Javascript

This lecture is based on materials from:

Eloquent JavaScript
A Modern Introduction to Programming
by Marijn Haverbeke

http://eloquentjavascript.net/

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

- 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)
- NOT related to Java other than by name and some syntactic similarities

JavaScript vs. Java

- Interpreted, not compiled
- Dynamically typed
- More relaxed syntax and rules
  - Variables don't need to be declared
  - Errors often silent (few exceptions)
- Key construct is the function rather than the class

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

Variables

- Declaration
  - Explicit var i = 12; // no type declaration
  - Implicit msg = "hello";

- Name
  - Cannot start with a digit or include spaces
  - Examples:
    - catch22
    - $
    - $_

Dynamic Typing

- Different than Java or C
- Variables can hold any type of value:
  - number (64 bit floating point)
    - 144, 9.81, 2.99e8
  - string
    - 'You ain\'t seen nothing yet!'
  - Boolean
    - FALSE: "", null, undefined, NaN, false
    - TRUE: everything else (e.g., true, "hi", -1, 3.5)
  - function (first-class data type)
  - object
  - string
  - undefined
- … and can hold values of different types at different times during execution

```
var somevariable = 0;
somevariable = "new value";
somevariable = {2,'hi!',3.1415};
```

Operators

- Arithmetic
  - + - * / %
- Logic
  - && || !
- Comparison
  - < > == != <= >= === !==
- Other
  - typeof

Examples

false == 0; →

“” == 0; →

“5” == 5; →

false === 0; →

“5” === 5; →

“Apollo” + 5 →

null + “ify” →

“5” * 5 →

“straberry” * 5 →

Control and Looping

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

Embedding in HTML

- Directly

```
<script>
  ……..
</script>
```

- Indirect

```
<script src="test.js" />
```

Example

```
<!DOCTYPE html>
<html lang="eng">
  <head>
    <title>Get Number</title>
    <meta charset="utf-8”>
    <script>
      var theNumber = Number(prompt("Pick as number"));
      if (!isNaN(theNumber))
        alert("Your number is the square root of " + theNumber*theNumber);
      else
        alert("This is not a number!");
    </script>
  </head>
</html>
```

Example

```
<!DOCTYPE html>
<html lang="eng">
    <head>
        <title>Loop</title>
        <meta charset="utf-8">
        <script>
            var theNumber = Number(prompt("Factorial of?"));
            var count = 1;
            var factorial = 1;
            while (!isNaN(theNumber) && count <= theNumber) {
                        factorial *= count++;
                        console.log(factorial);
            }
        </script>
    </head>
</html>
```

Functions

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

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

Example

```
<!DOCTYPE html>
<html lang="eng">
    <head>
        <title>Function Example</title>
        <meta charset="utf-8">
        <script>
            function factorial(num) {
              if (isNaN(num) || num ==0)
                    return 1;
              return num * factorial(num-1);
            }
            console.log(factorial(Number(prompt("Factorial of?"))));

        </script>

    </head>
</html>
```

Functions as Values

- Possible to return a function as a value

```
function createFunction() {
    return function() { console.log("Cool, Eh!"); };
}

var a = createFunction();

a();
```

→ "Cool, Eh!"

Scope

- Global
  - Declared outside functions
  - Any variable implicitly defined
- Local
  - Explicit declarations inside functions
  - Function arguments

Scope Chain

```
var x=1;
function f() {
  var y = 2;
  function g()
  {
    var z = 3;
  }
}
```

Closures

- A function can references a local variable created by a function that no longer exists

- Example

```
function createFunction() {
    var msg = "Really Cool!, Eh";
    return function() { console.log(msg); }
}

var a = createFunction();

a();
```

→ "Really Cool, Eh!"

Example

```
function makeAdder(amount) {
   var base = amount;
   return function(number) {
      return number + base;
   }
}

var addTwo = makeAdder(2);

addTwo(5);
```

→

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

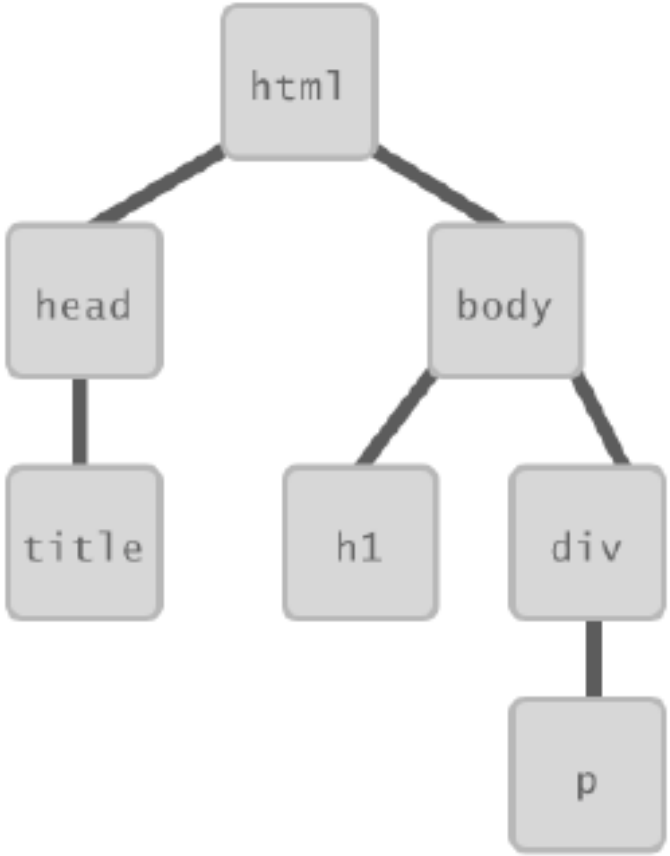
        <body onload="helloWorld()">

Document Object Model (DOM)

- W3C Standard
- Interface between document displayed by browser and application programs
- Platform-neutral and language-neutral collection of interfaces
- Documents have treelike structures
- 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

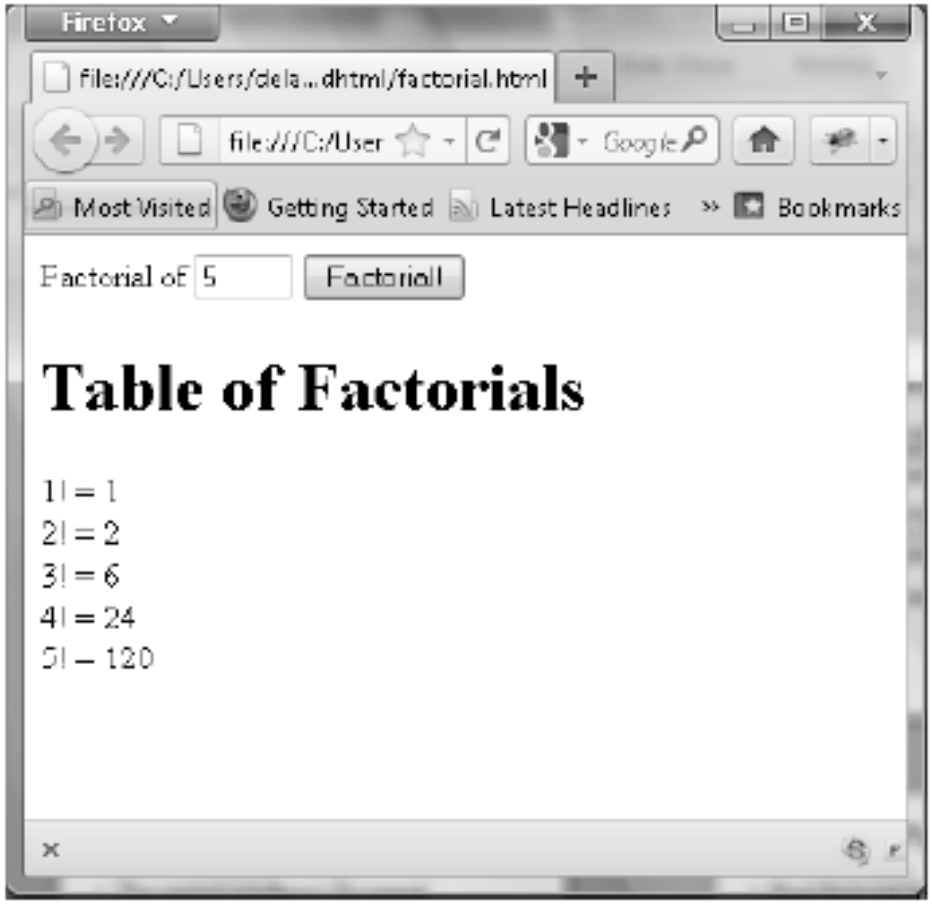
- Document
- Element
- Event

Document

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

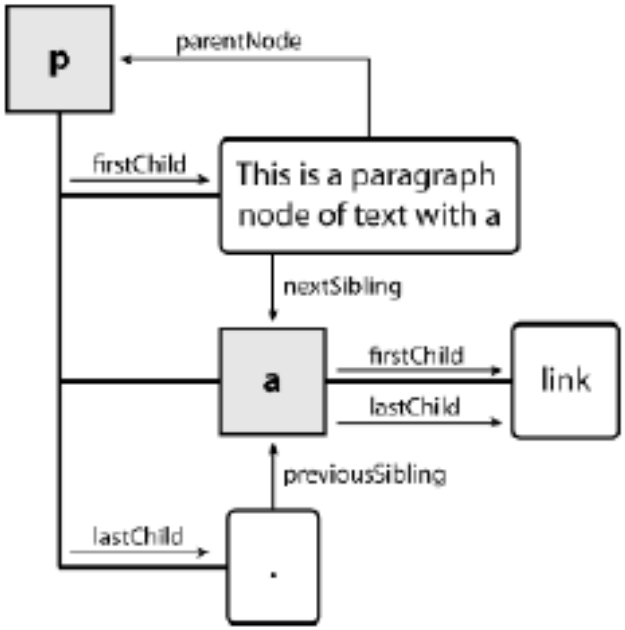
Example: Factorial

- Print factorial table

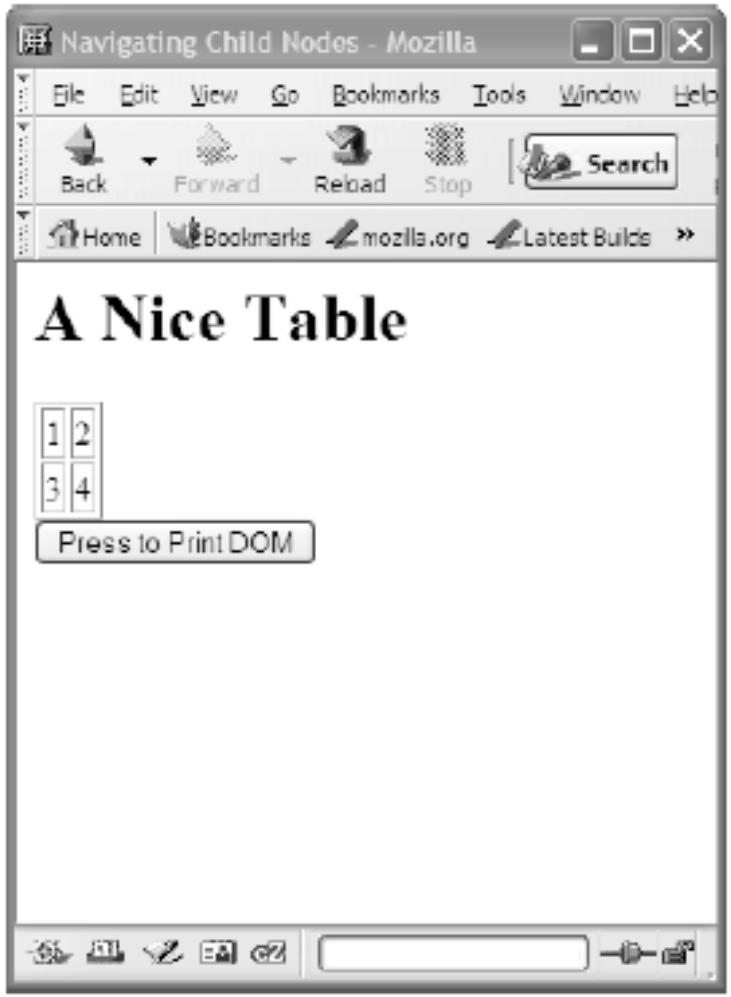


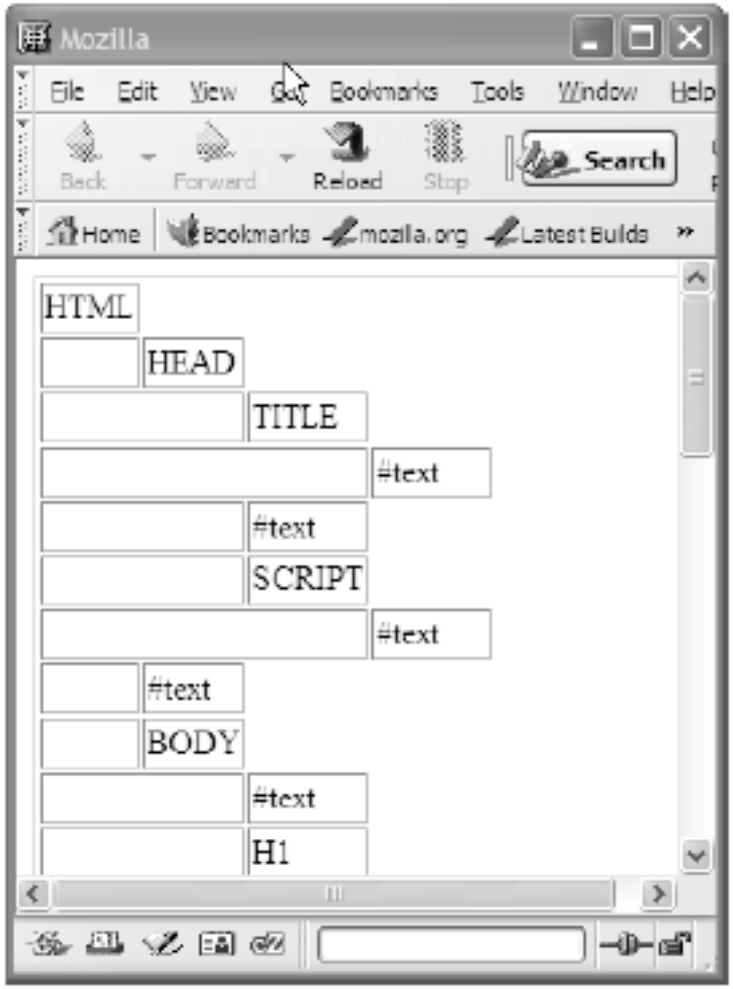
DOM Tree Traversal

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



Example: DOM Tree





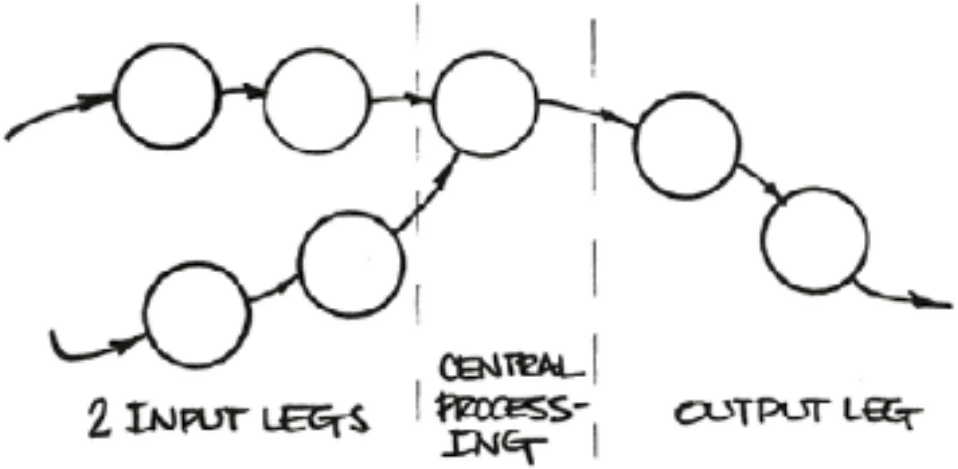
Modifying DOM

- 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 li
    - insertBefore(N,E) // Insert N in child list before E
    - cloneNode(deep) // Copy node. If deep=true copy
      // all descendents
- Removing nodes
  - Node
    - removeChild(N) // Removes N from child list

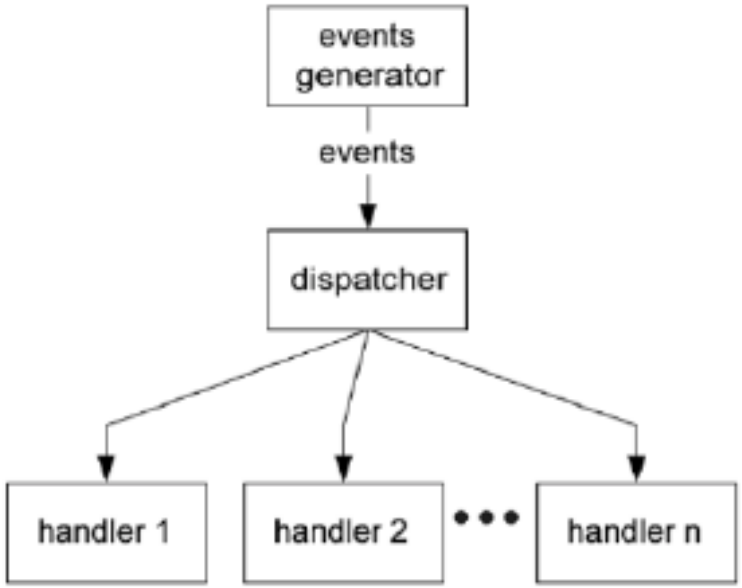
Example: Adding Table Rows

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

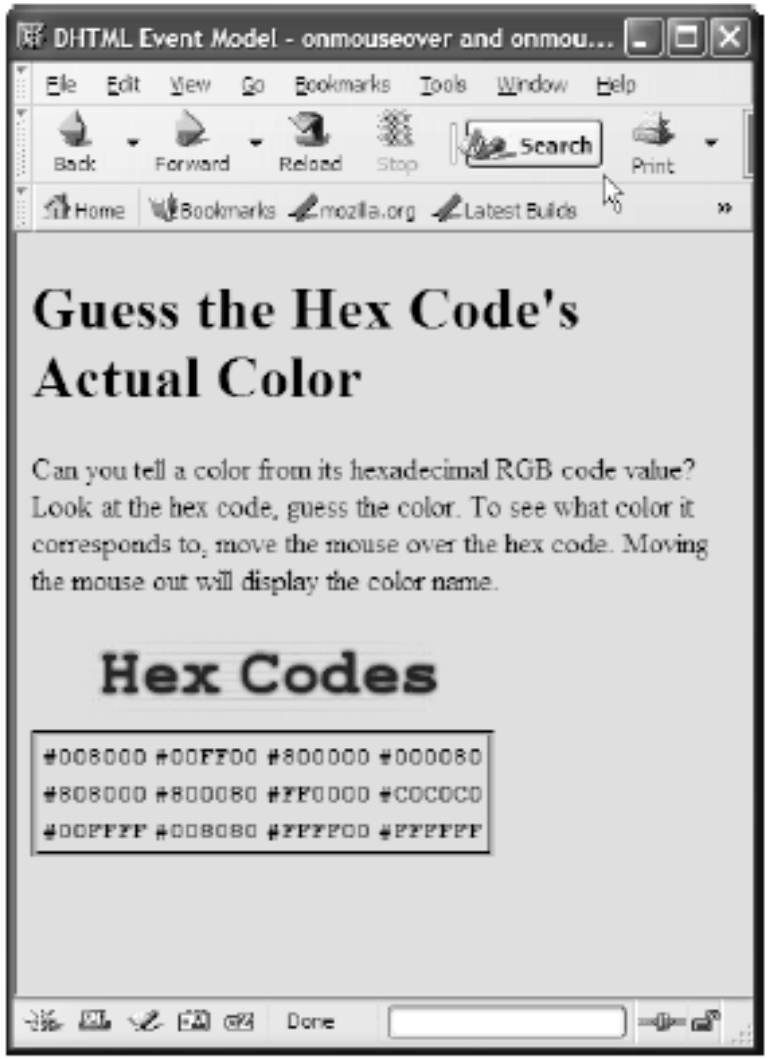
- Example: Count how many seconds have passed since page finish rendering

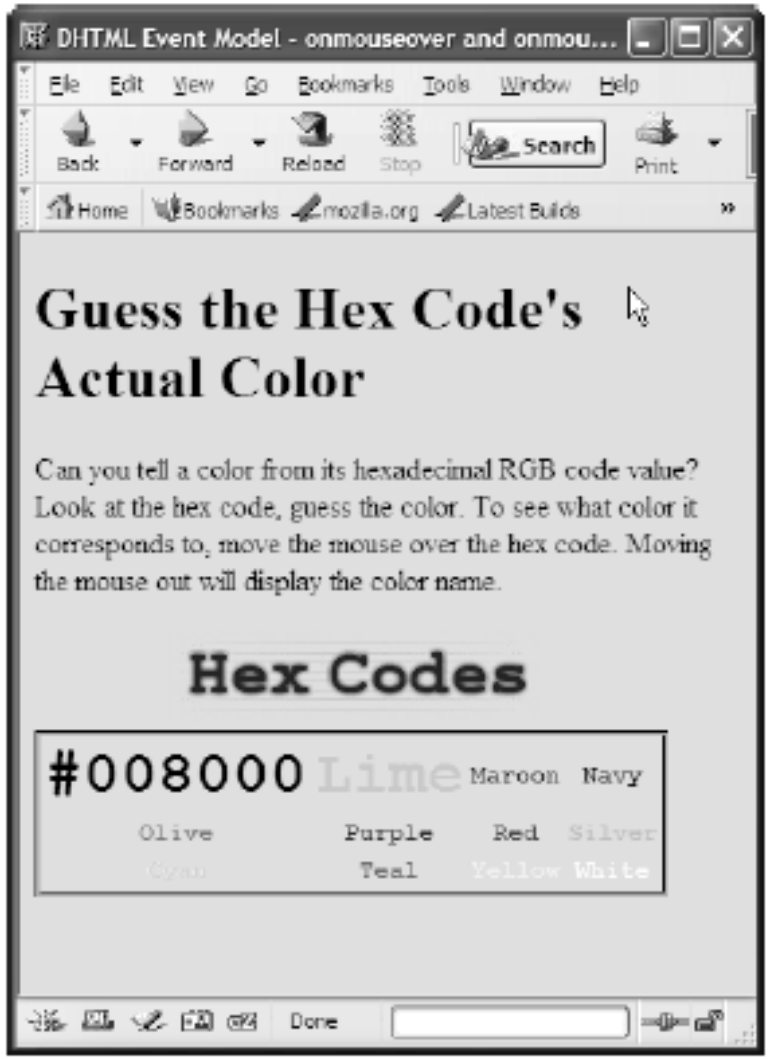
Event Bubbling

- 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





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