INTRODUCTION TO WEB DEVELOPMENT AND HTML

Lecture 15: JavaScript loops, Objects, Events - Spring 2011

Outline

- Selection Statements (if, if-else, switch)
- Loops (for, while, do..while)
- Built-in Objects:
  - Strings
  - Date
  - Math
  - Arrays

Selection Statements

if

- if statements allow code to be executed when the condition specified is true.
- if the condition is **true** then the code in the curly braces is executed.

```
if (condition)
{
    code to be executed if condition is true
}
```

What this code does?

```
<script type=”text/JavaScript”>
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
</script>
```

if-else

- If the conditions specified are met, run the first block of code; otherwise run the second block.

```
if (condition) {
   code to be executed if condition is true
}
else {
   code to be executed if condition is false
}
```

What this code does?

```
<script type="text/JavaScript">
  date = new Date();
  time = date.getHours();
  if (time < 12) {
    document.write('Good Morning');
  }
  else {
    document.write('Good Afternoon');
  }
</script>
```

switch

- A switch statement allows you to deal with several results of a condition.

```
switch (expression) {
  case option1: code to be executed if expression is what is
    written in option1
  break;
  case option2: code to be executed if expression is what is
    written in option2
  break;
  case option3: code to be executed if expression is what is
    written in option3
  break;
  default: code to be executed if expression is different from
    option1, option2, and option3
}
```

switch example

```
<p>Enter the name of your favorite type of animal that stars in a
  cartoon:</p>
<form name="frmAnimal">
<input type="text" name="txtAnimal" /><br />
<input type="button" value="Check animal" onclick="checkAnimal()" />
</form>
```

```
function checkAnimal() {
  switch (document.frmAnimal.txtAnimal.value){
    case "rabbit": alert("Watch out, it's Elmer Fudd!")
    break;
    case "coyote": alert("No match for the road runner - meep
    meep!")
    break;
    case "mouse": alert("Watch out Jerry, here comes Tom!")
    break;
    default : alert("Are you sure you picked an animal from a
    cartoon?");
  }
}
```

Zanastardust/ Flickr

Loops!

while

In a while loop, a code block is executed if a condition is true and for as long as that condition remains true.

The syntax is as follows:

```
while (condition)
{
    code to be executed
}
```

What this code does?

```
<script type="text/JavaScript">
 var i = 1;
  while (i < 11) {
    document.write(i + " x 3 = " + (i * 3) +
     "<br/>" );
    i++;
  }
</script>
```

while Example

do..while

- A do ... while loop executes a block of code once and then checks a condition.
- For as long as the condition is true it continues to loop. So, whatever the condition, the loop **runs at least once**
- Here is the syntax:

```
do
{
   code to be executed
}
while (condition)
```

What this code does?

```
<script type="text/JavaScript">
  var i = 12;
  do {
    document.write(i + " x 3 = " + (i * 3) + "<br
      />" );
    i++;
  }
  while (i < 11)
</script>
```

for

- The for statement executes a block of code a specified number of times.
- Use it when you know **how many times** you want the code to be executed rather than running while a particular condition is true/false.
- Syntax:

```
for (a; b; c)
{
    code to be executed
}
```

a is evaluated before the loop is run, and is only evaluated once.

b should be a condition that indicates whether the loop should be run again. if it returns true the loop runs again.

c is evaluated after the loop has run.

What this code does?

```
for (i=0; i<20; i++) {
  document.write(i + " x 3 = " + (i * 3)
  + "<br />" );
}
```

What about this one?

```
for (i=0; i<20; i++) {
  document.write(i + " x 3 = " + (i * 3)
  + "<br />" );
  if( i == 10 )
     break;
}
```

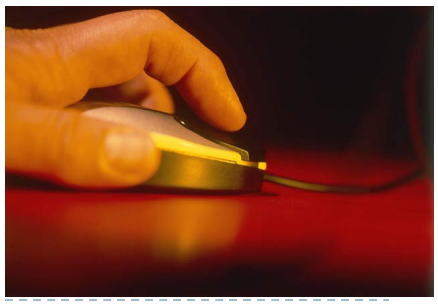
Events

Events

- **Window events**: which occur when something happens to a window.
  - For example, a page loads or unloads.
- **User events**: which occur when the user interacts with elements in the page using a mouse (or other pointing device) or a keyboard.

Events

- **onload**: Document has finished loading
- **onclick**: Button on mouse has been clicked over the element.
- **ondblclick**: Button on mouse has been double-clicked over the element.
- **onmousedown**: Button on mouse has been pressed (but not released) over the element.
- **onmouseup**: Button on mouse (or other pointing device) has been released over the element.



More Events

- **onmouseover**: Button on mouse has been moved onto the element.
- **onmousemove:** Button on mouse has been moved while over the element.
- **onmouseout**: Button on mouse has been moved off the element.
- **onkeypress:** A key is pressed and released over the element.
- **onkeydown:** A key is held down over an element.
- **onkeyup**: A key is released over an element. Most elements

Even more Events

- **onfocus**: Element receives focus either by mouse clicking it, tabbing order giving focus to that element, or code giving focus to the element.
- **onblur:** Element loses focus.

Forms:

- **onsubmit:** A form is submitted.
- **onreset**:A form is reset.
- **onselect**: User selects some text in a text field.
- **onchange**:A control loses input focus and its value has been changed since gaining focus.

Built-in Objects!

String, Date

String

- Allows you to deal with strings of text:

```
myString = new String('Here is some big text')
document.write(myString);
```

Why we just don't use ?

```
var someText = "Here is some big text";
document.write(someText);
```

String methods

- **charAt(index):** Returns the character at a specified position.
  - String text = new String("Good Morning");
  - text.charAt(3) would return the letter **d**.
- **link(targetURL):** Creates a link
  - text.link("http://www.google.com") →
  - Good Morning
- **substr(start, length):** Returns a substring starting at the position ***start*** and then ***length*** consecutive characters.

String methods

- **substring(startPosition, endPosition):** returns a substring from position ***startPosition*** to the position ***endPosition***

- **toLowerCase():** Converts a string to lowercase
  - String text = new String("John Smith")
  - text.toLowerCase() → john smith

- **toUpperCase():** Converts a string to uppercase
  - text.toUpperCase() → JOHN SMITH

Date

- Date object helps you work with dates and times
  - new Date()
- To set a specific date or time, you need to pass only one of these parameters:
  - **milliseconds**: number of milliseconds since 01/01/1970.
  - **yr_num, mo_num, day_num**: Represents year, month, and day.
  - **yr_num, mo_num, day_num, hr_num, min_num, seconds_num, ms_num**: Represents the years, days, hours, minutes, seconds, and milliseconds.

Date Methods

- getDate() : returns the date (from 1 to 31)
- getDay(): returns the day (from 0 to 6; Sunday, to Saturday)
- getMonth(): returns the month (from 0 to 11; January to December)
- getYear(): returns the year using two digits
- getFullYear(): returns the year using four-digit year

Date Methods (Cont'd)

- **getHours()**: returns the hour
- **getMinutes():** returns the minute
- **getSeconds():** returns the seconds
- **getTimezoneOffset()**: returns the time difference between the user's computer and GMT

Date Methods (Cont'd)

- **setDate**(): sets the date of the month (from 1 to 31).
- **setFullYear**(): sets the year (four digits).
- **setHours**(): sets the hour (from 0 to 23).
- **setMinutes**(): Sets the minute (from 0 to 59).
- **setMonth**(): Sets the month (from 0 to 11; 0=January, 1=February).
- **setSeconds**(): Sets the second (from 0 to 59).

Dates Examples

- With dateString, and will read Wed Apr 16 00:00:00 UTC+0100 1975:
  - var birthDate = new Date("April 16, 1975")
  - document.write(birthDate)

- With yr_num, mo_num, and day_num, and will read Mon May 12 00:00:00 UTC+0100 1975:
  - var birthDate = new Date(1975, 4, 28)
  - document.write(birthDate)

Math

- The math object helps in working with numbers.
- Example:

```
numberPI = Math.PI
document.write (numberPI)
```

- Another Example:

```
numberPI = Math.PI
numberPI = Math.round(numberPI)
document.write (numberPI)
```

Math methods

- abs(x): Returns the absolute value of *x*.
- acos(x): Returns the arccosine of *x*.
- asin(x): Returns the arcsine of *x*.
- atan(x): Returns the arctangent of *x*.
- atan2(y,x): Returns the angle from the x-axis to a point.
- ceil(x): Returns the nearest integer greater than or equal to *x*.
- cos(x): Returns the cosine of *x*.
- sqrt(x): Returns the square root of *x*.
- tan(x): Returns the tangent of *x*.

Math methods

- exp(x): Returns the value of E raised to the power of *x*.
- floor(x): Returns the nearest integer less than or equal to *x*.
- log(x): Returns the natural log of *x*.
- max(x,y): Returns the number with the highest value of *x and y*.
- min(x,y): Returns the number with the lowest value of *x and y*.
- pow(x,y): Returns the value of the number *x raised to the power of y*.
- random(): Returns a random number between 0 and 1.
- round(x): Rounds *x to the nearest integer*.
- sin(x): Returns the sine of *x*.

Arrays

- An *array can hold more than one value*.
- *These values* can be accessed individually.
- We need to use a Constructor to build up the array:

```
instruments = new Array("guitar", "drums","piano")
```

- The elements of the array are indexed using their ordinal number, starting at 0.
  - instruments[0] → returns the guitar.
  - instruments[1] → returns the drums.
  - instruments[2] → returns the piano.

Arrays (Cont'd)

- If you do not want to provide all the values when you create the array, you can just indicate how many elements you want to be able to hold:

  var instruments = new Array(3)

- To change the size of an array, just modify the length property, such as:

  instruments.length = 5

  document.write(instruments.length) → will print 5

Array Methods

- **concat()**: Joins two or more arrays to create one new one.
- **join(separator)**: Joins all of the elements of an array together separated by the character specified as a separator (the default is a comma);
- **reverse():** Returns the array reversed
- **slice():** Returns a specified part of the array
- **sort():** Returns a sorted array

Examples

- Find the examples in the course website…

Get Dojo !

Builds

Cloud hosting via CDN

You can utilize the full Dojo Toolkit from the services by including a script tag in your page:

Google CDN Yandex CDN (Europe)

```
<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js" type="text/javascript"></script>
```

<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js" type="text/javascript"></script>

Cool Examples with Dojo!!!

- See Examples in course site

Questions?