jQuery Rich Internet Applications

Text: EN

Software: EN

REALDOLMEN
to get there, together

Table of contents

1. Developing Rich Internet Applications using jQuery
    1.1. Goal
    1.2. Agenda
2. Introduction
    2.1. What is jQuery?
    2.2. Advantages of jQuery
    2.3. Downloading and Installing
    2.4. Preparing Your Pages
    2.5. Starting with jQuery
    2.6. jQuery and HTML
    2.7. jQuery vs. JavaScript
    2.8. Selecting Elements
    2.9. Your first Effects
3. Selecting
    3.1. The DOM
    3.2. The jQuery Alias $()
    3.3. CSS Selectors
    3.4. Other Selectors
    3.5. Attribute Selectors
    3.6. Custom Selectors
    3.7. Form Selectors
    3.8. Traversing the DOM
    3.9. Accessing DOM Elements
    3.10. Method Chaining
4. Decorating
    4.1. CSS with jQuery
    4.2. Reading CSS Properties
    4.3. Setting CSS Properties
    4.4. Adding and Removing Classes
5. Events and Functions
    5.1. Using Events
    5.2. Binding an Event
    5.3. Event Triggers
    5.4. Triggering Events
    5.5. The Event Object
    5.6. Removing Events
    5.7. Stopping an Event
    5.8. What does the following code do?
    5.9. Stopping Default Actions
    5.10. Using Event Bubbling
    5.11. Separating Scripts
    5.12. Creating Functions
    5.13. Passing Variables and Returning Values
    5.14. jQuery Loops
    5.15. Using Conditional Logic
6. Modifying the DOM
    6.1. DOM Traversal Revisited
    6.2. Removing and Detaching
    6.3. jQuery Arrays
    6.4. Replacing Elements
    6.5. Other Filter Methods

- 6.6. Wrapping Elements
- 7. Adding Effects
  - 7.1. Hiding and Revealing Elements
  - 7.2. Progressive Enhancement
  - 7.3. Adding New Elements
  - 7.4. Removing Elements
  - 7.5. Modifying Content
  - 7.6. Basic Animation
  - 7.7. Callback Functions
- 8. Animating
  - 8.1. Animating CSS Properties
  - 8.2. Color Animation
  - 8.3. Easing
  - 8.4. Bouncing
  - 8.5. The Animation Queue
  - 8.6. Chaining Actions
  - 8.7. Animated Navigation
  - 8.8. Using jQuery UI
- 9. Scrolling
  - 9.1. Scroll Event
  - 9.2. Floating Navigation
  - 9.3. Scrolling the Document
  - 9.4. Custom Scroll Bars
- 10. Resizing
  - 10.1. Resize Event
  - 10.2. Layout Switcher
  - 10.3. Resizable Elements
  - 10.4. Pane Splitter
- 11. Images and Slideshows
  - 11.1. Images and jQuery
  - 11.2. Custom Lightbox
  - 11.3. Slideshow
- 12. Navigating and Organizing Content
  - 12.1. Creating Rich Internet Applications (RIA)
  - 12.2. Menus
  - 12.3. Accordion Menus
  - 12.4. Tabs
  - 12.5. Tooltips
- 13. Code Practices
  - 13.1. Cleaner jQuery
  - 13.2. Client-side Templating
  - 13.3. Feature Detection
- 14. Adding Ajax
  - 14.1. What is Ajax?
  - 14.2. Loading Remote HTML
  - 14.3. Fetching JSON Data
  - 14.4. Ajax with jQuery
  - 14.5. Loading External Scripts
  - 14.6. GET and POST Requests
  - 14.7. jQuery Ajax Events
  - 14.8. Sending Form Data
- 15. Enhancing Forms
  - 15.1. Form Validation
  - 15.2. Using the Validation Plugin

15.3. Form Hints
15.4. Inline Editing
15.5. Autocomplete
15.6. Form Controls: Date Picker
15.7. Form Controls: Slider
15.8. Form Controls: Progress Bar
15.9. Dialogs and Notifications
16. Advanced Controls
16.1. Advanced Controls
16.2. Lists
16.3. Trees
16.4. Tables
17. Extending jQuery
17.1. Plugins
17.2. Adding Methods
17.3. Themes
17.4. Other Extensions
17.5. Next Steps
18. Appendix
18.1. References
18.2. Development Tools

1. Developing Rich Internet Applications using jQuery

1.1. Goal

Learn the fundamentals of jQuery to create compelling, interactive web applications

1.2. Agenda

- Introduction
- Selecting
- Decorating
- Events and Functions
- Modifying the DOM
- Adding Effects
- Animating
- Scrolling
- Resizing
- Images and Slideshows
- Navigating and Organizing Content
- Code Practices
- Adding Ajax
- Enhancing Forms
- Advanced Controls
- Extending jQuery
- Appendix

2. Introduction

2.1. What is jQuery?

- jQuery is a JavaScript Library
  - Allows you to change web pages on the fly
  - Can let you do in 3 lines of code what you would need 15 lines of JavaScript for
  - Enables you to work easily with Ajax
  - jQuery motto: "Write Less. Do More"
  - Has a robust, cross-platform compatibility
- jQuery has an active community
  - Hundreds of plugins have been created that extend its functionality
- jQuery exists in two versions, each a single compact file
  - Production
    - Smaller file size optimized for speed of execution
  - Development
    - To see behind the magic and for debugging

2.2. Advantages of jQuery

- HTML and CSS let you create great-looking, but static web pages
  - They give your pages structure and style
  - They tell the browser how the page is displayed
- To make the web pages more responsive you need
  - Interactivity
  - Action
  - Animation
  - Interaction
  - All kinds of effects
- You could load a new page for each action, but... real interactivity should be done without reloading
- To add behaviour to the web page, you need a scripting language, such as JavaScript
  - Every browser comes with a JavaScript interpreter
  - Follows the directions in the `<script></script>` tags
- jQuery is a specialised library that eases JavaScript development
- jQuery allows you to
  - Dynamically add elements to a web page without reloading
  - Change menu items when users hover their mouse over them
  - Validate form fields and notifying the user
  - Add animations and transitions to text and pictures
  - Load data from a server asynchronously, when needed
  - and much more!
- jQuery is executed on the client side

2.3. Downloading and Installing

- You can download jQuery from the website
  - http://www.jquery.com
  - Save it in a folder "scripts"

- Browsers may behave differently when executing jQuery
  - You will need to test jQuery code in multiple browsers
  - You want your pages to work well for most of your users

2.4. Preparing Your Pages

- You can load jQuery by adding the following line at the end of your page

```
<script src="scripts/jquery-3.1.1.min.js"></script>
```

- You can now start to add jQuery code inside a `<script>`

```
jQuery(function() { // ready event
    jQuery("#button1").click( function() { // click event
        $("#div").animate({top:30},200); // moves up
    });
    jQuery("#button2").click( function() {
        $("#div").animate({top:500},2000); // moves down
    });
    jQuery("#button3").click( function() {
        $("#div").css("color", "purple"); // changes color
    });
    jQuery("#button4").click( function() {
        $("#div").toggle("slow"); // hides / shows
    }); });
```

- While the browser displays changes, the original HTML and CSS will not change

2.5. Starting with jQuery

```
<script>
jQuery(document).ready(function(){
    jQuery("button").click(function(){
        jQuery("h1").hide("slow");
        jQuery("h2").show("fast");
        jQuery("img").slideUp();
    });
});
</script>
```

- The code above shows how concise jQuery can be
  - When the document is ready
  - When any button element is clicked on
  - Hide all h1 elements slowly
  - Show all h2 elements quickly
  - Let all img elements slide upward and disappear
- Accessing elements in a document

```
jQuery('div.content').find('p');
```

- Adding CSS classes dynamically

```
jQuery('ul > li:first').addClass('active');
```

- Alter the content of a document

```
jQuery('#container').append('<a href="more.html">more</a>');
```

- Respond to a user's interaction

```
jQuery('button.show-details').click(function() {
    $('div.details').show();
});
```

- Retrieve information from the server with Ajax

```
jQuery('div.details').load('more.html #content');
```

2.6. jQuery and HTML

- A browser uses the HTML structure in a file to build a document
  - Uses the HTML Document Object Model (DOM)
- The DOM can be visualised as a tree
  - Has a root
  - Each part of the tree is called a node
  - All HTML tags are called elements
  - Text nodes are not elements, they just contain text
- The browser then renders the page in its viewport
- The JavaScript interpreter will reference the DOM
  - It is able to make changes to the DOM without reloading the page
  - This will change the structure on the page

2.7. jQuery vs. JavaScript

- jQuery can be considered as a more approachable version of JavaScript
  - It is JavaScript, but easier to use

    ```
    document.getElementsByTagName("p")[0].innerHTML = "Update";
    ```

  - in jQuery:

    ```
    jQuery("p").html("Update");
    ```

- Fetch elements from the DOM

  ```
  for (i = 0; i <= 4; i++){
  document.getElementsByTagName("p")[i].innerHTML="Update";
  }
  ```

  - jQuery uses CSS selectors to fetch elements:

    ```
    jQuery("p").html("Update");
    ```

- jQuery uses JavaScript underneath, but hides the complexity of the DOM

2.8. Selecting Elements

- jQuery selects elements in the same way as CSS
- Examples:
  - An element selector: selects all corresponding elements

```
h1 {
  text-align: left;
}
```

```
jQuery("h1").hide();
```

- Examples:
  - A class selector: allows you to select a grouping of elements

```
.footer {
  position: absolute;
}
```

```
jQuery(".footer").slideUp();
```

- Examples:
  - An ID selector: to select one, and only one element

```
#header {
  color: #3300FF;
}
```

```
jQuery("#header").fadeOut();
```

- While CSS adds style to the selected elements, jQuery selectors will add behaviour to them

2.9. Your first Effects

- Sliding effect
  - Slide up to 0 height, or slide down from 0 to the height defined in CSS
  - Adding a parameter such as "slow" will change the default behaviour

```
jQuery("div").slideUp();
jQuery("div").slideDown();
jQuery("div").slideToggle();
```

- Fading effect
  - Fading in lets an element go from being invisible (transparent) to being visible (opaque)
  - Adding a parameter (in ms) specifies how fast the element fades

```
jQuery("div").fadeIn();
jQuery("div").fadeOut();
jQuery("div").fadeTo();
jQuery("div").fadeToggle();
```

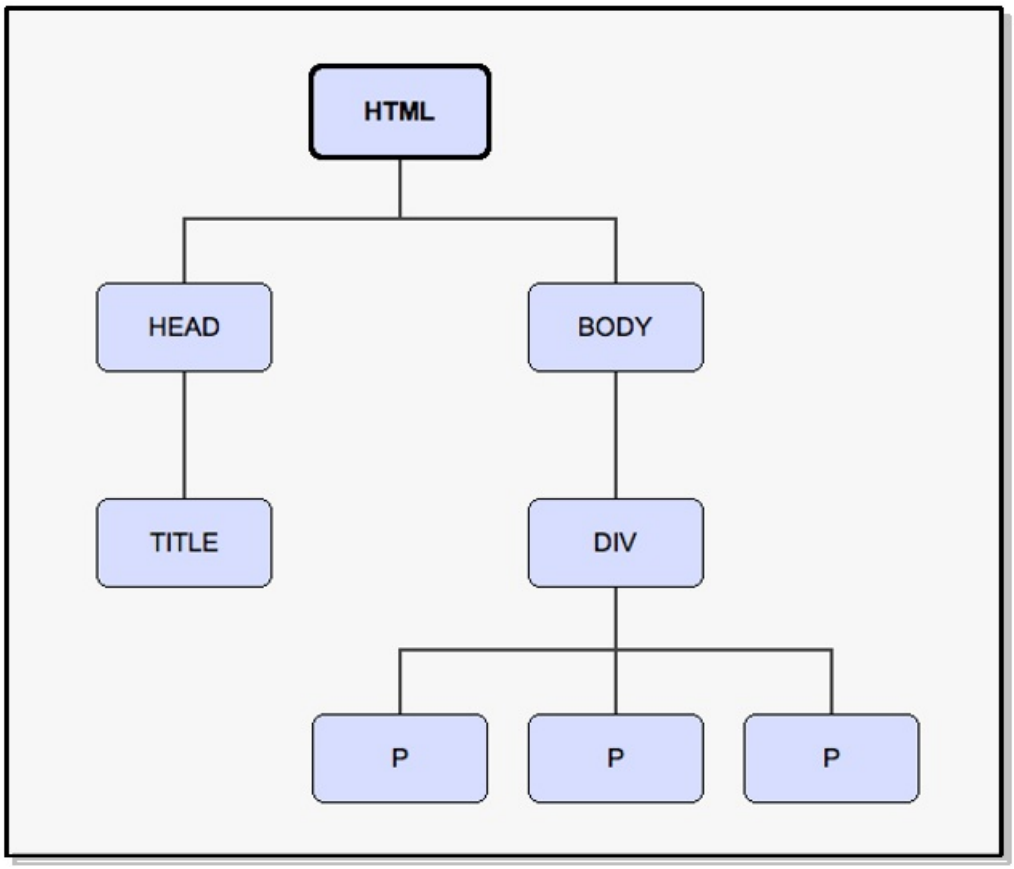
3. Selecting

3.1. The DOM

- Sample HTML code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>the title</title>
    </head>
    <body>
        <div>
            <p>This is a paragraph.</p>
            <p>This is another paragraph.</p>
            <p>This is yet another paragraph.</p>
        </div>
    </body>
</html>
```

- When starting with jQuery, it is important to visualize the DOM in your mind... or to use some tools
- Corresponding DOM tree



3.2. The jQuery Alias $()

- The `$()` is a shorter version of the `jQuery()` function
  - This saves you from writing those five extra characters
  - Is referred to as the jQuery wrapper
    - `jQuery()` becomes: `$()`
  - Both notations point to the same thing
- The `jQuery()` function accepts multiple things

    - CSS selectors
        - Returns the set of elements that match that selector
    - HTML
        - Will let you add DOM elements to the browser page on the fly
    - JavaScript objects
- Note:
    - When combining jQuery with other client-side scripting, the `jQuery()` function may help avoid naming conflicts

3.3. CSS Selectors

- CSS selectors let you grab elements from a page
- jQuery accepts the same selectors
    - You will always start with the `$()` function

Selector Type	CSS	JQuery
Tag name	`p { }`	`$('p')`
ID	`#div-id { }`	`$('#div-id')`
Class	`.class-name { }`	`$('.class-name')`

- The elements returned from the $() function are automatically and implicitly looped through!

3.4. Other Selectors

- You can select child elements with the child combinator

```
$('#nav > li').addClass('horizontal');
```

    - Find each list item (li) that is a child (>) of the element with an ID of nav (#nav)
- The following shows how you can select with the negative pseudo-class

```
$('#nav li:not(.horizontal)').addClass('sub-level');
```

    - Find each list item (li) that is a descendant of the element with ID of nav (#nav) that does not have a class of horizontal (:not(.horizontal))
- You can just select descendants by leaving a space

```
$('div p').remove();
```

    - Find all descendant p elements that are directly inside div elements and remove them
- The current element can be selected with the word `this`

```
var value = Math.round((Math.random() * 9) + 1);
var msg = '<p>A number between 1 and 10: ' + value +'</p>';
$(this).append(msg); // context sensitive!
```

    - On the currently selected element, add a p element with a random value between 1 and 10
    - Note the use of JavaScript methods, variables and concatenation
    - `append()` and `remove()` are jQuery methods
    - `this` removes the need for re-selecting the current element

- Selecting all elements of the DOM tree with *

```
$('*').hide();
```

  - All elements will be hidden...

3.5. Attribute Selectors

- You can also select elements by one of its HTML attributes

```
$('img[alt]')
```

  - Find all images that have an alt attribute
- Attribute selectors accept a wildcard syntax
  - Inspired by regular expressions
    - ^ : at the beginning of a string
    - $: at the ending of a string
    - * : at an arbitrary position within a string
    - ! : indicates a negated value
- Attribute selector examples

```
$('a[href^="mailto:"]').addClass('mailto');
```

- Find all anchor elements (a) with a href attribute ([href]) that begins with mailto: (^="mailto:")

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

- Find all anchor elements (a) with a href attribute ([href]) that ends with .pdf ($=".pdf")

```
$('a[href*="real"]').addClass('realdolmen');
```

- Find all anchor elements (a) with a href attribute ([href]) that contains the word real anywhere (*="real")

3.6. Custom Selectors

- If that is not enough, jQuery adds its own custom selectors
  - They enhance the capabilities of CSS to locate page elements
  - They are a lot slower than normal selecting with the DOM selector engine
- You would like the fetch the first element of list

```
$('div.horizontal:eq(0)')
```

  - Is zero-based, because a JavaScript array is zero-based
- You would like to select the div element that are the first child of their parent element

```
$('div:nth-child(1)')
```

- CSS is one-based
  - Better use:

```
$('div:first-child')
```

- You can select only the odd rows of a table to give them a different style

```
$('tr:even').addClass('alt');
```

- Use even for the odd rows
  - The :even and :odd selectors use JavaScript zero-based numbering
  - The first row (odd) counts as 0 (even)

- But, what if you have two tables on the page
  - The next table could begin with an odd row...

```
$('tr:nth-child(odd)').addClass('alt');
```

  - This selector is one-based, but counts an element's position relative to its parent element

- To select on content, you can use the :contains() selector

```
$('td:contains(Malaz)').addClass('highlight');
```

  - This selector is case-sensitive

3.7. Form Selectors

- jQuery custom selectors also allow you to fetch form elements

Selector	Match
:input	Input, textarea, select, and button elements
:button	Button elements and input elements with a type attribute equal to button
:enabled	Form elements that are enabled
:disabled	Form elements that are disabled
:checked	Radio buttons or checkboxes that are checked
:selected	Option elements that are selected

- Example

```
$('input[type="radio"]:checked')
```

  - Find all checked radio buttons, but not checkboxes

3.8. Traversing the DOM

- Sometimes, you will need to access the parent or ancestor element of a node
  - DOM traversal methods let you go up, down and all around the DOM tree

- Using the `filter()` method

```
$('tr').filter(':even').addClass('alt');
```

  - Is nearly identical to the :even selector, but filter is more powerful, as it accepts functions!

```
$('a').filter(function() {
return this.hostname && this.hostname != location.hostname;
}).addClass('external');
```

  - Find all anchor elements (a) that have a href attribute with a domain name (this.hostname), and the domain name must not match (!=) the domain of the current page (location.hostname)

- Using `next()`, you can select the element next to your current selection

```
$(document).ready(function() {
      $('td:contains(Real)').next().addClass('highlight');
});
```

- Style all the cells next to the cell containing the word 'Real'
- Replace `next()` by `nextAll()` to style all the following cells

- Other DOM traversal methods
  - The `next()` method counterparts: `prev()` and `prevAll()`
  - The method `siblings()` will give you all the elements at the same DOM level (before and after)
  - If you want to include the current element, you can use `addBack()`

```
$(document).ready(function() {
$('td:contains(Henry)').nextAll().addBack().addClass('highlight');
});
```

- You can also access the `parent()` of the current element or its `children()`

3.9. Accessing DOM Elements

- Sometimes, traversal is not enough
  - What if you wanted to know the tag name of an element
  - This is a property of the DOM element
  - For these rare situation, jQuery offers the `get()` method

```
var myTag = $('#my-element').get(0).tagName;
```

  - jQuery also gives you a shorthand notation

```
var myTag = $('#my-element')[0].tagName;
```

- If you want to select the text of a node, use `text()`

```
var text = $('#my-element').text();
```

3.10. Method Chaining

- jQuery lets you select elements and do multiple things on them, in a single line of code
  - Most jQuery methods, return a jQuery object, so you can apply extra methods to your result
  - This can improve performance, because you will not need to respecify another selector

- It is recommended to break a single line into multiple ones for readability

```
$('td:contains(Real)')       // find every cell containing "Real"
.parent()                    // select its parent
.find('td:eq(1)')            // find the 2nd descendant cell
.addClass('highlight')       // add the "highlight" class
.end()                       // return to the parent of the cell containing "Real"
.find('td:eq(2)')            // find the 3rd descendant cell
.addClass('highlight');      // add the "highlight" class
```

4. Decorating

4.1. CSS with jQuery

- Selecting elements in jQuery is the hard part
  - Changing styles and decorating is easy and fun!
  - Once you have selected your target, you can manipulate them to build effects
- jQuery does not only use CSS for selecting
  - jQuery also allows adding and removing styles, classes, and more
  - This is called decorating

4.2. Reading CSS Properties

- Before you try to change CSS properties, you have to access it
  - jQuery does this with the `css()` function

```
$(document).ready(function() {
var fontSize = $('h1').css('font-size');
alert(fontSize);
});
```

  - The `css()` function always returns the property for the first matched element, so you do not have to use it on multiple elements
- jQuery will return the calculated style
  - You will receive the value that has been rendered in the browser
  - This does not strictly correspond to what has been specified in the CSS file, but it is the actual value

4.3. Setting CSS Properties

- To change a CSS property inline, you only need to add an extra parameter to the `css()` function

```
$(document).ready(function() {
  $('h1').css('background-color','#dddddd');
});
```

  - You first choose the property you would like to set
  - Next you provide the value you wish to set for that property
- If you want to change multiple properties, best use an object literal

```
$('tr:even').css('background-color','#dddddd');
$('tr:even').css('color', '#666666');
```

```
$('tr:even').css({'background-color': '#dddddd',
  'color': '#666666'});
```

  - Notice the curly braces, and the key/value pairs separated by a colon, each pair separated by a comma

4.4. Adding and Removing Classes

- jQuery provides and easy way to give an element CSS classes and to remove them
  - This allows you to change how an element looks dynamically

```
$(document).ready(function() {
          $("#btn1").click( function(){
              $("#header").addClass("hover");
              $("#header").removeClass("no_hover");
```

```
        });
        $("#btn2").click( function(){
            $("#header").removeClass("hover");
            $("#header").addClass("no_hover");
    });
});
```

- jQuery also includes `toggleClass()`, that automatically checks for the presence of a class before applying or removing it

- This technique is often preferred over setting with css()
  - The `css()` function changes the style inline, while we prefer putting style in a separate CSS file

5. Events and Functions

5.1. Using Events

- An event is a mechanism that allows you to run a piece of code when something happens to the page
  - Examples: ready, click, ...
- The code that gets run is a function
  - Allows you to make your jQuery code more efficient and reusable
  - Functions that react on events are called handler functions
- Event listeners are what make events work
  - Are part of the DOM
  - Can be added to any element on the page, not just buttons and links
  - The browser will pay attention to the user's interactions, and execute the function specified

```
$("#showMessage")                          // to an element of ID showMessage
    .click(                                    // add a click event
        function() {                           // run this code
            alert('You Clicked Me!');
});
```

5.2. Binding an Event

- Adding an event to an element is called binding
- There are two ways of binding
  - The convenience method, for when the DOM elements already exist

```
$("#myElement").on( function() {
alert($(this).text());
});
```

```
$("#myElement").on('click', function() {
alert($(this).text());
});
```

  - ~~The bind() method, for binding events to elements that you dynamically add after the page is loaded~~
  - To attach an event for all elements that match the selector, now and in the future, use the on() method
- jQuery always registers event handlers in the bubbling phase of the model

5.3. Event Triggers

- Many things can trigger an event
- jQuery groups events in five different categories
  - Browser events
    - ~~error,~~ resize, scroll
  - Document loading events
    - load, ready, ~~unload~~
  - Form events
    - blur, change, focus, select, submit
  - Keyboard events
    - keydown, keypress, keyup
  - Mouse events

        - click, dblclick, focusin, focusout, mousedown, mouseenter, mouseleave, hover, mousemove, mouseover, mouseup

- There are nearly 30 different event types
    - Most will accept one event handler, while some of them, like hover, will accept two handlers

5.4. Triggering Events

- You can also trigger events in code
    - For submitting forms for validation
    - To hide pop-up boxes
    - ...
- To trigger the event, use the trigger() method

```
$(document).ready(function(){
    $("input").select(function(){
        $("input").after(" Text marked!");
    });
    $("button").click(function(){
        $("input").trigger("select");
    });
});
```

5.5. The Event Object

- Events are handled differently in different browsers
    - Internet Explorer supports different events than Chrome, Safari, Firefox, Opera
    - This is a result of the '90s browser wars...
- Luckily for us, jQuery handles these browser issues
    - jQuery knows which browser is in use and handles events accordingly
    - jQuery makes binding events easier
- The event object contains variables and functions
    - Can be used for programmatic triggering of events
    - Has interesting information
        - target, pageX, pageY, ...
- You can pass any function to a bound event as handler
- An event can also be unbounded

5.6. Removing Events

- You will often need to remove events from objects
    - You don't want people to click a submit button twice
    - You want people to only do something once
- After an event is bound, you can then remove it

```
$("#myElement").bind ('click', function() {
  alert($(this).text());
});
```

```
$("#myElement").unbind('click');
```

- Notice how the unbind() method takes the event as argument
- You can bind multiple events to an element, and remove them all at once

```
$("#myElement").bind('focus', function() { // focus event
    alert("I've got focus");
});
$("#myElement").click(function(){ // click event
    alert('You clicked me.');
});
```

```
$("#myElement").unbind();
```

- Notice how the unbind() method takes no arguments...

5.7. Stopping an Event

- Events will automatically bubble up to parent elements
  - This might result in unwanted behaviour
- To get access to the event, add the event as variable to the handler

```
$('#switcher').click(function(event) {
    $('#switcher button').toggleClass('hidden');
});
```

- Now you can use the event object in your code

```
$('#switcher').click(function(event) {
    if (event.target == this) {
        $('#switcher button').toggleClass('hidden');
    }
});
```

- You can also stop the event from propagating

```
$('#switcher button').click(function(event) {
    // more code
    event.stopPropagation();
});
```

5.8. What does the following code do?

```
$(document).ready(function(){
    $("span").click(function(){
        event.stopPropagation();
        alert("The span element was clicked.");
    });
    $("p").click(function(){
        alert("The p element was clicked.");
    });
    $("div").click(function(){
        alert("The div element was clicked.");
    });
});
```

5.9. Stopping Default Actions

- Some elements have default actions
  - When adding event handlers to an anchor element, the browser will also load a new page
  - When pressing the Enter key when editing a form, a submit event is triggered
- If you do not want the default action, use preventDefault()
  - Useful when you want to do specific checks, and only use the default action when everything is valid
- Event propagation and default actions are separate
  - You can disable them separately
  - If you want to halt both, then return false at the end of your event handler, which calls both stopPropagation() and preventDefault()

5.10. Using Event Bubbling

- Bubbling can also be very useful!
  - The technique that exploits bubbling is called event delegation
  - Imagine: a large table with information, each row containing an interactive item with a click handler
    - Adding handlers with implicit iteration is easy...
    - ... but performance will suffer from the internal jQuery loops
    - ... and memory will suffer from maintaining all the handlers
  - Instead, assign a single click handler to the ancestor element, and let the event bubble up to it!

```
$('#switcher').click(function(event) {
      if ($(event.target).is('button')) {
          // more code
          $(event.target).addClass('selected');

      }
});
```

  - Using event.target, we have access to the element where the event came from
  - The is() method checks if the element matches the selector

5.11. Separating Scripts

- When your page becomes heavy on script code, you should separate the jQuery script in a separate file
- Advantages
  - You can include it in more than one page (code reuse)
  - Your page will load faster
  - The HTML code you write will be cleaner and easier to read
- Steps
  - Create a new script file, ending in .js
  - Move the JavaScript and jQuery code to this file
  - Create a link to this file in your HTML page

```
<script src="scripts/scripts.js"></script>
```

- The browser will only ask for this file once, and cache it

5.12. Creating Functions

- Functions are blocks of code
  - They are separate from the rest of your code

    - You can execute them wherever you want in your script

- jQuery provides lots of functions
- Event handlers are also functions, but anonymous
- Creating your own functions allows you to organize your code by name, so that they can easily be reused
- There are two ways to create functions
    - Function declaration

```
function myFunc1(){
$("div").hide();
}
```

    - Function expression

```
var myFunc2 = function() {
$("div").show();
}
```

- The difference between the two ways is timing
    - A named function expression cannot be used until after it is encountered and defined
    - A named function declaration can be called whenever you want, even as an onload event handler

- Anonymous functions
    - Do not have names
    - Get called immediately when they are encountered in code
    - Any variables declared inside are only available when the function is running
    - Cannot be called anywhere else

- Named functions can be called and assigned as event handlers

```
myFunc1();                      // calls the function, hides the divs
$("#myElement").click(myFunc2); // calls when clicked, shows the divs
```

    - Notice how assigning does not required the use of parentheses

5.13. Passing Variables and Returning Values

- Functions can behave differently, depending on the information we give it
    - Functions can accept variables
    - When variables are passed into functions, they are called arguments (or sometimes referred as parameters)

```
function welcome (name) { // argument name
alert ("Hello " + name);
}
welcome("John");
```

    - By changing the variable, we can now change the behaviour of the function, without having to rewrite it

- Functions can also return values, representing results

```
function multiply (num1, num2) {
  var result = num1*num2;
  return result;
}
var total = multiply (6, 7);
```

    - If no return is present, the function returns undefined

5.14. jQuery Loops

- If you need to interact with a group of elements one by one, you can use a loop
  - Also known as iteration
  - While going through the group, you can do something to each element along the way

```
$(".nav_item").each(function(){
$(this).hide();
});
```

- In other words
  - each() uses the selector that calls it and creates an array of elements identified by that selector
  - It then loops through each element in the array sequentially
  - Note: If you return false from inside a loop, it will stop executing and move on
- The function passed to each() can also take two variables

```
$(".nav_item").each(function(index, value){
  // use index and value here
});
```

  - The index refers to where the element appears in the list
  - Indexes always start at 0
  - Value represents the current object the loop is working on
    - Is the same as using $(this) inside the each() loop

5.15. Using Conditional Logic

- jQuery uses the JavaScript conditional logic
  - This lets you run different code based on decisions you want your code to make, based on the information you provide

```
if( myBool == true ){
          // do something!
      } else {
          // otherwise do something else!
      }
```

- jQuery has a static method that checks if the first parameter contains whatever is in the second parameter

```
if($.contains(this, document.getElementById("my_element"))){
      // when true, do something!
}
```

  - Notice how an if does not require an else
  - Static means the function is part of the jQuery library instead of being from an object, and works simply with $ or jQuery

6. Modifying the DOM

6.1. DOM Traversal Revisited

- You have already seen how to traverse the DOM tree
  - next(), nextAll(): returns the next elements at the same DOM level
  - prev(), prevAll(): returns the previous elements at the same DOM level
  - parent(): returns the parent element
  - children(): returns the children of an element
  - siblings(): returns all the elements at the same DOM level
  - addBack(): to include the current node in the selection
- jQuery offers other interesting traversal options
  - parents(): let's you traverse all of the element's parent elements
  - closest(): climbs through the parents, but stops when a matching element is found

```
$("li").closest("ul")
```

6.2. Removing and Detaching

- With jQuery, you can remove elements from the DOM

```
$("img#thumbnail").remove();
```

  - This will remove the element from the DOM permanently
  - If you remove a parent element, its children will also be removed
- jQuery also offers the detach() method

```
$("img#thumbnail").detach();
```

  - This will take the element out, but holds on to it so that it can be reattached later
  - All elements matched by the selector will be detached
  - To store the detached elements and its contents, you can use a variable
- To get rid of something inside an element, you can use the empty() method

```
$("p").empty();
```

6.3. jQuery Arrays

- To store elements returned from jQuery, it is common practice to place a $-sign in front of the variable

```
$s = $(".sold-out").parent().parent().detach();
```

  - This variable will store multiple elements
  - jQuery puts the result in a JavaScript array
  - Using the $-sign is a coding convention for a jQuery array
  - Note the method chaining capability of jQuery
- To access values from an array, you use an index

```
$array[2] = 15;
```

  - This will put a value in the third slot
  - An array index is a sequential number that starts at 0
- To find elements elements in an array, you use find()

```
var $my_elements = $("li");
$my_elements.find("a");
```

6.4. Replacing Elements

- If you need to replace elements with new ones, you can use the replaceWith() method

```
$("h2").replaceWith("<h1>My Books</h1>");
```

  - The element will be taken out of the DOM to be replaced by the new code (creating new DOM nodes)
  - You can provide HTML in the parentheses to replace

- This works well for a one-to-one replacement
  - You can always put the replaced element back
- If you use this method on one-to-many substitution...
  - ... the elements will be replaced, but you would not know what was there before...
  - The DOM has forgotten about the replaced elements
- If you need to remember what was replaced, you will need two steps
  - Insert the new elements into the DOM before or after the selected element
  - Detach the selected elements and hold them in a variable
- jQuery has operations that allow to add elements

```
$(".sold-out").before("<li>Sold Out!</li>");
$(".sold-out").after("<li>Sold Out!</li>");
```

  - before() adds the element before the selection
  - after() adds the element after the selection

6.5. Other Filter Methods

- To narrow your selection, you can use filter methods
  - first(): will filter out everything but the first element in a selection
  - eq(): will filter out everything but the element whose index is equal to what you put in the parameter
  - last(): will filter out everything but the last element in a selection

```
$(".books").children().first();
$(".books").children().last();
$(".books").children().eq(0); // same as first
```

  - slice(): will filter out everything except elements with an index between the numbers you put in parentheses
    - You can also specify a negative starting number!
  - filter(): will filter out everything but elements that match the selector you put in parentheses (even accepts functions!)
  - not(): will filter out everything that does not match the selector

```
$(".books").children().slice(1,3); // returns two elements [1,3[
$(".books").parents().filter(".fantasy");
$("ul.books.fantasy").children().not(".expensive");
```

6.6. Wrapping Elements

- Sometimes, you would like to wrap an element inside another element

- jQuery gives you the wrap() method
  - Without it, it would take you multiple calls to detach, append, before, and so on … to put the element inside another

```
$("img#logo").wrap("<a href='http://www.realdolmen.com'></a>");
```

7. Adding Effects

7.1. Hiding and Revealing Elements

- To hide an element, you can use the hide() function

```
$('#hideButton').click(function() {
  $('#disclaimer').hide();
});
```

  - When the button is clicked, the element #disclaimer will disappear
- It can be confusing that jQuery uses methods for selecting, actions and also for events and effects...
  - The name of the action gives you a good clue to its purpose
  - Consult the jQuery API regularly to sort them out!
- To reveal a hidden element, you use the show() function

```
$('#showButton').click(function() {
  $('#disclaimer').show();
});
```

  - When the button is clicked, the element #disclaimer will appear
- You could combine the hide() and show() in one action

```
$('#toggleButton').click(function() {
  if ($('#disclaimer').is(':visible')) {
      $('#disclaimer').hide();
  } else {
      $('#disclaimer').show();
  }
});
```

  - The is() function lets you check if a selector matches the element it was called on... in this case:
    - Is the element also selected by the selector :visible?
  - The above construct is called a toggle, which is very useful!
  - But we need this toggle lots of times in our code...
- jQuery has a toggle version of most functions

```
$('#toggleButton').click(function() {
  $('#disclaimer').toggle();
});
```

7.2. Progressive Enhancement

- Some users do not have JavaScript...
  - Using very old computers
  - Using limited devices (like mobile phones...)
  - People with visual impairments who require screen readers
  - Those users who worry about JavaScript security and have it disabled
- Around 5 to 10% of users might be browsing without JavaScript
  - You don't want to alienate 10% of potential customers
- Solution: progressive enhancement
  - Provide an acceptable experience to those users
  - Beef up your website for the others!
- Show some elements by default... but hide, add or remove functionality with jQuery

7.3. Adding New Elements

- Any valid HTML string put inside the jQuery function will be created and made ready for page addition

```
$('<p>A new paragraph!</p>').addClass('new');
```

- After creating the element, there are several ways to add the element to the page
  - We have already seen append(), after() and before()
  - jQuery also has insertAfter() and insertBefore() to insert as siblings

```
$('<input type="button" value="toggle" id="toggleButton">')
.insertAfter('#disclaimer');
```

  - The difference lays in the fact that jQuery applies the function to the newly created element and adds to the parameter
  - The same can be said about appendTo() and prependTo()

```
$('<strong>START!</strong>').prependTo('#disclaimer');
```

  - The element will appear inside the existing element

7.4. Removing Elements

- We have already seen the remove() operation
  - But remove() can also take parameters to refine the selection further

```
$('#books tr').remove(':contains("Fantasy")');
```

  - This will remove only those rows that contain the text "Fantasy"
- Removing elements is one of the things you should do for progressive enhancement

```
<p id="no-script">
  We recommend that you have JavaScript enabled!
</p>
```

  - The paragraph will be visible to all users
  - Remove the paragraph for people with JavaScript enabled

```
$('#no-script').remove();
```

7.5. Modifying Content

- Adding, removing and replacing are fun, but if you want to replace the contents of an element... what then?
  - jQuery has it all!
  - You can use the html() and text() methods

```
$('p').html('good bye, cruel paragraphs!');
$('h2').text('All your titles are belong to us');
```

- Use text() if you need to add plain text, but html() to add HTML

```
$('p').html('<strong>Warning!</strong> Text has been replaced … ');
$('h2').text('<strong>Warning!</strong> Title elements can be …');
```

  - In this example, the paragraphs will contain bold-faced text, but the h2 tags will contain the text exactly as defined
  - The same methods can also be used to retrieve the contents

7.6. Basic Animation

- We have already worked with fadeIn() and fadeOut()
  - These methods accept as parameter the time it takes to complete
  - You can also use some predefined values: slow, fast or normal
  - Use these instead of show() and hide() to give your page more visual impressiveness
- jQuery does not have a fadeToggle()... but you can simply animate the toggle() method!
  - Pass a time span parameter to the method

```
$('#toggleButton').click(function() {
$('#disclaimer').toggle('slow');
});
```

- Further basic animation element include slideDown(), slideUp() and slideToggle()
- Animation is important
  - When things change on the page, the user should not get lost!

7.7. Callback Functions

- Many effects accept a callback function
  - This enables you to run some code when the effect has finished

```
$('#disclaimer').slideToggle('slow', function() {
alert('The slide has finished sliding!')
});
```

  - You pass the function in the same way as you provide event handlers
- You can use this to chain effects one after the other

```
$('#disclaimer').slideUp('slow', function() {
  $('#hideButton').fadeOut();
});
```

8. Animating

8.1. Animating CSS Properties

- Basic animation was easy, but what if you want more control?
  - jQuery lets you animate() a whole host of CSS properties to create your own effects

```
$('p').animate({
padding: '20px',
borderBottom: '3px solid #8f8f8f',
borderRight: '3px solid #bfbfbf'
}, 2000);
```

  - This code animates all paragraphs on the page, changing the padding, and adding a bevelled border over a period of 2 seconds
  - Pass an object literal to animate, with camel-cased properties
  - You can also use relative values with += and -=

```
$('#navigation li').hover(function() {
    $(this).animate({paddingLeft: '+=15px'}, 200);
}, function() {
    $(this).animate({paddingLeft: '-=15px'}, 200);
});
```

- animate() gives you more control over showing, hiding, and toggling

```
$('#disclaimer').animate({
  opacity: 'hide',
  height: 'hide'
}, 'slow');
```

- You should try to animate as much elements as possible
  - You will stumble upon some nice effects!
  - But... keep it sensible!
    - At one time, the tag was considered sensible...

8.2. Color Animation

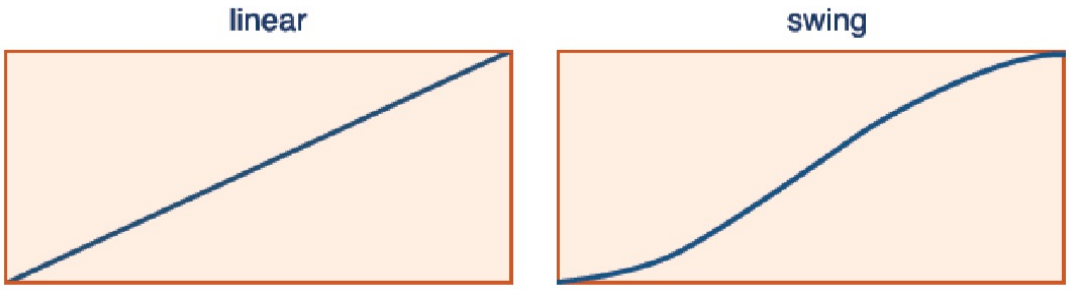
- Animating color is a bit tricky...
  - Color values in between start and end colors need to be calculated
  - This color calculating functionality is not included in jQuery
    - Most projects do not need this functionality
    - This is to keep the jQuery library size as small as possible
  - You will need to download the Color Animations plugin
    - http://plugins.jquery.com/color/ or https://github.com/jquery/jquery-color/
  - jQuery has lots of plugins you can download and use freely
  - You will just need to add the plugin to your page
- Once added to your page, you can now start animating colors

```
$('#disclaimer').animate({'backgroundColor':'#ff9f5f'}, 2000);
```

8.3. Easing

- Easing refers to the acceleration and deceleration that occurs during an animation
  - Gives the animation a more natural feel
  - Applies a mathematical algorithm to alter the speed of the animation as it progresses

- jQuery has two types of easing available
  - linear and swing
  - Any time you use an animation, you can add these as parameters to control the easing

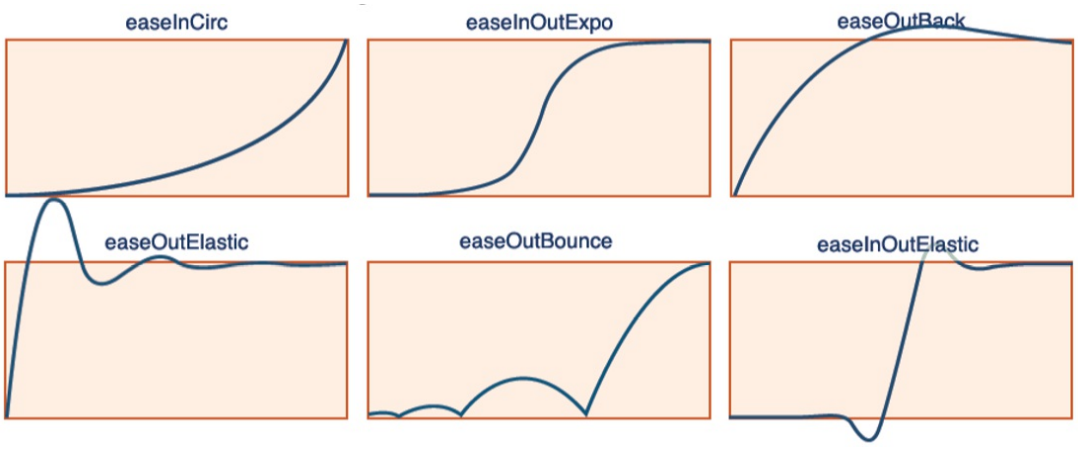



```
$('p:first').toggle(function() {
    $(this).animate({'height':'+=150px'}, 1000, 'linear');
}, function() {
    $(this).animate({'height':'-=150px'}, 1000, 'swing');
});
```

- What happens here?
  - We select only the first paragraph tag
  - A toggle event executes each passed function on successive clicks
  - Inside the handlers we use $(this) to reference the paragraph
  - The first handler uses the += format to grow the paragraphs height, using linear easing
  - The other handler shrinks the height with -= with swing easing
- You will notice the shrinking animation feels a bit more natural...

8.4. Bouncing

- There is a vast array of easing options including bouncing!
  - More than 30 options can be found in the easing plugin
  - http://plugins.jquery.com/jquery.easing/
  - The easing library is also included in the jQuery UI library
    - jQuery UI includes several common plugins: color animation, class transitions, easing, ...



- To use the algorithms, pass its name to the animate function

```
$('p:first').animate({height: '+=300px'}, 2000, 'easeOutBounce');
$('p:first').animate({height: '-=300px'}, 2000, 'easeInOutExpo');
$('p:first').animate({height: 'hide'}, 2000, 'easeOutCirc');
$('p:first').animate({height: 'show'}, 2000, 'easeOutElastic');
```

- The algorithms can be found at this website
  - http://www.robertpenner.com/easing/
  - The algorithms originated from Robert Penner's easing equations
- To see all the available equations just view the plugin's source code
  - If you use your text editor to open up the file you downloaded, you'll see a list of the functions you can use in jQuery animations

8.5. The Animation Queue

- The animate() function can also be called with extra options
  - These options are bundled together as object literal
  - This format is only necessary when you need the queue parameter

```
$('p:first').animate({
      height: '+=100px',
      backgroundColor: 'green'
      }, {
      duration: 'slow',
      easing: 'swing',
      complete: function() {alert('done!');},
      queue: false
});
```

  - The queue is the list of animations waiting to occur on a particular element
- Queued animations will be performed one at a time until everything is complete
  - But sometimes, you want multiple animations to happen at the same time
  - The queue is then not desirable
- The queue can be controlled using the queue option
  - Other alternatives: jQuery actions stop(), queue(), and dequeue()
  - This will give you super-fine control over how your animations will run

8.6. Chaining Actions

- We have already seen the advantages of method chaining
- You can apply chaining on animations as well

```
$('p:first').hide().slideDown('slow').fadeOut();
```

  - Be careful: chaining can become addictive!
  - You can add, and remove elements, or move around the DOM, still based on your initial selection... which creates weird statements
  - It is recommended to lay out your actions in separate lines for readability
- If you want your animation to pause briefly in the middle of a chain, use the delay() action

```
$('p:first')
.hide()
.slideDown('slow')
.delay(2000)
.fadeOut();
```

  - This code will slide down the paragraph, then wait 2 seconds before fading out
  - This can be a great way to control your animations more precisely!

8.7. Animated Navigation

- Many web sites use animated navigation on their pages
- You can use a Flash control that wobbles and zooms as the user interacts with it...
  - But you would not be here if you wanted to use Flash...
  - You can apply what you have learned to create animated navigation with free technologies: HTML, CSS and jQuery
- Step 1: Adding a navigation menu to the page

```
<ul id="navigation">
      <li><a href="#">Home</a></li>
      <li><a href="#">About Us</a></li>
      <li><a href="#">Buy!</a></li>
      <li><a href="#">Gift Ideas</a></li>
</ul>
```

- Step 2: Changing the navigation style

```
#navigation {
  margin-top: 10px;
  padding: 5px;
  list-style-type: none;
  position: relative;
  z-index: 1;
  background: #c0d0d0;
  float:none;
}
#navigation ul {
  margin: 0;
  padding: 0;
}
#navigation li {
  display: inline;
  margin: 0;
  padding: 0;
}

#navigation a {
  color: #000;
  display: inline-block;
  padding: 5px;
  text-decoration: none;
}
```

- Step 3 : Adding the background color blob
  - This div will be positioned behind whatever link the user is mousing over

```
$('<div id="navigation_blob"></div>').css({
width: $('#navigation li:first a').width() + 10,
height: $('#navigation li:first a').height() + 10
}).appendTo('#navigation');
```

- Step 4: Styling the background blob

```
#navigation_blob {
top: 5px;
background: #826c4b;
position: absolute;
z-index: -1;
}
```

- Step 5: Setting up the event handler functions

```
$('#navigation a').hover(function() {
  // Mouse over function
  ⋮
}, function() {
```

```
  // Mouse out function
  :
});
```

- Step 6: The fun part!

```
// Mouse over function
$('#navigation_blob').animate(
{width: $(this).width() + 10, left: $(this).position().left},
{duration: 'slow', easing: 'easeOutElastic', queue: false}
);
```

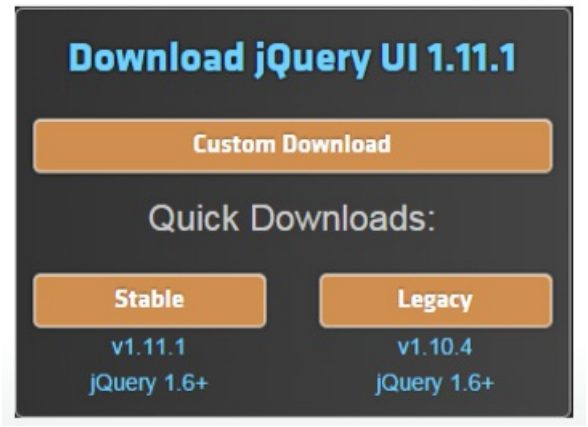
- Notice the position() method, which does nothing but exposes the top and left properties, so we can change the relative position
- Step 7: When the user moves the mouse off the link

```
// Mouse out function
var leftPosition = $('#navigation li:first a').position().left;
$('#navigation_blob').stop(true).animate(
{width:0},{duration:'slow', easing: 'easeOutCirc', queue: false}
).animate({left: leftPosition}, 'fast' );
```

- Two animations are chained together
  - The first one will hide the element with some easing applied
  - The second one will send it to the side
- Notice the stop() method
  - Two arguments: clearQueue and gotoEnd
  - clearQueue set to true will clear any queued animations
  - gotoEnd will let the element immediately go to the end of its animation

8.8. Using jQuery UI

- jQuery UI library
  - Is a collection of advanced jQuery widgets, effects, and interactions
  - Contains date pickers, accordions, and drag-and-drop functionality and more
  - Most elements included are widely applicable to web development
- Downloading jQuery UI
  - The full jQuery UI library is very big in size (more than 1Mb!)
  - The jQuery web site has a tool to create a custom version of jQuery UI that contains only the elements you need
  - http://www.jqueryui.com



- The download builder is broken into several sections

    - Core, Interaction, Widgets, Effects, and Themes
    - Core is the main jQuery UI library
    - Best practice: deselect everything and then add only what you require
        - If a component relies on another component to function,it will be automatically selected for you
    - While developing, it is safe to grab the whole library
        - Once happy with the result, you can then create a custom library with reduced file size
    - Themes will greatly influence the visual output
    - The downloaded library contains demo's and examples
- To get started, you will need
    - The jquery-ui.min.js file
    - The images directory
    - The jquery-ui.min.css file
- jQuery UI has an interesting effects package
    - It includes the color and easing plugins, so you no longer need to add them
    - It adds extra effects and functionality for advanced animation

```
$('p:first')
.effect('shake', {times:3}, 300)
.effect('highlight', {}, 3000)
.hide('explode', {}, 1000);
```

    - Some can only be used with effect(), others can be applied on hide(), show() and toggle()
        - blind, clip, puff, fold, and slide
- If these effects are still not enough, there are hundreds of them available from the plugin repository

9. Scrolling

9.1. Scroll Event

- Scrolling is like animation
  - The elements move on the page
  - But... the user is still in control!
  - jQuery lets you customize the scroll experience
- You will need to know when and what the user scrolls
  - jQuery has a scroll() event
  - This event fires updates as the user changes the scroll position
  - scroll() events are usually put on the window element...
- But you can also set up scrolling effects on other elements

```
#news {
  height: 100px;
  width: 300px;
  overflow: scroll;
}
```

- The following adds some text every time the scroll() event fires

```
$('#news').scroll(function() {
  $('#header')
  .append('<span class="scrolled">You scrolled!</span>');
});
```

  - This will be very annoying, ... but it is easy to implement!
- scroll() events will fire for multiple user interactions
  - Dragging the scroll bar
  - Using the mouse wheel
  - Clicking inside the scrollable zone and using the arrow keys

9.2. Floating Navigation

- A floating navigation pane is always present at the top part of the screen
  - This is regardless of the scroll position
  - It is as if the navigation menu follows the users as they scroll down
- Step 1: Set up CSS properties

```
#navigation {
  position: relative;
}
#main {
  height: 2000px;
}
```

  - The large height for the main content is to force it to have a scrollbar
- Step 2: Responding to the scroll() event

```
$(window).scroll(function() {
  $('#navigation').css('top', $(document).scrollTop());
});
```

  - Notice the use of scrollTop()
    - This returns the top offset of the matched element
    - In the above example, on $('document'), it returns the top of the screen

- The above example works... but is a little jumpy
  - Every time the user changes the scroll bar, it fires many events!

- Step 3: Adding some animation
  - It will add some refinement to our floating panel

```
$(window).scroll(function() {
$('#navigation')
.stop(true)
.animate({top: $(document).scrollTop()},'slow','easeOutBack');
});
```

  - stop() will prevent the jumpiness

9.3. Scrolling the Document

- It is common to have a hyperlinked list at the top of the page
  - Especially when a whole list of subjects has to displayed in the same HTML document
  - The links will let you jump to the correct position
  - After the content, you can have a link that takes you back to the top of the page

- This should be easy
  - Add a `<a href="#">Top</a>` at the end of the page
  - Animate with scrollTop()
  - Disable the default link action (or else the animation will not occur)

```
$('a[href=#]').click(function() {
$('html').animate({scrollTop: 0},'slow');
return false; // cancel the default link action
}
```

- The code works, but there is an issue!
  - The $('html') fails when the browser is in 'quirks' mode
  - http://reference.sitepoint.com/css/doctypesniffing
  - It can happen when you are working with legacy code...

- To make it work you can
  - Change the selector to $('body')
  - Or target both $('html, body')
    - But this in turn gives issues in Opera which tries to scroll both...

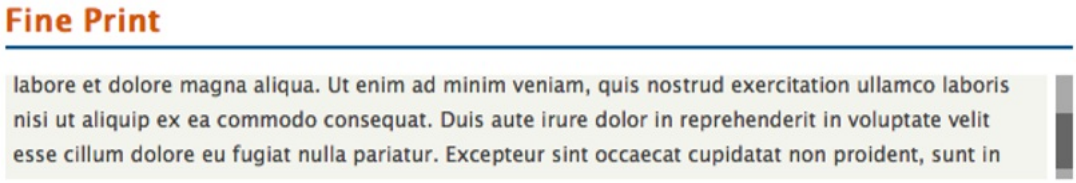
- But jQuery should handle all these cross-browser issues!
  - You're right! jQuery handles most of them
  - This is a trickier one that you need a plugin for, called ScrollTo
  - http://plugins.jquery.com/scrollTo/

```
$('a[href=#]').click(function() {
$.scrollTo(0,'slow'); // scroll the window
// $('div#scrolly').scrollTo(0, 'slow');
return false;
});
```

9.4. Custom Scroll Bars

- This can be interesting for internal elements like scrolling divs
  - But you have to be aware of some usability implications
    - People expect how certain elements of their OS will function
    - Breaking user's expectations can be extremely frustrating

        - This type of UI customization should be undertaken with extreme care!
    - It can still make a world of difference to the overall feel of the interface
- You do not have to build it from scratch
    - You can use the jScrollPane plugin
    - Allows you to replace the browser's default vertical scroll bars with custom ones in any element with overflowed content
    - https://github.com/vitch/jScrollPane
    - Demo's and documentation will help you get started
- Include the following files in your page
    - jquery.jscrollpane.css: styles needed by jScrollPane
    - jquery.mousewheel.js: optional, but recommended to allow users to scroll content with the mouse wheel
    - jquery.jscrollpane.min.js: the actual JavaScript for jScrollPane
- You can extend or overwrite the styles to customize
- The default scroll bars are already sleek-looking




```
$('#fine_print').jScrollPane({
  scrollbarWidth: 10,
  scrollbarMargin: 10,
  showArrows: false
});
```

10. Resizing

10.1. Resize Event

- Resizing has different meanings
  - Resizing the browser window (which is hard to do!)
  - Resizing of windows, images, etc... inside applications
- jQuery can gather information of user resizing, and gives resizing capabilities to any element on the page
- The resize() event is fired whenever a document view is resized

```
$(window).resize(function() {
  alert("You resized the window!");
});
```

  - Here an alert will be shown every time the user resizes the window...
  - Better to use resizing for more than that!

10.2. Layout Switcher

- Instead of debating the use of fluid or fixed-width layouts
  - Just give your users separate fixed-width layouts for different screen sizes
  - You could then switch them using the resize() event
- Step 1: Set up the wider CSS

```
body #main {
  width: 850px;
}
body #main p {
  width: 650px;
}
body #header {
  background-image: url('../../css/images/wide.png');
}
body #books table {
  width: 650px;
  margin-left: 5px;
}
```

  - This CSS is set up to 850px width instead of the default 650px
  - The use of the superfluous body makes sure these rules take precedence over the default because they are more specific
- Step 2: Switch layouts when resizing / at initialization

```
$(function() {
  stylesheetToggle();
  $(window).resize(stylesheetToggle);
});
function stylesheetToggle() {
  if ($('body').width() > 900) {
      $('<link rel="stylesheet" href="wide.css" type="text/css" />')
      .appendTo('head');
  } else {
      $('link[href=wide.css]').remove();
  }
}
```

  - Tests if the body width is greater than 900px at resize, then append the wide stylesheet, if not, remove the stylesheet
  - The function has been given to resize() as a parameter

10.3. Resizable Elements

- jQuery UI contains a Resizable plugin
  - Elements you select get a small handle to the bottom corner
  - This can be stretched around with the mouse
  - Very easy to use and highly configurable
- The basic use of the Resizable plugin is very easy

```
$('p').resizable();
```

  - Just call the resizable() function
  - Of course, jQuery UI must be included in your page

- This is very handy for resizing textarea elements

```
$('textarea').resizable({
  grid : [20, 20], // resizes by steps
  minWidth : 160, // min width
  minHeight : 30, // min height
  maxHeight : 220, // constraining the max
  containment: 'parent' // limit to the parent
});
```

  - Add a handles parameter with n, e, s, w, ne, se, sw, nw, and all to specify in which directions the element can be resized

---

10.4. Pane Splitter

- A splitter divides multiple areas on a page
  - Users are allowed to resize these elements
  - They can decide how much space they want to allot each area
  - Are commonplace in desktop applications
    - On the left: table of contents, tree structure, ...
    - On the right: the contents to show
- Step 1: Setting up the structure

```
<div id="splitter">
  <div class="pane" id="tocPane">
      <div class="inner">
      ⋮
      </div>
  </div>
  <div class="pane" id="contentPane">
      <div class="inner">
      ⋮
      </div>
  </div>
</div>
```

- Step 2: Adding some styles in a new CSS file

```
#splitter {
  height: 150px;
  margin-top: 30px; margin-bottom: 50px;
}
#splitter .pane {
  width: 50%; height: 100%;
  float: left;
}
#splitter h2 {
```

```
  margin-bottom: 0;
  padding-bottom: 0;
}
#tocPane {
overflow: hidden;
background: #d6dde5 url(../images/handle.png) no-repeat right center;
}
#tocPane .inner {
  width: 300px;
}
#contentPane {
  overflow: auto;
}
#contentPane .inner {
  padding: 0 5px;
}
```

- Step 3: The jQuery code
  - Make the first element resizable
  - Recalculate the width of the right element as the component is resized
  - Use outerwidth() to include padding and borders

```
$('#splitter > div:first').resizable({
handles: 'e',
minWidth: '100',
maxWidth: '400',
resize: function() {
  var remainingSpace = $(this).parent().width() -
      $(this).outerWidth();
  var divTwo = $(this).next();
  var divTwoWidth = remainingSpace - (divTwo.outerWidth() -
      divTwo.width());
  divTwo.css('width', divTwoWidth + 'px');
}
});
```

- While very useful, if you need more complex behaviour, you could use the jQuery Splitter plugin

11. Images and Slideshows

11.1. Images and jQuery

- The Internet would be fairly boring without images
  - Much of the content we receive is in the form of pictures and design elements
    - Borders, icons, gradients, ...
- Combine all of these elements with a dose of jQuery and you can add some startling effects
  - Let your creativity go wild!
    - Image galleries, slideshows, fading effects, ...
- jQuery lets you implement less common effects that would be rather difficult to do in JavaScript alone
- We will only look at two examples
  - Creating a Custom Lighbox
  - Creating a Slideshow
- More you can do with images...
  - Cross-fading slideshows, automatic scrolling with JavaScript timers, ...

11.2. Custom Lightbox

- Is used to display full-sized versions of an image thumbnail in a modal dialog
  - The entire background becomes darker to indicate it has been disabled
  - The user must interact with the image to continue working on the page, such as closing it
- Let's create our own custom lightbox!
  - There are some plugins, such as ColorBox, that are able to do all this, but we want to see what's under the hood!
    - http://plugins.jquery.com/colorbox/
  - Clicking on a link will pop up the image file, centered, and the surrounding area will be disabled and darkened
- Step 1: Adding the link

```
<a href="/images/book01.jpg" class="lightbox">Picture</a>
```

- Step 2: Making the screen go dark
  - Make a div as tall and wide as the page
  - Inside that div, you can add another div into which we load the image
  - Fade the opacity to give the div a shadowy effect
  - You can also add a spinning loader image as extra effect

```
#overlay {
position:absolute;
top:0;
left:0;
height:100%;
width:100%;
background:black url(../images/loader.gif) no-repeat scroll center center;
}
#container {
position:absolute;
}
```

- Step 3: Adding the click handler
  - When clicked, we'll add the dark overlay, an image container, and the image itself
  - The container is not strictly necessary, but helpful if you want to add borders, descriptions or next and previous buttons later...

```
$('a.lightbox').click(function(e) {
```

```
$('body').css('overflow-y', 'hidden'); // hide scrollbars!
$('<div id="overlay"></div>') // add overlay
  .css('top', $(document).scrollTop())
  .css('opacity', '0')
  .animate({'opacity': '0.5'}, 'slow') // create shadow
  .appendTo('body');
$('<div id="container"></div>').hide().appendTo('body');
$('<img />').attr('src', $(this).attr('href'))
.load(function() { // when image is loaded
  positionLightboxImage();
})
.click(function() { // remove the lightbox
  removeLightbox();
})
.appendTo('#container');
return false;
});
```

- Step 4: Positioning the lightbox

```
function positionLightboxImage() {
  var top = ($(window).height() - $('#container').height()) / 2;
  var left = ($(window).width() - $('#container').width()) / 2;
  $('#container').css({
      'top': top + $(document).scrollTop(),
      'left': left
  }).fadeIn();
}
```

  - Calculate the center point and fades in the image

- Step 5: Removing the lightbox when the image is clicked

```
function removeLightbox() {
  $('#overlay, #container')
  .fadeOut('slow', function() {
      $(this).remove();
      $('body').css('overflow-y', 'auto'); // scrollbars!
  });
}
```

- Use console.log() for troubleshooting!
  - But don't forget to remove them before going into production...

11.3. Slideshow

- We will create a slideshow widget for use on our pages
  - Mousing over the left or right side of the current image will scroll to the previous or the next image
  - This lets users flip casually through the gallery

- Step 1: Start with a list of images

```
<div id="covers">
  <a href="#" class="trigger">Cover Gallery</a>
     <ul id="covers_inner">
         <li>
          <img src="images/book01.jpg" />
     </li>
     <li>
         <img src="images/book02.jpg" />
     </li>
     ⋮
  </ul>
</div>
```

  - If JavaScript and CSS are disabled, the user will just see a stack of images

- Step 2: Enhancing with CSS

```
#covers {
  border: 1px solid #BEBEBE;
  height: 400px;
  overflow: hidden;
  position: relative;
  width: 268px;
}
#covers ul {
  left: 0;
  list-style-type: none;
  margin: 0;
  padding: 0;
  position: absolute;
  top: 0;
  width: 1072px;
}
#covers li {
  float: left;
}
#covers .trigger {
  left: 0;
  position: absolute;
  top: 0;
  z-index: 10;
  text-indent: -9999px;
  height: 400px;
  width: 268px;
  display: block;
}
```

  - Set the display constraints and hide too wide images
  - The width of the ul depends on the number of images
  - The last CSS lets the trigger cover the whole gallery, with its text hidden away

- Step 3: Creating a widget
  - Will let you reuse the code easily and will help you learn a few JavaScript concepts

```
var gallery = {};
```

  - This creates a JavaScript object literal
    - Put it outside the $(function(){})-block
    - It gives your code a named boundary, and limits the chance of code conflicts with other scripts included in the page
    - It allows you to use a shorter code notation

- Next, you can add some properties to the object inside the$(function(){})-block

```
gallery.trigger = $('#covers .trigger'); // the trigger
gallery.content = $('#covers_inner'); // the list of images
gallery.scroll = false;
gallery.width = 268;
gallery.innerWidth = gallery.content.width();
gallery.timer = false;
gallery.init();
```

- Step 4: Creating an offset() function
  - Sets how far to slide the gallery along
  - Put these function declarations outside the$(function(){})-block

```
gallery.offset = function() {
var left = gallery.content.position().left;
if (gallery.scroll == '>') { // check direction
if (left < 0) {
left += gallery.width; // generate new left value
}
}
else {
```

```
if (left <= 0 && left >= ((gallery.innerWidth * -1)
      + (gallery.width * 2))) {
left -= gallery.width; // generate new left value
}
}
return left + "px"; // return CSS value
}
```

- Step 5: Creating a slide() function
  - Animates the gallery sliding

```
gallery.slide = function() {
if (gallery.timer) { // check if timer set
clearTimeout(gallery.timer); // to be safe...
}
if (gallery.scroll) { // only scroll when needed
$(gallery.content).stop(true,true) // prevent piling up
  .animate({left: gallery.offset()}, 500);
// let the next slide happen
gallery.timer = setTimeout(gallery.slide, 1500)
}
}
```

  - Notice the use of JavaScript clearTimeout() and setTimeout()

- Step 6: Sorting out the direction to scroll to
  - Notice the use of the ternary operation ?:

```
gallery.direction = function(e,which) {
var x = e.pageX - which.offset().left;
gallery.scroll = (x >= gallery.width / 2) ? ">" : "<";
}
```

- Two parameters: the event and the trigger
- Step 7: Initializing the gallery

```
gallery.init = function() {
  $(gallery.trigger)
  .mouseout(function() {gallery.scroll = false;})
  .mousemove(function(e) {gallery.direction(e,gallery.trigger);})
  .mouseover(function(e) {
      gallery.direction(e,gallery.trigger);
      gallery.slide();
  });
}
```

  - This sets the event handlers, which in turn set the parameters for the sliding animation

12. Navigating and Organizing Content

12.1. Creating Rich Internet Applications (RIA)

- Static content is a shrinking part of the Web
  - More and more fully featured, highly functional, and impressive looking applications are sprouting up every day
  - jQuery allows you to create such Rich Internet Applications
- RIA need a user interface
  - Grouping content logically
  - Easy access with drop-down menus
  - Tabbed interfaces
  - Sliding panels
  - Tooltips
  - Accordion controls

12.2. Menus

- While we have created menus before, they have been no more than just simple, top-level navigation panes
- jQuery lets you create more intricate menus
  - Collapsible menus with indicators and hover effects
  - Drop-down menu
- In this chapter you will create a cross-browser compatible drop-down menu
  - By adding a little jQuery on top of the CSS, you will make it a little sleeker!
- Step 1: The menu markup

```
<ul id="menu">
  <li><a href="#">News</a>
      <ul class="active">
          <li><a href="#">Weekly promotions</a></li>
          <li><a href="#">Popular books</a></li>
          <li><a href="#">New books</a></li>
      </ul>
  </li>
  <li><a href="#">Book information</a>
      <ul>
          <li><a href="#">Book summaries</a></li>
          <li><a href="#">Book reviews</a></li>
  ⋮
```

  - The menu consists of multi-leveled unordered lists
- Step 2: Styling the menu with CSS

```
#menu {
position: absolute;
top: 10px;
left: 10px;
z-index: 1;
}
#menu, #menu ul {
padding: 0;
margin: 0;
list-style: none;
}
#menu li {
float: left;
background: #C0D0D0;
}
#menu li:hover {
background: #826C4B;
}
```

```
#menu a {
display: block;
padding: 5px;
width: 150px;
text-decoration: none;
}
#menu li ul {
position: absolute;
left: -999em;
width: 160px;
}
#menu li:hover ul,
  #menu li ul:hover {
left:auto;
}
```

- The menu will already be fully functional!
  - No JavaScript required
  - Based on the Suckerfish drop-down menu
    - http://www.alistapart.com/articles/dropdowns
- Step 3: Add jQuery for smoother animation

```
$('#menu li ul').css({
  display: "none",
  left: "auto"
});
$('#menu li').hover(function() {
      $(this)
      .find('ul')
      .stop(true, true)
      .slideDown('fast');
  }, function() {
      $(this)
      .find('ul')
      .stop(true,true)
      .fadeOut('fast');
});
```

- When JavaScript is enabled, jQuery takes over the CSS behaviour
- The menu can perhaps feel too eager to activate
  - The moment the mouse hovers over the element, the effects spring to life...
  - With menus, a false start can appear unnecessary and distracting
  - You should delay the effect until you're sure the user want to activate the menu
- Use the Hover Intent plugin
  - Can be used instead of the standard hover()
  - http://cherne.net/brian/resources/jquery.hoverIntent.html
  - Calculates the speed of the mouse pointer, and only lets the effects kick in if the mouse slows down enough
  - This lets the plugin think the user intends to stop there

```
$('#menu li').hoverIntent(function() {
⋮
}, function() {
⋮
});
```

12.3. Accordion Menus

- In accordion menus, the expansion of one area leads to the contraction of another
  - Accordions are usually fixed so that one the areas must be visible at all times
- You can easily create an accordion yourself, but jQuery UI offers you an accordion with many options

- Comes with a cost of file size and bandwidth use...
- ... but you don't have to implement it yourself

- Include jQuery UI and its theme, and then...

```
<div id="accordion">
  <h3><a href="#">News</a></h3>
  <div>
    News content here
  </div>
  <h3><a href="#">Books</a></h3>
  <div>
    Books content here
  </div>
  ⋮
```

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

12.4. Tabs

- Tabs provide a way to group content logically
  - Allow you to break content into multiple sections that can be swapped in order to save space
- You can create tabs easily with jQuery, but you will not be the first to attempt this...
- jQuery UI includes a very feature-rich tab widget

- Include jQuery UI and its theme, and then...

```
<div id="tabs">
 <ul>
   <li>
     <a href="#news">
       <span>News</span>
     </a>
   </li>
   ⋮
 </ul>
 <div id="news">
   <p>News information here</p>
 </div>
 ⋮
</div>
```

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

- jQuery UI tabs have many options

```
$('#info').tabs({
  event: 'mouseover',
  fx: {
      opacity: 'toggle',
      duration: 'fast'
  },
  spinner: 'Loading...',
  cache: true
});
```

  - The user only needs to hover to change tabs
  - A fade animation will fast fade the tabs when switching
  - Spinner defines the HTML to show when the tab is loading
  - Cache keeps a copy of the tab after it is loaded

- jQuery UI tabs can also be controlled with code

```
$('#info').tabs().tabs('rotate', 3500);
```

- Instructs jQuery to cycle through the tabs every 3.5 seconds

12.5. Tooltips

- A tooltip is an interface component that appears when the user hovers over a control
  - Most browsers already have them with the title attribute for a link or an alt attribute for an image
- You can replace the browser's default tooltips with jQuery
- Step 1: Reuse the title attribute

```
<a href="#" title="Check out all the … " class="promo">Promotions</a>
```

  - This lets the user see default tooltips when JavaScript is disabled
- You will probably want more advanced tooltips that allow to add any markup and even images
- Step 2: Styling the tooltips with CSS

```
.tooltip {
  display: none;
  position: absolute;
  border: 1px solid #333;
  background-color: #ffed8a;
  padding: 2px 6px;
}
```

- Step 3: Turning the tooltip on and off, and positioning with jQuery

```
$('.promo').hover(function(e) {
  var titleText = $(this).attr('title');
  $(this)
  .data('tipText', titleText)
  .removeAttr('title');
  $('<p class="tooltip"></p>')
  .text(titleText)
  .appendTo('body')
  .css('top', (e.pageY - 10) + 'px')
  .css('left', (e.pageX + 20) + 'px')
  .fadeIn('slow');
}, function() {
  $(this).attr('title', $(this).data('tipText'));
  $('.tooltip').remove();
}).mousemove(function(e) {
  $('.tooltip')
  .css('top', (e.pageY - 10) + 'px')
  .css('left', (e.pageX + 20) + 'px');
});
```

13. Code Practices

13.1. Cleaner jQuery

- JavaScript is a wonderful language!
  - But you have to follow some good coding practices
    - Encapsulation, reuse, minimize inline coding, using code libraries...
  - Adding Ajax functionality is not really an issue with jQuery
    - But you want to avoid to create a mess of unmaintainable code spaghetti
  - As the jQuery components and effects grow more complex, we will need to structure our code
- Tip 1: Add code comments as documentation
  - Make your code more reusable and maintainable

```
// Assign the value '3' to the variable 'count':
var count = 3;
```

```
/* An example of
a multiline
comment */
var count = 3;
```

- Tip 2: Encapsulate data in map objects

```
var id = $('input#id').val();
var name = $('input#name').val();
var age = $('input#age').val();
// prefer mapping as shown below
var data = {
  type: 'person',
  id: id,
  name: name,
  age: age
}
```

  - The data gets encapsulated, and you can pass it around where needed
  - To access the data, use a dot (.)
- Tip 3: Namespacing your code
  - Take the previous tip, and now add functions to it!
  - You can then execute the function on the object with the dot (.)
  - By giving a scope to your functions, their names will not conflict with other JavaScript on the page
  - Namespace names should be unique, short and helpful

```
function exclaim () {
alert("hooray");
}
exclaim();// hooray
```

```
var AMAZON = {};
AMAZON.exclaim = function () {
alert("hooray");
};
```

```
var AMAZON = {
name: "My Bookshop",
exclaim: function() {
  alert("hooray");
}
};
```

- Tip 4: Use scope effectively
  - Scope refers to the area where a variable exists

- Important when using callback methods with Ajax
  - These methods will usually run in a different context
- You will need to use closures!

```
var WIDGET = {};
WIDGET.delay = 1000;
WIDGET.run = function() {
alert(this.delay); // 1000 … good!
$(p).click(function() {
  alert(this.delay) // undefined! bad!
});
};
```

```
var WIDGET = {};
WIDGET.delay = 1000;
WIDGET.run = function() {
      alert(this.delay); // 1000 … good!
      var _widget = this;
      $(p).click(function() {
            alert(_widget.delay) // 1000 … yes!
      });
};
```

13.2. Client-side Templating

- Text on pages is likely to change
  - When using Ajax
  - When data is dynamic, based on what can be found in a database
- The simplest way to update the text is to replace the entire contents

```
$('#cartitems').html("<p>You have " + cart.items.length + " items in your cart.</p>");
```

- But for more elaborate contents...
  - You would need lots of concatenation and string manipulation
  - You will run into trouble when creating complex tables...
- The solution: client-side templating
  - Provide hooks in your HTML content

```
<div id='overlay'>
<p>You have <span id='num-items'>0</span> items in your cart.</p>
<p>Total cost is $<span id='total-cost'>0</span></p>
</div>
```

  - Update the data with jQuery as required

```
$(this).find('#num-items').text(cart.items.length);
$(this).find('#total-cost').text(cart.getTotalCost());
```

- You could also create an element that acts as a template
  - This object can then be copied and edited whenever you need it

```
<table id="cart">
      <thead>
          <tr>
              <th>Name</th>
              <th>Qty.</th>
              <th>Total</th>
          </tr>
      </thead>
      <tr class="template" style="display:none;">
          <td><span class="item_name">Name</span></td>
          <td><span class="item_qty">Quantity</span></td>
```

```
            <td><span class="item_total">Total</span>.00</td>
        </tr>
</table>
```

- Notice the row with display:none that will be used as template

- This helper function will do the templating for us

```
function template(row, cart) {
  row.find('.item_name').text(cart.name);
  row.find('.item_qty').text(cart.qty);
  row.find('.item_total').text(cart.total);
  return row;
}
```

- We can now use the template and add new rows to the DOM with new data inserted

```
var newRow = $('#cart .template').clone().removeClass('template');
var cartItem = {
  name: 'The Crippled God',
  qty: 1,
  total: 450
};
template(newRow, cartItem)
  .appendTo('#cart')
  .fadeIn();
```

13.3. Feature Detection

- Browser sniffing... is bad!
  - It is the process of using JavaScript to figure out which version of which web browser the user is browsing with
  - Once you know the browser, you can work around any known bugs that exist in it, to make your pages consistent...
  - This technique has become unreliable!
    - Old browsers are updated with patches
    - New versions are released
    - Completely new browsers are introduced...
  - Workaround code will quickly break or become redundant
- Use a library such as Modernizr to detect features

14. Adding Ajax

14.1. What is Ajax?

- Ajax stands for Asynchronous JavaScript and XML
  - But in fact stands for any technique or technology that lets a user's browser interact with a server without disturbing the existing page
  - Can be finicky to implement, unless you are using jQuery!
- Ajax mimics the feel of a desktop application
  - You can fire requests from the browser to the server without page reload
  - You can update a part of the page while the user continues on working
  - Gives the user a more responsive and natural experience
- jQuery makes sure you do not have to worry about browser differences
  - There are only a handful of Ajax functions in jQuery
  - Most of those are wrappers to help you out!

14.2. Loading Remote HTML

- jQuery has a load() method
  - This method will let you grab an HTML file off the server
  - Its contents can then be inserted in the current web page
  - You can load either static HTML files, or dynamic pages that generate HTML output

```
$('div:first').load('page.html');
```

  - This dynamically inserts the entire contents of the page.html (anything inside the tags) into the first div
  - You can use any selector to decide where it should go
  - You can even load into multiple locations at the same time
- Be careful!
  - Do not mistake this method for the load() event, which fires when an object (window or image) has finished loading
  - load() is limited to content coming from the same domain to prevent cross-site scripting
- Lets use the load() operation to intercept hyperlinks and load the information in the same page
- Step 1: Create separate HTML pages for basic content

```
<body>
      <h1>The Crippled God</h1>
      <p id="text">
          Savaged by the K'Chain Nah'Ruk, the Bonehunters    ...
      </p>
</body>
```

  - You will need one page per content link...
  - If you have lots of entries, you would load the data from a database, by passing a query string to a server-side script
- Step 2: Creating the central page with content links

```
<ul id="books">
      <li><a href="book01.html">Book 01</a></li>
      <li><a href="book02.html">Book 02</a></li>
      <li><a href="book03.html">Book 03</a></li>
      <li><a href="book04.html">Book 04</a></li>
</ul>
<div id="summary">
      Click on a book to found out more!
</div>
```

- Notice the div under the list of links
- This is where the content will be put after the Ajax call

- Step 3: Intercept the links and do some Ajax

```
$('#books a').click(function(e) {
  var url = $(this).attr('href') + ' #text';
  $('#summary').load(url);
  e.preventDefault();
});
```

- All the links are prevented to do their default action
- The original destination (href) from the link is passed to the load() function

- You may want to only load a part of the page, and not the whole contents...
  - Notice the selector in the code above and below

```
$('#summary').load('books.html div:first');
```

- If you want to show the page is still loading, you can use a quick solution

```
$('#summary').html('loading…').load(url);
```

- The load() function can further be tweaked
  - You could pass some information along to the server

```
$('div#results').load('search.php', 'q=jQuery&maxResults=10');
```

  - If you pass a string, jQuery will execute a GET request
  - If you pass an object, jQuery will execute a POST request

- You can also perform some extra processing when a request has finished by passing a callback function

```
$('div#result').load('feed.php', function(data, status, response) {
  // post-processing here
});
```

  - The data and callback function are optional

- As the content is loaded, you would like some effects to happen

```
$('p#text').mouseover(function() {
  $(this).css('background-color','yellow');
});
```

  - The code will fail to work, as the p#text does not exist, but is added later on with our Ajax call

- jQuery adds the on() method that binds on current but also on future elements

```
$('p#text').on('mouseover', function() {
  $(this).css('background-color', 'yellow');
});
```

  - If you can set on, you can also set off()

```
$('p#description').off('mouseover');
```

  - The off() operation will let you stop the event from occurring later

14.3. Fetching JSON Data

- For data interchange in Ajax, JSON (JavaScript Object Notation) has become very popular format
  - It is a lightweight alternative to XML
  - Data is structured as plain JavaScript objects

    - No parsing or interpretation is required, it is immediately ready to use in your scripts
- jQuery provides a $.getJSON() function
    - This accepts a URL and a callback function
    - The URL should point to a service returning JSON data
    - If the data is in JSONP format, you could even do cross domain requests

```
$.getJSON(
'http://feeds.delicious.com/v2/json/tag/books?callback=?',
function(data) {
  alert('Fetched ' + data.length + ' items!');
});
```

    - You will have to read the API at http://delicious.com/help/api
- The data returned from $.getJSON() can be difficult to decipher...
    - Different services return different data
    - The API documentation is not always of quality or even clarity
    - You will mostly have to figure it out for yourself
    - One way to examine the result, is to type the URL directly in your web browser and inspect the resulting JSON directly

14.4. Ajax with jQuery

- All of jQuery's Ajax functions are simply wrappers around the $.ajax() method
    - $.ajax() is the hearth of jQuery Ajax functionality
    - Is the most powerful and has the most options
- While the $.ajax() method has a more complex syntax, it is still easy to use

```
$.ajax({
  type: 'GET',
  url: 'getDetails.php',
  data: { id: 142 },
  success: function(data) {
      // grabbed some data!
  };
});
```

    - The above code specifies a GET request
    - The complexity emerges from the number of possible options you can provide
- jQuery allows you to specify global Ajax settings
    - This prevents you from having to type them for each Ajax call
    - Use the $.ajaxSetup() method

```
$.ajaxSetup({
type: 'POST'
url: 'send.php',
timeout: 3000
});
```

    - This really simplifies the code for actually sending the request, while still giving you the possibility to override them

```
$.ajax({
data: { id: 142 }
});
```

14.5. Loading External Scripts

- The $.getScript() function will load and execute a JavaScript file via Ajax
  - This decreases the download time as your application becomes larger
  - It keeps your application snappy and reactive
  - You will only load the script when you need it, starting with a minimum of code, then pulling in further files later on

```
$.getScript('http://view.jquery.com/trunk/plugins/color/jquery.color.js',
      function() {
          $('body').animate({'background-color': '#fff'}, 'slow');
      }
);
```

14.6. GET and POST Requests

- jQuery Ajax packs a few helpers for performing GET and POST requests
  - Those are simple wrappers around the $.ajax() but are more convenient to use
  - You just need to select the URL and any data you want to push

```
$.get(url, data, callback, dataType);
$.post(url, data, callback, dataType);
```

    - Only the URL is a required parameter
    - The data parameter should contain any data that needs to be sent
    - The callback we specify will be passed the response data and status
    - The type parameter lets you specify the data type: xml, html, script, json, jsonp, or text

```
$.get("getInfo.php", function(data) {
alert("got your data:" + data);
});
$.post("setInfo.php", {id: 2, name: "The Crippled God"});
```

- $.get() and $.post() are very easy and convenient, but...
- Beware!
  - These methods are supposed to be for quick, one-off requests
  - The callback function only fires when everything goes well
  - Unless you're watching with a global event handler you'll never know if a call fails
  - For most production-level functionality, the $.ajax() method is the only solution...

14.7. jQuery Ajax Events

- When making Ajax requests with jQuery, a number of events are fired
  - These can be handled anytime for some extra processing
  - Example: adding a "loading" message when a request begins and removing it when it concludes, or handling errors

- There are two types of Ajax events in jQuery
  - Local events: apply only to individual Ajax requests
    - You handle local events as callbacks

  - Global events: are broadcast to any handlers that are listening
    - To implement functionality that should apply to all requests

- Ajax Events:
  - error and ajaxError
  - succes and ajaxSuccess
  - complete and ajaxComplete
  - beforeSend and ajaxSend

    - ajaxStart and ajaxStop

- A local event

```
$.ajax({
  url: "test.html",
  error: function() {
      alert('an error occurred!');
  }
});
```

- A global event

```
$("#msg").ajaxError(function(event, request, settings) {
  $(this).html("Error requesting page " + settings.url + "!");
});
```

    - You add the event on the DOM node where you want to show the error
    - The handler will be called when the Ajax call fires an error

---

14.8. Sending Form Data

- jQuery allows you to quickly collect data from forms and send them with Ajax
    - You could read all the field values and concatenate them, but...
    - ...why would you do that if you have the serialize() method!

- The serialize() method sucks up input fields that have a name attribute attached to them

```
<form>
  <input type="text" name="name" />
  <input type="text" name="tags" />
  <input type="hidden" name="id" />
  <input type="button" value="update" />
</form>
```

    - Make sure you name your fields

```
var form_data = $("form").serialize();
```

    - The result:

```
name=Crippled+God&tags=books+fantasy&id=8
```

- You can also use the serializeArray() method
    - This returns an object containing key/value pairs of the form fields

- You can now send the data to the server

```
var form_data = $('form').serialize();
$.post(url, form_data, function() {
  $('#status').text('Update successful!');
});
```

    - There is no way to tell if this did succeed or not...
    - Better use the $.ajax() method instead

- Use the jQuery Ajax events to react on possible errors

```
$.ajax({
  type: "POST",
  url: url,
  data: form_data,
  beforeSend: function() {
      $('#ajaxDetails').addClass('progress');
  },
  error: function() {
```

```
        $('#status').text('Update failed—try again.').slideDown('slow');
    },
    success: function() {
        $('#status').text('Update successful!');
    },
    complete: function() {
        $('#ajaxDetails').removeClass('progress');
        setTimeout(function() {
            $('#status').slideUp('slow');
        }, 3000);
    }
});
```

- You can check the data being send with the JavaScript console
- You should probably have to create some server-side code...

15. Enhancing Forms

15.1. Form Validation

- While form validation often seems boring, it is essential
  - Proper and well designed forms influence the user's perception of your web site
- You should use client-side validation only to help the user
  - You cannot rely on it to prevent certain data being sent
  - Remember that the user can always disable JavaScript... so they can submit any data they want!
  - The server should thoroughly validate the sent data
- jQuery provides convenient methods to access and set form values
  - Using the :input filter

```
$('#form:input').css('background-color', 'lemonchiffon')
```

  - Other filters:
    - :text, :password, :radio, :checkbox, :submit, :button, :image (for image buttons), :file, :enabled, :disabled, :checked and :selected
- jQuery filters can be useful when doing conditional logic

```
if($(this).is(':checked')){}
```

- Once selected, you should use the value of the field, and validate
  - Checking if any text boxes in the form are empty

```
$(':submit').click(function(e) {
$(':text').each(function() {
  if ($(this).val().length == 0) {
      $(this).css('border', '2px solid red');
  }
});
e.preventDefault();
});
```

  - The val() method also works on select boxes and radio buttons

```
$(':radio[name=sex]').change(function() {
alert($(this).val());
});
```

- The change() and blur() events can be useful for inline validation
  - Testing for empty fields whenever the user moves to the next field

```
$(':input').blur(function() {
if ($(this).val().length == 0) {
  $(this).addClass('error')
  .after('<span class="error">This field must … </span>');
}
});
$(':input').focus(function() {
$(this).removeClass('error').next('span').remove();
});
```

  - While the example only checks if the fields are filled in, you can do any type of validation this way
- Warning: avoid over-validation!
  - The more rules you add, the more likely you'll have forgotten an edge case
  - Offer your user guidance, sample inputs, hints... instead of scaring away your users with lots of error messages
- You should also make use of the submit() event
  - Use this instead of a button click, as the user may use the Enter key to submit the form
  - If you return false, the form will not be submitted

```
$("form").submit(function() {
        var error = false;
        $(this).find(":text").each(function() {
        if ($(this).val().length == 0) {
            alert("Textboxes must have a value!");
            $(this).focus();
            error = true;
            return false; // only exits the "each" loop
            }
        });
        if (error) {
            return false;
        }
        return true;
});
```

15.2. Using the Validation Plugin

- There are lots of edge cases to consider when designing form validation
  - And you would like your validation to be bullet-proof!
  - You have to know regular expressions to verify the syntax of email addresses, credit cards...
  - Building your own inline validation is a difficult endeavour!
- The Validation plugin solves a lot of these problems
  - It lets you add sophisticated and customizable inline validation to your forms with minimal effort
  - You can download the Validation plugin from the following website
  - http://bassistance.de/jquery-plugins/jquery-plugin-validation
- Example form for testing the plugin

```
<form action="">
  <div>
      <label for="name">Name:</label>
      <input name="name" id="name" type="text"/>
  </div>
  <div>
      <label for="email">Email:</label>
      <input name="email" id="email" type="text"/>
  </div>
  <div>
      <label for="website">Web site URL:</label>
      <input name="website" id="website" type="text" />
  </div>
  <div>
      <label for="password">Password:</label>
      <input name="password" id="password" type="password" />
  </div>
  <div>
      <label for="passconf">Confirm Password:</label>
      <input name="passconf" id="passconf" type="password" />
  </div>
  <input type="submit" value="Submit!" />
</form>
```

- You just need to call validate() on a selection of our form
  - You then pass options, such as rules, to validation the input
  - There are many predefined rules available, and you can define your own
  - The plugin will add a new label element after each form field to contain the error message, with CSS class of error

```
$('#signup form').validate({
rules: {
    name: {required: true},
    email: {required: true,
        email: true
```

```
    },
    website: {url: true},
    password: {minlength: 6,
        required: true
    },
    passconf: {equalTo: "#password"}
},
success: function(label) {
    label.text('OK!');
}
});
```

15.3. Form Hints

- Adding hints to forms helps users fill in the form correctly
- You could, for example, display the remaining characters next to the form field
  - Think about limiting the number of characters to 140 "à la Twitter"

```
$('.maxlength')
.after("<span></span>") // add a span
.next() // move into it
.hide() // hide it
.end() // come back to the field
.keypress(function(e) { // add a key press event
    var current = $(this).val().length;
    if (current >= 140) { // check the length
        // accept delete and backspace
        if (e.which != 0 && e.which != 8){
            e.preventDefault();
        }
    }
    $(this).next().show().text(140 - current);
});
```

- Another trick, is to put the label for a field inside the input itself, to decrease the space the form takes up
  - When users move their focus to it, the label vanishes, allowing them to type
  - If they leave the field empty and move away, the label returns

```
$('input.clear').each(function() {
    $(this)
    .data('default', $(this).val())
    .addClass('inactive')
    .focus(function() {
        $(this).removeClass('inactive');
        if ($(this).val() == $(this).data('default') || '') {
            $(this).val('');
        }
    })
    .blur(function() {
        var default_val = $(this).data('default');
        if ($(this).val() == '') {
            $(this).addClass('inactive');
            $(this).val($(this).data('default'));
        }
    });
});
```

15.4. Inline Editing

- The most common approach is to allow the editing of non-form elements
  - Elements such as paragraph tags and title tags, for example
  - When the user clicks on the tag, the contents are replaced with a text box or textarea that the user can

interact with
- When the task is complete, the original tags are replaced with the new content
- This will require some markup as well as Ajax

- Step 1: The editable titles and paragraphs

```
<h3 id="book-01-title" class="editable">The Crippled God</h3>
<p id="book-01-summary" class="editable-area">
Savaged by the K'Chain Nah'Ruk, the Bonehunters march ...
</p>
```

- Step 2: The CSS styles

```
.over-inline {
background-color: #FFFACD;
}
body .active-inline {
background-color: inherit;
}
```

  - These will allow us to see if an element is editable or not

- Step 3: The events
  - The hover event, to apply the above CSS
  - The click event, to start editing
  - The blur event, to stop editing

```
$(".editable, .editable-area").hover(function() {
$(this).toggleClass("over-inline");
})
.click(function(e) {
})
.blur(function(e) {
});
```

- Step 4: Starting the inline editing

```
var $editable = $(this);
if ($editable.hasClass('active-inline')) {
  return;
}
```

  - The goal of the if-test is to check if we are already editing
    - We do not want to replace our edit box with another edit box...

- Step 5: Grabbing the contents of the element, then removing it

```
var contents = $.trim($editable.html());
$editable
.addClass("active-inline")
.empty();
```

  - The empty() method removes the children
  - Adding the CSS class indicates the editing is in progress

- Step 6: Insert the text box or textarea

```
// determine what kind of form element we need
var editElement = $editable.hasClass('editable') ?
  '<input type="text" />' : '<textarea></textarea>';
// replace the target with the form element
$(editElement)
.val(contents)
.appendTo($editable)
.focus()
.blur(function(e) {
  $editable.trigger('blur'); // manually fire the event
});
```

- Trigger shows to the reader you want to manually fire the event
- As the input is blurred, you tell the original element you are finished editing

- Step 7: End of the inline editing
  - Grab the value of the element
  - Send it to the server with $.post(), while adding a "Saving..." message
  - When the posting is done, replace with the updated value

```
.blur(function(e) {
var $editable = $(this);
var contents = $editable.find(':first-child:input').val();
$editable
.contents()
.replaceWith('<em class="ajax">Saving … </em>');
$.post('save',
{id: $editable.attr('id'), value: contents},
function(data) {
$editable
.removeClass('active-inline')
.contents()
.replaceWith(contents);
});
});
```

- Note: the POST has been faked with the 'save' file...

15.5. Autocomplete

- Autocomplete lets users quickly find and select from a pre-populated list of values
  - Is added to an input field
  - Leverages searching and filtering
- jQuery UI has all the functionality and styling we need
  - http://jqueryui.com/autocomplete/
- You can pull data in from a local and/or a remote source
  - Local is good for small data sets (like an address book with 50 entries)
  - Remote is necessary for big data sets, like a database with hundreds or millions of entries to select from
- Autocomplete can be customized to work with various data sources
  - An Array with local data
  - A String, specifying a URL, which will be called with Ajax
  - a Callback
- Expected data format
  - An Array of Strings: ["Choice1", "Choice2"]
  - An Array of Objects with label and value properties: [{ label: "Choice1", value: "value1" }, ...]
- Step 1: Add the field to your page

```
<label for="titles">Search Books:</label>
<input type="text" id="titles"/>
```

- Step 2: Adding the autocomplete with source

```
var titles = ['The Crippled God', 'Mistborn', 'The Girl with...', ...];
$('#titles').autocomplete({
  source: titles,
  autoFill: true,
  selectFirst: true,
  width: '240px'
```

```
});
```

- autoFill gives a nice type-ahead effect
- matchContains will cause a match substrings of words
- The autocomplete also fires a result() event

15.6. Form Controls: Date Picker

- jQuery UI adds more capabilities for interaction
- A popular control is a date picker
  - If you ever tried to write one yourself, you will be inclined to avoid ever doing it again
  - It is a lot of hard work, and can be frustrating when done wrongly
- jQuery UI contains a highly customizable and fully-featured date picker control
- A date picker is easy to add

```
<input type="text" id="date" />
```

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

  - While this date picker offers nothing new, it is fully packed with features such as localization, date formats, multiple selection, keyboard navigation and much much more!

- Some options:

```
$('#date').datepicker({
  showOn: 'button', // when will the calendar pop up
  buttonText: 'Choose a date',
  buttonImage: 'calendar.png', // show an icon
  buttonImageOnly: true, // show only the icon
  numberOfMonths: 2, // show two months of days
  maxDate: '0d', // specify max range of choices (today)
  minDate: '-1m -1w', // specify min range of choices
  showButtonPanel: true
});
```

  - Even though we specify a maximum date, users can still select a data outside the range by typing the date manually
  - You will need to add validation on the server side!
  - The date picker will only help the users to pick valid options

- jQuery UI also adds functions to help manipulating dates
  - $.datepicker.iso8601Week(): returns the week of the year
  - $.datepicker.parseDate(): pass a string, returns a date
  - $.datepicker.formatDate(): formats a date in a chosen format

15.7. Form Controls: Slider

- Lets your users easily select ranges of values
- jQuery UI lets you add the slider on select boxes
  - The jQuery UI slider has lots of options!
- Step 1: Adding the basic form

```
<div id="price-range">
    <form>
        <label for="min">Minimum Price:</label>
        <select id="min">
```

```
            <option value="0">0</option>
            <option value="10">10</option>
            ⋮
            <option value="90">90</option>
        </select>
        <br/>
        <label for="max">Maximum Price:</label>
        <select id="max">
            <option value="10">10</option>
            <option value="20">20</option>
            ⋮
            <option value="100">100</option>
        </select>
    </form>
</div>
```

- Step 2: Initiate the slider

```
var max = $('#max').val();
var min = $('#min').val();
var rangeSlider = $('<div></div>')
.slider({
    min: 0,
    max: 100,
    step: 10,
    values: [min, max], // the default values
    range: true, // for a slider with more handles
    animate: true,
    slide: function(e,ui) { // event handler
        $('#min')
        .val(ui.values[0]);
        $('#max')
        .val(ui.values[1]);
        showBooks(); // to filter the books
    }
})
.before('<h3>Drag the slider to filter by price:</h3>');
$('#price-range').after(rangeSlider).hide();
```

- Step 3: Filter the books

```
function showBooks() {
  var min = $('#min').val();
  var max = $('#max').val();
  $('.data tr').each(function() {
      var price = parseInt($(this).find('td:last').text()
          .substring(1)); // strips the $ or € signs
      if (price >= min && price <= max) {
          $(this).fadeIn();
      } else {
          $(this).fadeOut();
      }
  });
}
```

  - The above operation filters on the client side
  - Sliders are great, because they are intuitive
    - Users will know how to use them without being told
    - Lets the users easily and visually filter the data

15.8. Form Controls: Progress Bar

- A progress bar is one of the most recognizable messages a user can see
  - It effectively shows how far through a long-running process we are
  - It also shows how far we have to go

- jQuery UI has a progress bar widget
- Step 1: Adding an empty div

```
<form>
    <fieldset>
        <legend>Type your message</legend>
        <textarea id="text" rows=""></textarea>
        <div id="console">
            <div id="bar"></div>
            <div id="count">0</div>
        </div>
        <input type="submit" value="Send!" />
    </fieldset>
</form>
```

- Step 2: Adding the progress bar

```
$('#bar').progressbar();
```

- Step 3: Updating the progress bar as the user types

```
$('#text')
.val('')
.keyup(function(e) {
  var characters = 140;
  var value = $('#text').val();
  var count = value.length;
  if (count <= characters) {
      $('#bar').progressbar('value', (count / characters) * 100);
      $('#count').text(count);
  } else {
      $('#text').val(value.substring(0,characters));
  }
});
```

  - The progress bar is updated with a percentage value
  - When the maximum is reached, the user is not allowed to type any more characters

15.9. Dialogs and Notifications

- Dialogs lets you add information that pops up
  - Can be used for many reasons: validation messages, status updates, error handling messages...
  - They can be overwhelming and annoying, so only use them when interaction is essential
- Step 1: Markup for a simple modal dialog

```
<div id="overlay">
  <div id="blanket"></div>
</div>
<!-- the dialog contents -->
<div id="eula" class="dialog">
  <h4>End User License Agreement</h4>
  ⋮
  <div class="buttons">
      <a href="#" class="ok">Agree</a>
      <a href="#" class="cancel">Disagree</a>
  </div>
</div>
```

  - The CSS plays a big part on how effective the dialog looks
- Step 2: The CSS styling

```
#overlay {
    display:none;
    top: 0;
```

```
    right: 0;
    bottom: 0;
    left: 0;
    margin-right: auto;
    margin-left: auto;
    position: fixed;
    width: 100%;
    z-index: 100;
}
#blanket {
    background-color: #000000;
    top: 0;
    bottom: 0;
    left: 0;
    display: block;
    opacity: 0.8;
    position: absolute;
    width: 100%;
}
.dialog {
    display: none;
    margin: 100px auto;
    position: relative;
    width: 500px;
    padding: 40px;
    background: white;
    -moz-border-radius: 10px;
}
```

- Step 3: Opening the dialog

```
function openDialog(selector) {
  $(selector)
  .clone()
  .show()
  .appendTo('#overlay')
  .parent()
  .fadeIn('fast');
}
```

```
$("#eulaOpen").click(function() {
  openDialog("#eula");
});
```

  - We clone the dialog, so we can remove it later, and keep the original one available

- Step 4: Closing the dialog

```
function closeDialog(selector) {
  $(selector)
  .parents("#overlay")
  .fadeOut('fast', function() {
      $(this)
      .find(".dialog")
      .remove();
  });
}
```

  - When closing, we remove the cloned dialog

- Step 5: You should add different handlers for the buttons

```
$('#eula')
.find('.ok, .cancel')
.on('click', function() {
  closeDialog(this);
})
.end()
.find('.ok')
.on('click', function() {
  // Clicked Agree!
```

```
})
.end()
.find('.cancel')
.on('click', function() {
  // Clicked disagree!
});
```

- The on() method will keep the button handlers available no matter how often we clone and delete the dialogs

- Creating dialogs this way can be tiring... so use jQuery UI!
- jQuery UI Dialogs have all the bells and whistles, and offer many options to configure
- Step 1: Easy dialog markup for jQuery UI

```
<div id="dialog" title="Are you sure?">
  <p>You have just rated this book 0 out of 5 stars...</p>
  <p>Did you read it before providing a rating?</p>
</div>
```

- Where are the buttons?

- Step 2: Applying the dialog and adding buttons

```
$('#dialog').dialog({
    autoOpen: false, // only open on event
    height: 280,
    modal: true,
    resizable: false,
    buttons: {
        'Continue': function() {
            $(this).dialog('close');
            // Submit Rating
        },
        'Change Rating': function() {
            $(this).dialog('close');
            // Update Rating
        }
    }
});
```

- Step 3: Adding the event to open the dialog

```
$('#rating-0').click(function() {
  $('#dialog').dialog('open');
});
```

- Pass the string 'open' to the dialog() method to display itself
- This is a nice way to communicate with the dialog after its initialization

16. Advanced Controls

16.1. Advanced Controls

- Most sites need some kind of administration module
  - This area is traditionally the most neglected part of the site
  - It is not visible to the general public, so less thought is put into usability and user experience
  - The rise of the Rich Internet Application is changing that!
  - Clients expect web-based systems to rival desktop applications in usability and design
- jQuery lets you create impressively rich and interactive administration modules
  - While you have to do most of the work, the need for administration elements will let you find a solution quickly
  - Use "cookbooks" with jQuery recipes to help you out!

16.2. Lists

- jQuery offers capabilities to manipulate and improve the use of lists
  - They can be used as menus, navigation panels, tag clouds, ...
  - jQuery UI's selectable behavior gives the user the ability to lasso any of the element's children to select them
    - http://jqueryui.com/selectable/
  - jQuery lets you create reusable sorters that will let you sort lists on the client side
  - You can create swappable lists, in which you can select multiple options from the list on the left to put them in the list on the right
  - You can easily add searching functionality by just filtering on queried objects

16.3. Trees

- Trees are hard to do well, and there are few situations where they make sense
  - A nested set of categories is a valid use of a tree structure
  - A more common one is representing a directory structure
- Trees are lists!
  - jQuery will let you make sure where you are in the list so you can open and close branches
  - Once the markup is created, dynamically assigned CSS will let you style and add icons to the tree
- Event delegation will limit the creation of event handlers on each part of the tree
  - A single event handler intercepts the click, and figures out on which element the click was aimed at

16.4. Tables

- While tables have been misused in the past for hacking layouts into a website, we now use CSS for that
- Tables can be used for their original purpose: to show tabular data
  - jQuery lets you create fixed table headers, so when the user scrolls, the headers follow along
  - jQuery can also dynamically repeat the header at regular intervals, which is nice when printing the table
  - With jQuery, you can sort, filter, search, page, resize columns, and edit rows on a table
  - Use jQuery to create selectable rows in the table with column of checkboxes
  - Let the user select multiple rows by holding the Shift-key
  - If you need lots of features fast, use plugins!
    - http://www.datatables.net/index

17. Extending jQuery

17.1. Plugins

- Once you have a nice jQuery solution running, do not start using copy-paste!
  - This can let your beautiful web site degenerate into your worst JavaScript nightmare!
- The solution: create plugins!
  - We have used multiple plugins throughout the course
  - jQuery makes it easy to convert your existing code into a plugin
  - Once created, you can add options and callbacks
  - You can then include it in your page whenever you need it
- More information
  - http://learn.jquery.com/plugins/

17.2. Adding Methods

- jQuery provides a mechanism for extending and overwriting its core functionality
  - You can extend jQuery with your new functionality without creating a plugin, directly from inside your script
  - With $.fn.extend() you can pass an object that contains a new set of methods to extend jQuery
  - The net result is exactly the same as creating plugins, but also lets you overwrite default jQuery functionality
- Example:
  - Overwrite the trim() method to remove all spaces when clear = true

```
(function($) { // plugin shell
    var _trim = $.trim;
    $.extend({ // extending jQuery
        trim:function(text, clear) {
            if (clear) {return text.replace(/\s+/g,'');}
            return _trim.call(this, text);
        }
    });
})(jQuery); // end of plugin shell
```

17.3. Themes

- jQuery UI is very attractive to use, because...
  - ...out of the box the UI components look great!
- jQuery UI has a whole set of themes you can choose from
  - Take a look at the Theme Roller web application
    - http://jqueryui.com/themeroller/
  - This application lets you choose the theme you like
  - You can even roll your own theme by setting properties in the application and so modify every aspect of a jQuery UI theme
    - Changes will even be immediately reflected
  - Once finished, you can then download the packaged product

17.4. Other Extensions

- You can create your own selectors
- You can add your own custom events

- You can let jQuery react to non-standard events
  - Such as iPhone / Android / "the next big thing" events
- You can let treat standard JavaScript objects as jQuery objects
- You should make your own widgets as themeable as jQuery components
  - http://api.jqueryui.com/category/theming/

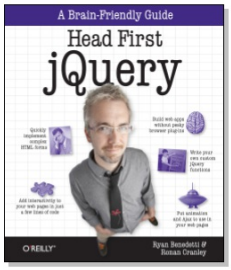
17.5. Next Steps

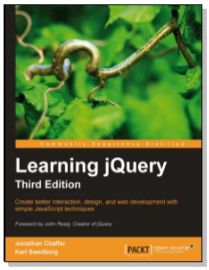
- Apply all what you have learned in a complete web application
  - Use jQuery UI to construct your page and forms
  - Add Ajax capabilities with server-side components in Java, PHP or .Net for fully functional and dynamic user interaction
  - Decide on effects that make the interactivity smooth and enjoyable
  - Add images and slide shows
  - Add plugins and advanced controls for an administration module
- Start learning more about jQuery!
  - It would be impossible to see everything in just a few days...
  - See the following chapter for very interesting references
  - Build up a library of code solutions that you can use in any project
- Do not wait until you have forgotten about this incredible JavaScript framework!

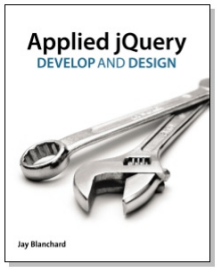
---

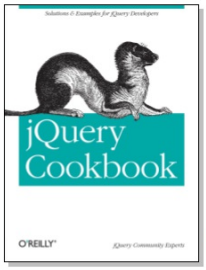
18. Appendix

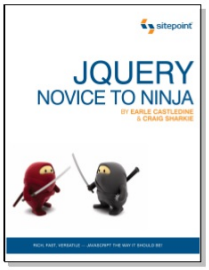
18.1. References











- Online resources
  - http://api.jquery.com
  - http://learn.jquery.com/
  - http://www.w3schools.com/jquery/default.asp
  - http://www.elated.com/articles/cat/jquery/

18.2. Development Tools

- To write jQuery code, you need:
  - A text editor: PSPad, Notepad++, Notepad
  - A browser: Google Chrome, Firefox
  - A web server: Apache, IIS, Tomcat
  - The jQuery library
- You will benefit from using browser development tools
  - Google Chrome
    - https://developer.chrome.com/devtools
  - Firebug in Firefox
    - http://getfirebug.com
  - Internet Explorer
    - http://msdn.microsoft.com/en-us/library/dd565628(v=vs.85).aspx
  - These tools will help you find errors, and have means for debugging your code

jQuery Rich Internet Applications Exercises

Text: EN

Software: EN

REALDOLMEN
to get there, together

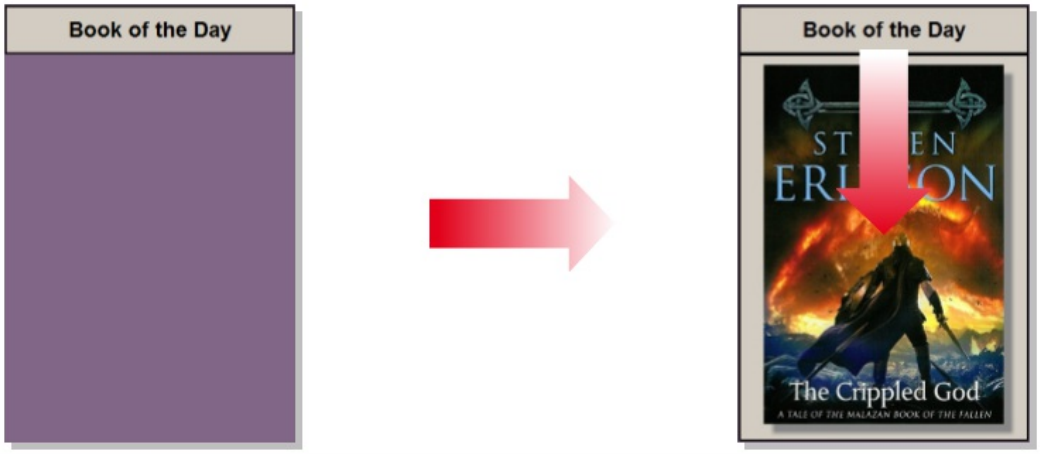
Table of contents

1. Introduction
    1.1. Exercise #01
2. Selecting
    2.1. Exercise #02
3. Decorating
    3.1. Exercise #03
4. Events and functions
    4.1. Exercise #04 : Update exercise #02 as follows
5. Modifying DOM
    5.1. Exercise #05
6. Adding Effects
    6.1. Exercise #06
7. Basic Animation
    7.1. Exercise #07
    7.2. Exercise #08
    7.3. Exercise #09
    7.4. Exercise #10
8. Resizing
    8.1. Exercise #11
9. Images slideshows
    9.1. Exercise #12
    9.2. Exercise #13
10. Navigation and Organizing Content
    10.1. Exercise #14
    10.2. Exercise #15
11. Adding Ajax
    11.1. Exercise #16
    11.2. Exercise #17
12. Enhancing Forms
    12.1. Exercise #18

1. Introduction

1.1. Exercise #01

- Create a page that slides down and fades in a picture when a user clicks on a part of the page
- Hints:
  - Use `<div>` elements with IDs
  - Use CSS to hide part of the page
  - Style the elements in any way you like (background, font-size, ...)



2. Selecting

2.1. Exercise #02

- Create a page that gives promotions to users
- Hints:
  - Add 4 sections with images using divs
  - When the user clicks on an image, a message is added under the image, giving the user a random promotion between 5 and 10
  - When the user clicks again, get rid of the old message, and create a new one

3. Decorating

3.1. Exercise #03

- Create a page containing a table
  - Use alternate colored rows
  - The rows containing a “Fiction” book, should have bold text
  - The word “Fiction” itself should be colored red
  - Clicking on a cell will colorize all rows that contain a cell with that same value yellow
  - Clicking on the same puts the table in its original state, while clicking on another cell re-colorizes the table
  - Developing Rich Internet Applications using jQuery

4. Events and functions

4.1. Exercise #04 : Update exercise #02 as follows

- Put your script code in a separate file
- Make sure the user can only click once to get a promotion code
- Replace the click handler function with a named function declaration
- Create a function that generates random numbers
- Use the function to generate random numbers to 1000
- Highlight the book the user is currently hovering over
- Choose one book at random that will contain the promotion
  - Add an empty span element
- Show which book contained the promotion, but tell the user if he got it right
  - Check if the element contains the hidden span
- Fade in the promotion code in a separate box

5. Modifying DOM

5.1. Exercise #05

- Create a page with a list of books
  - Add navigation buttons: cheap and restore
  - When cheap is clicked, do the following:
    - Remove all authors with expensive books
    - Replace the collection title with some custom text
    - Replace all sold-out books with “Sold Out!”
    - Add a green color to all the cheap books
  - When restore is clicked, put the list back to its original state

6. Adding Effects

6.1. Exercise #06

- Create a page with book summaries
  - Use progressive enhancement
    - The page should still be accessible to users who do not have JavaScript!
    - Add a “no-script” element to warn the users when they do not have JavaScript
    - Test by putting your script in comment, then reloading the page
  - The book summaries should be hidden behind a “spoiler” span tag
  - Add a “Let me read...” span tag to reveal the hidden summary
    - When hovering over “Let me read...”, change the background style
    - When clicking on “Let me read...”, fade out the tag and fade in the “spoiler”

7. Basic Animation

7.1. Exercise #07

- Adapt your page with book summaries
- Hide all contents of the book except from the first one
- Add a color animation on the book title when the page loads
- When clicking on a title, animate the height of the contents to toggle with an easeOutBounce
- The other book summaries must close when a title is clicked

7.2. Exercise #08

- Follow the instructions on the slides to add animated navigation on your page
- Change the animation by switching the easing
- Add some animation on the blob's color

7.3. Exercise #09

- Follow the steps in the slides to implement a floating navigation pane
- Use the previous exercise as starting point

7.4. Exercise #10

- Adapt your book summaries exercise to add custom scroll bars
- Start from exercise #06
- Add custom scroll bars for the main div
- Add custom scroll bars for the articles

8. Resizing

8.1. Exercise #11

- Adapt your previous exercise with scrollable articles
  - Remove the scrolling on your articles
  - Make your articles resizable

9. Images slideshows

9.1. Exercise #12

- Add a custom lightbox on your book summaries page
- Follow the instructions on the slides to create the lightbox
- Make use of console.log() if you notice something is not working right...

9.2. Exercise #13

- Implement the custom slideshow
- Follow the instructions on the slides
- Make sure your images all have the same size

10. Navigation and Organizing Content

10.1. Exercise #14

- Create a page with a drop-down menu
- Follow the instructions on the slides
- Include the Hover Intent plugin for improved interaction
- You can try to combine this with exercise #08 and #09

10.2. Exercise #15

- Add a jQuery accordion, tabs, and tooltips to your page
- You may need to restyle some elements...

11. Adding Ajax

11.1. Exercise #16

- Create a page that loads summaries with Ajax
  - When a link is clicked, the corresponding page should be loaded
  - The loaded page contains a paragraph with a mouseover event that should still function for future Ajax loads

11.2. Exercise #17

- Create a news page that fetches information from the delicious feed
  - Show the title, the author and add a link to the source article
  - You will need the $.getJSON() and $.each()
  - Append the results to the main content on your page

12. Enhancing Forms

12.1. Exercise #18

- Create a page with a form that will let you create a book
  - A book has
    - An isbn number of exactly 13 characters long
    - A title with a maximum of 100 characters
    - An author that can be chosen with an autocomplete
    - A publication date that can be chosen with a date picker
    - A small description with with a maximum of 500 characters
    - A number of pages that is chosen with a slider
  - All fields are required (use the Validation plugin)
  - Clicking on the button will pop up a confirmation dialog
    - Use serialize() to see the contents of the form
    - Prevent the default form submission to keep to the current page
  - Prevent the user from typing more than 500 characters in the description