ICT286
Web and Mobile Computing

Topic 5
DOM Manipulation with jQuery

Objectives

- Understanding the concept of Document Object Model
- Be able to use jQuery to
  - select the desired DOM elements using the right selector
  - define event handlers for common events
  - use effects methods to achieve desired visual effects
  - manipulate HTML elements, including getting and setting contents, values and attributes, adding and removing contents, elements and attributes
  - manipulate CSS, including getting property values and setting new property values
  - be familiar with methods for dimensions and traversing.

The Document Object Model (DOM)

- The Document Object Model is an cross-platform, language-independent Application Programming Interface (API) that defines a standard way for programs to access HTML document.
- Originally released by Netscape, then Microsoft developed its own version (similar but not exactly the same). W3C developed a series of standardised specifications:
  - DOM Level 0: prior to W3C specification, unofficial
  - DOM Level 1: 1998
  - DOM Level 2: 2000
  - DOM Level 3: 2004
  - DOM Level 4: 2015

The Document Object Model (DOM)

- Apart from JavaScript binding, several other language binding are available for accessing DOM, such as Java, Perl, PHP, and C#.
- The DOM represents an HTML document as a tree of nodes in which each node represents an element in the HTML document.
- In the JavaScript binding, an HTML element is represented as an object and the attributes of that element are represented as properties of the object.
  - e.g., `<input type = "text" name = "address">` would be represented as an object which contains (at least) two properties, `type` and `name`, with the values `"text"` and `"address"` respectively.
- The `document` object is the root of the DOM tree.

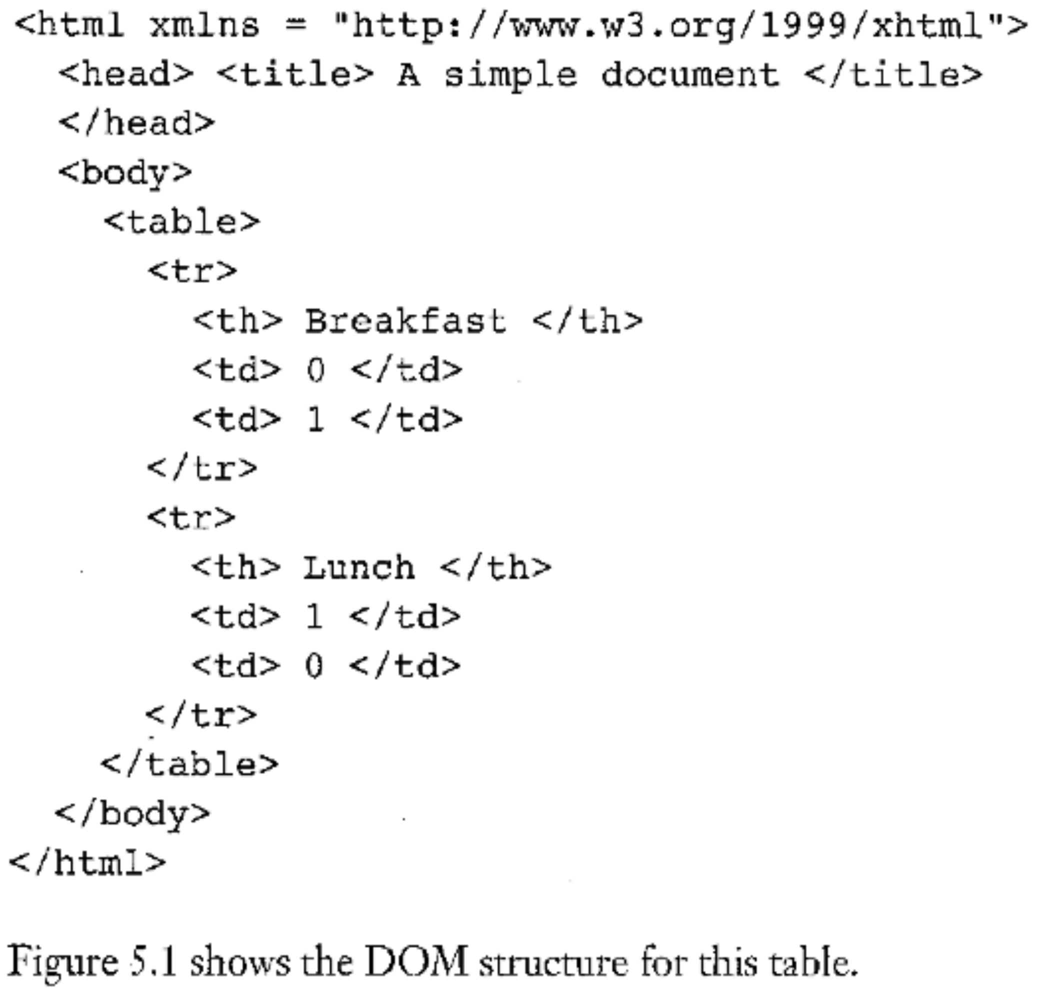
Some Document Methods

Method	Description
write(*string*)	Writes *string* to the output stream as HTML code.
writeln(*string*)	Writes *string* to the output stream as HTML code and adds a *newline* character.
open()	Opens an output stream to collect outputs from document.write or document.writeln;
close()	Closes the output stream opened with document.open and display the collected data.
getElementById(*string*)	Returns the reference to the first object with the specified id
getElementsByName(*string*)	Returns a collection of object references with the specified name
getElememtsByTagName(*string*)	Returns a collection of object references with the specified tag name

Some Document Properties

Property	Description
lastModified	The date and time that this document was last modified.
title	The title of the current document.
URL	The url of the current document
referrer	Returns the URL of the document that loaded the current document
cookie	Contains a string of the values of the cookies associated with this document.
forms	An array of forms in the document
links	An array of hyper links in the document
images	An array of the images in the document.

A Simple HTML Example



```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> A simple document </title>
  </head>
  <body>
    <table>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 5.1 shows the DOM structure for this table.

Sebesta page 192

DOM Structure

- The previous HTML document is represented internally in the browser using DOM object tree where each node is an object representing an element and can be accessed in JavaScript:

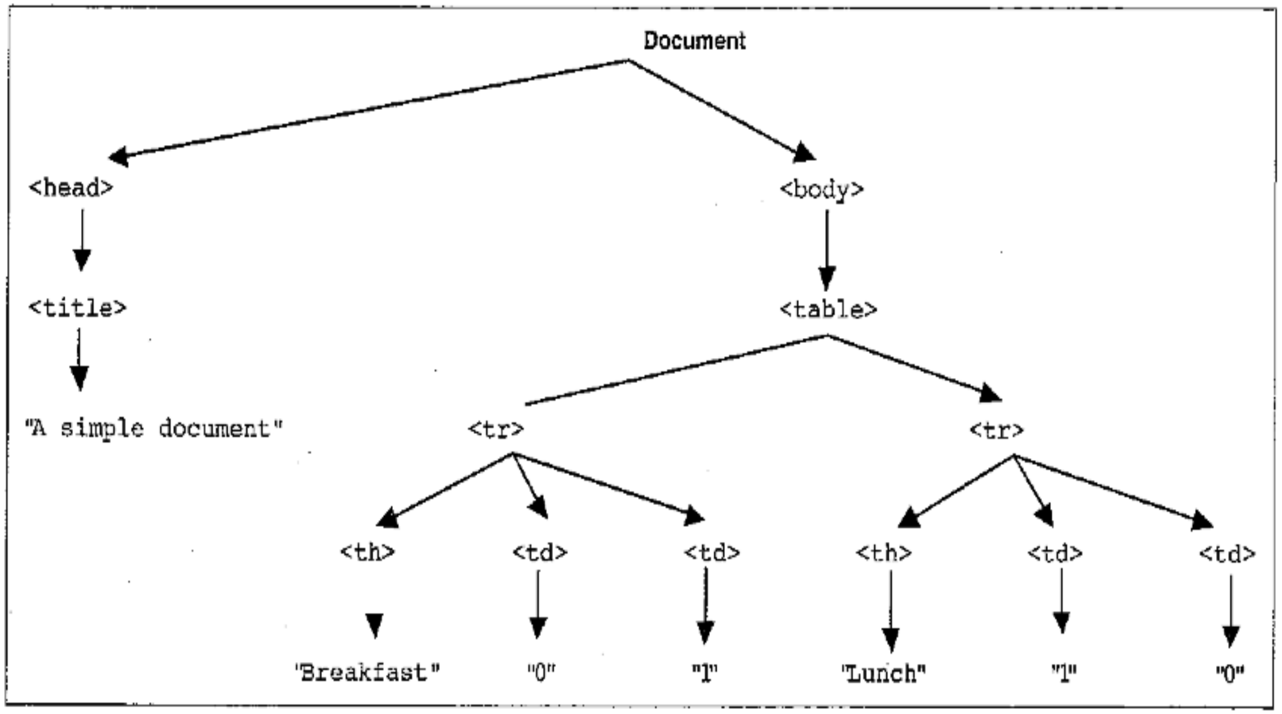
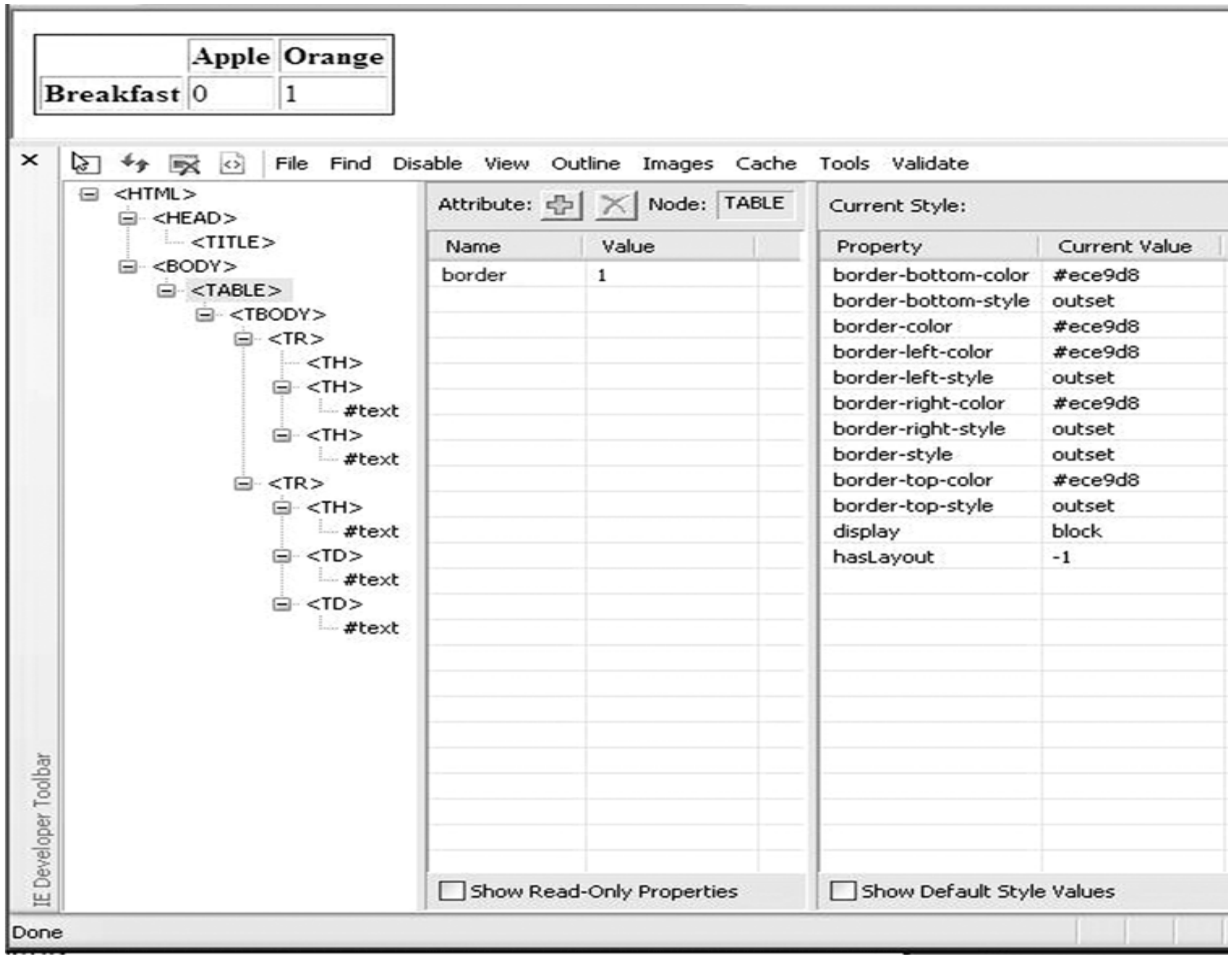


FIGURE 5.1 The DOM structure for a simple document

Sebesta page 192

The DOM Structure with IE7



	Apple	Orange
Breakfast	0	1

```
<HTML>
  <HEAD>
    <TITLE>
  <BODY>
    <TABLE>
      <TBODY>
        <TR>
          <TH>
          <TH>
            #text
          <TH>
            #text
        <TR>
          <TH>
            #text
          <TD>
            #text
          <TD>
            #text
```

Attribute: Node: TABLE

Name	Value
border	1

☐ Show Read-Only Properties

Current Style:

Property	Current Value
border-bottom-color	#ece9d8
border-bottom-style	outset
border-color	#ece9d8
border-left-color	#ece9d8
border-left-style	outset
border-right-color	#ece9d8
border-right-style	outset
border-style	outset
border-top-color	#ece9d8
border-top-style	outset
display	block
hasLayout	-1

☐ Show Default Style Values

Sebesta: Figure 5.1

Access DOM Objects

- Each element in an HTML document has a corresponding object in its DOM tree.
- We can access and modify these objects in JavaScript.
- It is possible to traverse through the DOM tree.
- In Topic 4, we have discussed a number of ways to obtain the reference of the object representing a particularly element.
- However, it often takes many lines of code to accomplish a simple task using the DOM interface.
- In practice, most people use a JavaScript library, jQuery, to query the DOM tree and take an action on the selected elements.

jQuery

- jQery is a client-side JavaScript library released in 2006 by John Resig.
- The latest version is v3.4 released in April 2019.
- jQuery is not a separate language. A jQuery code is a piece of JavaScript code.
- It takes common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- It also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

jQuery

- jQuery library brings together the following set of functionalities:
  - HTML/DOM navigation and manipulation
  - CSS manipulation
  - Event registration and handling
  - HTML effects and animations
  - Developing with AJAX
  - Other utilities

Include jQuery Library

- To start using jQuery, you must firstly include jQuery in your web page. One way to do it is download the jQuery library from jQuery.com
  - The jQuery library is a single JavaScript file (eg, jquery-3.3.1.js)
  - Place the downloaded file in the same directory as the pages where you wish to use it
  - You reference it with the HTML `<script>` tag which should be inside the `<head>` section:

```
 <head>
  <script src="jquery-3.3.1.js"> </script>
</head>
```

  - Note: the above script element adds the jQuery() function to the global namespace

Include jQuery Library

- Another way to include jQuery library is to reference it from a Content Delivery Network (CDN), such as Google
  - This is the recommended usage for this unit, as the latest updates are always readily available
  - The reference to the `<script>` tag should be inside the `<head>` element.

```
<head>

<script
   src="https://ajax.googleapis.com/ajax/libs/jquer
   y/3.3.1/jquery.min.js">
</script>

</head>
```

jQuery: Concept

- The central concept behind jQuery is:

  "find something, and then do something on them"

  ```
  jQuery( selector ).action( … );
  ```

- For example, select DOM elements from an HTML document and then do something with them using jQuery methods, such as

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

  `jQuery("p")` select all <p> elements

  then hide those elements (not displaying them)

- Because the function `jQuery` is used so often, jQuery provided a convenient alias, `$`, for it, so the above jQuery statement is often written as

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

jQuery: Example

```
<!DOCTYPE html>
<html>
<head><script src=“jquery-3.3.1.js"></script></head>
<body>
     <!-- the texts in following two paragraphs will be
       replaced with jQuery, and the background of
       the texts is changed to yellow -->
     <p>ICT286</p>
     <p>JavaScript</p>
     <script>
        $("p").text("jQuery").
              css("background-color","yellow");
     </script>
</body>
</html>
```

The jQuery Function

- jQuery library is centred on the function `jQuery`. This function has a short alias `$`.
  - towards the end of the jQuery library, you will see statements like

    ```
    window.jQuery=winodw.$=jQuery;
    ```
- This function can take the following arguments to select an DOM object:
  - A selector in the form of `$(expression, [context])`. It selects objects that meet the selection criteria
    - Eg: `$('input: radio', document.form[0]);`
  - A piece of HTML code, in the form of `$(html-code)`, which will create a mini DOM for the HTML code
    - Eg: `$('<div id="load">Loading… </div>');`
  - A reference to an existing DOM object, in the form of `$(DOM object reference)`
    - Eg: `$(document.body).css('background', 'red');`

jQuery Action Methods

- The jQuery function `$` select one or more elements and create a jQuery object.
- This object contains many methods to allow actions to be taken on those selected elements.
- These action methods can be used to
  - Get and set the contents of the selected elements
  - Get and set the attributes of the selected elements
  - Get and set the styles for the selected elements
  - Define the event handler for the selected elements
  - Achieve some visual effects on the selected elements
  - and do many other things

jQuery Selectors

- Selectors:
  - jQuery supports nearly all CSS selectors from CSS 1 through 3
- Types of Selectors:
  - Basic – such as element, #id, .class, .classA,.classB, el1,el2,el3
    - Eg: `$('p'),$('#id'),$('.English'),$('.English, .French'), $("h1,div,p")`
  - Hierarchy – such as ancestor descendent, parent > child, prev + next
    - Eg: `$('form input'),$('#main > *'),$('label + input')`

jQuery Selectors

- Form element selectors

Selectors	Matched Elements
:input	input, select, textarea and button elements
:text, :radio, :checkbox, :image, :submit, :reset, :password, :file	input element whose attribute is equal to the specified selectors
:button	button element, input element with type "button"

jQuery Selectors

- Basic filters
  - :first, :last, :not(selector), :even, :odd, :eq(index), :gt(index), :lt(index), :header, :animated
- Attribute filters
  - [attribute], [attribute!=value], [attribute^=value], [attribute$=value], [attribute*=value], [filter1][filter2]
- Select elements having specified attribute, where:
  - ^= value begins exactly with a given string
  - != does not contain given value
  - $= value ends exactly with a given string
  - For a list of jQuery selectors and examples see the following w3School page

https://www.w3schools.com/Jquery/jquery_ref_selectors.asp

The `ready` Event

- It is a good practice to wait for the DOM to be fully "loaded and ready" before working on it
- Code that manipulates the DOM can run in the handler for the `ready` event. This ensure that you only start to manipulate the DOM after the HTML is fully loaded in the browser and the DOM is created for the HTML document.

```
// DOM Ready Event
$(document).ready( function() {
  // the actual handler code for the ready event
  // goes here...



});
```

The `ready` Event

- The event is typically placed in the head section, as shown below:

```
<?DOCTTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml" >
<head>
    <title> jQuery ready handler </title>
    <script src="https://ajax.googleapis.com/ajax
        /libs/jquery/3.3.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            // Other jQuery code goes here
        });
    </script>
</head>
<body>
    <!-- HTML and JavaScript code goes here -->
</body>
</html>
```

Event Handler for ready Event

- To save space in this lecture notes, we will only show the handler code for the ready event, assuming the code is placed in the area marked by "`// other jQuery code goes here`"
- Example

```
// other jQuery code goes here
$("button").click( function() {
        $("#test").hide();
});
```

- Example

```
// other jQuery code goes here
$("button").click( function() {
    alert("the name is " + $(":text").val();
});
```

jQuery Chaining

- jQuery allows us to run multiple jQuery action methods on the same selected element) within a single statement
- This is achieved by *chaining* together these action methods
- To chain an action, you simply append the action to the previous action using the 'dot' notation
- The example in the next slide chains together the `css()`, `slideUp()`, and `slideDown()` methods
- The `"p1"` element first changes to red, then it slides up in 2000 miniseconds, and then it slides down in 2000 miniseconds.

jQuery Chaining

- Example:

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

- The above statement is equivalent to the following three statements, except that it only calls the jQuery function once instead of three times!

```
$("#p1").css("color","red");
$("#p1").slideUp(2000);
$("#p1").slideDown(2000);
```

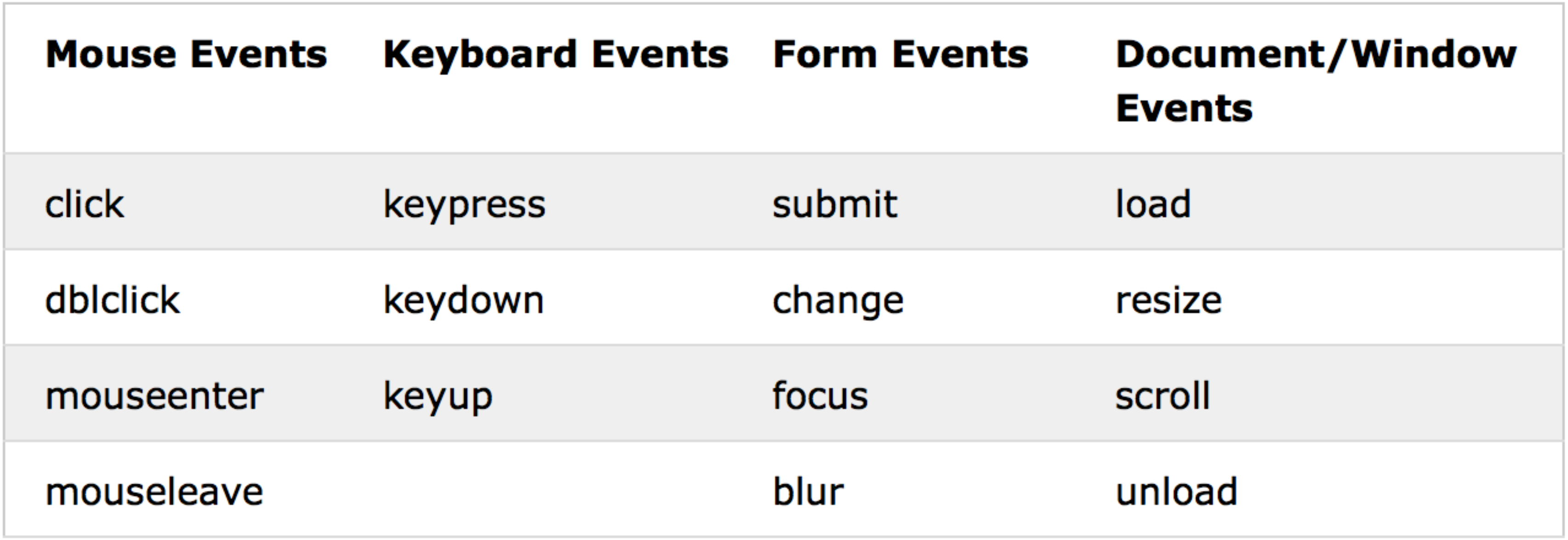
jQuery Chaining

- When chaining, the line of code could become quite long
- jQuery is not very strict on the syntax
    - You can format it like you want, including line breaks and indentations
- Eg:

```
$("#p1").css("color", "red")
    .slideUp(2000)
    .slideDown(2000);
```

jQuery Event Methods

- jQuery supports most DOM events. Each of these events has a corresponding method in the jQuery object.
- These methods allow us to define event handlers for the events and register them with the selected elements.
- Some common event/event methods (taken from w3school):



Mouse Events	**Keyboard Events**	**Form Events**	**Document/Window Events**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Example: Click Event

- For example, to define and register an event handler for the click event for all paragraphs such that when a paragraph is clicked, it is displayed in red colour.

```
$("p").click(function(){
    $(this).css("color", "red");
});
```

- In the above code,
  - `this` represents the reference of the DOM object for the p element that is clicked.
  - `$(this)` returns the reference to the jQuery object for that p element.
  - `css("color", "red")` method sets the colour property to red for that p element.

Example: Click Event

```
<?DOCTTYPE html>
<html>
<head>
    <title> jQuery click event example </title>
    <script src="jquery-3.3.1.js"></script>
    <script>
       $(document).ready(function(){
          $("p").click(function(){
              $(this).css("color", "red");
          });
       });
    </script>
</head>
<body>
    <p> Paragraph 1 </p>
    <p> Paragraph 2 </p>
    <p> Paragraph 3 </p>
</body>
</html>
```

Example: Mouse Hover Event

- The hover method takes two functions: the first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves that HTML element:
- Example

```
$("#p1").hover( function() {
   $("#feedback").val("You entered p1!");
},  // note: comma separating the two functions
function() {
   $("#feedback").val("Bye!");
});
```

- In the above example, the `val()` function places the text into the feedback textbox.

Example: Mouse Hover Event

```
<?DOCTTYPE html>
<html>
<head>
    <title> jQuery move hover event example </title>
    <script src="jquery-3.3.1.js"></script>
    <script>
       $(document).ready(function(){
          $("#p1").hover( function() {
             $("#feedback").val("You entered p1!");
          }, // note: comma separating the functions
          function(){
             $("#feedback").val("Bye!");
          });
       });
    </script>
</head>
<body>
   Feedback: <input id="feedback" size="30"/>
   <p id="p1"> Paragraph 1</p>
   <p> Paragraph 2</p>
</body>
</html>
```

Example: Window `load` Event

- The event `ready` is triggered once the DOM is created, but before the window loads
- So we do not have to wait for the `window.onload` to manipulate the DOM
- However, sometimes we might want to wait for `window.onload` event, so that code is executed once the entire page (including all assets) is completely rendered
- This is done by attaching a load event handler to the window object

```
$(window).load( function() {
     // methods go here...
});
```

jQuery `on()` Method

- The `on()` method attaches one or more event handlers for the selected elements
- Eg: attach a click event to `<p>` elements:

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

jQuery `on()` Method

- Eg: attach multiple event handlers to `<p>` elements:

```
$("p").on({
   mouseenter: function(){
     $(this).css("background-color", "gray");
   },
   mouseleave: function(){
     $(this).css("background-color", "blue");
   },
   click: function(){
     $(this).css("background-color", "red");
   }
});
```

jQuery Effects Methods

- jQuery provides a number of action methods that achieve certain visual effects.
- Basic visual effects, eg
  - show(), show(speed)
  - hide(), hide(speed), hide(speed, callback)
  - toggle(), toggle(speed)
- Fading, eg
  - fadeIn(speed),
  - fadeOut(speed),
  - fadeTo(speed, opacity)
- Slide
  - slideDown(), slide(speed, callback)
  - slideUp(), slideUp(speed, callback)
  - slideToggle(), slideToggle(speed, callback)
- More, such as animate, stop etc

jQuery Effects Example

- Example 1: hide the element with id=“test”

```
$("#test").hide();
```

- Example 2: show all p elements

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

- Example 3: hide the element with `id="test"` slowly and once it is hidden, display an alert message

```
$("#test").hide("slow", function(){
      alert("it is now hidden!");
});
```

In this example, the hide method also takes a *callback function* (in red colour), which will be executed once the element is completely hidden.

jQuery Callback Functions

- JavaScript statements are executed line by line
- However, with effects such as those just mentioned, the next line of code can be run even though the effect is not finished. Sometimes, this could result in an undesired outcome!
- To prevent these types of issues, you can define a callback function, which is executed after the current effect is 100% finished. The typical syntax:

```
$(selector).effect(speed, callback);
```

such as

```
$("#test").hide("slow", function(){
    alert("Dolly the sheep disappeared");
});
```

jQuery Effects Example

In the head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("#test").hide("slow",
                function(){
                    alert("Dolly the sheep disappeared");
                });
        });
    });
</script>
```

in the body:

```
<body>
    <button>Click to hide</button>
    <p id="test">Dolly the sheep</p>
</body>
```

Manipulate HTML

- jQuery comes with many DOM related methods that make it easy to access and manipulate DOM elements including contents, values, and attributes.
- The following four methods are particularly useful:
  - `text()` - returns the text content of all selected elements (HTML tags removed)
  - `text(`*`string`*`)` – set the contents of all selected elements to text *`string`* (if there are HTML tags, they are treated as normal text)
  - `html()` – return the content of the first of the selected elements (including HTML markup if there are)
  - `html(`*`string`*`)` - sets the contents of selected elements to *`string`* (including HTML markup)

Manipulate HTML

- `val()` - returns the value of the first of the selected form elements
- `val(`*`string`*`)` - sets the values of selected form elements to *`string`*
- `attr(`*`attr-name`*`)` – returns the value of the specified attribute in the first of the selected elements.
- `attr(`*`attr-name, attr-value`*`)` – set the value of the specified attribute *`attr-name`* in all selected elements to *`attr-value`*.
- The `attr` method also allows us to set several attributes at the same time:

```
attr( { attr-name1 : attr-val1,
        attr-name2 : attr-val2,
        . . . } )
```

Example: get text

- Example: return the text contents of p elements

```
<?DOCTTYPE html>
<html>
<head>
    <title> text get example </title>
    <script src="jquery-3.3.1.js"></script>
    <script>
       $(document).ready(function(){
          $("button").click(function(){
             alert($("p").text());
          });
       });
    </script>
</head>
<body>
<button> click </button>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
</body>
</html>
```

Example: set text

- Example: set the text content of all p elements to “jQuery”

```
<?DOCTTYPE html>
<html>
<head>
    <title> jQuery: set text </title>
    <script src="jquery-3.3.1.js"></script>
    <script>
       $(document).ready(function(){
          $("button").click(function(){
             $("p").text("jQuery");
          });
       });
    </script>
</head>
<body>
<button> click </button>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
</body>
</html>
```

Example: get html

- Example: get the HTML content of the first of the p elements.

In head:

```
<script>
    $(document).ready(function(){
       $("button").click(function(){
            var s = $("p").html();
            $("textarea").val( s );
       });
    });
</script>
```

In body:

```
<button> click </button>
<p>This is <strong>important</strong>! </p>
<p>Paragraph 2</p>
<textarea row=”5" col="30"> text area</textarea>
```

Example: set html

- Example: set the HTML content of all p elements to `<em>jQuery</em> is great!`

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("p").html("<em>jQuery</em> is great!");
          });
    });
</script>
```

In body:

```
<button> click </button>
<p>This is <strong>important</strong>! </p>
<p>Paragraph 2</p>
```

Example: get val

- Example: return the value of the first text box

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            alert($(":text").val());
        });
    });
</script>
```

In body:

```
<button> click </button> <br/>
choice1: <input type="text" value="pizza"/> <br/>
choice2: <input type="text" value="big mac"/>
```

Example: set val

- Example: set the value of all text boxes to “subway”

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
             $(":text").val(”subway");
        });
    });
</script>
```

In body:

```
<button> click </button> <br/>
choice1: <input type="text" value="pizza"/> <br/>
choice2: <input type="text" value="big mac"/>
```

Example: get attr

- Example: get the id of the first p element

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            alert($("p").attr("id"));
        });
    });
</script>
```

In body:

```
<button> click </button>
<p id="p1">Paragraph 1</p>
<p id="p2">Paragraph 2</p>
```

Example: set attr

- Example: change the colour of all p elements to yellow

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("p").attr("style", "color:yellow");
        });
    });
</script>
```

In body:

```
<button> click </button>
<p id="p1" style="color: red">Paragraph 1</p>
<p id="p2">Paragraph 2</p>
```

Example: set multiple attr

- Example: change the links in all a elements to "http://www.google.com" and the colour of the hyper texts to blue

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("a").attr( {"href":"http://www.google.com",
                 "style":"color:blue"} );
        });
    });
</script>
```

In body:

```
<button> click </button>
<a href="https://www.w3schools.com">w3school</a> <br/>
<a href="https://www.w3schools.com/jquery/">jQuery</a>
```

Manipulate HTML: callback functions

- All of the four jQuery methods: `text()`, `html()`, `val()` and `attr()` also come with a callback function option
- The callback function has two parameters:
  - The index of the current element in the list of elements selected, and
  - The original (old) value
- You then return the string you wish to use as the new value from the function
- See w3schools website for examples

Add Elements/Contents

- The following jQuery methods are used to add *new HTML content*:
  - `append()` - Inserts content at the *end* of the selected elements
  - `prepend()` - Inserts content at the *beginning* of the selected elements
  - `after()` - Inserts content *after* the selected elements
  - `before()` - Inserts content *before* the selected elements

Add Elements/Contents

- The `append()` and `prepend()` methods can also be used to add *new HTML elements*
  - They can take an infinite number of new elements as parameters
  - The new elements can be generated with text/HTML, with jQuery, or with JavaScript code and DOM elements
- `append()` adds the new elements and text to the *end* of the selected element; `prepend()` adds to the *beginning* of the selected element

Example: append

- Example: change the color of all p elements to yellow

In head:

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            var txt1 = "<p>Text 1</p>";
            var txt2 = $("<p></p>").text("Text 2");
            var txt3 = document.createElement("p");
            txt3.innerHTML = "Text 3";
            $("#test").append(txt1, txt2, txt3);
        });
    });
</script>
```

In body:

```
<button> click </button>
<p id="test"> the original text</p>
```

Remove Elements/Contents

- To remove elements and content, there are mainly two jQuery methods:
  - `remove()` - Removes the selected element AND its child elements
  - `empty()` - Removes the child elements FROM the selected element(s)
- Eg, the following statement removes the div1 element including its contents:

```
$("#div1").remove();
```

- Eg, the following statement only removes the content of the div1 element, but not the div1 element itself:

```
$("#div1").empty();
```

- Refer w3schools site for more examples

Remove Attributes

- Attributes can be removed with the method `removeAttr(name)`. Eg

In head:

```
<script>
 $(document).ready(function(){
     $("#btn1").click(function(){
         $("a").attr( {"href":"http://www.google.com",
                 "style":"background-color:yellow"} );
     });
     $("#btn2").click(function(){
         $("a").removeAttr("style");
     });
 });
</script>
```

In body:

```
<button id="btn1"> add attributes  </button> <br/>
<button id="btn2"> remove color attribute  </button> <br/>
<a href="https://www.w3schools.com">w3school</a> <br/>
<a href="https://www.w3schools.com/Jquery/">jQuery</a>
```

Manipulate CSS

- jQuery has several methods for CSS manipulation:
  - `addClass()` – Adds one or more classes to the selected elements
  - `removeClass()` - Removes one or more classes from the selected elements
  - `toggleClass()` - Toggles between adding/removing classes from the selected elements
  - `css()` - Sets or returns the style attribute
  - Refer w3schools for examples

Manipulate CSS

- To return the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

- To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

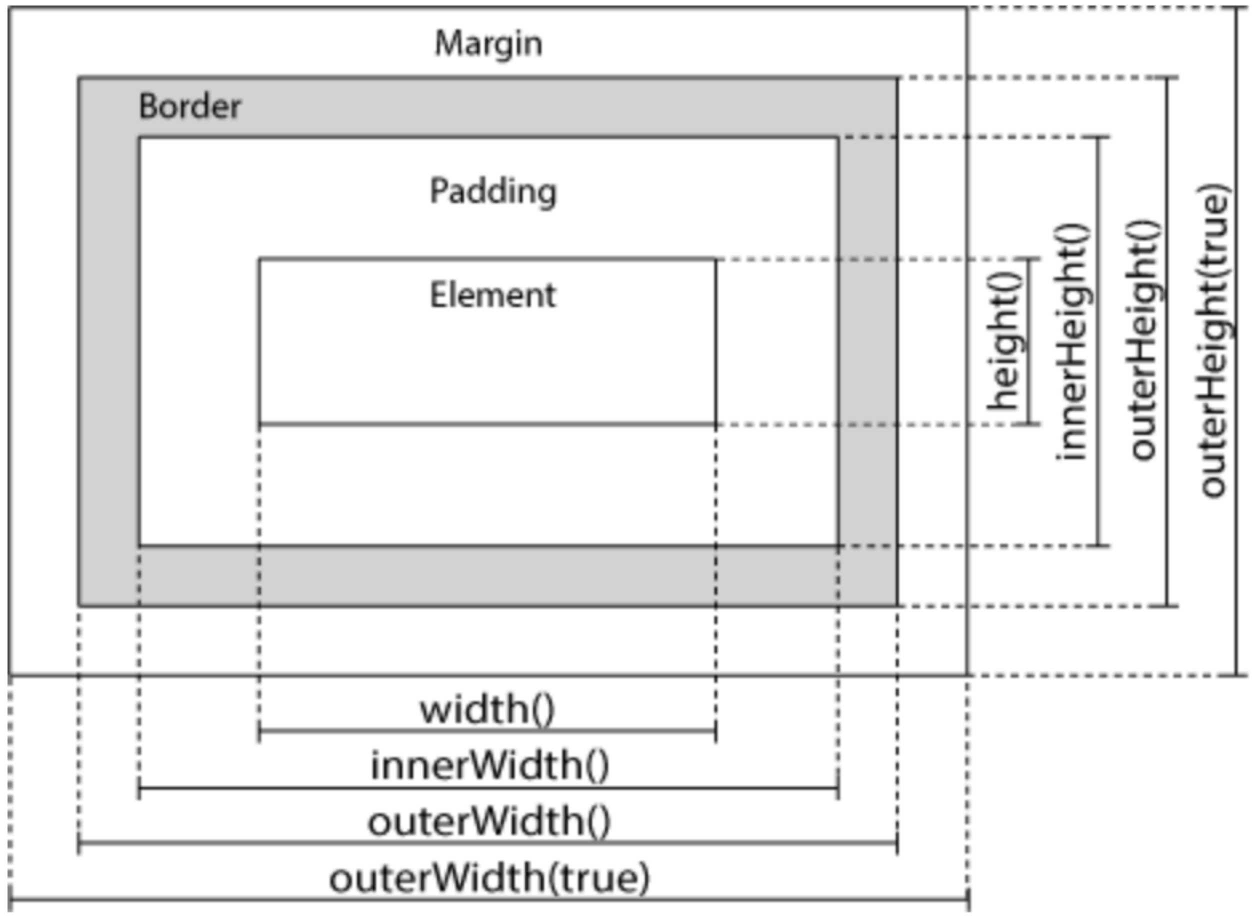
- To set multiple CSS properties, use the following syntax:

```
css({"propertyname1":"value1",
     "propertyname2":"value2",
      ...
});
```

Get and Set Dimensions

- jQuery has several important methods for working with dimensions:
  - `width()` - sets or returns the width of an element
  - `height()` – sets/returns the height of an element
  - `innerWidth()`
  - `innerHeight()`
  - `outerWidth()`
  - `outerHeight()`
  - Refer w3schools for examples

Get and Set Dimensions



(from: https://www.w3schools.com/jquery/jquery_dimensions.asp)

HTML Document Traversing

- jQuery traversing methods are used to "find" (or select) HTML elements based on their relative position to other elements.
- Think of the HTML document as a tree of elements. Starting with one of the elements in the tree, you can easily move up (ancestors), down (descendants) and sideways (siblings) within the tree structure.
  - An ancestor is a parent, grandparent, great-grandparent, and so on
  - A descendant is a child, grandchild, great-grandchild, and so on
  - Siblings share the same parent

Traversing Up

- Three useful jQuery methods for traversing up the DOM tree are:
  - `parent()` - returns the parent element of the selected element; i.e. a single step up the tree
  - `parents()` - returns all ancestor elements of the selected element; i.e. all the way up to the root element of the document tree
  - `parentsUntil()` - returns all ancestor elements between the selected element and a given argument

Traversing Down

- Two useful jQuery methods for traversing down the DOM tree are:
  - `children()` - returns all direct children of each occurrence of the selected element; i.e., a single step down the tree
    - You can filter the search with an optional parameter
  - `find()` - returns all descendant elements of the selected element all the way down to the last descendant

Traversing Sideway

- There are many useful jQuery methods for traversing sideways within the DOM tree:
  - `siblings()` - returns all sibling elements of the selected element
    - You can filter the search with an optional parameter
  - `next(),nextAll(),nextUntil()`
  - `prev(),prevAll(),prevUntil()`
- Refer w3schools for examples related to traversing using sibling methods

Filter Methods

- The most basic filtering methods are:
  - `first()`: returns the first element of the specified elements
  - `last()` : returns the last element of the specified elements
  - `eq()` : returns an element with a specific index number of the selected elements
  - The index number starts at 0

Filter Methods

- Other filtering methods:
  - `filter()` : allow you to select elements that match a certain criteria
  - `not()` : allow you to select elements that do not match a certain criteria
  - Refer w3schools for examples

References

- w3schools tutorial online:

  https://www.w3schools.com/Jquery/default.asp

- jQuery Succinctly, 2012. Lindley, C.
  - Available for free from www.syncfusion.com
- Learning jQuery, Fourth Edition 2013. Jonathan Chaffer and Karl Swedberg
- jQuery in Action, Third Edition 2015. Bear Bibeault, Yehuda Katz, and Aurelio De Rosa
- jQuery Documentation: https://api.jquery.com/