

JavaScript

Document Object Model

Client-Side Scripting

- So far, the browser has only passively displayed content.
- It is also possible to download a program and have it execute on the client browser
 - JavaScript / Jscript / ECMAScript
 - VBScript
 - TCL

JavaScript

- A lightweight programming language (scripting)
- Used to make web pages interactive
 - Insert dynamic text into HTML (ex: user name)
 - React to events (ex: page load user click)
 - Get information about a user's computer (ex: browser type)
 - Perform calculations on user's computer (ex: form validation)
- A web standard (but not supported identically by all browsers)
- NOT related to Java other than by name and some syntactic similarities

JavaScript vs. Java

- Interpreted, not compiled
- More relaxed syntax and rules
 - Fewer and "looser" data types
 - Variables don't need to be declared
 - Errors often silent (few exceptions)
- Key construct is the function rather than the class
 - (more procedural less object-oriented)
- Contained within a web page and integrates with its HTML/CSS content

JavaScript Security

- Language/API limitations:
 - No file/directory access defined in the language
 - No raw network access. Limited to either
 - load URLs
 - send HTML form data to
 - web servers, CGI scripts, e-mail addresses
 - 'same origin policy'
 - can only read props of documents and windows from the same place: host, port, protocol
- Privacy restrictions:
 - cannot read history
 - cannot hide/show menubar, status line, scrollbars
 - cannot close a window not opened by itself

Embedding in HTML

- Directly
`<script type="text/javascript">`
.....
`</script>`
- Indirect
`<script type="text/javascript" src="test.js" />`
- Location
 - `<head>` Definitions that are later called by elements in the document body
 - `<body>` Process while parsing

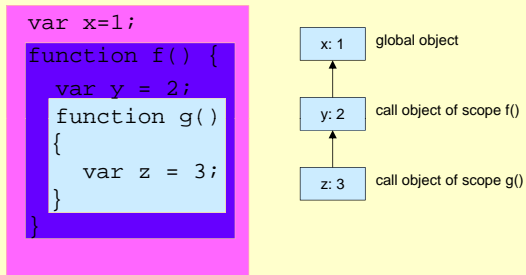
Dynamic Typing

- Different than Java or C
- Variables can hold any type of value:
 - Number (8-byte IEEE fp)
 - Boolean
 - Function (first-class data type)
 - Object
 - Array (elements can be of mixed types)
 - String
- ... and can hold values of different types at different times during execution

Variables

- Declaration
 - Explicit `var i = 12;` // no type declaration
 - Implicit `msg = "hello";`
- Scope
 - Global
 - Declared outside functions
 - Any variable implicitly defined
 - Local
 - Explicit declarations inside functions

Scope Chain



Functions

```
function functionName ([arg1] [...,argN])
{
  .....
  [return [value]];
}
```

- Arguments
 - Primitive types (number, boolean) are passed by value
 - Object types are passed by reference

Arguments Array

- Every function contains an array named arguments representing the parameters passed
- Can loop over them, print/alert them, etc.
- Allows you to write functions that accept varying numbers of parameters

```
function example() {
  for (var i = 0; i < arguments.length; i++) {
    alert(arguments[i]);
  }
}
example("how", "are", "you"); // alerts 3 times
```

Control and Looping

- Control
 - if
 - switch
- Looping
 - for
 - while
 - do..while
 - for .. in
 - for (*property* in *object*) { }

Evaluation and Execution

- Evaluation
 - As document is parsed, in order
 - Execution
 - Statement outside functions
 - When it is encountered
 - Statement inside function
 - When function is called
 - Event handler
- <body onload="helloWorld()">

Objects

- JavaScript is Object Based rather than Object Oriented
 - No *class* construct
 - No inheritance, polymorphism.
- Objects are most like associative arrays

```
var point = new Object();
point.x = 2;
point["x"] = 2;           // equivalent to previous
point.print();
point["print"]();        // equivalent to previous
```
- Note: Object's definition is determined at *run time*.
 - Possible to dynamically add new properties or methods and to change the binding of methods at runtime

Object Constructors

```
function Rectangle_area() {
    return this.width * this.height;
}
function Rectangle(w,h) {
    this.width = w;
    this.height = h;
    this.area = Rectangle_area;
}
var rec = new Rectangle(2,4);

rec.area();
rec["area"]();
rec.newFunction = newPredeclaredFunc;
```

Object Constructors

```
function Rectangle(w,h) {
    this.width = w;
    this.height = h;
    this.area = function () {
        return this.width * this.height;
    };
}
var rec = new Rectangle(2,4);

rec.area();
rec["area"]();
rec.newFunction = newPredeclaredFunc;
```

Predefined Objects

- Native/Built-in
 - Number
 - Boolean
 - String
 - Array
 - Host
 - navigator
 - Window
 - Document
- Primitive types are automatically coerced into Objects

Array Object

```
var a = new Array();           // empty array
var b = new Array("dog", 3, 8.4);
var c = new Array(10);        // array of size 10
var d = [2, 5, 'a', 'b'];

c[15] = "hello";             // implicit extension
```

Array Properties and Methods

- length
- join()
- reverse()
- sort()
- concat()
- slice()
- splice()
- push() / pop()
- shift() / unshift()
- toString()
-

Browser Objects

name	description
document	current HTML page and its content
history	list of pages the user has visited
location	URL of the current HTML page
navigator	info about the web browser you are using
screen	info about the screen area occupied by the browser
window	the browser window

navigator Object

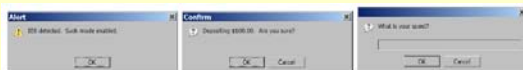
- Represents the Browser
- Properties
 - appName Browser name
 - appVersion Browser version
 - platfoms Client OS
 - cookieEnabled
 - cpuClass
 - Language
 - mimeTypes

window Object

- Browser window itself
- Methods
 - open(url,name,[options]) open a new window
 - close()
 - alert() popup alert
 - prompt() user input
- Properties
 - document
 - width, height
 - toolbar, menubar, scrollbar, status, location
 - rezisable
 - screenX, screenY

Popup Boxes

```
alert("message");    // message
confirm("message"); // returns true or false
prompt("message");  // returns user input string
```

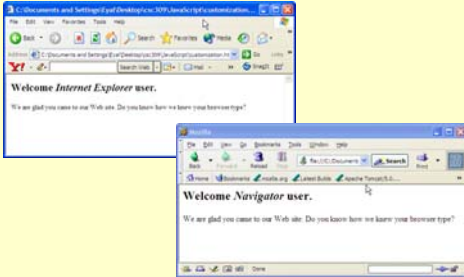


document Object

- Container for all HTML-related objects
 - Content nested inside <html> tag
- Methods
 - open opens output steam to document
 overwrites document
 - write appends text to document
- Properties
 - URL

Example: Customization

- Content customization based on browser

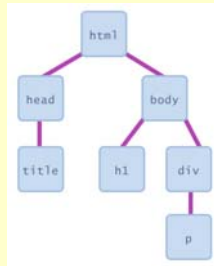


Document Object Model (DOM)

- W3C Standard
 - Implemented differently by every browser
- Interface between document displayed by browser and application programs
 - Object oriented
- Platform-neutral and language-neutral collection of interfaces
 - Can program with a variety of languages (C,Java,VB)
- Documents have treelike structures
 - One interface for every XML node type
 - element,attribute,text,PI,comment
- Create documents, move around document structure (parse), and change, add, or delete elements.

DOM and JavaScript

- A set of JavaScript objects that represent each element on the page .
- Most JS code manipulates elements on an HTML page
- Examine the state of the elements, e.g. whether a box is checked
- Change state, e.g. putting text into a div
- Change styles, e.g. make a paragraph red



Key Interfaces

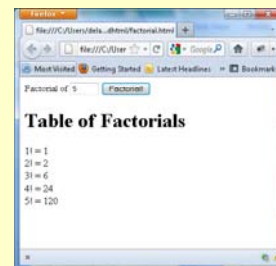
- Document
- Element
- Event

Document

- The central interface is Document
- Create new elements, attributes and text nodes
- Access existing elements
 - getElementByName(stringName)
 - getElementById(stringId)

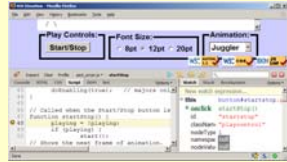
Example: Factorial

- Print factorial table



JavaScript Debugging

- Firebug JS debugger can set breakpoints, step through code, examine values (Script tab)
- Interaction pane for typing in arbitrary JS expressions (Console tab; Watch tab within Script tab)



JSLint

- <http://www.jshint.com/>
- JSLint: an analyzer that checks your JS code, much like a compiler, and points out common errors
- When your JS code doesn't work, paste it into JSLint first to find many common problems



Debugging Checklist

- Are you sure the browser is even loading your JS file at all?
Put an alert at the top of it and make sure it appears.
- When you change your code, do a **full browser refresh (Shift-Ctrl-R)**
- Check bottom-right corner of Firefox for Firebug syntax errors.
- Paste your code into our [JSLint](#) tool to find problems.
- Type some test code into Firebug's console or use a breakpoint.

Element

- All tags in document inherit from Element
- Navigate document
- Change document tree structure

Types of DOM Nodes

```
<p>
This is a paragraph of text with a
<a href="/path/to/another/page.html">link</a>.
</p>
```

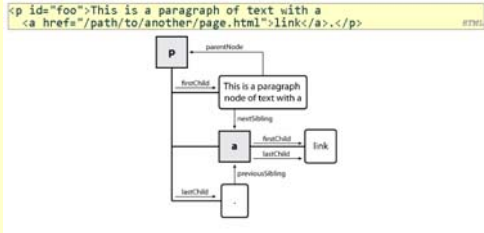
- **element nodes** (HTML tag)
 - can have children and/or attributes
- **text nodes** (text in a block element)
- **attribute nodes** (attribute/value pair)
 - text/attributes are children in an element node
 - they cannot have children or attributes

Traversing the DOM Tree

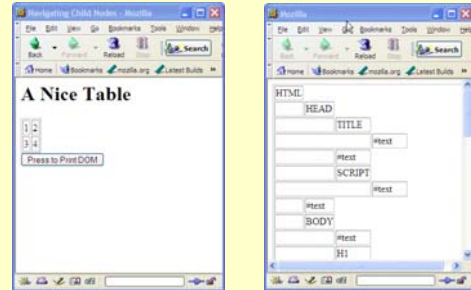
- Every node's DOM object has the following properties:

name(s)	description
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

DOM Tree Traversal



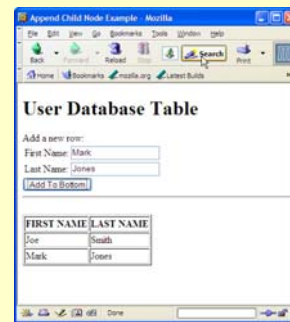
Example: DOM Tree



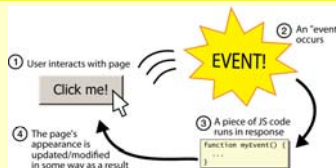
Modifying DOM

- Creating new nodes
 - Document
 - createElement(tag) // Create an element of type tag
 - createTextNode(string) // Creates text node with string
 - Element
 - appendChild(N) // Add the N to the end of child list
 - insertBefore(N,E) // Insert N in child list before E
 - cloneNode(deep) // Copy node. If deep=true copy // all descendants
- Removing nodes
 - Node
 - removeChild(N) // Removes N from child list

Example: Adding Table Rows



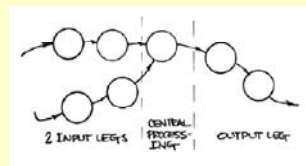
Event-Driven Programming



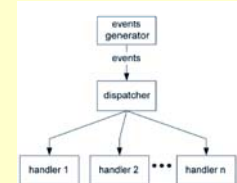
- Most languages' programs start with a main method and run sequentially
- JavaScript programs wait for user actions called events and respond to them
 - Events are associated with the various objects that make up a Web page
 - For example, onClick, OnDbClick, onKeyDown, onLoad, onMouseOver, onSubmit, onResize, ...
- Event-driven programming: writing programs driven by user events

Structured vs. Event Driven

Structured Program



Event Driven Architecture



Partial List of Events

- Clipboard
 - oncopy, oncut, onpaste
- Keyboard
 - onkeydown, onkeyup, onkeypress
- Mouse
 - onmousedown, onmouseup, onmousemove
- Other
 - onfocus, onblur,

Associating Events with Elements

- In the HTML
 - As value of attributes

```
<a href="..." onmouseover="popupFunc();" />
```
- In a script
 - Explicit reference to object's event handler

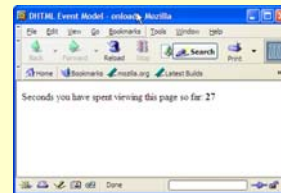
```
document.onmouseover = functionFoo;
```

onload & timers

- onload
 - Fires when element (an all children) finish loading
 - Used in the <body> to execute script after page has been rendered

Example: Onload & Times

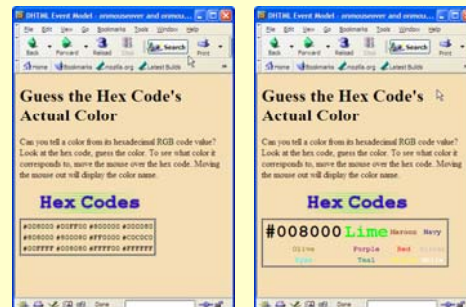
- Example: Count how many seconds have passed since page finish rendering



Event Bubbling

- Event "fired" by child elements "bubble" up to their parent elements.
- Event delivery order
 - First to element that fired event
 - Then to parent
- To cancel bubbling, set event property `event.cancelBubble = true`

Example: Event Bubbling



Strings and Regular Expressions

- `match(regExpObj)`
Verify input

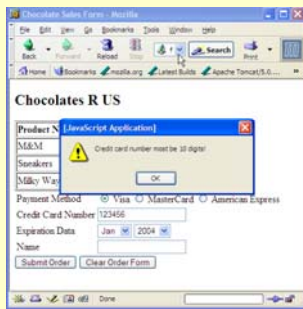
```
var phone = "416-4403467";  
if (phone.match(/d{3}-d{7}/))  
    return true;
```
- `Replace(regExpObj, str)`
Replace part of a string

```
var str = "One elephant and two zebras";  
var matches = str.replace(/two/, "three");
```

Regular Expressions

- Compatible with Perl regular expression syntax
- `\w` Alphanumeric
- `\d` Numerical digit
- `\s` White spaces
- `.` Anything other than newline
- `[abcde]` Any of a,b,c,d,e
- `[a-e]` Same as above
- `[^a-e]` Anything but a to e
- `exp?,exp+,exp*` 0 or 1, 1 or more, 0 or more
- `exp{x,y}` At least x but less than y
- `expA | expB` expA or expB

Example: Form Validation

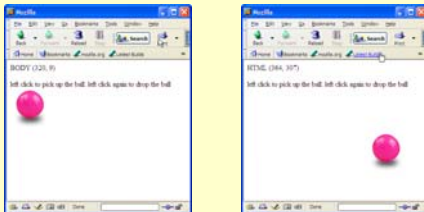


Changing Style Attributes

- CSS is scriptable from JavaScript
 - allows HTML elements to float around and grow and shrink.

Tracking Mouse Movements

- Track mouse position on screen
- Drag and drop ball on click
- Events `onmousemove` and `onclick`



Error Handling

- Specify function to handle script errors
 - `window.error = errorHandler`

