JavaScript

Client-side dynamic documents

Smart Browsers

- Most *browsers* support a <SCRIPT> tag that is used to include executable content in an HTML document.
- There are a number of *scripting* languages that are supported

Client-Side Script Languages

- Netscape and others
  - JavaScript
- Internet Explorer
  - Jscript (MS name for JavaScript)
  - VBScript
  - PerlScript

JavaScript Capabilities

- Add content to a web page dynamically.
- Alter a web page in response to user actions.
- React to user events.
- Interact with frames.
- Manipulate HTTP cookies

JavaScript is not Java

- JavaScript is a very simple scripting language.
- Syntax is similar to a subset of Java.
- Interpreted language.
- Uses objects, but doesn't really support the creation of new object types*

*It almost does, but it's cumbersome.

General Format

- <!doctype ...>
- <html>
- <Head>
- <Title> Name of web page </title>
- <script type="text/javascript">
- ...script goes here
- </script>
- </head
- <body>
- ...page body here: text, forms, tables
- ...more JavaScript if needed
- ...onload, onclick, etc. commands here
- </body>
- </html>

Characteristics

- Case sensitive
- Object oriented
- Produces an HTML document
- Dynamically typed
- Standard operator precedence
- Overloaded operators
- Reserved words

Characteristics

- Division with / is not integer division
- Modulus (%) is not an integer operator
- 5 / 2 yields 2.5
- 5.1 / 2.1 yields 2.4285714285714284
- 5 % 2 yields 1
- 5.1 % 2.1 yields 0.8999999999999995

Characteristics

- " and ' can be used in pairs
- Scope rules for variables
- Strings are very common data types
- Rich set of methods available
- Arrays have dynamic length
- Array elements have dynamic type
- Arrays are passed by reference
- Array elements are passed by value

Language Elements

- Variables
- Literals
- Operators
- Control Structures
- Objects

JavaScript Variables

- Untyped!
- Can be declared with var keyword:

```
var foo;
```

- Can be created automatically by assigning a value:

```
foo=1;    blah="Hi Dave";
```

Variables (cont.)

- Using **`var`** to declare a variable results in a *local* variable (inside a function).

- If you don't use **`var`** – the variable is a global variable.

Literals

- The typical bunch:
  - Numbers **`17   123.45`**
  - Strings **`"Hello Dave"`**
  - Boolean: **`true   false`**
  - Arrays: **`[1,"Hi Dave",17.234]`**

Arrays can hold anything!

Operators

- Arithmetic, comparison, assignment, bitwise, boolean (pretty much just like C).

```
+ - * / % ++ -- == != > <
&& || ! & | << >>
```

Control Structures

- Again – pretty much just like C:

```
if  if-else  ?: switch

for  while do-while
```

- And a few not in C

```
for (var in object)

with (object)
```

Objects

- Objects have attributes and methods.
- Many pre-defined objects and object types.
- Using objects follows the syntax of C++/Java:

```
objectname.attributename
objectname.methodname()
```

Array Objects

- Arrays are supported as objects.
- Attribute **`length`**
- Methods include:

  `concat join pop push reverse sort`

Array example code

```
var a = [8,7,6,5];

for (i=0;i<a.length;i++)
  a[i] += 2;

b = a.reverse();
```

Many other pre-defined object *types*

- **`String`**: manipulation methods
- **`Math`**: trig, log, random numbers
- **`Date`**: date conversions
- **`RegExp`**: regular expressions
- **`Number`**: limits, conversion to string

Predefined Objects

- JavaScript also includes some objects that are automatically created for you (always available).
  - **`document`**
  - **`navigator`**
  - **`screen`**
  - **`window`**

The **`document`** object

- Many attributes of the current document are available via the **`document`** object:

Title	Referrer
URL	Images
Forms	Links
Colors	

`document` methods

- **`document.write()`** like a print statement – the output goes into the HTML document.

```
document.write("My title is" +
  document.title);
```

string concatenation!

JavaScript Example

```
<HEAD>
<TITLE>JavaScript is Javalicious</TITLE>
</HEAD>
<BODY>
<H3>I am a web page and here is my
  name:</H3>
<SCRIPT>
document.write(document.title);
</SCRIPT>
<HR>
</BODY>
```

JavaScript and HTML Comments

```
<SCRIPT>
<!--
document.write("Hi Dave");
document.bgColor="BLUE";
-->
</SCRIPT>
```

HTML comment

```
(or //-->)
```

JavaScript Functions

- The keyword **`function`** used to define a function (subroutine):

```
function add(x,y) {
   return(x+y);
}
```

JavaScript Events

- JavaScript supports an event handling system.
  - You can tell the browser to execute javascript commands when some event occurs.
  - Sometimes the resulting *value of the command* determines the browser action.

Simple Event Example

```
<BODY BGCOLOR=WHITE onUnload="restore()">
<H5>Hello - I am a very small page!</H5>
<SCRIPT>
savewidth =  window.innerWidth;
saveheight = window.innerHeight;
function restore() {
  window.innerWidth=savewidth;
  window.innerHeight=saveheight;
}
// Change the window size to be small
window.innerWidth=300; window.innerHeight=50;
document.bgColor='cyan';
</SCRIPT>
```

Buttons

- You can associate buttons with JavaScript events (buttons in HTML forms)

```
<FORM>
<INPUT TYPE=BUTTON
VALUE="Don't Press Me"
onClick="alert(‘now you are in trouble!’)">
</FORM>
```

Some Events (a small sample)

`onUnLoad`

`onLoad`

Window events

`onClick`

`onMouseUp`

`onMouseDown`

Button events

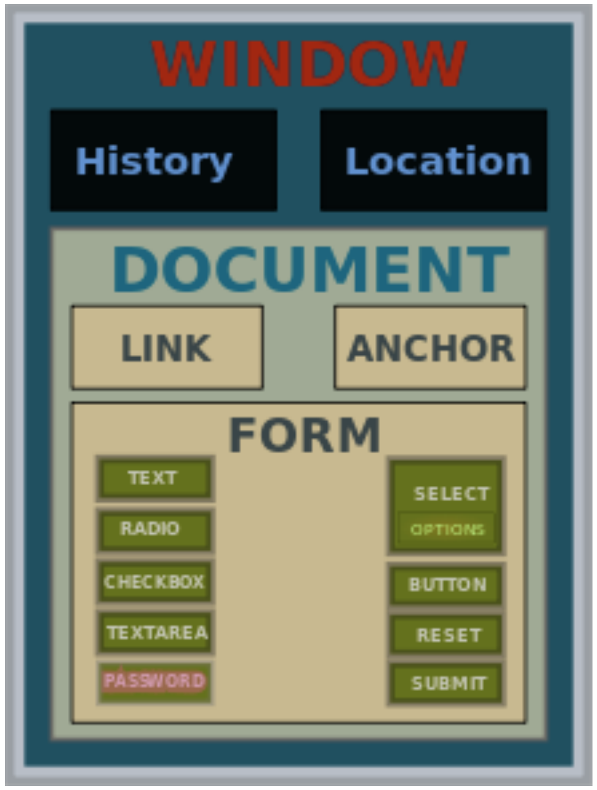
`onDblClick`

`onMouseOver`

Link events

Document Object Model

- Naming hierarchy used to access individual elements of a HTML document.
- Netscape D.O.M. is a little different than IE D.O.M.
- Easy to use if you name all entities:
  - Forms, fields, images, etc.



Things are getting better all the time – there are standard DOMs defined by The W3C

DOM example

```
<FORM ID=myform
  ACTION="action_page.php">
Please Enter Your Age:
<INPUT TYPE=TEXT ID=age NAME=age><BR>
And your weight:
<INPUT TYPE=TEXT ID=weight
  NAME=weight><BR>
</FORM>
```

From javascript you can get at the age input field as: **`document.myform.age.value`**

Form Field Validation

- You can have JavaScript code that makes sure the user enters valid information.
- When the submit button is pressed the script checks the values of all necessary fields:
  - You can prevent the request from happening.

Checking Fields

```
<script}
function checkform() {
  if (document.myform.age.value == "") {
     alert("You need to specify an age");
     return(false);
  } else {
     return(true);
  }
}
</script>
```

Needs to return true or false!

The Form

Needed to prevent the browser from submitting!

```
<FORM METHOD=GET
  ACTION=action_page.php
 NAME=myform
 onSubmit="return(checkform())">


AGE: <INPUT TYPE=TEXT NAME=age>
<INPUT TYPE=SUBMIT>
</FORM>
```

Important Note about Form Validation

- It's a good idea to make sure the user fills out the form before submitting.
- Users can bypass your form – they can create requests manually (or their own forms).
- Your CGI programs cannot rely (soley) on Client-Side JavaScript to validate form fields!

More on Javascript – HTML DOM Methods

- HTML DOM methods are actions you can perform (on HTML Elements).
- HTML DOM properties are values (of HTML Elements) that you can set or change.

```
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

More on Javascript – HTML DOM Methods, Events, …

Reference:

www.w3schools.com/js