CS2204 Fundamentals of Internet Applications Development

8. Javascript - Part III

Application of Javascript:

- Introduction
- Dynamic Content
- Multimedia programming
- Common Techniques for Games
- Canvas for Drawing
- JS Library

In this course only introduce selected techniques to enable you to learn more advanced Javascript later.

8.1. Introduction

Before HTML5

- with CSS2, can do quite a lot but require lots of scripting, e.g.
  - form checking
  - dynamic content
  - animation & movement

HTML5 & New Web Technologies

- new development to address previous limitations
  - CSS3 - transform & animate
  - drag & drop
  - scripting of video & audio
  - drawing with canvas
- more JS based technologies, not directly related to HTML5 (sometimes called HTML5 API) such as offline (local) storage, geolocation, Web socket & Web worker, etc.

8.2. Dynamic Content

Common ways to build dynamic content :

- inline script
- show or hide
    - display
    - visibility
    - z-index
    - positioning
- innerHTML property
- DOM api
    - a set of methods to create elements and append to the DOM tree
    - more "formal" way of accessing the DOM but quite tedious to use

8.2.1. inline script

8.2.1. inline script

HTML can be created dynamically when the Web page is loading e.g.

```
document.write("<h1>writing out html now</h1>")
```

usually checking different conditions and create HTML accordingly, or something that is not known at HTML writing time, e.g. current date time.

Be careful where you execute the document.write(), HTML will be created at the location where you scripts are running. One technique to give a little better maintenance is to use a function call in the inline script and maintain the function in embedded or external script.

8.2.2. Hide & Show

JS allows you to change the CSS properties of an element. First select the element e.g. document.querySelector(), then make objects appear or disappear by changing the display property, e.g.

- object.style.display="none" - hide
- object.style.display="block" - show as block element

Visibility property

- object.style.visibility="hidden"
- object.style.visibility="visible"

Z-index - prepare all elements and stack them up with overlapping; change z-index to show either one by putting it on top.

With JavaScript, you can control all CSS properties dynamically by using object.style.property name. Note that the property name could be different from those used in CSS

```
object.style.width="120px"
object.style.backgroundColor="red"
```

the style properties set by CSS cannot be obtained by JS. If you want to use style properties in conditions, the properties must be initialized with JS first.

8.2.3. innerHTML | Example - http://courses.cs.cityu.edu.hk/cs2204/example/html/33-DynamicContent.html

8.2.3. innerHTML

This is not a standard property from W3C but all browsers support it. It allows access of the content of an element (or actual HTML) as a string, e.g.

```
<div id="mydiv">
  <div>
  …
  </div>
</div>
```

can change the HTML inside the div as

```
document.querySelector("#mydiv").innerHTML="<h2>Hello World</h2>";
```

after running this, all HTML inside div will be replaced by the h2.

8.2.4. Ajax

Ajax stands for

- asynchronous
- javascript
- and
- xml

Asynchronous - communicate with server in the background, javascript codes not executed in sequence. User may not be aware of the communication as there is no page reload.

XML - is the original format of data exchange between Web page and server. It is common now to use JSON as well.

This can be understood by referring back to the basic browser functioning with many connections back to the Web server during page load. After page initial load, use of Javascript to communicate further with the Web server in the background is Ajax. It is a more flexible way then external script to access data from (any) server/database but there are complications needed to be understood and handled:

- asynchronous execution of Javascript - similar to event driven

- CORS - Cross Origin Resource Sharing. Browsers would block access if the http request is to domains different from that of the Web page. This make testing with local files not possible.

An object is provided for Ajax programming:

```
let xhttp = new XMLHttpRequest();
http.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
     //responseText is in JSON format
       return init(JSON.parse(this.responseText));
       }
  };
xhttp.open("GET",
           "https://courses.cs.cityu.edu.hk/cs2204/example/js/movies.js", true);
xhttp.send();
//
// ajax will not be completed here because of asynchronous execution
// init function used to start other actions
function init() { ... }
```

8.3. Multimedia Programming

Before HTML, although video and audio are supported with the <object> tag, there was no scripting ability. Image is the only media that can be scripted.

In HTML5, video & audio scripting API are provided in addition to the usual dynamic content techniques for :

- video
- audio
- slide show
- animation

8.3.1. Video

Video & audio are timed media, i.e. have to be played according to a time interval. Introduced in HTML5 and provide many useful properties, methods and events in JS, e.g.

- load()
- play()
- pause() – no stop method
- .duration
- .control
- .oncanplay
- .onended

Programming the video can best be illustrated with the VCR (video camera and recorder) operations in the example below. Note the logic :

- use methods to play and pause the video
- stop is to set the play time to the beginning and pause
- fast forward is add video duration/10 to current play time, reset to 0 if larger than duration
- fast backward is minus current play time

Also check out :

- how to set up various events for different operations?
- how to avoid timing problem – use of buttons before the video is ready?

How to switch video by JS?

- can manipulate different HTML and their properties/attributes
- 3 ways shown in the example below
    - change .src property of <video> tag
    - change the whole <video> tag by .innerHTML of container
    - change .src property of <source> tag

8.3.2. Audio

Audio is similar to video except that it does not really need a visual interface

- audio can exist as a JS object without the <audio> tag
- e.g. var a= new Audio();
- especially useful for storing multiple audios for different events

Note some of the techniques in the example below

- creating the audio object with a URL
- storing many audio objects in an array
- pre-loading the audio files so they could be played faster

8.3.3. Slide Show

Slide show is in fact showing images one by one at a certain interval of time.

2 ways to execute Javascript periodically

```
setInterval(function, interval)
```

- e.g. setInterval("myfunction( )", 500)
- to execute the function myfunction every 500 milliseconds

```
setTimeOut(function, interval)
```

- execute the function/code after interval millisec
- to execute repeatedly, need to use the method recursively in a function

To stop execution after the Javascript is started

- clearInterval(id) or clearTimeOut(id) can stop the execution
- where id is the return value of the setInterval( ) or setTimeout( ) methods
- e.g. id = setTimeout(function, interval); clearTimeout(id); there may be more than one setTimeOut/setInterval in your code and their return ids are different

8.3.4. Animation

Show a sequence of images at a certain speed

- same as slide show
- require a sequence of action images
- setInterval or setTimeout again

In the example :

- use setInterval in this example
- animation speed (frames per second) determined by the time interval

8.4. Common Techniques for games

In game programming, common techniques usually involve :

- movement & layering
- object location tracking
- collision detection
- keyboard handling
- drag and drop
- local storage

8.4.1. Movement & Layer

Movement is a continuous change of positions. Positioning schemes other than normal flow use CSS properties: left, right, top, bottom for precise control of positions. 2 functions setInterval and setTimeout can execute Javascript continuously or at a fixed time. Combining the two can achieve continuous change of positions.

Layer can be thought of containing characters and background to form scenes. Different scenes can be stacked up with overlapping using z-index and non-static positioning schemes. They can then changed rapidly by changing the z-index.

The example below demonstrates some of the idea :

- relative positioning is used for the circle
- the circle and the alternate vertical bars have different z-index
- circle moves from left to right, bottom to top, passing on top and below of vertical bars

8.4.2. Object location tracking

Movement using position schemes

- fixed, absolute & relative
- all involve manipulation of top, left, right & bottom properties
- keep track of these values can determine object location
- usually done with self-defined Javascript variables

By using object method

```
x = obj.getBoundingClientRect()
```

where obj is an element

- return x as an object with properties x.top, x.left, x.right & x.bottom
- values are relative to window, require adjustment if page scrolled

  ```
  myElement.getBoundingClientRect().top + window.scrollY;
  ```

  but scrollY could be incorrect in some browsers

8.4.3. Collision Detection

Object collision is when two moving objects start to overlap, could be complicated because it can occur in different directions and objects can have different shapes.

One simplified algorithm

- use the tracked locations to do the calculation
- use one or more corners of the moving object, (x,y)
- check whether it is inside another object
    - assume absolute positioning
    - left, left + width; top, top + height
    - inside ?

```
if (x > left && x < left + width) &&
   (y > top && y < top + height)
```

- only work for rectangle

8.4.4. Keyboard Handling

Frequently used in gaming, need to know

- key has been pressed and released
- which key?

More than one events are triggered

- onkeydown (for all keys)
- onkeypress
    - some keys won’t fire this event
    - e.g. alt, ctrl, shift, esc
- onkeyup

Key codes

- finding which key is pressed using key code is not standard across browsers / versions of a browser
- eventobject.charCode

- ◦ complicated if more than one key is pressed, e.g. shift a

Better test your browsers first.

8.4.5. Drag and Drop

Processing Model

- drop a source object to a destination container
- source set to be dragable
- a series of events (and corresponding actions) when the source is selected and dragged
    - event ondragstart – get the id of the source object
    - event ondragover – source is now on top of another (destination) object
        - whether it is a destination depends the event handler
        - default action is not allowed to drop
        - if it is destination object – turn off default by preventDefault()
    - event ondrop – dropping on the destination
        - get back source object id
        - move it inside the destination container and update the DOM
        - the result is in fact moving the HTML of source to the destination

8.4.6. Drag and Drop Example

8.4.6. Drag and Drop Example

Drag an image from one division to another

- image is the source object, need to set id
- another div is the destination container

Once drag and drop starts

- an event object is created and passed to event handlers
- ondragstart event handler – for image
  - `function drag(event) {event.dataTransfer.setData(“Text”, event.target.id); }`
  - the image is the target
  - ondragover event handler – for destination div

```
function allowDrop(event) {event.preventDefault();}
```

  - ondrop event handler – for destination div
  - note the border color changed showing the image is now actually inside the destination div

  -
    ```
    function drop(event) {
      event.preventDefault(); // default action is open a window to show the im
    ```

```
    var data=event.dataTransfer.getData("Text");
    //get back id of image
    event.target.appendChild(document.querySelector("#"+data));
    //update the DOM by moving the image inside the destination div
  }
```

8.5. Local Storage

Information in a Web page

- information include variables and objects
- temporary – go away when page is reloaded or loaded with other page
- need to store information permanently or at least hold information until browser is closed (a session)
- database? could be an overkill

Local storage

- store information in local computer so that the browser can access
- available in HTML5, not in old versions
- in the form of objects
  - window.localStorage – store data with no expiration date
  - window.sessionStorage – store data for one session, lost when the tab is closed
  - localStorage.setItem('key', value string), localStorage.getItem('key'), localStorage.clear() ...
  - can store objects, but more complicated,JSON.stringify()

Example

- instead of submitting the form data, save to local storage
- can get back even after closing the tab

8.6. Canvas

A HTML element/tag

- serves as a drawing board inside the Web page
- draw by calling a set of predefined JS functions (Canvas API)
- work with a co-ordinate (x-y) system
  - note its co-ordinate system is defined by the width and height attributes
  - not the same as CSS styles width and height - conversion will be done if they are not the same
  - left upper corner is (0, 0)
- need to get a context (an object) before drawing
  - 2D only, 3D is still under development

Many complex functions - only introduce a few basic functions here.

8.6.1. Line

8.6.1. Line

Imagine a invisible pen

- “ink” functions
  - can actually draw on the context, creating lines, circles and rectangles, etc.
- functions with no “ink”
  - move the pen around in the x-y co-ordinate system
  - begin and end paths (a sequence of drawings)
  - set style of the pen (line width/color, fill color, etc.)

Line

- move to a start point - ctx.moveTo(x, y), where ctx is the context object
- sketch the line by the end point - ctx.lineTo(x, y)
- actually draw the line - ctx.stroke()
- new lines connected together unless the current path is ended and a new one created - ctx.beginPath()

8.6.2. Circle | Example - http://courses.cs.cityu.edu.hk/cs2204/example/html/37-CanvasLineCircle.html

8.6.2. Circle

Use method .arc() of the context

```
context.arc(x,y,r,sAngle,eAngle,counterclockwise)
```

select the centre point - x, y, determine the radius - r

Determine the portion of the circle to be drawn in degree in radian

- horizontal line through centre is 0 radian (i.e. 3 o'clock position)
- 360^0 = 2*Math.PI radian
- full circle - ctx.arc(x, y, r, 0, Math.PI * 2)
- half circle - ctx.arc(x, y, r, 0, Math.PI)

Direction of draw

- true - anti-clockwise
- false - clockwise
- lower half circle - ctx.arc(x, y, 0, Math.PI, false)
- upper half circle - ctx.arc(x, y, 0, Math.PI, true)

Fill

- line only - ctx.stroke() after ctx.arc(...)
- with fill - ctx.fill() after ctx.arc(...)

8.6.3. Rectangle & Text

Rectangle

```
ctx.strokeRect(top-left point, bottom-right point)
ctx.fillRect(top-left point, bottom-right point)
```

Text

- set font size and type
- stroke or fill with the text and the top-left start point

```
ctx.font = "bold 1.8em sans-serif";
ctx.fillText("HELLO MY NAME IS", 12, 40);
```

8.6.4. Image

Create an image object, set its source

```
var img = new Image();
img.src = “../images/mypic.jpg”;
```

Draw at top-left point

```
ctx.drawImage(img, x, y)
```

Drawback of this method

- the image may not be fully loaded when drawn
- will not be seen unless keep reloading the page
- use an onload event handler to draw, see example below

8.7. Use of Library

External script

- when some of your codes are placed externally and used repeatedly - you have created a library
- lots of people are providing codes for others to use - 3rd party library

API - application programming interface

- expose objects and methods for users to use your library
- also need to consider objects that you don't want other people to access

Different kinds of library

- help to write DOM accessing JS easier - jQuery, YUI
- widget/GUI library - jQuery UI
- specialized library - Google Map, ... & others

8.8. jQuery

Full learning with jQuery

- a very popular and powerful library
- requires strong background in basic JS

Just pick some examples as introduction

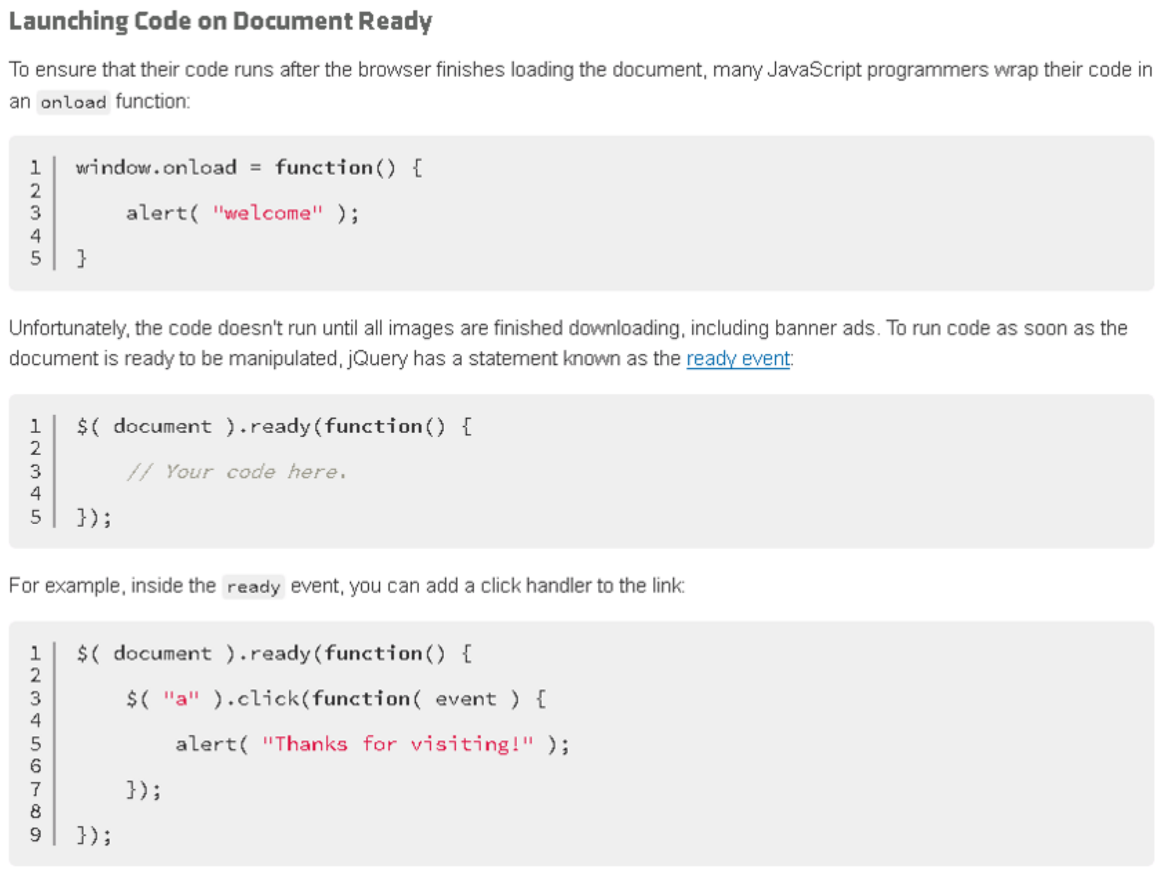
- that do things similar to what have been learnt
- see how it simplifies things (or make it difficult to understand?!)

How to use

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" /><title>Demo</title>
</head>
<body><a href="http://jquery.com/">jQuery</a>
<script src="jquery.js"></script>
<script>// Your code goes here.
```

```
</script>
</body>
</html>
```

8.8.1. Launching code



Launching Code on Document Ready

To ensure that their code runs after the browser finishes loading the document, many JavaScript programmers wrap their code in an `onload` function:

```
window.onload = function() {

    alert( "welcome" );

}
```

Unfortunately, the code doesn't run until all images are finished downloading, including banner ads. To run code as soon as the document is ready to be manipulated, jQuery has a statement known as the ready event:

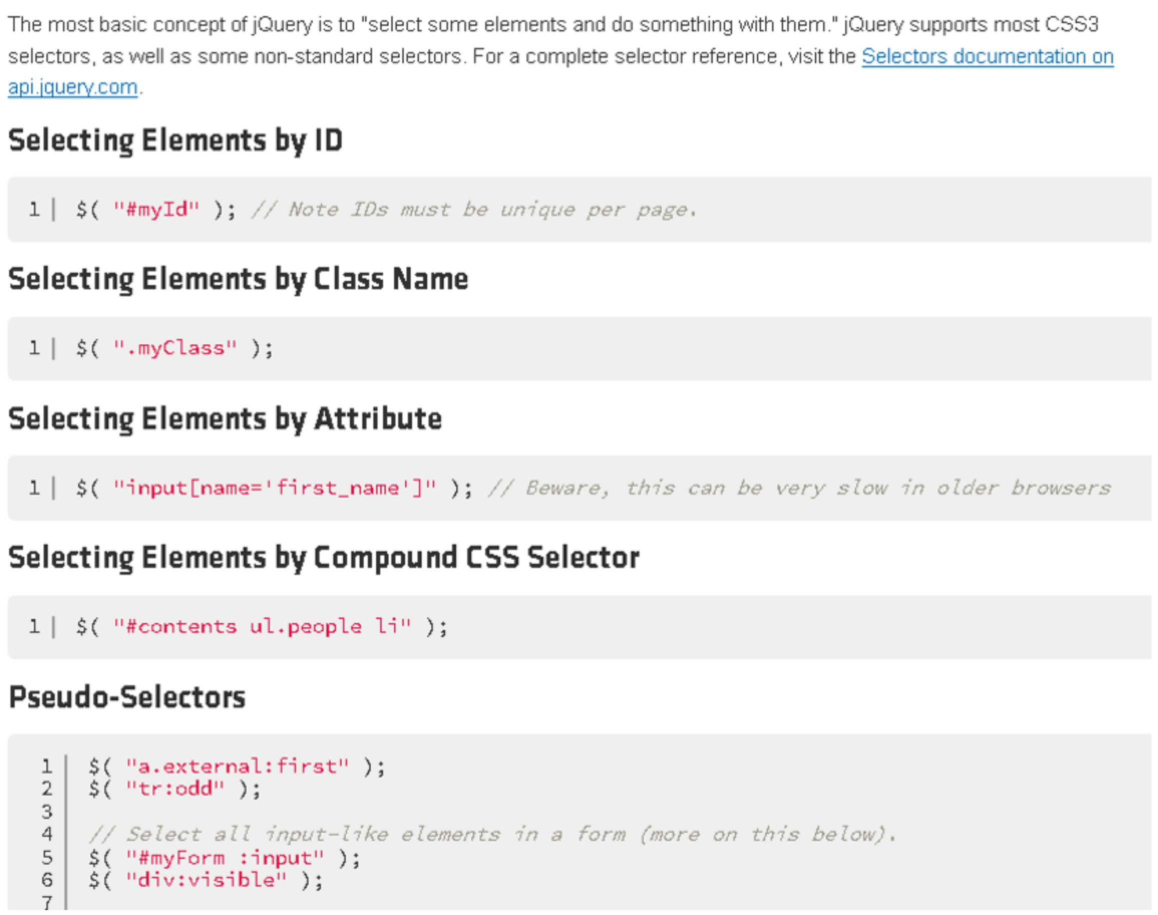
```
$( document ).ready(function() {

    // Your code here.

});
```

For example, inside the `ready` event, you can add a click handler to the link:

```
$( document ).ready(function() {

    $( "a" ).click(function( event ) {

        alert( "Thanks for visiting!" );

    });

});
```

8.8.2. Selecting Elements

8.8.2. Selecting Elements



The most basic concept of jQuery is to "select some elements and do something with them." jQuery supports most CSS3 selectors, as well as some non-standard selectors. For a complete selector reference, visit the Selectors documentation on api.jquery.com.

Selecting Elements by ID

```
$( "#myId" ); // Note IDs must be unique per page.
```

Selecting Elements by Class Name

```
$( ".myClass" );
```

Selecting Elements by Attribute

```
$( "input[name='first_name']" ); // Beware, this can be very slow in older browsers
```

Selecting Elements by Compound CSS Selector

```
$( "#contents ul.people li" );
```

Pseudo-Selectors

```
$( "a.external:first" );
$( "tr:odd" );

// Select all input-like elements in a form (more on this below).
$( "#myForm :input" );
$( "div:visible" );
```

8.8.3. Working with Selections

8.8.3. Working with Selections

Getters & Setters

jQuery "overloads" its methods, so the method used to set a value generally has the same name as the method used to get a value. When a method is used to set a value, it's called a setter. When a method is used to get (or read) a value, it's called a getter. Setters affect all elements in a selection. Getters get the requested value only for the first element in the selection.

```
// The .html() method used as a setter:
$( "h1" ).html( "hello world" );
```

```
// The .html() method used as a getter:
$( "h1" ).html();
```

The `.attr()` method acts as both a getter and a setter. As a setter, `.attr()` can accept either a key and a value, or an object containing one or more key/value pairs.

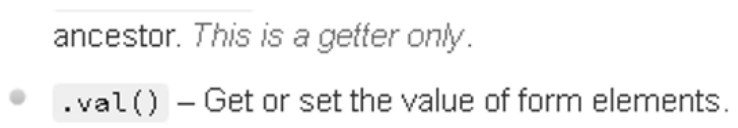
`.attr()` as a setter:

```
$( "a" ).attr( "href", "allMyHrefsAreTheSameNow.html" );

$( "a" ).attr({
    title: "all titles are the same too!",
    href: "somethingNew.html"
});
```

`.attr()` as a getter:

```
$( "a" ).attr( "href" ); // Returns the href for the first a element in the document
```

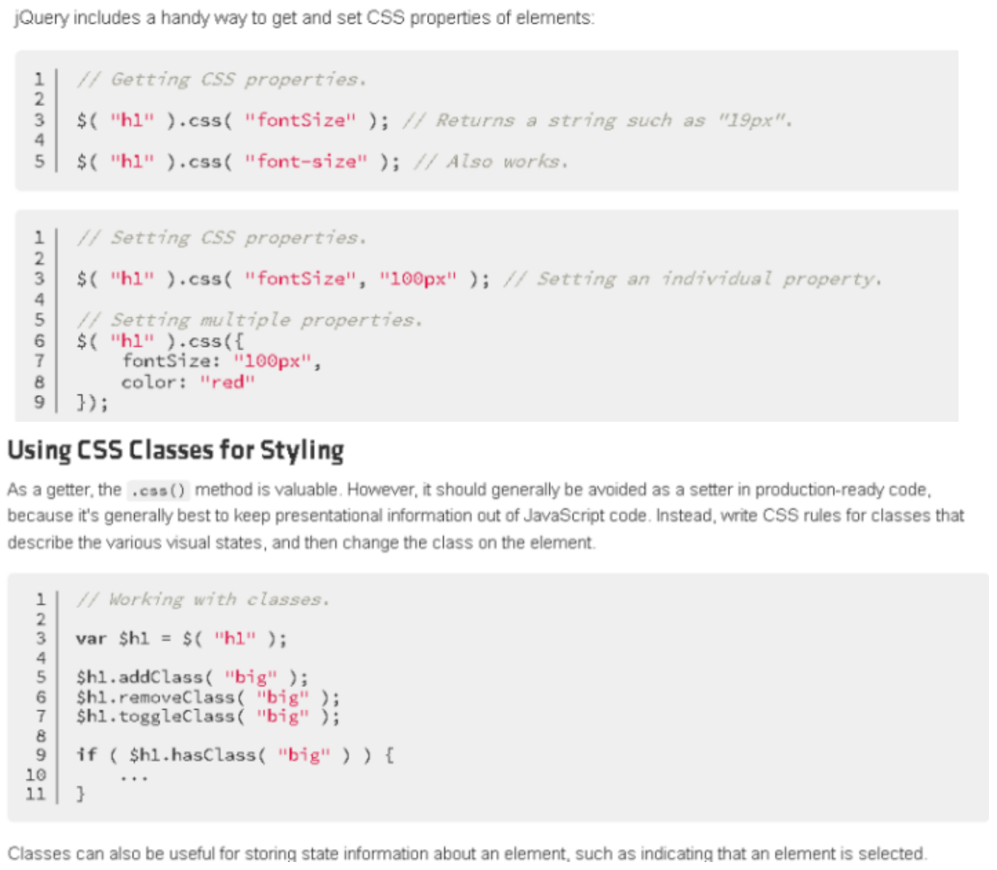
- `.html()` – Get or set the HTML contents.
- `.text()` – Get or set the text contents; HTML will be stripped.
- `.attr()` – Get or set the value of the provided attribute.
- `.width()` – Get or set the width in pixels of the first element in the selection as an integer.
- `.height()` – Get or set the height in pixels of the first element in the selection as an integer.
- `.position()` – Get an object with position information for the first element in the selection, relative to its first positioned


ancestor. *This is a getter only*.

- `.val()` – Get or set the value of form elements.

8.8.4. Styling & CSS

8.8.4. Styling & CSS



jQuery includes a handy way to get and set CSS properties of elements:

```
// Getting CSS properties.

$( "h1" ).css( "fontSize" ); // Returns a string such as "19px".

$( "h1" ).css( "font-size" ); // Also works.
```

```
// Setting CSS properties.

$( "h1" ).css( "fontSize", "100px" ); // Setting an individual property.

// Setting multiple properties.
$( "h1" ).css({
    fontSize: "100px",
    color: "red"
});
```

Using CSS Classes for Styling

As a getter, the `.css()` method is valuable. However, it should generally be avoided as a setter in production-ready code, because it's generally best to keep presentational information out of JavaScript code. Instead, write CSS rules for classes that describe the various visual states, and then change the class on the element.

```
// Working with classes.

var $h1 = $( "h1" );

$h1.addClass( "big" );
$h1.removeClass( "big" );
$h1.toggleClass( "big" );

if ( $h1.hasClass( "big" ) ) {
    ...
}
```

Classes can also be useful for storing state information about an element, such as indicating that an element is selected.

8.8.5. Event Handling & Effects

8.8.5. Event Handling & Effects

jQuery offers convenience methods for most native browser events. These methods — including `.click()`, `.focus()`, `.blur()`, `.change()`, etc. — are shorthand for jQuery's `.on()` method. The on method is useful for binding the same handler function to multiple events, when you want to provide data to the event hander, when you are working with custom events, or when you want to pass an object of multiple events and handlers.

```
// Event setup using a convenience method
$( "p" ).click(function() {
    console.log( "You clicked a paragraph!" );
});
```

```
// Equivalent event setup using the `.on()` method
$( "p" ).on( "click", function() {
    console.log( "click" );
});
```

8.9. Conclusions

8.9. Conclusions

This course only shows using Javascript in Web page which should be a start for you to learn more complete and comprehensive knowledge of Javascript.

Extension readings :

- text Book
- Javscript 101
- jQuery UI effect
- other library/framework
- prototype.js, angular.js, ... & many more