller
 - Choose an existing theme, ready to be used
- Finally, you need to include the CSS and the library in the HTML file.

```
<link href="jquery-ui-1.8.10.custom.css" type="text/css" rel="stylesheet"/>
<script src="libs/jquery-ui-1.8.10.custom.min.js"></script>
```

Note

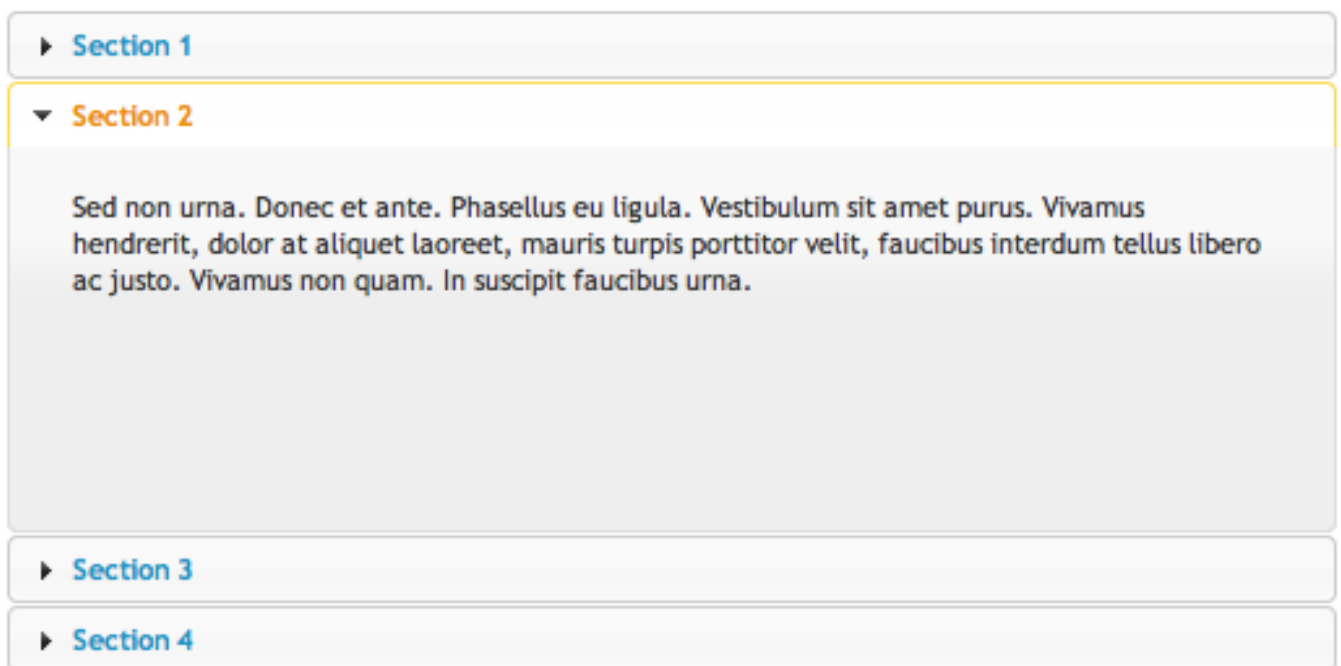
Do not forget to include the `images` folder included with the theme, this folder and the CSS should be contained in the same folder.

10.3 JQuery Widgets

- jQuery UI provides you a lot of cool UI widgets that:
 - Enhance navigation (such as accordion and tabs)
 - Add some forms controls (such as autocomplete, special button, datepicker and slider)
 - Other controls (such as progress bar and dialogs)

10.4 The Accordion Widget

- In your HTML page you might have a part representing a list of "headers" along with their related section of information.
- The Accordion widget turns this part of the page into an accordion widget where only one section can be seen at a time:



- Of course, you could implement your own accordion using the core jQuery library, however the Accordion widget gives you a lot of options, including:
 - Changing icons
 - Choosing event that trigger section opening
 - Settings sections height to the height of the biggest section or not
 - Etc...
 - This widget has some HTML semantics requirements.
 - The container that is being turned into an accordion must contain pairs of "headers" along with their sections like this:
-

```

<div id="accordion">
  <h3><a href="#">First header</a></h3> ❶
  <div>First content</div> ❷
  <h3><a href="#">Second header</a></h3> ❸
  <div>Second content</div> ❹
</div>

```

- ❶, ❷ A "header" along with its section of information
 - ❸, ❹ Another "header" and information pair
- As soon as you have the right markup, you are ready to apply the accordion to the main element containing header/section pairs (here in this example: `<div id="accordion">`):

```

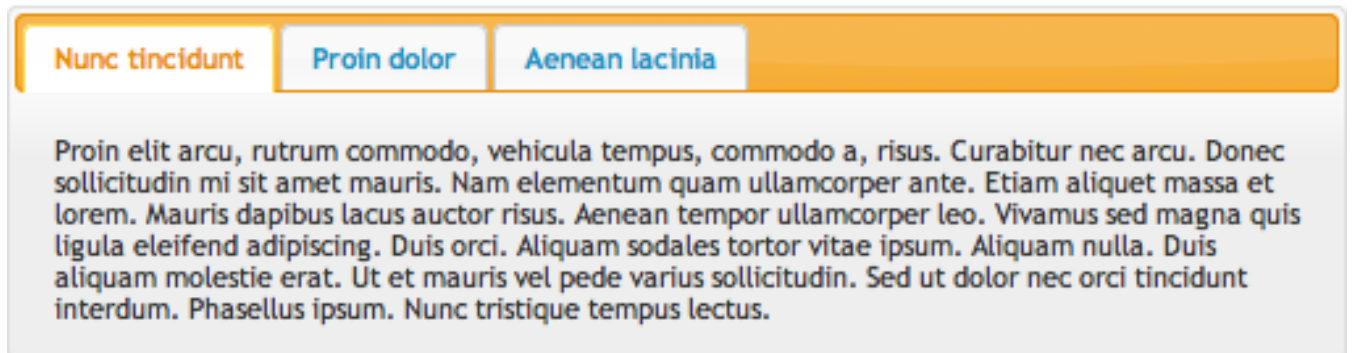
$('#accordion').accordion({ header:'h3' }); // ❶

```

- ❶ The `accordion` method takes an object as a parameter. This component is flexible and you can specify many options such as the `header` option that tells the accordion widget to consider `<h3>` elements as headers

10.5 The Tabs Widget

- Tabs are used to separate the content in different sections while saving space on the page, and allows users to easily switch from one section of content to another.



- Using specific semantics on the HTML side, you can easily turn parts of your page into tabs.
- Let us take a look at this example:

```

<div id="tabs">
  <ul> ❶
    <li><a href="#tabs-1">First</a></li> ❷
    <li><a href="#tabs-2">Second</a></li> ❸
    <li><a href="#tabs-3">Third</a></li> ❹
  </ul>
  <div id="tabs-1"> ❺
    Lorem ipsum dolor....
  </div>
  <div id="tabs-2"> ❻
    Phasellus mattis tincidunt...
  </div>
  <div id="tabs-3"> ❼
    Nam dui erat...
  </div>
</div>

```


- ❶ This is the element containing the tabs
 - ❷, ❸, ❹ Each tab has an anchor to its corresponding element (the `<div>` in this example)
 - ❺, ❻, ❽ These elements represent the content for each tab
- Now on the JavaScript side, you would just have to call the `.tabs()` method on the element containing the tabs and content:

```
$('#tabs').tabs();
```

- There are a lot of options for this Tab widget that can be found at <http://jqueryui.com/demos/tabs/#options>
 - These options would be passed to the Tabs widget like this:

```
$('#tabs').tabs({
  option1: 'value',
  option2: 'value'
});
```

- The following are the events that you can use to listen on the Tabs widget:
 - `tabselect`, `tabload` and `tabshow` (fired in this exact same order)
 - `tabsadd` and `tabsremove`
 - `tabsenable` and `tabdisable`

Note

Event listeners can directly be defined when creating the tab element. For instance for the `tabselect` event:

```
$('#tabs').tabs({
  select: function(event, ui) { ... } // ❶
});
```

- ❶ The `ui` object contains several properties:
 - `ui.tab` for the anchor element representing the clicked tab
 - `ui.panel` representing the element containing the content of the selected tab
 - `ui.index` representing the index of the clicked tab (zero based index)
- Tabs can also load content dynamically using Ajax.
 - Tab links that have an `href` attribute with a relative URL to resources on the server will dynamically load the data in the content panel when clicked.

Note

Of course, do not forget about the same origin policy. In order to dynamically load content into your tabs, the page and the resources it is trying to access must be on the same server.

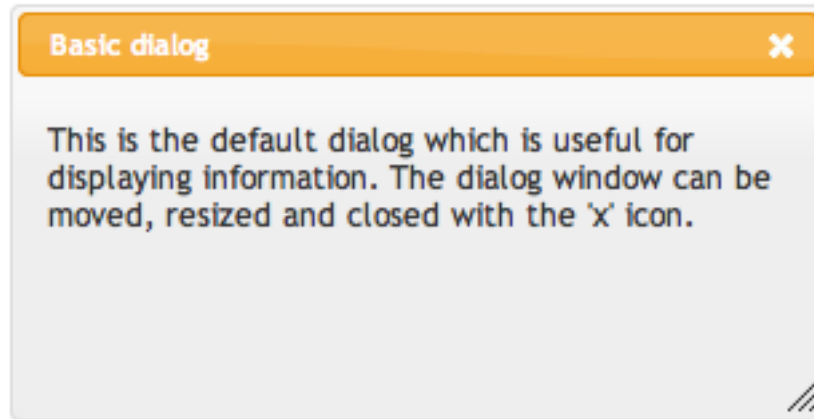
10.6 Dialogs

- jQuery UI provides you with the ability to easily create dialogs (modal and non-modal dialogs).

Note

The difference between a modal and a non-modal dialog is that a modal dialog prevents the user from interacting with the rest of the page until it is closed.

- A non-modal dialog looks like this in jQuery UI:
-



- A dialog is created from a `<div>` element having a `title` attribute representing the title of the window.

```
<div id="dialog" title="Basic dialog">
  <p>Welcome on Yamba</p>
  <p>Yet Another Micro-blogging App</p>
</div>
```

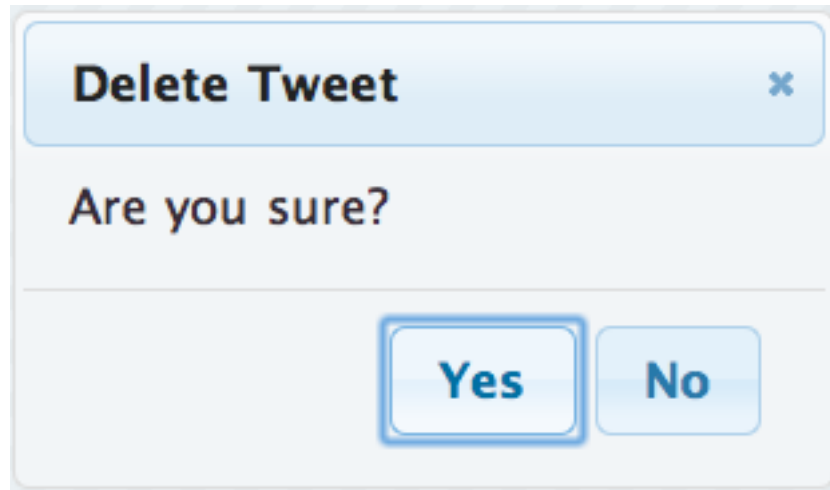
- Next, using jQuery UI, we can quickly turn this `<div>` element into a basic non-modal dialog:

```
$( '#dialog' ).dialog(); // ❶
```

- ❶ The fastest way to create a new dialog. Of course, we can specify many options (such as height, width, if the dialog is modal or not, the position of the dialog, etc...) as a form of object ⇒ <http://jqueryui.com/demos/dialog/#options>
- Another type of dialog that is often use is the modal dialog, which is used to get confirmation from the user before performing some action.
- This example will show you how to create a modal confirmation dialog using the following HTML markup:

```
$( "#confirm" ).dialog({
  title: "Delete Tweet", // ❶
  resizable: false, // ❷
  modal: true, // ❸
  buttons: { // ❹
    "Yes": function() {
      $(this).dialog("close"); // ❺
      alert("You clicked on yes");
    },
    "No": function() {
      $(this).dialog("close");
      alert("You clicked on no");
    }
  }
});
```

- ❶ This is the title of the dialog
- ❷ This states that the dialog cannot be resized
- ❸ This states that the dialog is modal
- ❹ This defines all of the buttons along with their text and handlers
- ❺ This closes the dialog
- This would produce the following result:



10.7 JQuery UI Controls

- jQuery UI comes with new controls that can be used in your form, such as:
 - The date picker
 - The slider
 - The progress bar
 - Etc. . .
- The drawbacks of creating these controls yourself:
 - A lot of time required
 - Error prone

10.8 The Datepicker

- The Datepicker is a component that allows you to display a calendar, allowing the user to choose a date.
- The Datepicker:
 - Is tied to a standard form input field
 - Is opened when the user focuses on this same input (click or tab)
 - Returns the date to the input field as soon as the user chooses a date



- In the HTML code, create a normal input:

```
<input type="text" id="datepicker">
```

- Then on the JavaScript side using jQuery UI, you would do the following:

```
$("#datepicker").datepicker();
```

- This date picker is highly customizable. You can:

- Format the date shown to the user in the input
- Restrict the date range
- Use different languages for your calendar
- Directly embed the calendar on the page
- Display month and year in a list
- Display multiple months at a time
- Etc...

Note

More details on <http://jqueryui.com/demos/datepicker/>

10.9 The Slider

- The jQuery UI `.slider()` method turns a `<div>` element into a slider:



Figure 10.1: title

- On the HTML side, you would create an empty `<div>` element with an id like this:

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

- On the JavaScript side, you would have to apply the `.slider()` method to the `<div>` like this:
-

```
$( "#slider" ).slider();
```

- There are of course a lot of options that you can configure here, such as having multiple handles and ranges, specifying the range, etc...

Note

Those handles can be moved with the mouse as well as with the direction key from the keyboard

- There are also some events that you can use to listen. Event handlers receive two parameters:
 - The original browser event
 - The prepared UI object

Note

More information on the slider ⇒ <http://jqueryui.com/demos/slider/>

10.10 The Progress Bar

- The progress bar is a good visual way to show the state of completion for a process (percentage of completion):



- To create a progress bar, you need to first create the HTML structure like this:

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

- Using jQuery UI and the `.progressbar()` method, you can quickly turn this structure into a jQuery UI widget:

```
$( "#progressbar" ).progressbar({  
  value: 50  
});
```

- Like any other widget, there are options and events that you can listen for. Details are available at <http://jqueryui.com/demos/progressbar/>

10.11 JQuery UI Interactions

- jQuery UI also allows you to define "behaviors" on elements.
 - Using jQuery UI you can make your elements:
 - Draggable
 - Droppable
 - Resizable
 - Selectable
 - Sortable
-

10.12 Drag & Drop

- Drag & drop allows you to make any element "draggable" on to other elements that are "droppable".
- This is what you can do with a "draggable" element (An element that a user moves):
 - Constrain the movement (like dragging vertically or horizontally)
 - Delay the dragging movement depending on the dragging distance or time in order to avoid accidental dragging
 - Fine-grained snapping of the dragged element
 - Possibility to revert the dragging movement on a certain condition (if not dropped at the right place for instance)
 - Provide a visual helper to the user when dragging an element
 - Position the mouse where you want in the dragged element
- This is what you can do with a "droppable" element (an element that "receives" the "draggable" element):
 - Choose accepted elements
 - Provide feedback when the user drags and/or drops and element on the "droppable" location
- You can also listen for events:
 - "draggable" events: `create`, `start`, `drag`, `stop`
 - "droppable" events: `create`, `activate`, `deactivate`, `over`, `out`, `drop`

Note

All of these events receive two arguments: The original browser event and a prepared UI object.

For more information on drag and drop ⇒ <http://jqueryui.com/demos/draggable/> and <http://jqueryui.com/demos/droppable/>

10.13 Drag & Drop: Example

- Let us see an example on how to do drag & drop using jQuery UI:
- This will be our HTML:

```
<div class="draggable"> ❶  
  <p>An element to drag on the page</p>  
</div>  
  
<div class="draggable"> ❷  
  <p>An element to drag on the page</p>  
</div>  
  
<div id="droppable"> ❸  
  <p>Drop element here</p>  
</div>
```

- ❶, ❷ These elements are the one that will be dragged on the page
 - ❸ This element will be the one that will be receiving the draggable element. This is the droppable element
- Time to add draggable and droppable interactions to these elements:
-

```

$('.draggable').draggable({ // ❶
  distance:20, // ❷
  opacity:0.7, // ❸
  zIndex:1000, // ❹
  revert:'invalid' // ❺
});

//Make the trash being droppable
$('#droppable').droppable({ // ❻
  accept:'.draggable', // ❼
  activeClass:'ui-state-hover', // ❽
  hoverClass:'ui-state-active', // ❾
  tolerance:'pointer', // ❿
  drop: function(event,ui) { // ⓫
    var $element = ui.draggable; // ⓬
    $element.fadeOut('slow',function() {
      $(this).remove();
    });
  }
});

```

- ❶ This will turn your elements with the class `draggable` into draggable elements
- ❷ `distance` specifies the distance in pixels that the user has to move before the drag starts
- ❸ `opacity` sets the opacity of the dragged element
- ❹ `zIndex` brings the element to the foreground or the background of the page depending on its value
- ❺ `revert` allows you to choose if you want to revert the dragged element to its initial position when dragging stops on a valid position or not
- ❻ This will turn your element with the id `droppable` into a droppable element
- ❼, ❽ `acceptClass` sets the class to use for the droppable when a user starts to drag an element
- ❾ `hoverClass` sets the class to use for the droppable element when a user drags an element over the droppable element
- ❿ `tolerance` allows you to specify what mode is used to determine whether or not a draggable is "over" a droppable.
- ⓫ `drop` allows you to specify the function to call when a draggable has been dropped over a droppable
- ⓬ `ui.draggable` refers to the draggable element

Note

The `.draggable()` method does not apply to elements that are created dynamically. You would need to use event delegation on the `mouseenter` event and then enable the draggable behavior on the clicked element

10.14 Sortable elements

- jQuery enables you to make your elements sortable using the `sortable` method.
- Take for example, an HTML list like this one:

```

<ul id="sortable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>

```

- Next, to make the list items sortable, you would have to use the `.sortable()` method:
-

```
$('#sortable').sortable(); // ❶
```

- ❶ Of course this method can take several options passed as a JavaScript object. Those options are available here ⇒ <http://jqueryui.com/demos/sortable/#options>. Sortable items also share draggable properties

10.15 Selectable Elements

- Using jQuery UI, you can enable an element or a group of elements to be selectable.
 - You can click on them to select them
 - Ctrl/meta key plus click and/or drag also works for selectable
- Imagine an HTML list like this one:

```
<ul id="selectable">  
  <li>Item 1</li>  
  <li>Item 2</li>  
  <li>Item 3</li>  
  <li>Item 4</li>  
</ul>
```

- Then to make the list items selectable, you would have to use the `.selectable()` method:

```
$('#selectable').selectable(); // ❶
```

- ❶ Of course this method can take several options passed as a JavaScript object. Those options are available here ⇒ <http://jqueryui.com/demos/selectable/#options>

10.16 Resizable Elements

- jQuery provides you the ability to resize any DOM element by calling the `.resizable()` method on this element. Of course, you can specify more options:
 - Preserve the aspect ratio
 - Set the minimum/maximum width and height
 - Provide visual feedback using a transparent ghost of the element being resized
 - Synchronize resizing between elements
 - Animate the element on resizing etc...
- There are also events that you can listen for, such as `create`, `start`, `resize` and `stop`
- Let us take the following HTML structure as an example:

```
<div id="resizable">  
  <!-- Content of the div -->  
</div>
```

- Using jQuery, you can make this element to be resizable:

```
$('#resizable').resizable(); // ❶
```

- ❶ This makes the element resizable. For more options and events ⇒ <http://jqueryui.com/demos/resizable>

10.17 jQuery UI animation

- jQuery UI adds a lot more functionality for animations. It adds:
 - New effects (<http://jqueryui.com/demos/effect/#default>)
 - New easing methods (<http://jqueryui.com/demos/effect/#easing>)
 - The `.effect()` method that applies an effect on an element without the show/hide logic
 - The `.switchClass()` method that replaces an existing class with a new one (with the possibility of animation if the duration is specified)
- jQuery UI also extends existing methods such as:
 - The `.animate()` method to support color animation when animating the following properties: `backgroundColor`, `borderBottomColor`, `borderLeftColor`, `borderRightColor`, `borderTopColor`, `color` and `outlineColor`
 - The `.addClass()`, `.removeClass()` and `.toggleClass()` methods now support class animation by adding a second parameter (the time of the animation)
 - The `.toggle()`, `.show()` and `.hide()` methods have been enhanced to accept the jQuery UI advanced effects

10.18 jQuery UI: Labs

For these labs, start with the file `labs/yamba9.html`. It includes a `<script>` element to load a file named `jquery_ui.js`, which you should edit in the `labs` directory.

1. Transform the side menu into an accordion using jQuery UI.
2. Transform the `<div id="tweets">` into tab navigation using jQuery UI.
3. Improve the tab navigation so when you click on "Public timeline", public statuses will be dynamically loaded using the `.getJSON()` method.

- URL to get public statuses: `http://yamba.marakana.com/api/statuses/public_timeline.json?callback=`
- Remember that you need to first catch which tab has been selected. For that supply a callback function to handle the `select` event as an `init` option like this:

```
<your selection>.tabs({
  select: function(event, ui) {
    //Make your JSONP call and display result
  }
});
```

4. Display the `<div id="dialog">` as a jQuery simple dialog when the page is ready.
5. Display the `<div id="confirm">` as a jQuery modal confirmation dialog when the user chooses to delete a tweet.
 - This modal dialog must have two buttons: "Yes" and "No"
 - If the user clicks on "Yes", close the dialog and remove the tweet (this is a simulation, do not make any AJAX calls) from the page using a `fadeOut` effect.
 - If the user clicks on "No", just close the dialog.
 - This dialog will have the title: "Delete Tweet".
 - This dialog must not be resizable.
6. It would be nice to drag our tweets in to a "trash" area. When our tweets are dragged over the trash, we will fade them out and remove them from the DOM. Here are the requirements, using jQuery UI:

- Our draggable elements will be all `<li class="status">` elements:
 - Do not start dragging unless we have tried to drag the element on a distance of 20px.
 - When dragged, the opacity of our tweet has to be 0.7.
 - To make sure that the tweet will be displayed in the foreground, give it a high `zIndex`, like 1000.
 - When the tweet is dropped on an invalid place, make sure it is reverted to its original position.
 - Our droppable element will be the `<div id="trash">` element.
 - This droppable can only accept our tweets and nothing else.
 - In terms of styling, the active class will be `"ui-state-hover"` and the hover class will be `"ui-state-active"`.
 - We consider each draggable to be "over" a droppable when the mouse is over the droppable.
-

Chapter 11

Advanced Concepts

11.1 Best Practices: Loops

- Cache the length property in loops:

```
var arrayLength = myArray.length;

for (var i = 0; i < arrayLength; i++) {
    //DO SOMETHING
}
```

- Append content outside of loops

```
//BAD
$.each(myArray, function(index, item) {
    var newTweet = '<li>' + item + '</li>';
    $('#statuses').append(newTweet);
});

//GOOD
var newTweet = '';

$.each(myArray, function(index, item) {
    newTweet += '<li>' + item + '</li>';
});

$('#statuses').html(newTweet);
```

11.2 Best Practices: Avoid Anonymous Functions

- Anonymous functions are often used when binding event handlers.
- However, anonymous functions are difficult to debug, maintain, test, or reuse.
- It's better to use object literals:

```
//BAD
$(document).ready(function() {
    $('.status .delete').click(function(e) {
        //Do something
    });
});
```

```
$('.status').dblclick(function(e) {
    //Do the same thing
});
});

//GOOD
var bindingObject = {
    onReady : function() {
        $('.status .delete').click(bindingObject.deleteTweet);
        $('.status').dblclick(bindingObject.deleteTweet);
    },
    deleteTweet : function(e) {
        //Do something
    }
};

$(document).ready(bindingObject.onReady);
```

11.3 Best Practices: Optimizing Selectors

- Here are some tips to keep in mind when using selectors:
 - Id selections are really fast. They are handled by `document.getElementById()`

```
// This is fast
$('#statuses .status');
```

```
//This is faster
$('#statuses').find('.status');
```

- Sizzle, the jQuery selector engine works bottom-up (right to left). Be more specific on the right side of your selector rather than on the left

```
//GOOD
$('li.status .delete');
```

```
//BETTER
$('.status img.delete');
```

11.4 JQuery Utility Methods

- Most of the jQuery methods act on selections. Those methods are in the `$.fn` namespace (also called the jQuery prototype).
 - Receive and return the selection as this.
- The other jQuery methods do not act on selections and are in the `$` namespace.

Note

You need to be aware that in some cases, while these two namespaces might have the same method name, they are in fact different methods (for instance, `$.fn.each` to iterate over each selected element in a selection and `$.each` to iterate over an array or an object).

- The utility methods that jQuery provides are very useful for doing routine programming tasks.
 - Here are some common jQuery utility methods. For more information, visit <http://api.jquery.com/category/utilities/>
-

- `$.trim()`: This method removes leading and trailing whitespaces

```
$.trim('  lots of extra whitespace  '); // ❶
```

- ❶ Will return *lots of extra whitespace*

- `$.each()`: This method iterates over objects and arrays

```
$.each(['apple','banana','peach'], function(index, value) {
  console.log('index: ' + index + ' value: ' + value);
});

$.each({ firstname : 'laurent', lastname : 'tonon' }, function(key, value) {
  console.log(key + ' : ' + value);
});
```

- `$.inArray()`: This method checks if a value is contained in an array. If found it returns the value's index of the array. Returns -1 if the value is not found

```
var myArray = [ 23, 46, 31, 12 ];
if ($.inArray(31, myArray) !== -1) { // ❶
  console.log('found it!');
}
```

- ❶ This returns 2

- `$.extend()`: This method merges the properties of the first passed object using the properties of the other passed objects as arguments. The first object is modified and returned by the method

```
var myFirstObject = { fruit : 'apple', vegetable: 'carrot' };
var mySecondObject = { fruit : 'banana' };
var resultingObject = $.extend(myFirstObject, mySecondObject); // ❶
console.log(myFirstObject.fruit); // ❷
console.log(resultingObject.fruit); // ❸
```

- ❶ Modifies the first object and returns the modified object
- ❷, ❸ Both print *banana*

- jQuery also provides methods to determine the type of a variable (`$.isFunction()`, `$.isPlainObject()`, `$.isArray()`)

11.5 Avoiding Conflict With Other JavaScript Libraries

- As we have seen, jQuery makes extensive use of the `$` variable.
- However, other JavaScript libraries like prototype make use of the `$` variable sign as well, and might create conflicts.
- Of course, you could use the `jQuery` variable instead (the `$` variable is an alias to the `jQuery` variable) but it is obviously less convenient to type this in every instance.
- The following examples are the same:

```
//Using the $ variable
$('.status').show();

//Using the jQuery variable
jQuery('.status').show();
```

- To avoid conflicts, you can put jQuery in no-conflict mode as soon as it is loaded in the page using the `.noConflict()` method:

```
<script src="prototype.js"></script>
<script src="jquery.js"></script>
<script>var $j = jQuery.noConflict();</script> ❶
```

- ❶ Create a new alias to jQuery: `$j`

11.6 Queuing And Dequeuing Animations

- Animations performed on an element are asynchronous.
- Consider this example where we would have a `<div>` to animate:

```
$('#button').click(function () {
  $('#div').animate({left:'+=200px'}, 2000);
  $('#div').animate({top:'0px'}, 600);
  $('#div').css('backgroundColor','red'); // ❶
  $('#div').animate({left:'10px', top:'30px'}, 700);
});
```

- ❶ One might guess that changing the background color would happen after the previous two animations have finished. However, this is not the case. The background would change almost at the same time as the first animations starts.

- Each element has its own animation queue, `fx` (default queue), which is an array of functions.
- Those queues are FIFO (First In First Out).
- Using the following animation queue function, you are in total control of your animations:
 - You can add a function to the queue using `.queue()` (putting the function at the end of the stack)
 - Calling `.dequeue()` removes the top function from the stack and executes it
- You could place your background changing operation on the `fx` stack like this:

```
$('#button').click(function () {
  $('#div').animate({left:'+=200px'}, 2000);
  $('#div').animate({top:'0px'}, 600);
  $('#div').queue(function(){ // ❶
    $(this).css('backgroundColor','red');
    $(this).dequeue(); // ❷
  });
  $('#div').animate({left:'10px', top:'30px'}, 700);
});
```

- ❶ Adds a function in the animation stack to be executed
- ❷ This allows the sequence of animation to continue. The next function in the queue is removed from the queue and then executed

- You can get the animation queue of an element by calling `.queue()` on the matched selection. It would return an array of functions.
- You could watch the state of the stack like this:

```
setInterval(function(){
  $('#span').html('Current Animation Queue Length for the div is ' + $('#div').queue('fx').length);
}, 100);
```