

CSC 551: Web Programming

Fall 2001

JavaScript objects and events

- predefined classes & objects
 - String, Array, Date, document, navigator
- user-defined classes
 - encapsulation, but no data hiding
- event-driven programming
 - onLoad/onUnload
 - onClick, onMouseOver, onMouseOut

Classes vs. objects

in programming terminology:

a *class* is a definition of a new type (a.k.a. ADT)

- encapsulates both data (a.k.a. properties) and functions (a.k.a. methods)
- usually, can hide data & force the use of functions

an *object* is an instance of a class

e.g., in C++, the String class encapsulates

- a sequence of characters and
- various methods (e.g., length, [], substring, find, ...)

```
String str = "foobar";    // str is a String object
cout << str.length();    // applies length method to str object
```

JavaScript predefined classes

we have already seen 2 predefined JavaScript classes

- String: encapsulates a sequence of characters, indexed
properties include: length
methods include: charAt, substring, toUpperCase, indexOf
- Array: encapsulates a sequence of elements, indexed
properties include: length
methods include: [], reverse, sort, slice, pop, push

to create and allocate an object in JavaScript, use new

```
numbers = new Array(4);           // calls Array constructor
firstName = new String("Dave");   // calls String constructor
```

for both String and Array, can assign constant values directly, hide call to new

```
numbers = [1, 2, 3, 4];           // creates and initializes
firstName = "Dave";               // same as above, but cleaner
```

Date class

the Date class can be used to access the date and time

- constructors include:

```
today = new Date();               // sets to current date & time
newYear = new Date(2001,0,1);     //sets to Jan 1, 2001 12:00AM
```

- methods include:

```
newYear.getYear()                 can access individual components of a date
newYear.getMonth()
newYear.getDay()
newYear.getHours()
newYear.getMinutes()
newYear.getSeconds()
newYear.getMilliseconds()

newYear.toString()                can convert date to printable String
```

Date example

```
<html>
<!-- Dave Reed  js14.html  9/20/01  -->

<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script language="JavaScript">
    now = new Date();

    document.write(now.toString() + "<br><br>");

    document.write(now.getHours() + ":" +
      now.getMinutes() + ":" +
      now.getSeconds());
  </script>
</body>
</html>
```

Here, set to the current date and time using the default constructor

toString displays the full date using month and day names

using the get methods, can display desired components of the date/time

[view page in browser](#)

```
<html>
<!-- Dave Reed  js15.html  9/20/01  -->

<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script language="JavaScript">
    now = new Date();

    time = "AM";
    hours = now.getHours();

    if (hours > 12) {
      hours -= 12;
      time = "PM"
    }
    else if (hours == 0) {
      hours = 12;
    }

    document.write(hours + ":" +
      now.getMinutes() + ":" +
      now.getSeconds() + " " +
      time);
  </script>
</body>
</html>
```

Date example (cont.)

suppose we don't like military time

instead of 0:15:22
we want 12:15:22 AM

we must perform the conversions

- need a variable for "AM" or "PM"
- need to adjust hours past noon
- need to handle 12 AM special

[view page in browser](#)

```

<html>
<!-- Dave Reed  js16.html  9/20/01 -->

<head>
  <title>Time page</title>
</head>

<body>
  Time in the new millenium:
  <script language="JavaScript">
    now = new Date();
    newYear = new Date(2001,0,1);

    secs = Math.round((now-newYear)/1000);

    days = Math.floor(secs / 86400);
    secs -= days*86400;
    hours = Math.floor(secs / 3600);
    secs -= hours*3600;
    minutes = Math.floor(secs / 60);
    secs -= minutes*60

    document.write(days + " days, " +
      hours + " hours, " +
      minutes + " minutes, and " +
      secs + " seconds.");
  </script>
</body>
</html>

```

Another example

you can add and subtract Dates:
the result is a number of
milliseconds

here, determine the number of
seconds since New Year's day

divide into number of days, hours,
minutes and seconds

possible improvements?

[view page in browser](#)

document object

Both IE and Netscape allow you to access information about an
HTML document using the document object (*Note: not a class!*)

```

<html>
<!-- Dave Reed  js17.html  9/20/01 -->

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

<body>
  <table width="100%">
    <tr>
      <td><small><i>
        <script language="JavaScript">
          document.write(document.URL);
        </script>
      </i></small></td>
      <td align="right"><small><I>
        <script language="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 saved

[view page in browser](#)

navigator object

can access information about the browser being used to access the Web page using the `navigator` object (*Again: not a class!*)

`navigator.appName` property gives the browser name, e.g.,

```
"Netscape"
"Microsoft Internet Explorer"
```

`navigator.appVersion` property gives the browser version, e.g.,

```
"4.0 (compatible; MSIE 5.0; Windows 98; DigExt) "
```

`navigator.plugins` property gives an array of all of the installed plug-ins

navigator example

can have a separate style sheet file for each browser

use `navigator.appName` to find out which browser used

dynamically write the LINK tag with corresponding HREF

```
<!-- 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>
<!-- Dave Reed js18.html 9/20/01 -->
<head>
<title>Dynamic Style Page</title>
<script language="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 in browser](#)

Event-driven programs

in C++, programs are serially executed

- start with main function, execute sequentially from first statement
- may loop or skip sections of code, but the program 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>
<!-- Dave Reed  js20.html  9/20/01  -->

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

<script language="JavaScript">
  function Hello()
  {
    userName=prompt("Welcome to my page. " +
                    "What is your name?","");
  }

  function Goodbye()
  {
    userName=alert("So long, " + userName +
                  " 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

[view page in browser](#)

```

<html>
<!-- Dave Reed  js21.html  9/20/01 -->

<head>
  <title>Anchor events</title>
  <script language="JavaScript"
    src="http://www.creighton.edu/~csc551/
JavaScript/random.js">
  </script>

  <script language="JavaScript">
    function GetNumber(maxNum)
    {
      var number = randomInt(1, maxNum);
      alert("The lucky number for today is " +
        number)
    }
  </script>
</head>

<body>
  For today's lucky number,
  <a href="#" onClick="GetNumber(100);" >
  click here</a>

  <p>
  Or better yet,
  <a href="javascript:GetNumber(100);" >
  try here</a>
</body>
</html>

```

User-driven events

can add event-handling to anchors and images

- ONCLICK attribute specifies what is to occur if the user clicks on the element
- ONMOUSEOVER attribute specifies what is to occur when the mouse passes over the element
- ONMOUSEOUT attribute specifies what is to occur when the mouse moves off the element

[view page in browser](#)

Image events

```

<html>
<!-- Dave Reed  html35.html  8/22/01 -->

<head>
  <title>Title for Page</title>
</head>

<body>
  <a href="javascript:alert('Do NOT click on me!');" >
  </a>

  <p><br>

  
</body>
</html>

```

recall sneak-preview example (Week 3)

- can use an image to simulate a link
 - can change the image when the mouse moves over or out
 - can perform an act (e.g., open a new window, change the file in a frame) when clicked
- when used in a menu, clickable images have some advantages over links

[view page in browser](#)

MORE EVENT-DRIVEN PROGRAMMING WHEN WE CONSIDER FORMS

Next week...

MIDTERM EXAM

- designed to be completed in 60 – 90 minutes, will allow full 150 minutes

types of questions:

- TRUE/FALSE
- short answer, discussion
- explain/modify/augment/write Web page
- explain/modify/augment/write JavaScript code

study advice:

- review online lecture notes
- review text & online readings
- reference other sources for examples, different perspectives
- look over quizzes