JavaScript and jQuery

Trainer: Duyen Dang
Title: Senior Software Engineer

Introduction

- Your role
- Your background and experience in the subject
  - HTML, CSS, JavaScript
- What do you want from this course

OBJECTIVES

- At the end of the course, you will have acquired sufficient knowledge to:
  - Understand the core of JavaScript and jQuery
  - Build or modify dynamic web pages and web applications.

PREREQUISITE

- Have knowledge
  - Html(5), Css(3)
- Editor:
  - VS, Eclipse, Notepad++, Brackets, Sublime Text …
- Developer tool:
  - Firebug (Firefox), Chrome developer tool…

Assessment Disciplines & Timetable

- **Assessment Disciplines**
  - Class Participation : Required
  - Assignment Completion : 100%
  - Pass Criteria: >=70%
- **Course Timetable**
  - Lecture Duration: 6 hours

AGENDA

I. JavaScript Overview
II. Programming with JavaScript
III. JavaScript Objects
IV. JavaScript Functions
V. JavaScript Prototype
VI. Ajax
VII. JQuery

JavaScript Overview

What is JavaScript?

- An interpreted programming language with object-oriented capability (optional JIT Compilation support).
- Lightweight and runs on the client side and interpreted by browser
- Loosely typed and can be embedded directly into HTML pages
- Everyone can write/run without license
- NOT JAVA

What can client-side JavaScript do?

- Contains an extended set of functionality to interface with the web browser DOM (Document Object Model)
- Event detection and handling
- Support pre-defined Objects (window, document, location etc).
- Manipulate HTML elements
- Make ajax request to server
- Others capabilities, like form validation,

  cookies storage, etc.

Pros and Cons of JavaScript

- Pros
  - Speed: Fast executed on client-side
  - Simplicity: Simple to learn and implement
  - Versatility: Play nicely with other languages
  - ServerLoad: Reduce the demand on web server.
- Cons
  - Security: can be exploited for malicious purpose.
  - Render Varies: may rendered differently in old browser engines.

LOCATION OF JS IN HTML

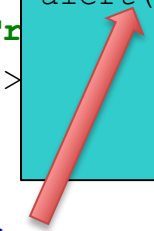
- JavaScript is put inside HTML between <script>
- <script> tag could be put in any places inside a html page (at <head>, <body>…)
- Can be kept in an external JavaScript file and referenced in HTML code

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

LOCATION OF JS (EXAMPLE)

/js/script.js

```
alert("This is external JavaScript");
```



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Tr
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>JavaScript Location Example</title>
    <script type="text/javascript" src="js/script.js"></script>
    <script type="text/javascript">
        alert("This is inline JavaScript in <head> tag.");
    </script>
</head>
<body>
    <script type="text/javascript">
        alert("This is inline JavaScript in <body> tag.");
    </script>
    <h1>This is a JavaScript Location Example</h1>
</body>
</html>
```

EVENTS

- Html event can be something browser does, or something user does.
- Syntax:

```
<some-HTML-element some-event="some JavaScript">
```

<button onclick="displayDate()">The time is?</button>

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JS OUTPUT

- JavaScript does NOT have any built-in print or display functions.
- JavaScript can "display" data in different ways:
  - Writing into an alert box, using **window.alert()** .
  - Writing into the HTML output using **document.write()**.
  - Writing into an HTML element, using **innerHTML**.
  - Writing into the browser console, using **console.log()**.

Programming with Javascript

VARIABLES

- Begin with a letter or $, _
- Case sensitive
- There are global scope and functional (lexical) scope
- Declare with ***var*** for global and local variables. No ***var*** for function parameters
- No type declaration

```
//Declare Number variables
var whole = 3;
var decimal = 3.14159265;
//Variable decimal2 implicitly declared
decimal2 = decimal - whole;
//Declare String variables
var single = 'Just single quotes';
var double = "Just double quotes";
var singleEscape = 'He said \'RUN\' ever so softly.';
var doubleEscape = "She said \"hide\" in a loud voice.";
//String operators
var name = "Khanh Nguyen";
var sentence = "My name is " + name;
//Declare Boolean variables
var lying = true;
//Assign new type
whole = "3";
whole = true;
```

VARIABLES HOISTING

- A variable can be used before it has been declared
- Hoisting (JS behavior) moves all declarations to the top of current scope.

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

- Best practice:
  - Declare variables on top
  - “use strict”

OPERATORS

- Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

OPERATORS (CONT)

- Assignment Operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

OPERATORS (CONT)

- Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

OPERATORS (CONT)

- Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

CONDITIONALS

if... statement

if...else statement

if...else if....else statement

```
if (condition1) {
    conditional 1 code execute;
} else if (condition2){
   conditional 2 code execute;
} else {
   conditional 3 code execute if 2
  above not match;
}
```

CONDITIONALS (CONT)

switch statement

```
switch (n)
{
  case 1:
    conditional 1 code execute;
    break;
  case 2:
    conditional 1 code execute;
    break;
  default:
    code execute if n is different from 1 and 2;
}
```

LOOPS

for and for...in loop

```
for (var i=0; i <= 8; i++) {
  code;
}

var x;
var txt="";
var person={fname:"John",lname:"Doe",age:25};
for (x in person) {
   txt=txt + person[x];
}
```

LOOPS (CONT)

while and do… while loop

```
while (var i < 8) {
  code to be executed;
}


do {
  code to be executed;
} while (var i < 8)
```

ERROR HANDLING

- The try...catch… statement
  - The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

```
try {
  // code to be executed and can be
 generated errors;
} catch (err) {
  // Handle errors here
}
```

Popup Boxes

```
alert("someText");
confirm(someText);
prompt("someText", "defaultValue");
```

COMMENTS

The syntax of **comments** is the same as in C++ and in many other languages

```
// a one line comment
/* this is a longer,
multi-line comment */
```

JAVASCRIPT OBJECTS

JavaScript Datatypes

- Primitive
  - Number
  - String
  - Boolean
- Trivial
  - Undefined
  - Null
- Object

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof false                  // Returns boolean
typeof [1,2,3,4]              // Returns object
typeof {name:'John', age:34}  // Returns object
```

WHAT IS AN OBJECT?

- Collections of properties and methods
- Object literal syntax:

{ name1 : value1 , ... , nameN : valueN }

```
var obj = {
  shiny: true,
  isShiny: function() {
    return this.shiny;
  }
};
obj.isShiny(); // true
```

Note: When a JavaScript variable is declared with the keyword "new", the variable is created as an object

3 WAYS TO CREATE OBJECT

- Use object literal

```
var course = { number: "CIT597", teacher="Dr. Dave" }
```

- Use new to create blank object, then add fields to it later

```
var course = new Object();
    course.number = "CIT597";
    course.teacher = "Dr. Dave";
```

- Use a constructor

```
function Course(n, t) {
     this.number = n;
     this.teacher = t;
}
var course = new Course("CIT597", "Dr. Dave");
```

ARRAY LITERALS

- No variable type declaration
- Array literal syntax:

  [value1, ... , valueN]
- Arrays are zero-based

```
var color= [“red”, , , “blue”, “green”];
=> Color has 5 elements
```

4 WAYS TO CREATE ARRAY

- Use array literals

```
var color = [“red”, “green”, “blue”];
```

- Use new Array() to create empty array

```
var color = new Array();
color[0] = “red”;
```

- Use new Array(n) with numeric argument to create empty array with that size

```
var color = new Array(3);
```

- Use new Array(…) with initial values

```
var color= new Array(“red”, “blue”, “green”);
```

ARRAYS AND OBJECTS

- Arrays are objects.

```
var person = {name: "John", age: 30};
```

  So, person.name is the same as person["name"].

- If you know the name of a property, use dot notation

```
person.name
```

- If you don't know the name of property, but you have it in variable, you must use array notation.

```
var prop= "name"; → person[prop]
```

ARRAY FUNCTIONS

Functions	Description
sort()	Sort the array alphabetically
reverse()	Reverse the array elements
push(object)	Add any number of new elements to the end of the array, and increase the array's length
pop()	Remove and return the last element of the array, and decrease the array's length
toString()	Return a string containing the values of the array elements, separated by commas.

JSON

- **J**ava**S**cript **O**bject **N**otation
- Is lightweight data interchange format
- Use Object and Array Literals
- Quotes required for Properties

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

JSON

```
<script>
var text = '{"employees":[' +
'{"firstName":"John","lastName":"Doe" },' +
'{"firstName":"Anna","lastName":"Smith" },' +
'{"firstName":"Peter","lastName":"Jones" }]}';

obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

JavaScript Functions

FUNCTIONS

- Function is an object.
- It has properties and methods.
- It can be copied, deleted, augmented.
- It is invokable.

```
function boo(what) {
  return what;
}
```

or

```
var boo = function(what) {
  return what;
};
```

FUNCTIONS

- All functions return a value.
- If they don’t explicitly, they return undefined implicitly.
- Functions can return other functions.

FUNCTIONS

- A function will be executed by an event or by a call to the function.
- A function can be declared nested with child function inside.

```
function function_name(var1, var2,…, varN) {
  // code to be executed inside function;
}

function function_name(var1, var2,…, varN) {
  // code to be executed inside function;
  function sub_function_name(…) {
    // code to be executed inside function;
  }
}
```

Function Hoisting

- A function can be used before it is declared
- Hoisting (JS behavior) moves all declarations to the top of current scope.

```
myFunction(5);

function myFunction(y) {
    return y * y;
}
```

Function Hoisting

- A function can be used before it is declared
- Hoisting (JS behavior) moves all declarations to the top of current scope.

```
function toCelsius(fahrenheit) {
    return (5/9) * (fahrenheit-32);
}
document.getElementById("demo").innerHTML = toCelsius;
```

VARIABLE SCOPE

- Local variable
  - Can only be accessible from inside a function where it is defined.
  - Is hidden from other functions and other scripting code.

```
function myFunction() {
    var a = 4;
    return a * a;
}
```

VARIABLE SCOPE

- Global variable
  - Can be accessible and changed from everywhere in the page.
  - Belongs to the window object in the web page.

```
var a = 4;
function myFunction() {
    return a * a;
}
```

- Notes:

  Variables created **without** the keyword **var**, are always **global**, even if they are created inside a function.

Self-invoked Function

- Immediately Invoked Function Expression (IIFE)
  - Is invoked automatically, without being called.
  - Will execute automatically if the expression is followed by ().

```
(function () {
    var x = "Hello!!";      // I will invoke myself
})();
```

CONSTRUCTOR FUNCTION

- When invoked with **`new`** keyword, function returns an object known as **`this`**.
- We can modify properties and methods of **`this`** before it's returned.

```
var Person = function(name) {
  this.name = name;
  this.say = function() {
    return "Je m'appelle " + this.name;
  };
};
```

CONSTRUCTOR FUNCTION

- An object created with Constructor.

```
var julien = new Person("Julien");
julien.say();
→ "Je m'appelle Julien"
```

CONSTRUCTOR FUNCTION

- **constructor** property

```
function Person(){};
var jo = new Person();
jo.constructor === Person
→ true

var o = {};
o.constructor === Object
→ true

[1,2].constructor === Array
→ true
```

Constructor Function

- Built-in constructor functions
  - Object
  - Array
  - Function
  - RegExp
  - Number
  - String
  - Boolean
  - Date
  - Error, SyntaxError, ReferenceError…

CONSTRUCTOR

Use this	Not that
var o = {};	var o = new Object();
var a = [];	var a = new Array();
var re = /[a-z]/gmi;	var re = new RegExp('[a-z]', 'gmi');

var fn = function(a, b){
  return a + b;
}
	var fn = new Function('a, b',
	'return a+b');

JavaScript Prototype

PROTOTYPE

- Every JavaScript object has a prototype.
- Prototype is also an object.

CREATE PROTOTYPE

- Create prototype using constructor function

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
```

- Create new objects from same prototype

```
var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");
```

USING PROTOTYPE PROPERTY

- Add new properties to existing prototype

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
Person.prototype.nationality = "English";
```

Using prototype Property

- Add new methods to existing prototype

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
Person.prototype.name = function() {
    return this.firstName + " " + this.lastName;
};
```

COOKIE

- Cookies are data, stored in small text files, on your computer.
- Cookies were invented to solve the problem "how to remember information about the user“
  - When a user visits a web page, his name can be stored in a cookie
  - Next time the user visits the page, the cookie "remembers" his name.

```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

AJAX

PROBLEM WITH TRADITIONAL WEB?

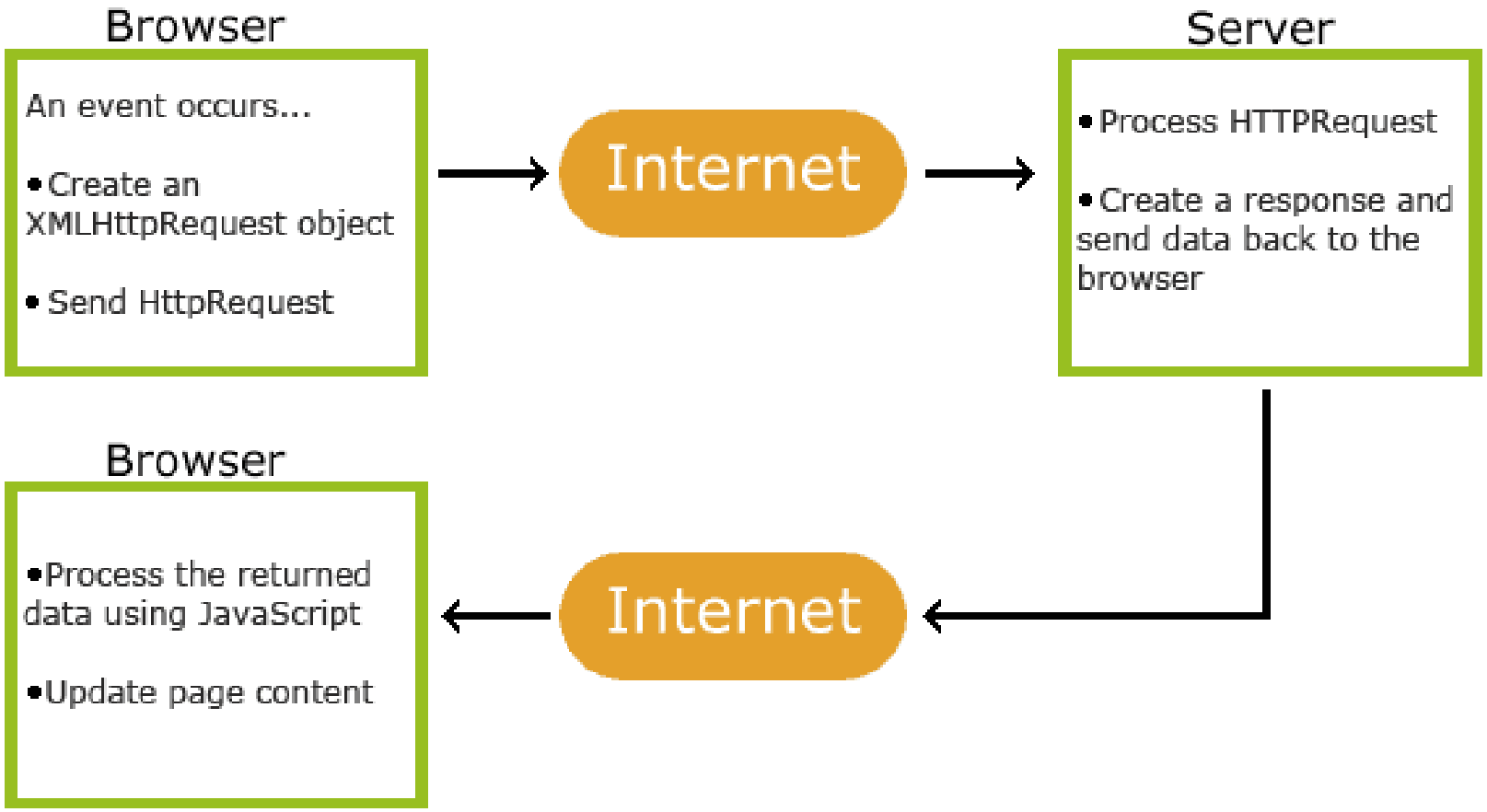
- In traditional web, to get information from server:
  - Make html form
  - GET or POST data to the server (like Submit button)
  - The browser loads a result page.

- Traditional web app run slowly and tend to be less user friendly.

WHY AJAX COME IN?

- AJAX = Asynchronous JavaScript and XML
- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

How Ajax works?



AJAX – SEND REQUEST

- Create XmlHttpRequest object

```
variable = new XMLHttpRequest();
```

- Send request to server

Method	Description
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

AJAX – GET OR POST ?

- Get is simpler and faster than POST

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

- Use POST when
  - Send large amount of data to server (no size limit)
  - Sending user input (robust and secure than GET)

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

AJAX – SERVER RESPONSE

- Use responseText or responseXML property of XMLHttpRequest object.

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

AJAX – EVENT & CALLBACK

- When a request to server is sent, perform some actions based on the response.

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

AJAX – EXAMPLE

```
<div id="demo"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadDoc()">Change Content</button>
```

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
     document.getElementById("demo").innerHTML = xhttp.responseText;
    }
  }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

JAVASCRIPT

- Demo and do exercises

JQUERY

What is jQuery?

- jQuery is a JavaScript Library
- jQuery is designed to change the way that you write JavaScript
- jQuery takes a lot of 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

jQuery features

- Write less, do more
- Awesome features:
  - HTML element selections
  - HTML element manipulation
  - CSS manipulation
  - HTML event functions
  - JavaScript Effects and animations
  - HTML DOM traversal and modification
  - AJAX

JQUERY – HOW TO GET STARTED

- Download the jQuery library from jQuery.com

```
<head>
<script src="jquery-1.12.4.min.js"></script>
</head>
```

- Include jQuery from a CDN (Content Delivery Network), like Google, Microsoft

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.4.min.js"></script>
```

jQuery – How to use

- Basic syntax is: **$(selector).action()**
  - A dollar sign ($) to define jQuery
  - A (selector) to query (or find) elements
  - A jQuery action() to be performed on the element(s)

    $(this).hide() - hides the current element.

    $("p").hide() - hides all <p> elements.
- Entry point **ready()** function

```
$(document).ready(function(){
        // jQuery methods go here...
});
```

JQUERY – SELECTOR

- Start with **$ or jQuery**

Selectors	Descriptions
$(“*")	selects all elements.
$(“p")	selects all <p> elements.
$("p.intro")	selects all <p> elements with class="intro".
$("p#intro")	selects the first <p> elements with id="intro".
$(":button")	selects all <button> and <input> elements of type="button".
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first

JQUERY – EVENT

- The jQuery event handling methods are core functions in jQuery.
- Event handlers are methods that are called when something happens in HTML. The term **triggered (or fired) by an event** is often used.

JQUERY – EVENT

Method	Description
bind()	Attaches event handlers to elements
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
focus()	Attaches/Triggers the focus event
submit()	Attaches/Triggers the submit event

JQUERY – EFFECT

Method	Description
hide()	Hides the selected elements
slideDown()	Slides-down (shows) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods
slideUp()	Slides-up (hides) the selected elements
toggle()	Toggles between the hide() and show() methods

JQUERY – HTML MANIPULATION

- jQuery methods for DOM manipulation are:
  - **text**() - Sets or returns the text content of selected elements
  - **html**() - Sets or returns the content of selected elements (including HTML markup)
  - **val**() - Sets or returns the value of form fields
  - **attr()** – Set or returns attribute values.

JQUERY – HTML MANIPULATION (CONT)

- More methods for HTML manipulation:

Method	Description
before()	Inserts content before selected elements
empty()	Removes all child nodes and content from selected elements
after()	Inserts content after selected elements
append()	Inserts content at the end of selected elements
remove()	Removes the selected elements (including data and events)
prepend()	Inserts content at the beginning of selected elements

JQUERY – 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

JQUERY – CSS MANIPULATION (CONT)

- Examples

```
$("button").click(function(){
    $("#div1").addClass("important blue");
});
```

```
$("button").click(function(){
    $("h1, h2, p").removeClass("blue");
});
```

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

jQuery – Traversing

Method	Description
find()	Returns descendant elements of the selected element
first()	Returns the first element of the selected elements
has()	Returns all elements that have one or more elements inside of them
last()	Returns the last element of the selected elements
parent()	Returns the direct parent element of the selected element
parents()	Returns all ancestor elements of the selected element

jQuery – Ajax

Method	Description
$.ajax()	Performs an async AJAX request
$.get()	Loads data from a server using an AJAX HTTP GET request
$.getJSON()	Loads JSON-encoded data from a server using a HTTP GET request
$.getScript()	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
$.post()	Loads data from a server using an AJAX HTTP POST request
load()	Loads data from a server and puts the returned data into the selected element

JQUERY

- Demo and do exercises

REVISION HISTORY

Date	Version	Description	Updated by	Reviewed and Approved By
Dec 01, 2015	1.0.0	Change styles follow new CSC theme	Tri Phan	
Jul 12, 2016	2.0.0	Modified knowledge - add more basic js knowledge and remove advance information to better suit to Fresher level	Duyen Dang	


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING