CSC-318
Web Technology
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

JavaScript Cookies

- Cookies let you store user information in web pages.
- Cookies are data, stored in small text files, on your computer.
- When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user
- Cookies were invented to solve the problem "how to remember information about the user":
  - When a user visits a web page, his/her name can be stored in a cookie
  - Next time the user visits the page, the cookie "remembers" his/her name

Cookies

- Cookies are saved in name-value pairs like:

```
username = John Doe
```

- When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users
- **Note : cookies do not work if your browser has local cookies support turned off**

Create a Cookie

- JavaScript can create, read, and delete cookies with the document.cookie property
- With JavaScript, a cookie can be created like this:

```
document.cookie = "username=John Doe";
```

- You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:

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

- With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

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

Read a Cookie

- With JavaScript, cookies can be read like this:

```
var x = document.cookie;
```

- document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;

Change a Cookie

- With JavaScript, you can change a cookie the same way as you create it:

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

- The old cookie is overwritten.

Delete a Cookie

- Deleting a cookie is very simple.
- You don't have to specify a cookie value when you delete a cookie.
- Just set the expires parameter to a passed date:

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC;
path=/;";
```

- You should define the cookie path to ensure that you delete the right cookie
- Some browsers will not let you delete a cookie if you don't specify the path

The Cookie String

- The document.cookie property looks like a normal text string. But it is not
- Even if you write a whole cookie string to document.cookie, when you read it out again, you can only see the name-value pair of it
- If you set a new cookie, older cookies are not overwritten. The new cookie is added to document.cookie, so if you read document.cookie again you will get something like:

```
cookie1 = value; cookie2 = value;
```

- If you want to find the value of one specified cookie, you must write a JavaScript function that searches for the cookie value in the cookie string

JavaScript Cookie Example

- In the example to follow, we will create a cookie that stores the name of a visitor
- The first time a visitor arrives to the web page, he/she will be asked to fill in his/her name. The name is then stored in a cookie
- The next time the visitor arrives at the same page, he/she will get a welcome message
- For the example we will create 3 JavaScript functions:
  - A function to set a cookie value
  - A function to get a cookie value
  - A function to check a cookie value

A Function to Set a Cookie

- First, we create a function that stores the name of the visitor in a cookie variable:

```
function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

- The parameters of the function above are the name of the cookie (cname), the value of the cookie (cvalue), and the number of days until the cookie should expire (exdays)
- The function sets a cookie by adding together the cookiename, the cookie value, and the expires string

A Function to Get a Cookie

- Then, we create a function that returns the value of a specified cookie:
- Take the cookiename as parameter (cname)
- Create a variable (name) with the text to search for (cname + "=")
- Decode the cookie string, to handle cookies with special characters, e.g. '$'
- Split document.cookie on semicolons into an array called ca (ca = decodedCookie.split(';'))
- Loop through the ca array and read out each value c = ca[i])
- If the cookie is found (c.indexOf(name) == 0), return the value of the cookie (c.substring(name.length, c.length)
- If the cookie is not found, return ""

```
function getCookie(cname) {
  var name = cname + "=";
  var decodedCookie = decodeURIComponent(document.cookie);
  var ca = decodedCookie.split(';');
  for(var i = 0; i <ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

A Function to Check a Cookie

- Last, we create the function that checks if a cookie is set
- If the cookie is set it will display a greeting
- If the cookie is not set, it will display a prompt box, asking for the name of the user, and stores the username cookie for 365 days, by calling the setCookie function

```
function checkCookie() {
  var username = getCookie("username");
  if (username != "") {
   alert("Welcome again " + username);
  } else {
    username = prompt("Please enter your name:", "");
    if (username != "" && username != null) {
      setCookie("username", username, 365);
    }
  }
}
```

JSON

- JSON: JavaScript Object Notation
- JSON is a syntax for storing and exchanging data
- JSON is a lightweight data-interchange format
- JSON is text, written with JavaScript object notation
- When exchanging data between a browser and a server, the data can only be text
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations
- JSON is language independent

JSON : Sending Data

- If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

JSON : Receiving Data

- If you receive data in JSON format, you can convert it into a JavaScript object:

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

- JSON uses JavaScript syntax, but the JSON format is text only. Text can be read and used as a data format by any programming language

Why Use JSON ?

- Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.
- JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects:
  - JSON.parse()
- So, if you receive data from a server, in JSON format, you can use it like any other JavaScript object

JSON Syntax Rules

- JSON syntax is derived from JavaScript object notation syntax:
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays
- JSON data is written as name/value pairs.
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

```
"name":"John"
```

- JSON names require double quotes. JavaScript names don't.

JSON - Evaluates to JavaScript Objects

- The JSON format is almost identical to JavaScript objects.
- In JSON, *keys* must be strings, written with double quotes:

```
{ "name":"John" }
```

- In JavaScript, keys can be strings, numbers, or identifier names:

```
{ name:"John" }
```

JSON Values

- In JSON, values must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null
- In JavaScript values can be all of the above, plus any other valid JavaScript expression, including:
  - a function
  - a date
  - undefined

JSON Values

- In JSON, string values must be written with double quotes:

```
{ "name":"John" }
```

- In JavaScript, you can write string values with double or single quotes:

```
{ name:'John' }
```

JSON Uses JavaScript Syntax

- Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.
- With JavaScript you can create an object and assign data to it, like this:

```
var person = { name: "John", age: 31, city: "New York" };
```

- You can access a JavaScript object like this:

```
// returns John
person.name;
```

- It can also be accessed like this:

```
// returns John
person["name"];
```

JSON.parse() : Parsing JSON

- A common use of JSON is to exchange data to/from a web server
- When receiving data from a web server, the data is always a string
- Parse the data with JSON.parse(), and the data becomes a JavaScript object
- Imagine we received this text from a web server:

```
'{ "name":"John", "age":30, "city":"New York"}'
```

- Use the JavaScript function JSON.parse() to convert text into a JavaScript object

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

- Make sure the text is written in JSON format, or else you will get a syntax error.

JSON From the Server

- You can request JSON from the server by using an AJAX request
- As long as the response from the server is written in JSON format, you can parse the string into a JavaScript object.

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj.name;
  }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

JSON.stringify()

- A common use of JSON is to exchange data to/from a web server.
- When sending data to a web server, the data has to be a string.
- Convert a JavaScript object into a string with JSON.stringify().
- Imagine we have this object in JavaScript:

```
var obj = { name: "John", age: 30, city: "New York" };
```

- Use the JavaScript function JSON.stringify() to convert it into a string

```
var myJSON = JSON.stringify(obj);
```

- The result will be a string following the JSON notation.
- myJSON is now a string, and ready to be sent to a server:

```
var obj = { name: "John", age: 30, city: "New York" };
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
```

Stringify a JavaScript Array

- It is also possible to stringify JavaScript arrays:
- Imagine we have this array in JavaScript:

```
var arr = [ "John", "Peter", "Sally", "Jane" ];
```

- Use the JavaScript function JSON.stringify() to convert it into a string.

```
var myJSON = JSON.stringify(arr);
```

- The result will be a string following the JSON notation.
- myJSON is now a string, and ready to be sent to a server:

```
var arr = [ "John", "Peter", "Sally", "Jane" ];
var myJSON = JSON.stringify(arr);
document.getElementById("demo").innerHTML = myJSON;
```

JSON Objects

```
{ "name":"John", "age":30, "car":null }
```

- JSON objects are surrounded by curly braces {}.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma

Accessing Object Values

- You can access the object values by using dot (.) notation:

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;
```

- You can also access the object values by using bracket ([]) notation:

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

- You can loop through object properties by using the for-in loop:

```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += x;
}
```

- In a for-in loop, use the bracket notation to access the property values:

```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += myObj[x];
}
```

Nested JSON Objects

- Values in a JSON object can be another JSON object.
- You can access nested JSON objects by using the dot notation or bracket notation:

```
x = myObj.cars.car2;
// or:
x = myObj.cars["car2"];
```

```
myObj = {
  "name":"John",
  "age":30,
  "cars": {
    "car1":"Ford",
    "car2":"BMW",
    "car3":"Fiat"
  }
}
```

Delete Object Properties

- Use the delete keyword to delete properties from a JSON object:

```
var myObj, i, x = "";
myObj = {
  "name":"John",
  "age":30,
  "cars": {
  "car1":"Ford",
  "car2":"BMW",
  "car3":"Fiat"
  }
}
delete myObj.cars.car2;

for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}
```

Ford
Fiat

Introduction to jQuery

- jQuery is a JavaScript Library
- ›jQuery greatly simplifies JavaScript programming
- ›jQuery is easy to learn
- ›The purpose of jQuery is to make it much easier to use JavaScript on your website
- ›Prerequisite : HTML, CSS, Javascript

Introduction to jQuery

- lightweight, "write less, do more", JavaScript library
- ›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
- ›also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation

Introduction to jQuery

- The jQuery library contains the following features:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities
- Many of the biggest companies on the Web use jQuery, such as
  - Google
  - Microsoft
  - IBM

jQuery Syntax

- Basic syntax is: $(selector).action()
  - a $ sign to define/access jQuery
  - a (selector) to "query (or find)" HTML elements
  - a jQuery action() to be performed on the element(s)
- Examples :
  - $(this).hide() - hides the current element
  - $("p").hide() - hides all <p> elements
  - $(".test").hide() - hides all elements with class="test"
  - $("#test").hide() - hides the element with id="test"
- All the jQuery code should be inside $(document).ready(function()

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

jQuery Selectors

- **Element Selector :**
  - selects elements based on the element name
  - When a user clicks on a button, all <p> elements will be hidden

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
<body>
    <h2>This is a heading</h2>
    <p>This is a paragraph.</p> <p>This is another paragraph.</p>
    <button>Click me to hide paragraphs</button>
</body>
```

jQuery Selectors

- **#Id Selector :**
  - uses the id attribute of an HTML tag to find the specific element
  - When a user clicks on a button, the element with id="test" will be hidden

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("#test").hide();
        });
    });
</script>
<body>
    <h2>This is a heading</h2>
    <p>This is a paragraph.</p>
    <p id="test">This is another paragraph.</p>
    <button>Click me</button>
</body>
```

jQuery Selectors

- **.class Selector :**
  - finds elements with a specific class
  - When a user clicks on a button, the elements with class="test" will be hidden

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $(".test").hide();
        });
    });
</script>
<body>
    <h2 class="test">This is a heading</h2>
    <p class="test">This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <button>Click me</button>
</body>
```

jQuery Events

- In jQuery, most DOM events have an equivalent jQuery method

```
<script>
    $("p").click(function(){
      // action goes here!!
    });
</script>
```

jQuery Events

- Commonly Used jQuery Event Methods
- Click :

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

- Dblclick :

```
$(document).ready(function(){
    $("p").dblclick(function(){
        $(this).hide();
    });
});
```

jQuery Events

- Commonly Used jQuery Event Methods
- Mouseenter:

```
$(document).ready(function(){
    $("#p1").mouseenter(function(){
        alert("You entered p1!");
    });
});
```

- Mouseleave :

```
$(document).ready(function(){
    $("#p1").mouseleave(function(){
        alert("Bye! You now leave p1!");
    });
});
```

jQuery Events

- **Commonly Used jQuery Event Methods**
  - Mouseenter
  - Mouseleave
  - Mousedown
  - Mouseup
  - Hover
  - Focus
  - Blur

jQuery Events

- **Commonly Used jQuery Event Methods**
  - "On" function can be used to handle one or more events
  - Eg. Single event

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

jQuery Events

- **Commonly Used jQuery Event Methods**
  - Eg. Multiple events

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

Exercise

- 1) Download the code from GitHub: https://github.com/CSY1018/Topic2 either clone the branch Exercise1-Exercise or download it as a zip
- Use jQuery so that when the button is clicked, its background colour is set to blue

Exercise Solution

```
<head>
    <title>Exercise 1</title>
    <link rel="stylesheet" href="style.css" />
    <!-- <script type="text/javascript"
    src="jquery-3.3.1.min.js">
    </script> -->
    <script src="https://ajax.googleapis.com/ajax/
    libs/jquery/3.3.1/jquery.min.js"></script>
    <script type="text/javascript" src="ex1.js"></
    script>
</head>
<body>
    <div id="circle">
        Click Me
    </div>
</body>
```

```
$(document).ready(function(){
    $('#circle').click(function(){
        $(this).css('background-color', 'blue')
    });
});
```

jQuery Effects

- jQuery Show/Hide

```
$(document).ready(function(){
    $("#hide").click(function(){
        $("p").hide();
    });
    $("#show").click(function(){
        $("p").show();
    });
});
```

jQuery Effects

- **jQuery Fading**
  - fadeIn
  - fadeOut
  - fadeToggle
  - fadeTo

jQuery Effects

- **jQuery Slide**
  - Slide Down
  - Slide Up
  - Slide Toggle

jQuery Effects

- Slide Down

```
$(document).ready(function(){
    $("#flip").click(function(){
        $("#panel").slideDown("slow");
    });
});
```

jQuery Effects

- Slide Toggle

```
$(document).ready(function(){
    $("#flip").click(function(){
        $("#panel").slideToggle("slow");
    });
});
```

jQuery Effects

- **Slide Animation**
- Lets you create custom animations.

```
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({left: '250px'});
    });
});
```

jQuery Effects

- **Slide Animation**
- Lets you create custom animations.

```
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({
            left: '250px',
            opacity: '0.5',
            height: '150px',
            width: '150px'
        });
    });
});
```

jQuery Effects

- **Slide Animation**
- Lets you create custom animations.

```
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({
            left: '250px',
            height: '+=150px',
            width: '+=150px'
        });
    });
});
```

jQuery HTML : Get

- Three simple, but useful, 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
- Get Attribute value

```
alert($("#w3s").attr("href"));
```

jQuery HTML : Set

- Three simple, but useful, jQuery methods for DOM manipulation are:
- **text(some text)** - Sets or returns the text content of selected elements
- **html(some html)** - Sets or returns the content of selected elements (including HTML markup)
- **val(some value)** - Sets or returns the value of form fields
- Set Attribute value

```
$("#w3s").attr("href", "https://www.w3schools.com/jquery/");
```

```
$("#w3s").attr({
    "href" : "https://www.w3schools.com/jquery/",
    "title" : "W3Schools jQuery Tutorial"
});
```

jQuery HTML : Add

- Append function : Appends at the end of the element

```
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").append(" <b>Appended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").append("<li>Appended item</li>");
    });
});
```

jQuery HTML : Add

- Append function : Appends at the start of the element

```
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").prepend("<b>Prepended text</b>. ");
    });
    $("#btn2").click(function(){
        $("ol").prepend("<li>Prepended item</li>");
    });
});
```

jQuery HTML : Remove

- Removing an element

```
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").remove();
    });
});
```

- Removing all the child elements

```
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").empty();
    });
});
```

jQuery HTML : Remove

- This removes all the paragraphs with class “test”

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").remove(".test");
    });
});
```

- This removes all the paragraphs with class “test” and “demo”

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").remove(".test, .demo");
    });
});
```

jQuery HTML : CSS

- **CSS Manipulating functions include :**
  - addClass
  - removeClass
  - toggleClass
  - css

jQuery HTML : CSS

- addClass

```
$(document).ready(function(){
    $("button").click(function(){
        $("h1, h2, p").addClass("blue");
        $("div").addClass("important");
    });
});
```

- removeClass

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

jQuery HTML : CSS

- toggleClass

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

- CSS

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

Exercise 2

- Create a HTML page with a single <button> element and an empty
- When the button is clicked, generate a random number from 1-100 and add it to the <ul> element as a
  - Each time the button is pressed a new number should be added to the page
- Use an array to keep track of which numbers have already been picked
- Use jQuery to accomplish this exercise.

Exercise 2 Solution

```
var pickedNumbers = [];
$(document).ready(function(){
    $('button').on('click',function(){
        var r = newRandom(pickedNumbers, 10);
        pickedNumbers.push(r);
        var text = '<li>'  + r + '</li>';
        $('ul').append(text);
    });
});
```

```
function arrayContains(array, value) {
    var found = false;
    for (var i = 0; i < array.length; i++) {
        if (array[i] == value) {
        found = true;
        }
    }
    return found;
}
function newRandom(array, max){
    var newNumber = false;
    while (newNumber == false) {
        var r = Math.floor(Math.random() * max);
        var alreadyPicked = arrayContains(array, r);
        if (alreadyPicked == false) {
            newNumber = true;
        }
    }
    return r;
}
```