# Introduction to jQuery

Maktab Sharif Front-End Bootcamp
spring - 2018
Alireza Riahi and Ramin Afhami

# jQuery Intro

With a basic understanding of JavaScript and some of it’s foundations, it is time to take a look at jQuery.

- jQuery is an open source JavaScript library.
- jQuery simplifies the interaction between HTML, CSS, and JavaScript.
- jQuery greatly simplifies JavaScript programming.
- jQuery is easy to learn.
- What has made jQuery so popular is it’s ease of use, with selections resembling CSS and a comprehensible separation of behavior.
- The benefits of jQuery are massive, however for our purpose we will only be considered about the ability to find elements and perform actions with them.

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

1. Download a local copy.

- Compressed and uncompressed copies of jQuery files are available.
- The uncompressed file is best used during development or debugging.
- the compressed file saves bandwidth and improves performance in production.
- jQuery 3.3.1 is now available!
- Both versions can be downloaded from jQuery.com.

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

1. Download a local copy.

```
project/
 |-- css/
 |     |-- style.css
 |-- js/
 |     |-- jquery-3.3.1.min.js
 |     |-- scripts.js
 |-- index.html
```

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

1. Download a local copy.

```
<html lang="en">

 <head>
       <title>jQuery Demo</title>
       <link rel="stylesheet" href="css/style.css">
</head>

<body>

<script src=“js/jquery-3.3.1.min.js"></script>
<script src="js/scripts.js"></script>
</body>

</html>
```

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

2. Link to a file via Content Delivery Network (CDN).

- A Content Delivery Network (CDN) is a system of multiple servers that deliver web content to a user based on geographical location.
- CDNs can offer a performance benefit by hosting jQuery on servers spread across the globe.
- This also offers an advantage that if the visitor to your webpage has already downloaded a copy of jQuery from the same CDN, it won't have to be re-downloaded.

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

2. Link to a file via Content Delivery Network (CDN).

```
project/
 |-- css/
 |     |-- style.css
 |-- js/
 |     |-- scripts.js
 |-- index.html
```

# Getting Started with jQuery

jQuery is a JavaScript file that you will link to in your HTML. There are two ways to include jQuery in a project:

2. Link to a file via Content Delivery Network (CDN).

```
<html lang="en">

 <head>
      <title>jQuery Demo</title>
      <link rel="stylesheet" href="css/style.css">
</head>

<body>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
<script src="js/scripts.js"></script>
</body>

</html>
```

# Getting Started with jQuery

- jQuery Object

jQuery comes with it's own object, the dollar sign, $, also known as jQuery. The $ object is specifically made for selecting an element and then returning that element node to perform an action on it. These selections and actions should be written in a new file, referenced outside of the actual jQuery library.

```
$()
jQuery()
```

Script.js

- Document Ready

Before trigging any jQuery to traverse and manipulate a page it is best to wait until the DOM is finished loading. By placing all of our other custom written jQuery inside of this function we can guarantee that it will not be executed until the page has loaded and the DOM is ready.

```
$(document).ready(function() {
        // jQuery code
});
```

Script.js

# jQuery Syntax

A Standard jQuery Syntax is as given below:
$( "selector").operation()
$() : To call the jQuery Library.
selector : The selector to select the element on which the jQuery operation will be performed.
operation: The operation to be performed on the selected item.

We're now going to put our custom "Hello, World!" program inside the jQuery ready() wrapper.

Index.html

```
<body>
<p id="demo"></p>
…
```

Script.js

```
$(document).ready(function() {
        $("#demo").html("Hello, World!");
 });
```

Once you've saved the file, you can open your index.html file in your browser. If everything worked properly, you will see the output Hello, World!.

# jQuery Selectors

- Selectors are how we tell jQuery which elements we want to work on.
- Most jQuery selectors are the same as what you're familiar with in CSS, with a few jQuery-specific additions.
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
- You can view the full list of jQuery selectors on their official documentation pages.
- To access a selector, use the jQuery symbol $, followed by parentheses ().

```
$("selector")
```

- Double-quoted strings are preferred by the jQuery Style guide, though single-quoted strings are often used as well.

# jQuery Selectors

Below is a brief overview of some of the most commonly used selectors

| Selector | Name | Description |
|---|---|---|
| $("*") | Wildcard | selects every element on the page |
| $("p") | Tag | selects every instance of the <p> tag |
| $(".example") | Class | selects *every* element that has the example class applied to it |
| $("#example") | Id | selects a *single instance* of the unique example id. |
| $(‘p, i') | Multiple | selects every instance of the <p> and <i> tags |
| $('li strong'); | Descendant | selects every instance of the  tags in <li> tags |
| $("[type='text']") | Attribute | selects any element with text apptred to the typeattribute |
| $('p:nth-child(1)') | Pseudo Element | selects the first <p> |
| $(this) | Current | selects the current element being operated on within a function |

# jQuery Selectors

- This Selection Keyword

When working inside of a jQuery function you may want to select the element in which was referenced inside of the original selector. In this event the this keyword may be used to refer to the element selected in the current handler.

```
$("div").click(function(event) {
        $("this").hide();
});
```

# jQuery Selectors

- Traversing

jQuery provides a handful of methods for traversing up and down the DOM tree, filtering and selecting elements as necessary.
To get started with filtering elements inside the DOM a general selection needs to be made, from which will be traversed from relatively. In the example below the original selection finds all of the div elements in the DOM, which are then filtered using the .not() method. With this specific method all of the div elements without a class of type or collection will be selected.

```
$("div").not(".type, .collection ");
```

- Chaining Methods

For even more control as to which elements are selected different traversing methods may be chained together simply by using a dot in-between them.
The code sample below uses both the .not() method and the .parent() method. Combined together this will only select the parent elements of div elements without a class of type or collection.

```
$("div").not(".type, .collection ").parent();
```

# jQuery Selectors

- Traversing Methods

jQuery has quite a few traversing methods available to use. In general, they all fall into three categories, filtering, miscellaneous traversing, and DOM tree traversing. The specific methods within each category may be seen below.

1. Filtering

- .not(): Remove elements from the set of matched elements.

```
<div></div>
<div id="blueone"></div>
<div></div>
<div class="green"></div>
<div class="green"></div>
<div class="gray"></div>
<div></div>
```

Index.html

```
$( "div" ).not( ".green, #blueone" )
  .css( "border-color", "red" );
```

Script.js

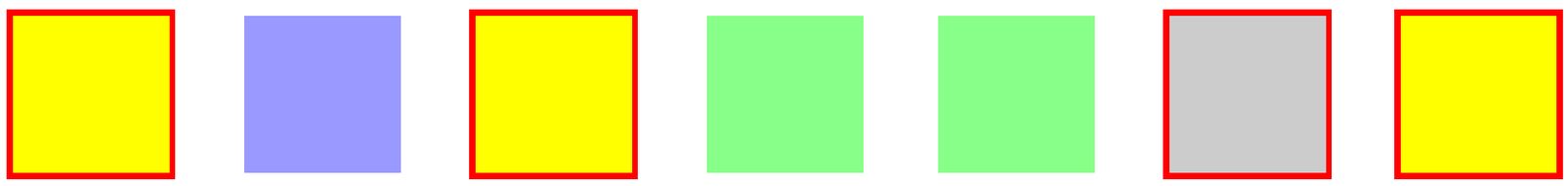
# jQuery Selectors

- Traversing Methods

jQuery has quite a few traversing methods available to use. In general, they all fall into three categories, filtering, miscellaneous traversing, and DOM tree traversing. The specific methods within each category may be seen below.

1. Filtering

- .not(): Remove elements from the set of matched elements.



Demo

# jQuery Selectors

.eq(): Reduce the set of matched elements to the one at the specified index.

```
.blue {
    background: blue;
 }
```

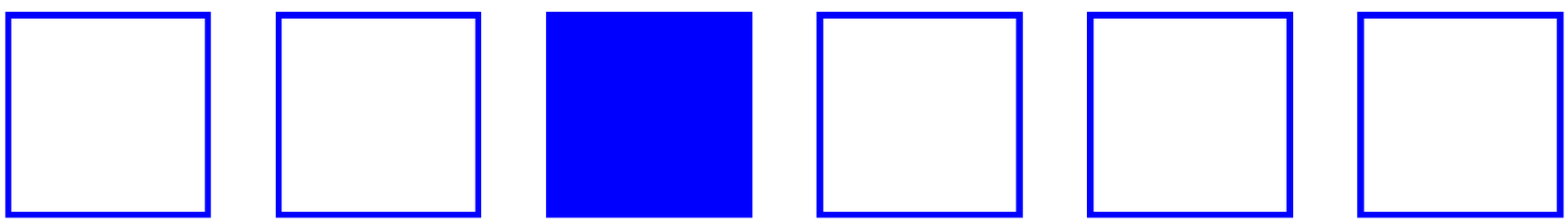
Style.css

```
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
```

Index.html

```
$( "body" ).find( "div" ).eq( 2 ).addClass( "blue" );
```

Script.js



Demo

# jQuery Selectors

- .filter(): Reduce the set of matched elements to those that match the selector or pass the function's test.
- .first(): Reduce the set of matched elements to the first in the set.
- .has(): Reduce the set of matched elements to those that have a descendant that matches the selector or DOM element.
- .is(): Check the current matched set of elements against a selector, element, or jQuery object and return true if at least one of these elements matches the given arguments.
- .last(): Reduce the set of matched elements to the final one in the set.
- .map(): Pass each element in the current matched set through a function, producing a new jQuery object containing the return values.
- .slice(): Reduce the set of matched elements to a subset specified by a range of indices.

# jQuery Selectors

## 2. Miscellaneous Traversing

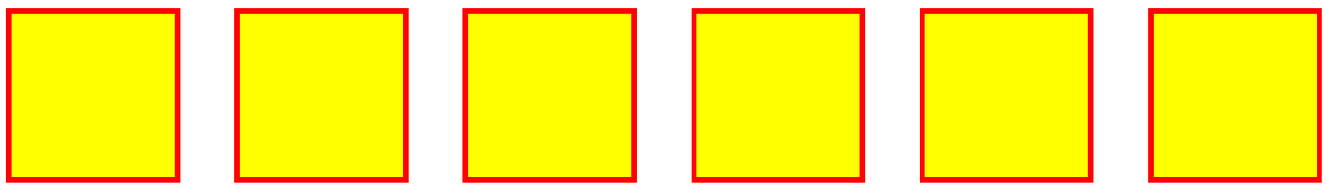
- .add(): Create a new jQuery object with elements added to the set of matched elements.

```
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<p>Added this... (notice no border)</p>
```

Index.html

```
$( "div" ).css( "border", "2px solid red" )
  .add( "p" )
  .css( "background", "yellow" );
```

Script.js



Demo

# jQuery Selectors

- .contents(): Get the children of each element in the set of matched elements, including text and comment nodes.

```
<div>
      <em> Hello world! What a beautiful day!</em>
</div>
```

Index.html

```
$("div").contents().filter("em").css( "border", "2px solid red" );
```

Script.js

*Hello world! What a beautiful day!*

Demo

- .addBack(): Add the previous set of elements on the stack to the current set, optionally filtered by a selector.
- .andSelf(): Add the previous set of elements on the stack to the current set.
- .end(): End the most recent filtering operation in the current chain and return the set of matched elements to its previous state.

# jQuery Selectors

## 3. DOM Tree Traversal

- .find(): Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or element.

```
<body class="ancestors"> body
  <div> span
    <div> div
     <span> span
          <span> span </span>
     </span>
    </div>
  </div>
</body>
```

Index.html

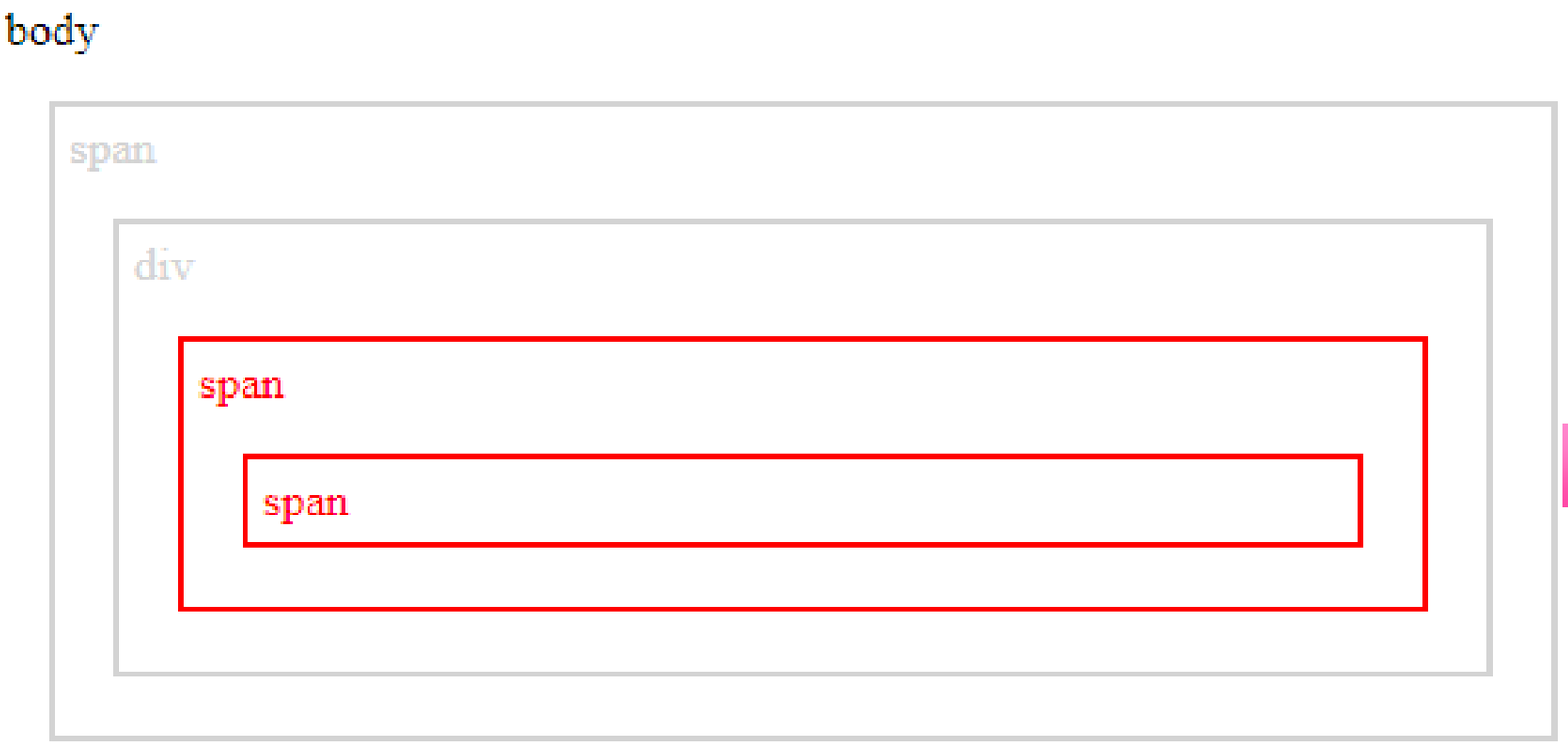
```
$("div").find("span").css({"color": "red", "border": "2px solid red"});
```

Script.js

# jQuery Selectors

## 3. DOM Tree Traversal

- .find(): Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or element.




Demo

# jQuery Selectors

## 3. DOM Tree Traversal

- .parent(): Get the parent of each element in the current set of matched elements, optionally filtered by a selector.

```
<body class="ancestors"> body (great-great-grandparent)
  <div> div (great-grandparent)
    <ul> ul (grandparent)
      <li> li (direct parent)
        <span> span </span>
      </li>
    </ul>
  </div>
</body>
```

Index.html

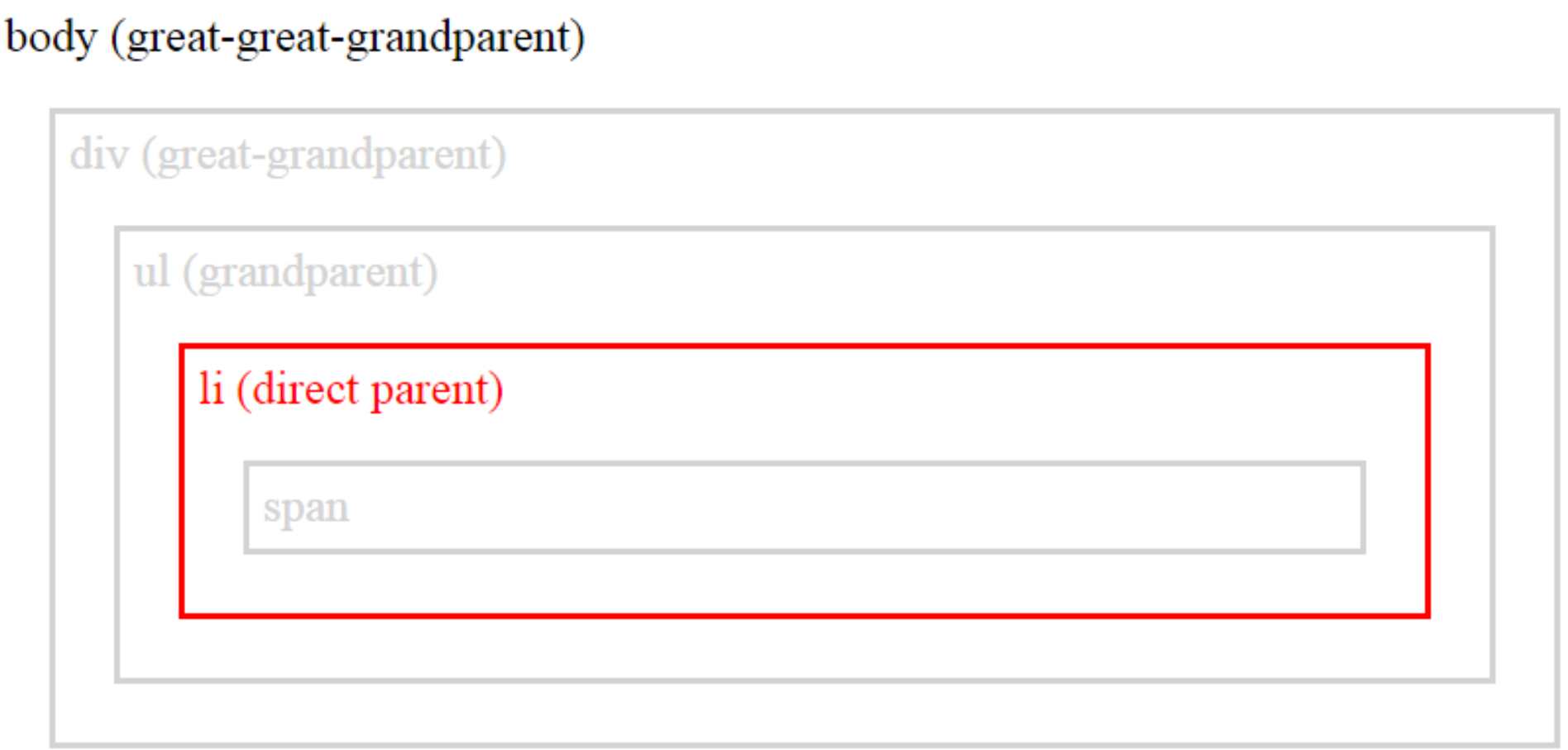
```
$("span").parent().css({"color": "red", "border": "2px solid red"});
```

Script.js

# jQuery Selectors

## 3. DOM Tree Traversal

- .parent(): Get the parent of each element in the current set of matched elements, optionally filtered by a selector.




Demo

# jQuery Selectors

- .children(): Get the children of each element in the set of matched elements, optionally filtered by a selector.

- .closest(): For each element in the set, get the first element that matches the selector by testing the element itself and traversing up through its ancestors in the DOM tree.

- .next(): Get the immediately following sibling of each element in the set of matched elements. If a selector is provided, it retrieves the next sibling only if it matches that selector.

- .nextAll(): Get all following siblings of each element in the set of matched elements, optionally filtered by a selector.

- .nextUntil(): Get all following siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object passed.

- .offsetParent(): Get the closest ancestor element that is positioned.

# jQuery Selectors

- .parents(): Get the ancestors of each element in the current set of matched elements, optionally filtered by a selector.

- .parentsUntil(): Get the ancestors of each element in the current set of matched elements, up to but not including the element matched by the selector, DOM node, or jQuery object.

- .prev(): Get the immediately preceding sibling of each element in the set of matched elements. If a selector is provided, it retrieves the previous sibling only if it matches that selector.

- .prevAll(): Get all preceding siblings of each element in the set of matched elements, optionally filtered by a selector.

- .prevUntil(): Get all preceding siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object.

- .siblings(): Get the siblings of each element in the set of matched elements, optionally filtered by a selector.

# Manipulation

Selecting and traversing elements in the DOM is only part of what jQuery offers, one other major part is what is possible with those elements once found. One possibility is to manipulate these elements, by either reading, adding, or changing attributes or styles. Additionally, elements may be altered in the DOM, changing their placement, removing them, adding new elements, and so forth.

## 1. Attribute Manipulation

One part of elements able to be inspected and manipulated are attributes. A few options include the ability to add, remove, or change an attribute or its value.

- .attr(): Get the value of an attribute for the first element in the set of matched elements or set one or more attributes for every matched element.

Index.html

```
<img src="img_pulpitrock.jpg" alt="Pulpit Rock“
        width="284" height="213">
```

Script.js

```
var a = $('img').attr('alt');
 $("img").attr("width", "500");
```

# Manipulation

- .addClass(): Adds the specified class(es) to each element in the set of matched elements.

```
.selected {
      color: red;
 }
 .highlight {
   background: yellow;
 }
```

Style.css

```
<p> Hello </p>
<p> and </p>
<p> Goodbye </p>
```

Index.html

```
$( "p:last" ).addClass( "selected highlight" );
```

Script.js

Hello

and

Goodbye

Demo

# Manipulation

- .removeClass(): Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

```css
.blue {
    color: blue;
 }
.under {
    text-decoration: underline;
 }
.highlight {
    background: yellow;
 }
```

Style.css

```html
<p class="blue under"> Hello </p>
<p class="blue under highlight"> and</p>
```

Index.html

```
$( "p:eq(1)" ).removeClass();
```

Script.js

Hello

and

Demo

# Manipulation

- .hasClass(): Determine whether any of the matched elements are assigned the given class.
- .prop(): Get the value of a property for the first element in the set of matched elements or set one or more properties for every matched element.
- .removeAttr(): Remove an attribute from each element in the set of matched elements.
- .removeProp(): Remove a property for the set of matched elements.
- .toggleClass(): Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the state argument.
- .val(): Get the current value of the first element in the set of matched elements or set the value of every matched element.

# Manipulation

## 2. Style Manipulation

On top of manipulating attributes, the style of an element may also be manipulated using a variety of methods. When reading or setting the height, width, or position of an element there are a handful of special methods available, and for all other style manipulations the .css() method can handle any CSS alterations.

- .css(): Get the value of a computed style property for the first element in the set of matched elements or set one or more CSS properties for every matched element.

- .height(): Get the current computed height for the first element in the set of matched elements or set the height of every matched element.

- .innerHeight(): Get the current computed inner height (including padding but not border) for the first element in the set of matched elements or set the inner height of every matched element.

# Manipulation

- .offset(): Get the current coordinates of the first element, or set the coordinates of every element, in the set of matched elements, relative to the document.

- .outerHeight(): Get the current computed outer height (including padding, border, and optionally margin) for the first element in the set of matched elements or set the outer height of every matched element.

- .outerWidth(): Get the current computed outer width (including padding, border, and optionally margin) for the first element in the set of matched elements or set the outer width of every matched element.

- .position(): Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

- .scrollLeft(): Get the current horizontal position of the scroll bar for the first element in the set of matched elements or set the horizontal position of the scroll bar for every matched element.

# Manipulation

- .scrollTop(): Get theof the scroll bar for the first element in the set of matched elements or set the vertical position of the scroll bar for every matched element.

- .width(): Get the current computed width for the first element in the set of matched elements or set the width of  current vertical position every matched element.

```
$('h1 span').css('font-size', 'normal');
$('div').css({
  fontSize: '13px',
  background: '#f60'
});
$('header').height(200);
$('.extend').height(30 + 'em');
```

# Manipulation

3. DOM Manipulation

Lastly, we are able to inspect and manipulate the DOM, changing the placement of elements, adding and removing elements, as well as flat out altering elements. The options here are deep and varied, allowing for any potential changes to be made inside the DOM.

- .after(): Insert content, specified by the parameter, after each element in the set of matched elements.
- .append(): Insert content, specified by the parameter, to the end of each element in the set of matched elements.
- .appendTo(): Insert every element in the set of matched elements to the end of the target.
- .before(): Insert content, specified by the parameter, before each element in the set of matched elements.

# Manipulation

- .clone(): Create a deep copy of the set of matched elements.
- .detach(): Remove the set of matched elements from the DOM.
- .empty(): Remove all child nodes of the set of matched elements from the DOM.
- .html(): Get the HTML contents of the first element in the set of matched elements or set the HTML contents of every matched element.
- .insertAfter(): Insert every element in the set of matched elements after the target.
- .insertBefore(): Insert every element in the set of matched elements before the target.
- .prepend(): Insert content, specified by the parameter, to the beginning of each element in the set of matched elements.

# Manipulation

- .prependTo(): Insert every element in the set of matched elements to the beginning of the target.
- .remove(): Remove the set of matched elements from the DOM.
- .replaceAll():Replace each target element with the set of matched elements.
- .replaceWith(): Replace each element in the set of matched elements with the provided new content and return the set of elements that was removed.
- .text(): Get the combined text contents of each element in the set of matched elements, including their descendants, or set the text contents of the matched elements.
- .unwrap(): Remove the parents of the set of matched elements from the DOM, leaving the matched elements in their place.
- .wrap(): Wrap an HTML structure around each element in the set of matched elements.

# Manipulation

- .wrapAll(): Wrap an HTML structure around all elements in the set of matched elements.
- .wrapInner(): Wrap an HTML structure around the content of each element in the set of matched elements.

# Manipulation

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Index.html

```
$("p").after("<p> Hello world! </p>");
```

Script.js

This is a paragraph.

Hello world!

This is another paragraph.

Demo

Hello world!

```
<p>This is a paragraph.</p>
<ol>
  <li> List item 1</li>
</ol>
```

Index.html

```
$("p").append(" Appended text.");
$("ol").append("<li> Appended item </li>");
```

Script.js

This is a paragraph. Appended text.

1. List item 1
2. Appended item

Demo

# Events

One of the beauties of jQuery is the ability to easily add event handlers, which are methods that are called only upon a specific event or action taking place. For example, the method of adding a class to an element can be set to only occur upon that element being clicked on.

1. Browser Events

.resize() .scroll()

2. Document Loading

.ready()

3. Event Handler Attachment

.off() .on() .one() jQuery.proxy() .trigger() .triggerHandler() .unbind() .undelegate()

4. Event Object

event.currentTarget event.preventDefault() event.stopPropagation() event.target event.type

# Events

One of the beauties of jQuery is the ability to easily add event handlers, which are methods that are called only upon a specific event or action taking place. For example, the method of adding a class to an element can be set to only occur upon that element being clicked on.

### 5. Form Events

.blur() .change() .focus() .select() .submit()

### 6. Keyboard Events

.focusin() .focusout() .keydown() .keypress() .keyup()

### 7. Mouse Events

.click() .dblclick() .focusin() .focusout() .hover() .mousedown()
.mouseenter() .mouseleave() .mousemove() .mouseout()
.mouseover() .mouseup()

# Events

```
$("p").click(function(){
    $(this).hide();
});
```

```
$("p").on("click", function(){
    $(this).hide();
});
```

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    }
});
```

# Effects

Next to events, jQuery also provides a handful of customizable effects. These effects come by the way of different methods, including event methods for showing and hiding content, fading content in and out, or sliding content up and down. All of these are ready to use methods and may be customized as best see fit.

## 1. Effect Syntax

each effect method has it's own syntax which can be found in the jQuery effects documentation. The duration, easing, and callback parameters outlined here are common, but not available on every methoded, each effect method has it's own.

- Basic Effects

.hide()  .show()  .toggle()

- Custom Effects

.animate()  .clearQueue()  .delay()  .dequeue()  jQuery.fx.interval
jQuery.fx.off  .queue()  .stop()

# Effects

Next to events, jQuery also provides a handful of customizable effects. These effects come by the way of different methods, including event methods for showing and hiding content, fading content in and out, or sliding content up and down. All of these are ready to use methods and may be customized as best see fit.

## 1. Effect Syntax

each effect method has it's own syntax which can be found in the jQuery effects documentation. The duration, easing, and callback parameters outlined here are common, but not available on every methoded, each effect method has it's own.

- Fading Effects

.fadeIn()  .fadeOut()  .fadeTo()  .fadeToggle()

- Sliding Effects

.slideDown()  .slideToggle()  .slideUp()

# Effects

## 2. Effect Duration

Using the .show() method as an example, the first parameter available to optionally pass in to the method is the duration, which can be accomplished using a keyword or milliseconds value. The keyword slow defaults to 600 milliseconds, while the keyword fast defaults to 200 milliseconds. Using a keyword value is fine, but millisecond values may also be passed in directly. Keyword values must be quoted while millisecond values do not.

```
$('.error').show();
$('.error').show('slow');
$('.error').show(500);
```

# Effects

## 3. Effect Easing

In addition to setting the duration in which an effect takes place the easing, or speed at which an animation progresses at during different times within the animation, may also be set. By default jQuery has two keyword values for easing, the default value is swing with the additional value being linear. The default swing value starts the animation at a slow pace, picking up speed during the animation, and then slows down again before completion. The linear value runs the animation at one constant pace for the entire duration.

```
$('.error').show('slow', 'linear');
$('.error').show(500, 'linear');
```

# Effects

4. Effect Callback

   When an animation is completed it is possible to run another function, called a callback function. The callback function should be placed after the duration or easing, if either exist. Inside this function new events or effects may be placed, each following their own required syntax.

```
$('.error').show('slow', 'linear', function(event){
  $('.error .status').text('Continue');
});
```

# Finished

for deep information you can visit

www.w3schools.com

Alireza Riahi and Ramin Afhami
ali.r.riahi@gmail.com
afhami.ramin@yahoo.com

Maktab Sharif - spring 2018