

Lab 7 - Java Script (April 19, 2018)

Content:

- Generating HTML Dynamically
- Monitoring User Events
- JavaScript Syntax
- Using JavaScript
- Exercises

Duration: 3 hours

Part 1: Generating HTML Dynamically

- JavaScript code contained inside a SCRIPT element is executed as the page is loaded, with any output the code generates being inserted into the document at the place the SCRIPT occurred.
- Template for generating HTML with JavaScript:

```
<BODY>
Regular HTML

<SCRIPT TYPE="text/javascript">
<!--
Build HTML Here
// -->
</SCRIPT>

More Regular HTML
</BODY>
```

Part 2: Monitoring User Events

Use Various onXxxx Attributes:

- onClick
- onLoad
- onMouseOver

- onFocus
- ...

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Simple JavaScript Button</TITLE>

  <SCRIPT TYPE="text/javascript">
  <!--
  function dontClick() {
    alert("I told you not to click!");
  }
  // -->
</SCRIPT>

</HEAD>
<BODY BGCOLOR="WHITE">
<H1>Simple JavaScript Button</H1>

<FORM>
  <INPUT TYPE="BUTTON"
        VALUE="Don't Click Me"
        onClick="dontClick()">
</FORM>

</BODY>
</HTML>
```

Part 3: JavaScript syntax

- Function declaration:

```
Function funcName(...) {
  //...
  [return ...]
}
```

Functions can be passed and assigned to variables

- Fields can be added On-the-Fly, ex:

```
var test = new Object();
test.field1 = "value 1";
test.field2 = 7;
```

- Can create objects using a shorthand “literal” notation of the form

```
var obj = { field1:val1, field2:val2, ... , fieldN:valN }
```

- The “for/in” statement iterates over properties:

```
for(fieldName in object) {  
    doSomethingWith(fieldName);  
}
```

- A “Constructor” is just a function that assigns to “this”.

```
function Ship(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
}
```

- Arrays:

```
var squares = new Array(5);  
for(var i=0; i<squares.length; i++) {  
    vals[i] = i * i;  
}  
// Or, in one fell swoop:  
var squares = new Array(0, 1, 4, 9, 16);  
var array1 = new Array("fee", "fie", "fo", "fum");  
// Literal Array notation for creating an array.  
var array2 = [ "fee", "fie", "fo", "fum" ];
```

Part 4: Using JavaScript:

- Adjusting to the browser window size: `window.innerWidth` and `window.innerHeight` (supported by Netscape 4.0)
- Using JavaScript to make pages dynamic, ex:

```
<IMG SRC="cool-image.jpg" NAME="cool"  
    WIDTH=75 HEIGHT=25>
```

```
function improveImage() {
    document.images["cool"].src = "way-cool.jpg";
}
```

- Making pages dynamic: moving layers
 - o JavaScript 1.2 lets you access layers via the *document.layers* array, each element of which is a Layer object with properties corresponding to the attributes of the LAYER element
 - o A named layer can be accessed via *document.layers["layer name"]* rather than by using an index, or simply by using *document.layerName*
 - o LAYER only supported Netscape 4
 - o Example: Camps.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Camps on K-3</TITLE>

<SCRIPT TYPE="text/javascript">
<!--

function hideCamps() {
    // Netscape 4 document model.
    document.layers["baseCamp"].visibility = "hidden";
    document.layers["highCamp"].visibility = "hidden";
    // Or document.baseCamp.visibility = "hidden";
}

function moveBaseCamp() {
    baseCamp.moveBy(1, 3);
    if (baseCamp.pageX < 130) {
        setTimeout("moveBaseCamp()", 10);
    }
}

// Hide camps, position base camp near top-left corner,
// make it visible, then have it slowly drift down to
// final position.

function showBaseCamp() {
    hideCamps();
    baseCamp = document.layers["baseCamp"];
    baseCamp.moveToAbsolute(0, 20);
    baseCamp.visibility = "show";
    moveBaseCamp();
}

function moveHighCamp() {
    highCamp.moveBy(2, 1);
    if (highCamp.pageX < 110) {
        setTimeout("moveHighCamp()", 10);
    }
}
```

```

}

// Hide camps, position high camp near top-left corner,
// make it visible, then have it slowly drift down to
// final position.

function showHighCamp() {
    hideCamps();
    highCamp = document.layers["highCamp"];
    highCamp.moveToAbsolute(0, 65);
    highCamp.visibility = "show";
    moveHighCamp();
}

// -->
</SCRIPT>
</HEAD>
<BODY>

<IMG SRC="images/peak4.gif" WIDTH=511 HEIGHT=600 ALIGN="LEFT">
<H1>Camps on K-3</H1>
The High Peaks Tours trip to the summit:
<UL>
    <LI>Day 1: Travel to Base Camp
    <LI>Day 2: Climb to High Camp
    <LI>Day 3: Ascend summit, return to High Camp
    <LI>Day 4: Descend to Base Camp
    <LI>Day 5: Return Home
</UL>
<BR CLEAR="ALL">

<!--          LAYER only supported Netscape 4          -->
<LAYER id="highCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
    <TABLE>
        <TR><TH BGCOLOR="WHITE" WIDTH=50>
            <FONT SIZE="+2">High Camp</FONT>
            <TD><IMG SRC="images/Arrow-Right.gif">
        </TR>
    </TABLE>
</LAYER>

<!--          LAYER only supported Netscape 4          -->
<LAYER id="baseCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
    <TABLE>
        <TR><TH BGCOLOR="WHITE" WIDTH=50>
            <FONT SIZE="+3">Base Camp</FONT>
            <TD><IMG SRC="images/Arrow-Right.gif">
        </TR>
    </TABLE>
</LAYER>

<FORM>
    <INPUT TYPE="Button" VALUE="Show Base Camp"
        onClick="showBaseCamp()">
    <INPUT TYPE="Button" VALUE="Show High Camp"
        onClick="showHighCamp()">
    <INPUT TYPE="Button" VALUE="Hide Camps"
        onClick="hideCamps()">

```

```
</FORM>

</BODY>
</HTML>
```

- Using JavaScript to validate CGI forms
 - o Accessing forms:

```
var firstForm = document.forms[0];
// Assumes <FORM NAME="orders" ...>
var orderForm = document.forms["orders"];
// Assumes <FORM NAME="register" ...>
var registrationForm = document.register;
```

- o Accessing elements within forms

```
var firstElement = firstForm.elements[0];
// Assumes <INPUT ... id="quantity">
var quantityField = orderForm.elements["quantity"];
// Assumes <INPUT ... id="submitSchedule">
var submitButton = registrationForm.submitSchedule;
```

- Using JavaScript to store and examine Cookies
 - o Using document.cookies

```
document.cookie = "name1=val1";
document.cookie = "name2=val2; expires=" + someDate;
document.cookie = "name3=val3; path=/; domain=test.com";
```

- o Parsing Cookies

```
function cookieVal(cookieName, cookieString) {
    var startLoc = cookieString.indexOf(cookieName);
    if (startLoc == -1) {
        return(""); // No such cookie
    }
    var sepLoc = cookieString.indexOf("=", startLoc);
    var endLoc = cookieString.indexOf(";", startLoc);
    if (endLoc == -1) { // Last one has no ";"
        endLoc = cookieString.length;
    }
    return(cookieString.substring(sepLoc+1, endLoc));
}
```

- Using JavaScript to interact with Frames
 - o Displaying a URL in a particular frame, ex:

```
parent.displayFrame.location = url;
```

- o Giving a frame the input focus, ex:

```
parent.displayFrame.focus();
```

- Accessing Java from JavaScript:

- o Calling java methods directly

```
java.lang.System.out.println("Hello Console");
```

- o Controlling Applets from JavaScript.

Example: MoldSimulation.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Mold Propagation Simulation</TITLE>

  <SCRIPT TYPE="text/javascript">
    <!--

    // Start simulation for all applets in document.

    function startCircles() {
      for(var i=0; i<document.applets.length; i++) {
        document.applets[i].startCircles();
      }
    }

    // Stop simulation for all applets in document.

    function stopCircles() {
      for(var i=0; i<document.applets.length; i++) {
        document.applets[i].stopCircles();
      }
    }

    // -->
  </SCRIPT>
</HEAD>
<BODY BGCOLOR="#C0C0C0">
  <H1>Mold Propagation Simulation</H1>

  <APPLET CODE="RandomCircles.class" WIDTH=100 HEIGHT=75>
</APPLET>
  <P>
  <APPLET CODE="RandomCircles.class" WIDTH=300 HEIGHT=75>
</APPLET>
  <P>
  <APPLET CODE="RandomCircles.class" WIDTH=500 HEIGHT=75>
```

```

</APPLET>
<FORM>
<INPUT TYPE="BUTTON" VALUE="Start Simulations"
        onClick="startCircles()">
<INPUT TYPE="BUTTON" VALUE="Stop Simulations"
        onClick="stopCircles()">
</FORM>

</BODY>
</HTML>

```

- Accessing JavaScript from Java, steps:
 - o Obtain and install the JSObject class.
 - o Import the class in your applet.
 - o From the applet, obtain a JavaScript reference to the current window.
 - o Read the JavaScript properties of interest.
 - o Set the JavaScript properties of interest.
 - o Call the JavaScript methods of interest.
 - o Give the applet permission to access its Web page.

(Note: you can run examples from the eBook Core Web Programming, 2nd Edition to check results)

Part 5: Exercises

1. Write a web page as below:

Register to become a member of the website

Full name:

Email:

Username:

Choose a password:

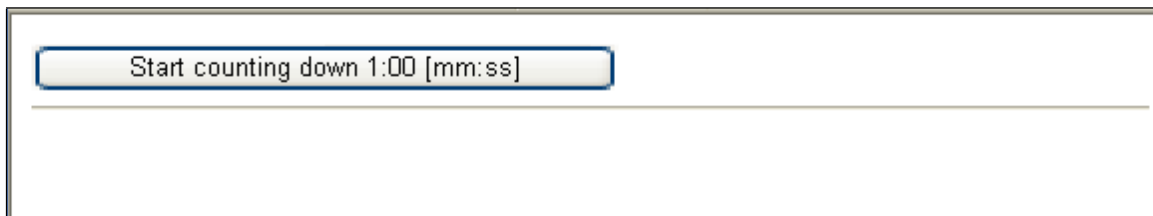
Re-enter password:

- Focus on full name on loading
- Check input values:

- o “Full name” is not empty.
- o “Email” is valid, ex: email@gmail.com
- o “Username” is not empty.
- o Passwords match.
- Store the values into cookies when submitting and can show them again on loading

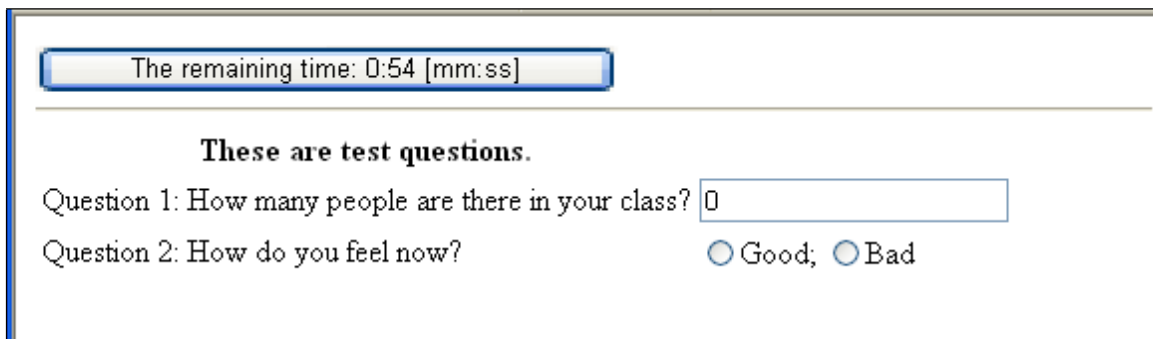
2. Using method `window.setInterval` to implement the following web page:

- A button used to count down from 60 seconds is displayed on loading page



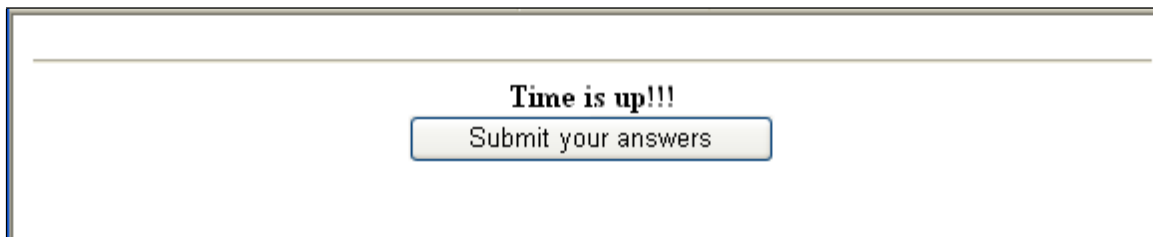
A screenshot of a web page with a single button at the top. The button has a light blue background and a dark blue border, containing the text "Start counting down 1:00 [mm:ss]". Below the button is a horizontal line.

- The clock starts counting down after clicking the button; and content of questions are showed like below



A screenshot of a web page showing the state after the button is clicked. At the top, a button displays "The remaining time: 0:54 [mm:ss]". Below it is a horizontal line. The main content area contains the text "These are test questions." followed by two questions. "Question 1: How many people are there in your class?" is followed by a text input field containing the number "0". "Question 2: How do you feel now?" is followed by two radio buttons labeled "Good;" and "Bad".

- When time is up, the content disappears and the submit button is displayed



A screenshot of a web page showing the state when time is up. The page is mostly empty, with a horizontal line near the top. In the center, the text "Time is up!!!" is displayed in a bold font. Below it is a button with a light blue background and a dark blue border, containing the text "Submit your answers".

Example:

The following code displays the current time in a Text object. In the startclock function, the setInterval method causes the showtime function to be called every second to update the clock. Notice that the startclock function and setInterval method are each called only one time.

```
<SCRIPT LANGUAGE="JavaScript">
var timerID = null
var timerRunning = false

function stopclock(){
    if(timerRunning)
        clearInterval(timerID)
        timerRunning = false
}
function startclock(){
    // Make sure the clock is stopped
    stopclock()
    timerID = setInterval("showtime()",1000)
    timerRunning = true
}

function showtime(){
    var now = new Date()
    var hours = now.getHours()
    var minutes = now.getMinutes()
    var seconds = now.getSeconds()
    var timeValue = "" + ((hours > 12) ? hours - 12 : hours)
    timeValue += ((minutes < 10) ? ":0" : ":") + minutes
    timeValue += ((seconds < 10) ? ":0" : ":") + seconds
    timeValue += (hours >= 12) ? " P.M." : " A.M."
    document.clock.face.value = timeValue
}
</SCRIPT>

<BODY onLoad="startclock()">
<FORM NAME="clock" onSubmit="0">
    <INPUT TYPE="text" NAME="face" SIZE=12 VALUE ="">
</FORM>
</BODY>
```

3. Refer and implement all exercises in chapter 9 (Vietnamese-Book).