CSE417: WEB ENGINEERING

Daffodil International University

I hear, and I forget.
I see, and I remember.
I do, and I understand.
-Chinese Proverb

Learning Outcome

- ✔ Use DOM to manipulate your content
- ✔ Make your page dynamic
- ✔ Use Forms
- ✔ Handle events

Contents

- Window Object
- DOM and Events
- Class and Objects
- HTML Form
- Events

Window Object

- Represents an open window in a browser
- Many window object properties and methods area available
- No public standard but major browsers support it
- If a document contain frames, then there is
  - one window object, window, for the HTML document • and one additional window object for each frame,
  - accessible via an array window.frames
- Methods provided by a window object include
    - close() - closes a browser window/tab
    - focus() - give focus to a window (bring the window to the front)
    - blur() - removes focus from a window (moves the window behind others)
    - print() - prints (sends to a printer) the contents of the current window
- Window Object: Dialog Boxes
  - Example:

```
alert("Local time: " + (new Date).toString())
```

- And many moreAnd many more…

navigator Object

`navigator.appName`
property that gives the browser name

`navigator.appVersion`
property that gives the browser version

```
<!-- MSIE.css  -->

a {text-decoration:none;
   font-size:larger;
   color:red;
   font-family:Arial}
a:hover {color:blue}
```

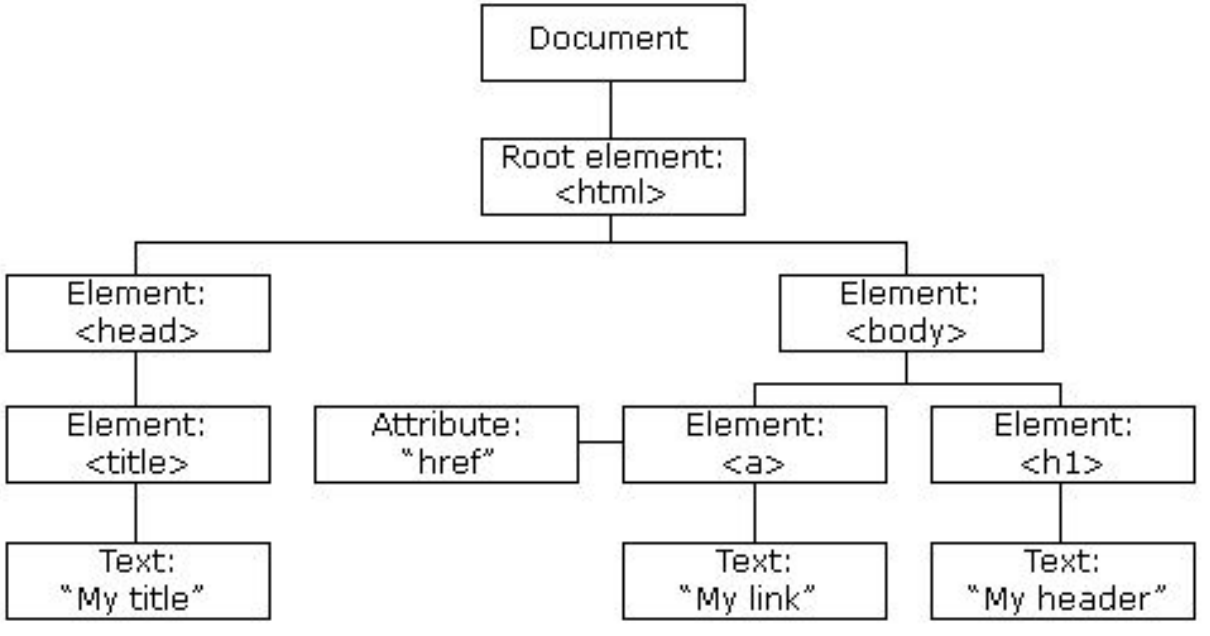
```
<!-- Netscape.css  -->

a {font-family:Arial;
   color:white;
   background-color:red}
```

```
<html>
<!-- CSE391  js14.html   -->

<head>
  <title>Dynamic Style Page</title>

  <script type="text/javascript">
    if (navigator.appName == "Netscape") {
      document.write('<link rel=stylesheet '+
        'type="text/css" href="Netscape.css">');
    }
    else {
      document.write('<link rel=stylesheet ' +
        'type="text/css" href="MSIE.css">');
    }
  </script>
</head>

<body>
Here is some text with a
<a href="javascript:alert('GO AWAY')">link</a>.
</body>
</html>
```

view page

The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:



- Standard **object** model and **programming interface** for HTML.
- It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

Document Object

Acces information about an HTML document using the `document` object *(Note: not a class!)*

```
<html>
<!-- CSE391  js13.html   -->

<head>
  <title>Documentation page</title>
</head>

<body>
  <table width="100%">
    <tr>
      <td><small><i>
        <script type="text/javascript">
            document.write(document.URL) ;
        </script>
      </i></small></td>
      <td style="text-align: right;"><small><i>
        <script type="text/javascript">
            document.write(document.lastModified) ;
        </script>
      </i></small></td>
    </tr>
  </table>
</body>
</html>
```

`document.write(…)`
method that displays text in the page

`document.URL`
property that gives the location of the HTML document

`document.lastModified`
property that gives the date & time the HTML document was last changed

view page

Reacting to Events

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
  - a page loads, user clicks a button, press any key, closing a window, resizing a window, etc.
- Developers can use these events to execute JavaScript coded responses,
  - which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.
- Events are a part of the DOM
- JavaScript can be executed when an event occurs
  - To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

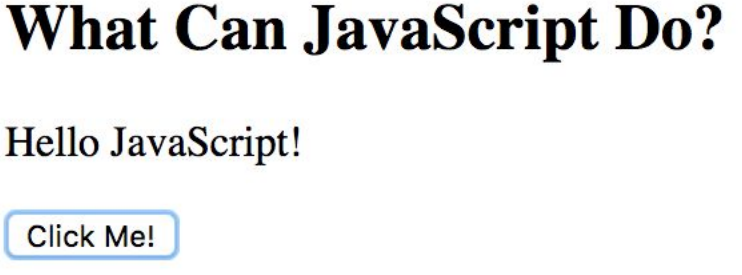
    onclick=*JavaScript'*
- onsubmit , onmouseover and onmouseout etc are different event type.
- One example in next slide!

Example

```
<!DOCTYPE html>
<html>
<body>
      <h2>What Can JavaScript Do?</h2>
      <p id="demo">JavaScript can change HTML content.</p>
   <button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>
</body>
</html>
```

Before

After



User-Defined Classes

- JavaScript is an object-oriented language, but one without classes
- Instead of defining a class, we can define a function that acts as object constructor
  - specify data fields & methods using `this`
  - no data hiding: can't protect data or methods

```
// CSE391        Die.js        //
// Die class definition
////////////////////////////////////////

function Die(sides)
{
   this.numSides = sides;
   this.numRolls = 0;
   this.Roll = Roll;   // define a pointer to a
function
}

function Roll()
{
    this.numRolls++;
    return Math.floor(Math.random()* this.numSides) + 1;
}
```

define `Die` function (i.e., constructor)

initialize data fields in the function, preceded with this

similarly, assign method to separately defined function (which uses this to access data)

```
<html>
<!-- js15.html   -->

<head>
  <title>Dice page</title>

  <script type="text/javascript"
          src="Die.js">
  </script>
</head>

<body>
 <script type="text/javascript">
    die6 = new Die(6);
    die8 = new Die(8);

    roll6 = -1;     // dummy value to start loop
    roll8 = -2;     // dummy value to start loop
    while (roll6 != roll8) {
      roll6 = die6.Roll();
      roll8 = die8.Roll();

      document.write("6-sided: " + roll6 +
                     "    " +
                     "8-sided: " + roll8 + "<br
/>");
    }

    document.write("<br />Number of rolls: " +
                   die6.numRolls);
  </script>
</body>
</html>
```

Class Example

create a Die object using new (similar to String and Array)

here, the argument to Die initializes numSides for that particular object

each Die object has its own properties (numSides & numRolls)

Roll(), when called on a particular Die, accesses its numSides property and updates its NumRolls

ECMAScript 2015

- ES6, also known as ECMAScript2015, introduced classes.
- A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.
- Class Definition
  - Use the keyword class to create a class, and always add the constructor() method.
  - The constructor method is called each time the class object is initialized.

```
class Car {
      constructor(brand) {
        this.carname = brand;
        }
  }
mycar = new Car("Ford")
```

- *We will not go into further details…*
- *Question to ponder:*
  - *What are the other OOP properties you can have?*

    https://www.w3schools.com/js/js_classes.asp

Again, what is ECMAScript?

Event-driven programs

- with C++ or Java, programs are usually serially executed
  - start with main function, execute sequentially from first statement
  - may loop or skip sections of code, but the program generally proceeds step-by-step

    the programmer specifies the sequence in which execution occurs (with some variability due to input values)

    there is a beginning and an end to program execution

- computation within a Web page is rarely serial instead, the page *reacts* to events such as mouse clicks, buttons, …
  - much of JavaScript's utility is in specifying actions that are to occur in the page as a result of some event

    the programmer may have little or no control over when code will (if ever) be executed, e.g., code that reacts to a button click

    there is no set sequence, the page waits for events and reacts

OnLoad & OnUnload

```
<html>
 <!-- form01.html  12.10.2006 -->

 <head>
  <title>Hello/Goodbye page</title>

  <script type="text/javascript">
     function Hello()
     {
       globalName=prompt("Welcome to my page. " +
                         "What is your
name?","");
     }

     function Goodbye()
     {
       alert("So long, " + globalName +
             " come back real soon.");
     }
  </script>
 </head>

 <body onload="Hello();" onunload="Goodbye();">
    Whatever text appears in the page.
 </body>
</html>
```

the simplest events are when the page is loaded or unloaded

- the onload attribute of the <body> tag specifies JavaScript code that is automatically executed when the page is loaded
- the onunload attribute similarly specifies JavaScript code that is automatically executed when the browser leaves the page

HTML forms

- most event-handling in JavaScript is associated with form elements
- an HTML form is a collection of elements for handling input, output, and events in a page

```
<form name="FormName">
…
</form>
```

- form elements might include:

  for input: button, selection list, radio button, check box, password, …

  for input/output: text box, text area, …

Button Element

- the simplest form element is a button
  - analogous to a real-world button, a click can be used to trigger events

```
<input type="button" value="LABEL" onclick="JAVASCRIPT_CODE"/>
```

```
<html>
 <!-- form02.html   12.10.2006 -->
 <head>
  <title> Fun with Buttons</title>

  <script type="text/javascript"
     src="JS/random.js">
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Lucky Number"
           onclick="var  num = RandomInt(1, 100);
                    alert('The lucky number for the day is ' + num);  " />
  </form>
 </body>
</html>
```

Buttons & Functions

```
<html>
 <!-- form03.html   13.10.2006 -->
 <head>
  <title>Fun with Buttons</title>

  <script type="text/javascript">
      function Greeting()
      // Results: displays a time-sensitive
greeting
      {
        var now = new Date();
        if (now.getHours() < 12) {
          alert("Good morning");
        }
        else if (now.getHours() < 18) {
          alert("Good afternoon");
        }
        else {
          alert("Good evening");
        }
      }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Greeting"
           onclick="Greeting();" />
  </form>
 </body>
</html>
```

for complex tasks,

should define function(s) and have the onclick event trigger a function call

Buttons & Windows

- alert boxes are fine for displaying short, infrequent messages
  - not well-suited for displaying longer, formatted text
  - not integrated into the page, requires the user to explicitly close the box

QUESTION: could we instead use document.write ?

NO -- would overwrite the current page, including form elements

- but could open a new browser window and write there

```
var OutputWindow = window.open();           // open a window and assign
                           //   a name to that object
                           //   (first arg is an HREF)
OutputWindow.document.open();            // open that window for
                           //   writing
OutputWindow.document.write(" WHATEVER");   // write text to that
                           //   window as before
OutputWindow.document.close();              // close the window
```

Window Example

```
<html>
 <!--form04.html   13.10.2006 -->

 <head>
  <title> Fun with Buttons </title>
  <script type="text/javascript">
    function Help()
    // Results: displays a help message in a separate window
    {
      var OutputWindow = window.open();
      OutputWindow.document.open();

      OutputWindow.document.write(" This might be a context- " +
                "sensitive help message, depending on the  " +
                "application and state of the page. ");

      OutputWindow.document.close();
    }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Help"
           onclick="Help();" />
  </form>
 </body>
</html>
```

Text Boxes

- a text box allows for user input
  - unlike prompt, user input persists on the page & can be edited

```
<input type="text" id="BOX_NAME" name="BOX_NAME"… />
```

optional attributes: size : width of the box (number of characters)
value : initial contents of the box

JavaScript code can access the contents as `document.BoxForm.userName.value`

```
<html>
 <!--  form06.html   13.10.2006 -->
 <head>  <title> Fun with Text Boxes </title>  </head>
 <body>
  <form name="BoxForm">
    Enter your name here:
    <input type="text"  name="userName" id="userName" size="12" value="" />
    <br /><br />
    <input type="button" value="Click Me"
    onclick="alert('Thanks, ' +  document.BoxForm.userName.value +
                                 ', I needed that.');" />
  </form>
 </body>
</html>
```

Read/Write Text Boxes

- similarly, can change the contents with an assignment

  Note: the contents are raw text, no HTML formatting

  Also: contents are accessed as a string, must `parseFloat` *or* `parseInt` *if want a number*

```
<html>
 <!--form07.html   13.10.2006 -->

 <head>
  <title> Fun with Text Boxes </title>
 </head>

 <body>
   <form name="BoxForm">
     Enter a number here:
     <input type="text"  size="12" name="number" value="2" />
     <br /><br />
     <input type="button" value="Double"
            onclick="document.BoxForm.number.value=
                     parseFloat(document. BoxForm.number.value) * 2;"
/>
   </form>
 </body>
</html>
```

Text Box Events

```
<html>
 <!-- CSE391  form08.html   13.10.2006 -->

 <head>
  <title> Fun with Text Boxes </title>
  <script type="text/javascript">
    function FahrToCelsius(tempInFahr)
    // Assumes: tempInFahr is a number (degrees Fahrenheit)
    // Returns: corresponding temperature in degrees Celsius
    {
      return (5/9)*(tempInFahr - 32);
    }
  </script>
 </head>

 <body>
  <form name="BoxForm">
    Temperature in Fahrenheit:
    <input type="text" name="Fahr" size="10" value="0"
     onchange="document.BoxForm.Celsius.value =
       FahrToCelsius(parseFloat(document.BoxForm.Fahr.value));"
/>
      <tt>----></tt>  
    <input type="text" name="Celsius" size="10" value=""
           onfocus="blur();" />
    in Celsius
  </form>
 </body>
</html>
```

onchange
triggered when the contents of the box are changed

onfocus
triggered when the mouse clicks in the box

`blur()`
removes focus

Text Areas

- a TEXT box is limited to one line of input/output

- a TEXTAREA is similar to a text box in functionality, but can specify any number of rows and columns

```
<textarea name="TextAreaName" rows="NumRows" cols="NumCols">
Initial Text
</textarea>
```

  - *Note:* unlike a text box, a TEXTAREA has a separate closing tag
    initial contents of the TEXTAREA appear between the tags

  - as with a text box, no HTML formatting of TEXTAREA contents

Better (and easier?) methods to access data

- So far, we have been accessing data input fields by giving them names, and using the “dotted” names from the Document Object Model tree structure.

- What if someone modifies the HTML document?
- Then, all those multiply referenced items can no longer be accessed.

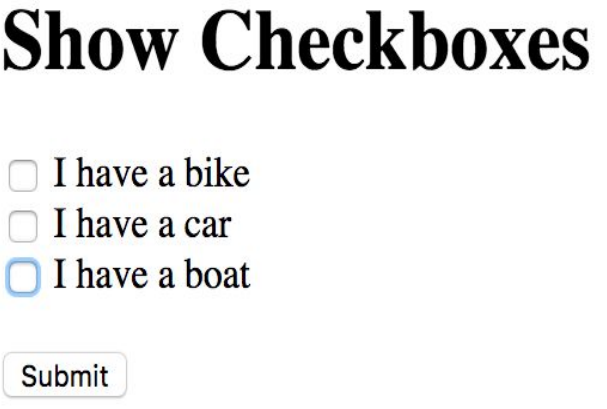
- A more reliable manner (more resistant to changes in the webpage code) would be to give each element an ID (using the “id” attibute) and use the JavaScript getElementById method.

Using getElementById

```
<html>
 <!--form09.html   16.10.2008 -->

 <head>
  <title> Fun with Text Boxes </title>
  <script type="text/javascript"
      src="JS/verify.js">
  </script>

  <script type="text/javascript">
    function FahrToCelsius(tempInFahr)
    {
      return (5/9)*(tempInFahr - 32);
    }
  </script>
 </head>

 <body>
  <form name="BoxForm">
    Temperature in Fahrenheit:
    <input type="text" id="Fahr" size="10" value="0"
       onchange="if (VerifyNum(this)) {  // this refers to current element
                 var F=document.getElementById('Fahr');
                 document.BoxForm.Celsius.value =
                     FahrToCelsius(parseFloat(F.value));
                 }" />
      <tt>----></tt>  
    <input type="text" name="Celsius" size="10" value=""
                     onfocus="getElementById('F').focus();" />
    in Celsius
  </form> </body>
</html>
```

Check Boxes and Radio buttons

```
<!DOCTYPE html>
<html>
<body>

<h1>Show Checkboxes</h1>

<form action="/action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```



- Radio buttons are similar to check boxes, but only one of them can be selected at any time.
- They are defined by <input type="radio"> tags (similar to the checkbox tags in the previous example, with similar properties) and accessed in the same manner.

JavaScript & Timeouts

- the setTimeout function can be used to execute code at a later time

  setTimeout(*JavaScriptCodeToBeExecuted*, *MillisecondsUntilExecution*)

- example: forward link to a moved page

```
<html>
 <!-- form13.html   13.10.2006 -->

 <head>
  <title> Fun with Timeouts </title>
  <script type="text/javascript">
    function Move()
    // Results: sets the current page contents to be newhome.html
    {
        self.location.href = "newhome.html";
    }
  </script>
 </head>

 <body onload="setTimeout('Move()', 3000);">
   This page has moved to <a
   href="newhome.html">newhome.html</a>.
 </body>
</html>
```

Exercise

- **Exercise**
- Design a form which calculate sum of two integers given by the user
- How to access Cookie with JS?
- **READINGS/Practice**
  - M Schafer: Ch. 19, 20, 22
  - https://www.w3schools.com/js/default.asp
  - http://www.csc.liv.ac.uk/~martin/teaching/comp519/NOTES/JavaScript.pdf

Acknowledgement

- This module is designed and created with the help from following sources-
  - https://cgi.csc.liv.ac.uk/~ullrich/COMP519/
  - http://www.csc.liv.ac.uk/~martin/teaching/comp519/
  - Anup Majumder, Jahangirnagar University