

JavaScript basics

SLIDES COURTESY:

DENNIS HIGGINS

STATE COLLEGE AT ONEONTA

EXAMPLES ARE IN A DIRECTORY MOSTLY LINKED AT

[HTTP://EMPLOYEES.ONEONTA.EDU/HIGGINDM/JAVASCRIPT/SCRIPTEXAMPLES.HTML](http://employees.oneonta.edu/higgindm/javascript/scriptexamples.html)



Useful Resources and Interesting Examples

JavaScript Tutorials

- <http://www.w3schools.com/>
- Some examples from W3Schools
 - [JavaScript Examples http://www.w3schools.com/js/js_examples.asp](http://www.w3schools.com/js/js_examples.asp)
 - [JavaScript Object Examples http://www.w3schools.com/js/js_examples_2.asp](http://www.w3schools.com/js/js_examples_2.asp)
 - [JavaScript HTML DOM Examples http://www.w3schools.com/js/js_examples_3.asp](http://www.w3schools.com/js/js_examples_3.asp)
- More examples
 - <http://employees.oneonta.edu/higgindm/javascript/scriptexamples.html>

JavaScript DHTML GUI Components

- <http://www.java2s.com/Code/JavaScript/GUI-Components/CatalogGUI-Components.htm>

EditGrid

- <http://www.editgrid.com/>

Javascript info

<https://javascript.info/>



Introduction

Introduction to HTML and DOM

What is it?

How does it work?

What is Java?

Learning JavaScript

- JavaScript Statements
- JavaScript and HTML forms

HTML Background

Many “markup” languages in the past

SGML: Standard Generalized Markup Language

- HTML (Hypertext Markup Language) based on SGML

XML (eXtensible Markup Language) “replaces” SGML

- XHTML is replacing HTML

Tags and Elements

Example of an element:

```
<name attr1="attrval">content</name>
```

Begin and end tags set off a section of a document

- Has a semantic property by tag-name
- Modified by attributes

“content” can contain other elements

- Elements nest, don’t “overlap”

Empty-elements: no end tag

- `
` ``
- Note space before `/>`

Basic HTML Structure

Comments:

`<!-- ... -->`

Example:

`<html>`

`<head>`

`...`

`</head>`

`<body>`

`....`

`</body>`

`</html>`

`<--- title, meta-tags,
etc. (not displayed)`

`<--- main content
(displayed)`

Larger Example

```
<html>
<head>
<title>An Example</title>
</head>
<body>
<h3><hr>An Example</h3>
<p align="left">
  <font face="Comic Sans MS" size="4"><b>
    Hello World!</b></font>
</p>
<p align="right">
  <font size="5"><u>I am 21.</u></font>
</p>
<!-- see next column -->
```

```
<p>
  <ol type="I" start=7>
    <li><font
      color=#00FF00>Green</font>
    </li>
    <li>Yellow</li>
    <ul type=square>
      <li>John</li>
      <li>Mike</li>
    </ul>
    </ol>
  </p>
</body>
</html>
```

Displays As...



More HTML

Learn on your own

You may never code in “raw” HTML

You may need to tweak HTML files created by a tool

You will need to understand HTML to code in JavaScript etc.

You will need to understand HTML to know limitations on how docs on the web can be structured

Question:

You're writing software to process an HTML page

- A web-browser engine, for example

What data structure would best represent an HTML document?

- Why?

Document Object Model (DOM)

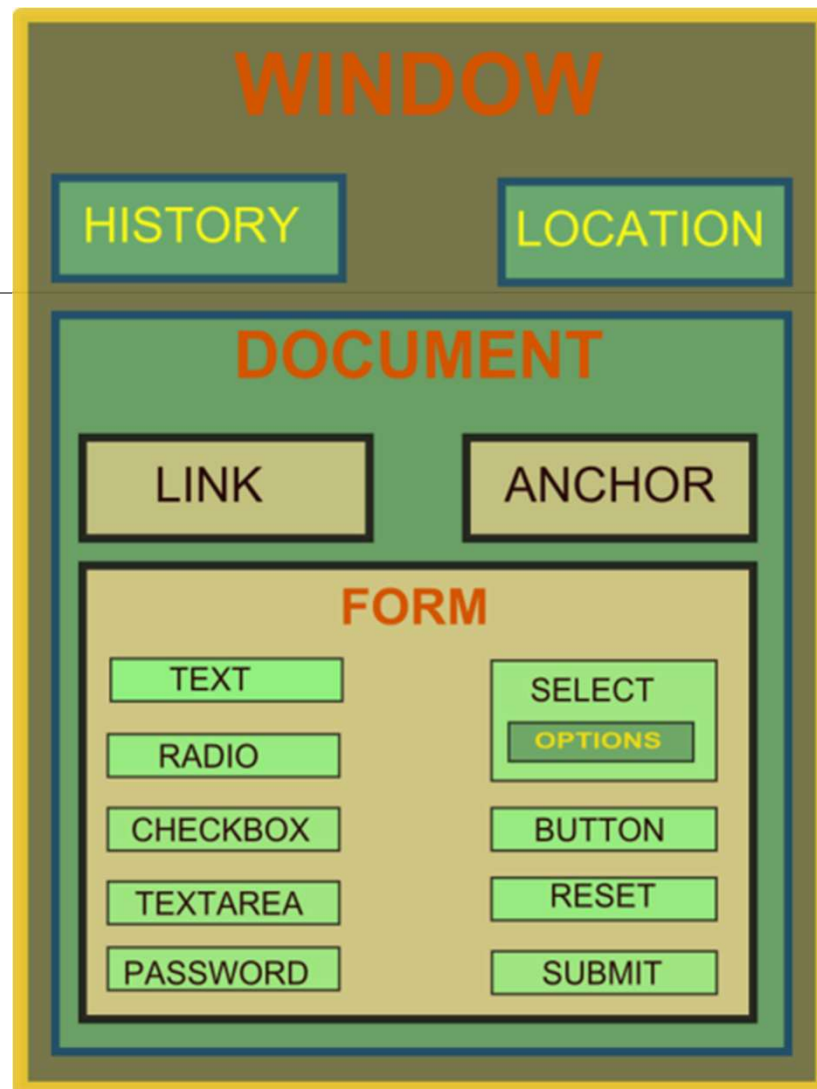
An model for describing HTML documents (and XML documents)

- A standard (ok, standards)
- Independent of browser, language
 - (ok, mostly)
- A common set of properties/methods to access everything in a web document

APIs in JavaScript, for Java, etc.

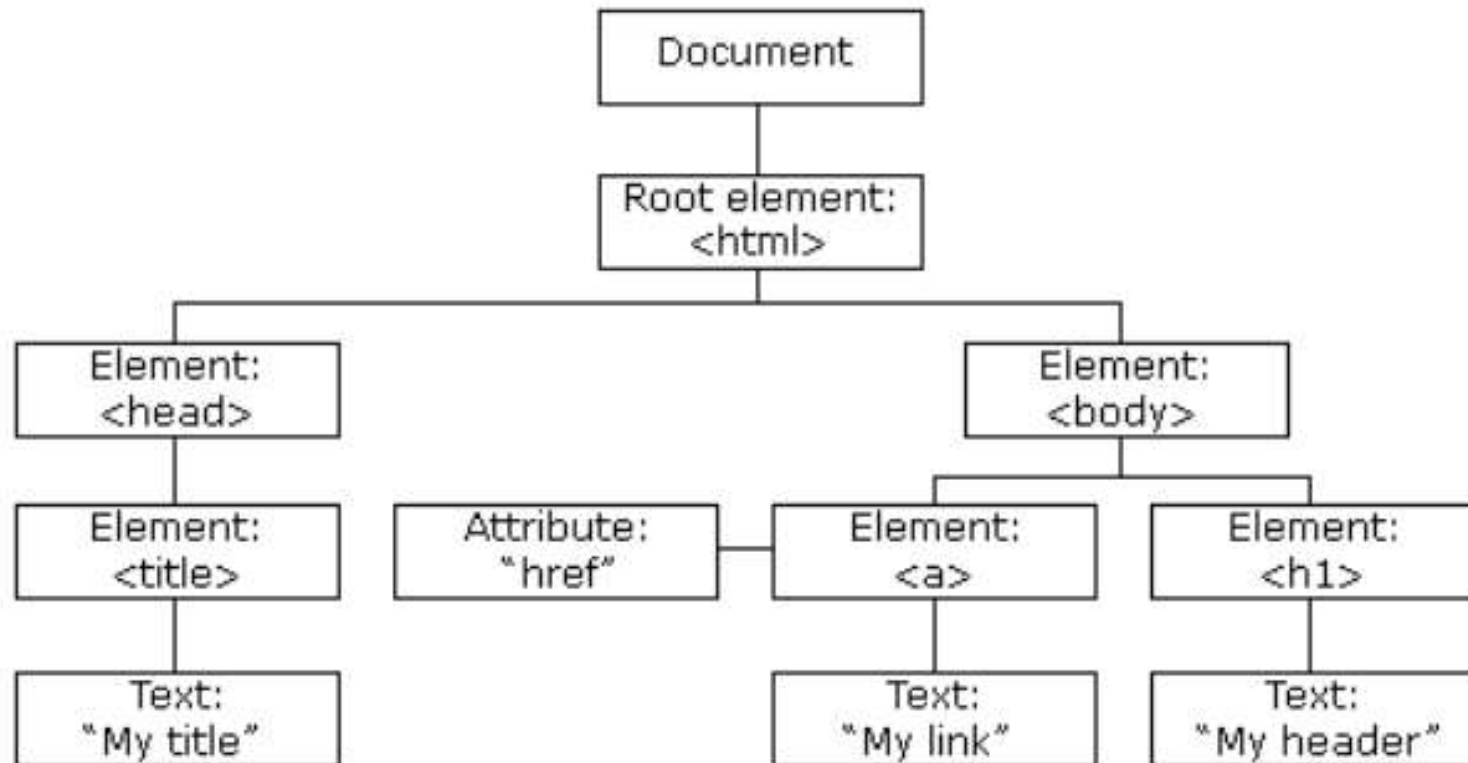
DOM

You get
anything you
want from...



More info: http://en.wikipedia.org/wiki/Document_Object_Model

The HTML DOM Tree of Objects



Background

- JavaScript was originally called LiveScript (pre-'95) and was developed by Netscape.
- It changed its name to JavaScript and became a joint venture with Sun.
- ECMA developed a standard for JavaScript in the late 90s which is also an ISO standard and the official name of the language is ECMAScript.
- ECMA is at version 3 which corresponds to Netscape's 1.5.

What is JavaScript?

Browsers have limited functionality

- Text, images, tables, frames

JavaScript allows for interactivity

Browser/page manipulation

- Reacting to user actions

A type of programming language

- Easy to learn
- Developed by Netscape
- Now a standard exists –
`www.ecma-international.org/publications/standards/ECMA-262.HTM`

JavaScript Allows Interactivity

Improve appearance

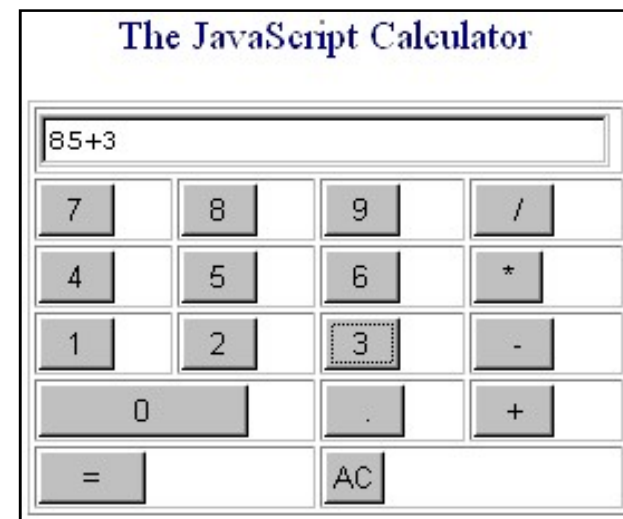
- Especially graphics
- Visual feedback

Site navigation

Perform calculations

Validation of input

Other technologies



How Does It Work?

Embedded within HTML page

- View source

Executes on client

- Fast, no connection needed once loaded

Simple programming statements combined with HTML tags

Interpreted (not compiled)

- No special tools required



Java vs. JavaScript

Requires the JDK to create the applet

Requires a Java virtual machine to run the applet

Applet files are distinct from the XHTML code

Source code is hidden from the user

Programs must be saved as separate files and compiled before they can be run

Programs run on the server side

Requires a text editor

Required a browser that can interpret JavaScript code

JavaScript can be placed within HTML and XHTML

Source code is made accessible to the user

Programs cannot write content to the hard disk

Programs run on the client side

Learning JavaScript

Special syntax to learn

Learn the basics and then use available code (lots of free sites)

Write it in a text editor, view results in browser

You need to revise your HTML

You need patience and good eyesight!

Examples:

<http://employees.oneonta.edu/higgindm/javascript/scriptexamples.html>

JavaScript Statements

```
<html>  
<head><title>My Page</title></head>  
<body>  
  <script language="JavaScript">
```

```
    document.write('This is my first →  
    JavaScript Page');
```



Note the symbol for
line continuation

```
  </script>  
</body>  
</html>
```

JavaScript Statements

```
<html>
```

```
<head><title>My Page</title></head>
```

```
<body>
```

```
<script language="JavaScript">
```

```
document.write('<h1>This is my first →
```

```
JavaScript Page</h1>');
```

```
</script>
```

```
</body>
```

```
</html>
```



HTML written
inside JavaScript

JavaScript Statements

```
<html>
<head><title>My Page</title></head>
<body>
<p>
<a href="myfile.html">My Page</a>
<br />
<a href="myfile.html"
onmouseover="window.alert('Hello');"
My Page</A>
</p>
</body>
</html>
```



An Event

JavaScript written
inside HTML

Example Statements

```
<script language="JavaScript">
```

```
window.prompt('Enter your name:', '');
```

```
</script>
```

Another event

```
<form>
```

```
<input type="button" Value="Press"  
onClick="window.alert('Hello');">
```

```
</form>
```

Note quotes: " and '



HTML Forms and JavaScript

JavaScript is very good at processing user input in the web browser

HTML `<form>` elements receive input

Forms and form elements have unique names

- Each unique element can be identified
- Uses JavaScript Document Object Model (DOM)

Naming Form Elements in HTML

Name:

Phone:

Email:

```
<form name="addressform">  
Name:  <input name="yourname"><br />  
Phone: <input name="phone"><br />  
Email: <input name="email"><br />  
</form>
```

Forms and JavaScript

`document.formname.elementname.value`

Thus:

`document.addressform.yourname.value`

`document.addressform.phone.value`

`document.addressform.email.value`

A diagram of a web form with a black border. Inside, there are three labels on the left and three corresponding input fields on the right. The labels are 'Name:', 'Phone:', and 'Email:'. The input fields are rectangular boxes. Three red lines originate from the code snippets above: one from 'yourname' pointing to the 'Name' field, one from 'phone' pointing to the 'Phone' field, and one from 'email' pointing to the 'Email' field.

Name:	<input type="text"/>
Phone:	<input type="text"/>
Email:	<input type="text"/>

Using Form Data

Personalising an alert box

Enter your name:



```
<form name="alertform">
```

Enter your name:

```
<input type="text" name="yourname">
```

```
<input type="button" value="Go"  
onClick="window.alert('Hello ' + →  
document.alertform.yourname.value);">
```

```
</form>
```

Parts of JavaScript

Core: operators, expressions, statements and subprograms

Client-side: browser control and user interaction (ie., mouse-clicks, input)

Server-side: language features for use on a server – not covered in this text.

Client-side JavaScript

XHTML-embedded scripting language.

Client-side JavaScript can substitute for some server-side functionality – especially things like checking form data to make sure phone numbers have an area code or ss numbers have the right format.

The browser interprets the JavaScript.

Client-side JavaScript cannot substitute for server-side file networking and database operations.



Interaction

Mouseclicks and form data can trigger java script functions.

DOM allows JavaScript to access and modify the CSS properties and content of any element of a displayed XHTML – enabling dynamic presentation.

Interpreter

Browsers use their JavaScript interpreters to process scripts embedded in html.

Client-side input data checking is an important role for JavaScript. Without this, form information goes to the server, which must locate errors and relay them back to the client.



Placement in head or body of html

Typically function definitions or code for handling events is placed in the head (and accessed if/as needed).

Script meant to be processed just once as part of document content goes in the body. Scripts in the head are cataloged but not interpreted at that time. Scripts in the body are processed as encountered.



Reserved words

break	delete	function	Return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with



More about reserved words

Look at <http://www.ecma.ch> to see a list of future reserved words.

Also some predefined words: ***alert, open, self, java***

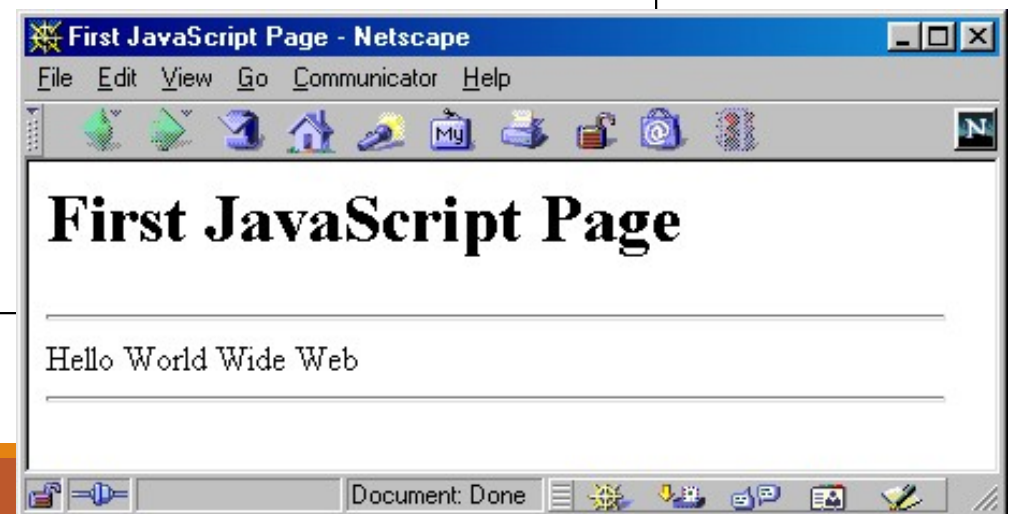
// on a line signals comment to end

C++/Java multiline comment syntax: ***/*...*/***

Most browsers now do recognize JavaScript so placing script in special comments is less important, except it may cause validation problems for XHTML validators if the JavaScript includes recognized tags like ***

A Simple Script

```
<html>
<head><title>First JavaScript Page</title></head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
    document.write("<hr>");
    document.write("Hello World Wide Web");
    document.write("<hr>");
</script>
</body>
</html>
```



Embedding JavaScript

```
<html>
<head><title>First JavaScript Program</title></head>
<body>
<script type="text/javascript"
        src="your_source_file.js"></script>
</body>
</html>
```

[Inside your source file.js](#)

```
document.write("<hr>");
document.write("Hello World Wide Web");
document.write("<hr>");
```

Use the **src** attribute to include JavaScript codes from an external file.

The included code is inserted in place.

Embedding JavaScript

The scripts inside an HTML document is interpreted in the order they appear in the document.

- Scripts in a function is interpreted when the function is called.

So where you place the `<script>` tag matters.



Hiding JavaScript from Incompatible Browsers

```
<script type="text/javascript">
<!--
    document.writeln("Hello, WWW");
// -->
</script>
<noscript>
    Your browser does not support JavaScript.
</noscript>
```

```
<script type="text/javascript">  
alert("This is an Alert method");  
confirm("Are you OK?");  
prompt("What is your name?");  
prompt("How old are you?", "20");  
</script>
```

nd **prompt()**



alert() and confirm()

```
alert("Text to be displayed");
```

Display a message in a dialog box.

The dialog box will block the browser.

```
var answer = confirm("Are you sure?");
```

- Display a message in a dialog box with two buttons: "OK" or "Cancel".
- `confirm()` returns `true` if the user click "OK". Otherwise it returns `false`.

prompt ()

```
prompt("What is your student id number?");  
prompt("What is your name?", "No name");
```

Display a message and allow the user to enter a value

The second argument is the "default value" to be displayed in the input textfield.

Without the default value, "undefined" is shown in the input textfield.

If the user click the "OK" button, **prompt ()** returns the value in the input textfield as a string.

If the user click the "Cancel" button, **prompt ()** returns null.

Identifier

Same as Java/C++ except that it allows an additional character – '\$'.

Contains only 'A' – 'Z', 'a' – 'z', '0' – '9', '_', '\$'

First character cannot be a digit

Case-sensitive

Cannot be reserved words or keywords

Variable and Variable Declaration

```
<head><script type="text/javascript">
  // We are in the default scope - the "window" object.
  x = 3;    // same as "window.x = 3"
  var y = 4; // same as "y = 4" or "window.y = 4"

  { // Introduce a block to create a local scope
    x = 0;    // Same as "window.x = 0"
    var y = 1; // This is a local variable y
  }

  alert("x=" + x + ", y=" + y); // Print x=0, y=4
</script></head>
```

Local variable is declared using the keyword 'var'.

Dynamic binding – a variable can hold any type of value

If a variable is used without being declared, the variable is created automatically.

- If you misspell a variable name, program will still run (but works incorrectly)

Data Types


Primitive data types

- **Number**: integer & floating-point numbers
- **Boolean**: true or false
- **String**: a sequence of alphanumeric characters

Composite data types (or Complex data types)

- **Object**: a named collection of data
- **Array**: a sequence of values (an array is actually a predefined object)

Special data types

- **Null**: the only value is "null" – to represent nothing.
 - **Undefined**: the only value is "undefined" – to represent the value of an uninitialized variable
- 

Strings

A string variable can store a sequence of alphanumeric characters, spaces and special characters.

Each character is represented using 16 bit

- You can store Chinese characters in a string.

A string can be enclosed by a pair of single quotes (') or double quote (").

Use escaped character sequence to represent special character (e.g.: `\`" , `\n` , `\t`)

typeof operator

```
var x = "hello", y;  
alert("Variable x value is " + typeof x );  
alert("Variable y value is " + typeof y );  
alert("Variable x value is " + typeof z );
```

An unary operator that tells the type of its operand.

- Returns a string which can be "number", "string", "boolean", "object", "function", "undefined", and "null"
- An array is internally represented as an object.

Object

An object is a collection of [properties](#).

Properties can be variables (Fields) or Functions (Methods)

There is no "Class" in JavaScript.

Array

An array is represented by the **Array** object. To create an array of N elements, you can write


```
var myArray = new Array(N) ;
```

Index of array runs from 0 to N-1.

Can store values of different types

Property "**length**" tells the # of elements in the array.

Consists of various methods to manipulate its elements. e.g., **reverse()**, **push()**, **concat()**, etc




```
var Car = new Array(3);  
Car[0] = "Ford";  
Car[1] = "Toyota";  
Car[2] = "Honda";  
  
// Create an array of three elements with initial  
// values  
var Car2 = new Array("Ford", "Toyota", "Honda");  
  
// Create an array of three elements with initial  
// values  
var Car3 = ["Ford", "Toyota", "Honda"];
```

```
// An array of 3 elements, each element is undefined
var tmp1 = new Array(3);

// An array of 3 elements with initial values
var tmp2 = new Array(10, 100, -3);

// An array of 3 elements with initial values
// of different types
var tmp3 = new Array(1, "a", true);

// Makes tmp3 an array of 10 elements
tmp3.length = 10; // tmp[3] to tmp[9] are undefined.

// Makes tmp3 an array of 100 elements
tmp3[99] = "Something";
// tmp[3] to tmp[98] are undefined.
```

Null & Undefined

An undefined value is represented by the keyword "**undefined**".

- It represents the value of an uninitialized variable

The keyword "**null**" is used to represent “nothing”

- Declare and define a variable as “null” if you want the variable to hold nothing.
- Avoid leaving a variable undefined.

Weakly Typed Language

JavaScript is considered a "weakly typed" or "untyped" language.

JavaScript will figure out what type of data you have and make the necessary adjustments so that you don't have to redefine your different types of data.

For example, Performing calculations involving money requires floating-point numbers.


However, as soon as a dollar sign is added to the number, it becomes a string variable.

In strongly typed languages, you need to convert the floating-point number to a string and then concatenate it with the dollar sign.



Weaklytype.html

```
<html>
<head>
<script language="JavaScript">
var apples=1.43;
var oranges=2.33;
var pears=4.32;
var tax=.04;
var shipping=2.75;
var subtotal=apples + oranges + pears;
var total=subtotal + (subtotal * tax) + shipping;
var message="Your total is $";
var deliver= message + total + ".";
document.write(deliver);
</script>
</head>
<body bgcolor="indianred">
</body>
</html>
```



Type Conversion (To Boolean)

The following values are treated as false

- null
- undefined
- +0, -0, NaN (numbers)
- "" (empty string)

Type Conversion

Converting a value to a number

```
var numberVar = someVariable - 0;
```

Converting a value to a string

```
var stringVar = someVariable + "";
```

Converting a value to a boolean

```
var boolVar = !!someVariable;
```

Operators

Arithmetic operators

- +, -, *, /, %

Post/pre increment/decrement

- ++, --

Comparison operators

- ==, !=, >, >=, <, <=
- ===, !== (Strictly equals and strictly not equals)
 - i.e., Type and value of operand must match / must not match


```
// Type conversion is performed before comparison
var v1 = ("5" == 5);    // true

// No implicit type conversion.
// True if only if both types and values are equal
var v2 = ("5" === 5);   // false

var v3 = (5 === 5.0);   // true

var v4 = (true == 1);   // true (true is converted to 1)

var v5 = (true == 2);   // false (true is converted to 1)

var v6 = (true == "1")  // true
```

Logical Operators

! – Logical NOT

&& – Logical AND

- **OP1 && OP2**
- If OP1 is true, expression evaluates to the value of OP2. Otherwise the expression evaluates to the value of OP1.
- Results may not be a boolean value.

|| – Logical OR

- **OP1 || OP2**
- If OP1 is true, expression evaluates to the value of OP1. Otherwise the expression evaluates to the value of OP2.

```
var tmp1 = null && 1000;    // tmp1 is null
var tmp2 = 1000 && 500;      // tmp2 is 500
var tmp3 = false || 500;    // tmp3 is 500
var tmp4 = "" || null;      // tmp4 is null
var tmp5 = 1000 || false;   // tmp5 is 1000

// If foo is null, undefined, false, zero, NaN,
// or an empty string, then set foo to 100.
foo = foo || 100;
```

Operators (continue)

String concatenation operator

- `+`
- If one of the operand is a string, the other operand is automatically converted to its equivalent string value.

Assignment operators

- `=, +=, -=, *=, /=, %=`

Bitwise operators

- `&, |, ^, >>, <<, >>>`

Conditional Statements

“if” statement

“if ... else” statement

“?:” ternary conditional statement

“switch” statement

The syntax of these statements are similar to those found in C and Java.

Looping Statement

“for” Loops

“for/in” Loops

“while” Loops

“do ... while” Loops

“break” statement

“continue” statement

All except "for/in" loop statements have the same syntax as those found in C and Java.

“for/in” statement

```
for (var variable in object) {  
    statements;  
}
```

To iterate through all the properties in "object".

"**variable**" takes the name of each property in "object"

Can be used to iterate all the elements in an Array object.

```
var keys = "", values = "";
var mylist = new Array("Chinese", "English", "Jap");
mylist.newField1 = "Something";

for (var key in booklist) {
    keys += key + " ";
    values += booklist[counter] + " ";
}

// keys becomes "0 1 2 newField1"
// values becomes "Chinese English Jap Something"
```



```
var obj = new Object(); // Creating an object

// Adding three properties to obj
obj.prop1 = 123;
obj.prop2 = "456";
obj["prop3"] = true; // same as obj.prop3 = true

var keys = "", values = "";
for (var p in obj) {
    keys += p + " ";
    values += obj[p] + " ";
}

alert(keys);
// Show "prop1 prop2 prop3 "

alert(values);
// Show "123 456 true "
```

Example: Using for ... in loop with object

Functions (Return Values)

```
// A function can return value of any type using the  
// keyword "return".
```

```
// The same function can possibly return values  
// of different types
```

```
function foo (p1) {  
    if (typeof(p1) == "number")  
        return 0;    // Return a number  
    else  
        if (typeof(p1) == "string")  
            return "zero"; // Return a string
```

```
    // If no value being explicitly returned  
    // "undefined" is returned.
```

```
}
```

```
foo(1);        // returns 0  
foo("abc");    // returns "zero"  
foo();         // returns undefined
```

Variable Arguments

```
// "arguments" is a local variable (an array) available
// in every function
// You can either access the arguments through parameters
// or through the "arguments" array.
function sum ()
{
    var s = 0;
    for (var i = 0; i < arguments.length; i++)
        s += arguments[i];
    return s;
}

sum(1, 2, 3);           // returns 6
sum(1, 2, 3, 4, 5);     // returns 15
sum(1, 2, "3", 4, 5);   // returns ?
```

Built-In Functions

eval (expr)

- evaluates an expression or statement
 - `eval("3 + 4");` // Returns 7 (Number)
 - `eval("alert('Hello')");` // Calls the function `alert('Hello')`

isFinite (x)

- Determines if a number is finite

isNaN (x)

- Determines whether a value is “Not a Number”

Built-In Functions

`parseInt(s)`

`parseInt(s, radix)`

- Converts string literals to integers
- Parses up to any character that is not part of a valid integer
 - `parseInt("3 chances")` // returns 3
 - `parseInt(" 5 alive")` // returns 5
 - `parseInt("How are you")` // returns NaN
 - `parseInt("17", 8)` // returns 15

`parseFloat(s)`

- Finds a floating-point value at the beginning of a string.
 - `parseFloat("3e-1 xyz")` // returns 0.3
 - `parseFloat("13.5 abc")` // returns 13.5

OO

JavaScript is not OO, but Object based. There are no classes, polymorphism or object inheritance.

However, documents and elements are modeled with objects and so there is an object-sense to JavaScript.

JavaScript objects are collections of properties, roughly corresponding to C++ or Java objects. Each is a data or method property. Data properties may be primitives or *references*, although in JavaScript the latter are themselves usually called *Objects*.

The text will usually use *properties* to mean data properties and *methods* to mean method properties.

Creating Objects

JavaScript is not an OOP language.

"prototype" is the closest thing to "class" in JavaScript.

Next few slides show several ways to create objects

It is also possible to emulate "inheritance" in JavaScript.

- See **JavaScript and Object Oriented Programming (OOP)**
(<http://www.javascriptkit.com/javatutors/oopjs.shtml>)

Creating objects using **new Object()**

```
var person = new Object();

// Assign fields to object "person"
person.firstName = "John";
person.lastName = "Doe";

// Assign a method to object "person"
person.sayHi = function() {
    alert("Hi! " + this.firstName + " " + this.lastName);
}

person.sayHi(); // Call the method in "person"
```


Creating objects using Literal Notation

```
var person = {  
    // Declare fields  
    // (Note: Use comma to separate fields)  
    firstName : "John",  
    lastName  : "Doe",  
  
    // Assign a method to object "person"  
    sayHi : function() {  
        alert("Hi! " + this.firstName + " " +  
            this.lastName);  
    }  
}  
  
person.sayHi(); // Call the method in "person"
```

Creating objects using Literal Notation (Nested notation is possible)

```
var triangle = {  
  // Declare fields (each as an object of two fields)  
  p1 : { x : 0, y : 3 },  
  p2 : { x : 1, y : 4 },  
  p3 : { x : 2, y : 5 }  
}  
  
alert(triangle.p1.y);    // Show 3
```

Object Constructor and prototyping

```
function Person(fname, lname) {  
    // Define and initialize fields  
    this.firstName = fname;  
    this.lastName = lname;  
  
    // Define a method  
    this.sayHi = function() {  
        alert("Hi! " + this.firstName + " " +  
            this.lastName);  
    }  
}  
  
var p1 = new Person("John", "Doe");  
var p2 = new Person("Jane", "Dow");  
  
p1.sayHi();    // Show "Hi! John Doe"  
p2.sayHi();    // Show "Hi! Jane Dow"
```

Adding methods to objects using prototype

```
// Suppose we have defined the constructor "Person"
// (as in the previous slide).

var p1 = new Person("John", "Doe");
var p2 = new Person("Jane", "Dow");

// Attaching a new method to all instances of Person
Person.prototype.sayHello = function() {
    alert("Hello! " + this.firstName + " " +
          this.lastName);
}

// We can also introduce new fields via "prototype"

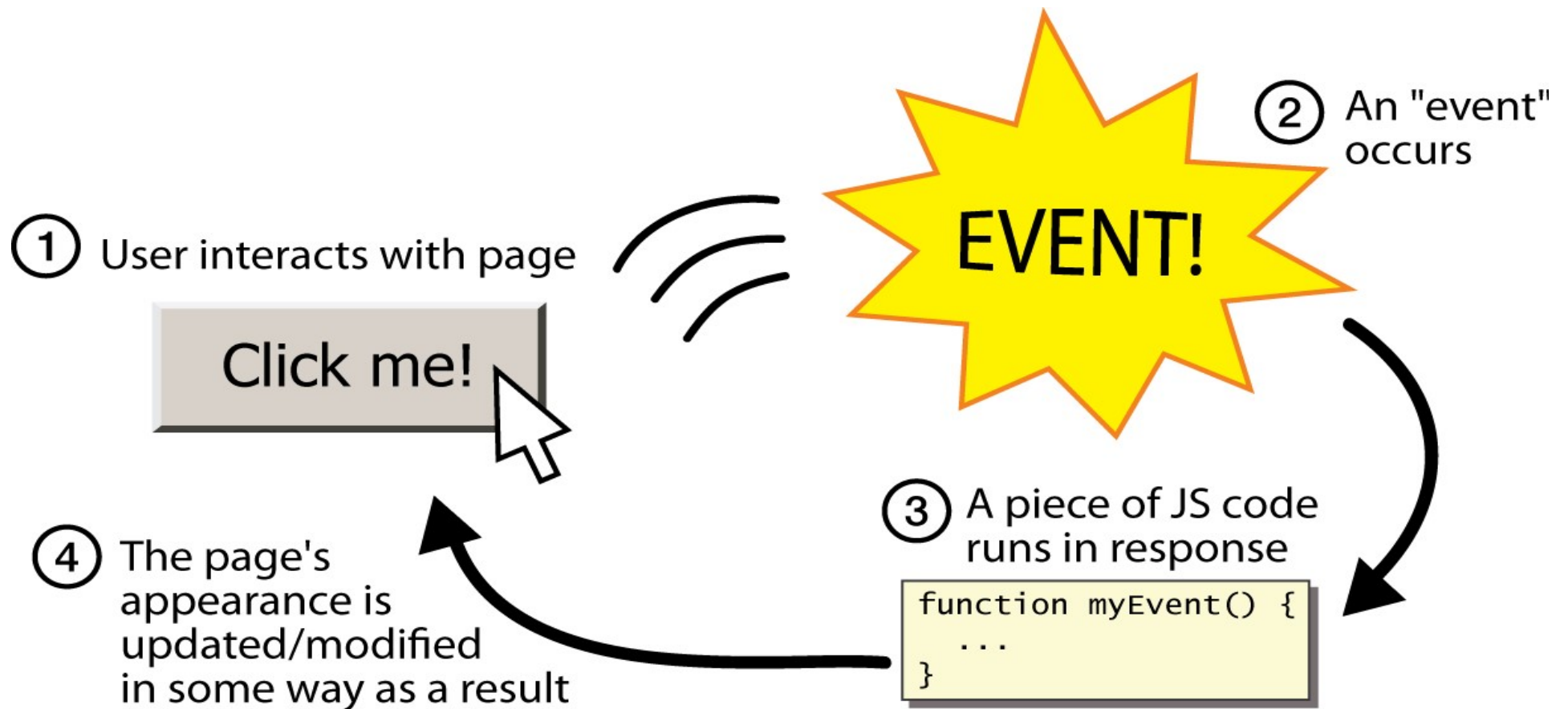
p1.sayHello(); // Show "Hello! John Doe"
p2.sayHello(); // Show "Hello! Jane Dow"
```

Events

An event occurs as a result of some activity

- e.g.:
 - A user clicks on a link in a page
 - Page finished loaded
 - Mouse cursor enter an area
 - A preset amount of time elapses
 - A form is being submitted

Event-driven programming



Event-driven programming

- ❑ you are used to programs start with a main method (or implicit main like in PHP)
- ❑ JavaScript programs instead wait for user actions called *events* and respond to them
- ❑ event-driven programming: writing programs driven by user events
- ❑ Let's write a page with a clickable button that pops up a "Hello, World" window...

Event Handlers

Event Handler – a segment of codes (usually a function) to be executed when an event occurs

We can specify event handlers as attributes in the HTML tags.

The attribute names typically take the form "**onXXX**" where **XXX** is the event name.

- e.g.:

```
<a href="..." onClick="alert('Bye') ">Other  
Website</a>
```


Buttons

```
<button>Click me!</button>
```

HTML

button's text appears inside tag; can also contain images

To make a responsive button or other UI control:

1. choose the control (e.g. button) and event (e.g. mouse 1. click) of interest
2. write a JavaScript function to run when the event occurs
3. attach the function to the event on the control

JavaScript functions

```
function name() {  
  statement ;  
  statement ;  
  ...  
  statement ;  
}
```

JS

```
function myFunction() {  
  alert("Hello!");  
  alert("How are you?");  
}
```

JS

- ❑ the above could be the contents of `example.js` linked to our HTML page
- ❑ statements placed into functions can be evaluated in response to user events

Event handlers

```
<element attributes onclick="function();">...
```

HTML

```
<button onclick="myFunction();">Click me!</button>
```

HTML

JavaScript functions can be set as event handlers

- when you interact with the element, the function will execute

onclick is just one of many event HTML attributes we'll use

but popping up an alert window is disruptive and annoying

- A better user experience would be to have the message appear on the page...

Event Handlers

Event Handlers	Triggered when
onChange	The value of the text field, textarea, or a drop down list is modified
onClick	A link, an image or a form element is clicked once
onDbIcIck	The element is double-clicked
onMouseDown	The user presses the mouse button
onLoad	A document or an image is loaded
onSubmit	A user submits a form
onReset	The form is reset
onUnLoad	The user closes a document or a frame
onResize	A form is resized by the user

For a complete list, see http://www.w3schools.com/html/dom/dom_obj_event.asp

onClick Event Handler

```
<html>
<head>
<title>onClick Event Handler Example</title>
<script type="text/javascript">
function warnUser() {
    return confirm("Are you a student?");
}
</script>
</head>
<body>
<a href="ref.html" onClick="return warnUser()">
<!--
    If onClick event handler returns false, the link
    is not followed.
-->
Students access only</a>
</body>
</html>
```

onLoad Event Handler Example

```
<html><head>
<title>onLoad and onUnload Event Handler
Example</title>
</head>
<body
  onLoad="alert('Welcome to this page')"
  onUnload="alert('Thanks for visiting this page')"
>
Load and UnLoad event test.
</body>
</html>
```

onMouseOver & onMouseOut Event Handler

```
<html>
<head>
<title>onMouseOver / onMouseOut Event Handler
Demo</title>
</head>
<body>
<a href="http://www.cuhk.edu.hk"
  onMouseOver="window.status='CUHK Home'; return true;"
  onMouseOut="status=' '"
>CUHK</a>
</body>
</html>
```

- When the mouse cursor is over the link, the browser displays the text "CUHK Home" instead of the URL.
- The "return true;" of `onMouseOver` forces browser not to display the URL.
- `window.status` and `window.defaultStatus` are disabled in Firefox.

onSubmit Event Handler Example

```
<html><head>
<title>onSubmit Event Handler Example</title>
<script type="text/javascript">
    function validate() {
        // If everything is ok, return true
        // Otherwise return false
    }
</script>
</head>
<body>
<form action="MessageBoard" method="POST"
    onSubmit="return validate();"
>
...
</form></body></html>
```

- If **onSubmit** event handler returns false, data is not submitted.
- If **onReset** event handler returns false, form is not reset

Build-In JavaScript Objects

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Error	Returns run-time error information
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

- See online references for complete list of available methods in these objects: <http://javascript-reference.info/>

String Object (Some useful methods)

length

- A string property that tells the number of character in the string
-

charAt(idx)

- Returns the character at location "idx"

toUpperCase(), toLowerCase()

- Returns the same string with all uppercase/lowercase letters

substring(beginIdx)

- Returns a substring started at location "beginIdx"

substring(beginIdx, endIdx)

- Returns a substring started at "beginIdx" until "endIdx" (but not including "endIdx")

indexOf(str)

- Returns the position where "str" first occurs in the string



Error and Exception Handling in JavaScript

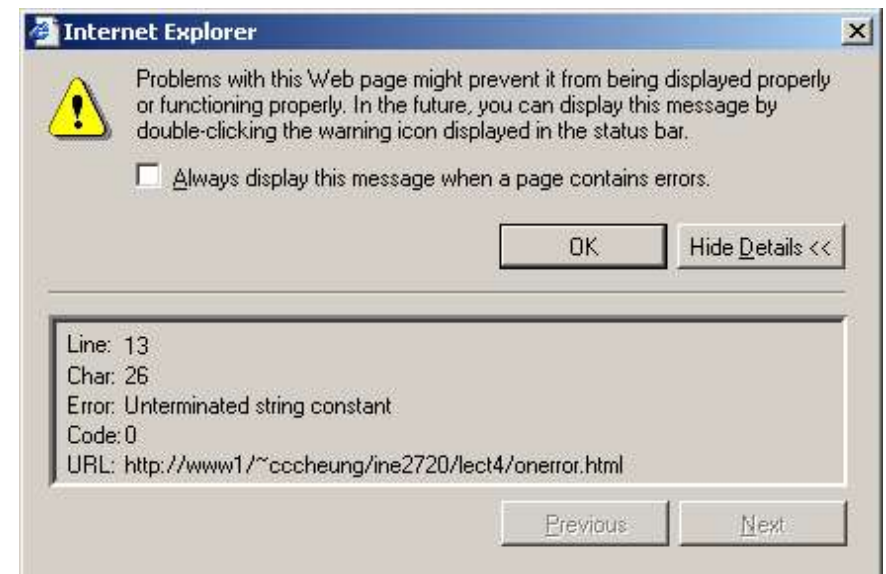
Javascript makes no distinction between Error and Exception (Unlike Java)

Handling Exceptions

- The **onError** event handler
 - A method associated with the window object.
 - It is called whenever an exception occurs
- The **try ... catch ... finally** block
 - Similar to Java try ... catch ... finally block
 - For handling exceptions in a code segment
- Use **throw** statement to throw an exception
 - You can throw value of any type
- The **Error** object
 - Default object for representing an exception
 - Each Error object has a **name** and **message** properties

How to use “onError” event handler?

```
<html>
<head>
<title>onerror event handler example</title>
<script type="text/javascript">
function errorHandler() {
    alert("Error Occured!");
}
// JavaScript is casesensitive
// Don't write onerror!
window.onError = errorHandler;
</script>
</head>
<body>
<script type="text/javascript">
    document.write("Hello there;
</script>
</body>
</html>
```



try ... catch ... finally

```
try {  
    // Contains normal codes that might throw an exception.  
  
    // If an exception is thrown, immediately go to  
    //    catch block.  
  
} catch ( errorVariable ) {  
    // Codes here get executed if an exception is thrown  
    //    in the try block.  
  
    // The errorVariable is an Error object.  
  
} finally {  
    // Executed after the catch or try block finish  
  
    // Codes in finally block are always executed  
}  
// One or both of catch and finally blocks must accompany the try  
// block.
```

try ... catch ... finally example

```
<script type="text/javascript">
try{
  document.write("Try block begins<br>");
  // create a syntax error
  eval ("10 + * 5");

} catch( errVar ) {
  document.write("Exception caught<br>");
  // errVar is an Error object
  // All Error objects have a name and message properties
  document.write("Error name: " + errVar.name + "<br>");
  document.write("Error message: " + errVar.message +
    "<br>");
} finally {
  document.write("Finally block reached!");
}
</script>
```

Throwing Exception

```
<script type="text/javascript">
try{
  var num = prompt("Enter a number (1-2):", "1");
  // You can throw exception of any type
  if (num == "1")
    throw "Some Error Message";
  else
    if (num == "2")
      throw 123;
    else
      throw new Error ("Invalid input");
} catch( err ) {
  alert(typeof(errMsg) + "\n" + err);
  // instanceof operator checks if err is an Error object
  if (err instanceof Error)
    alert("Error Message: " + err.message);
}
</script>
```

DOM element objects

HTML

```
<p>  
  Look at this octopus:  
    
  Cute, huh?  
</p>
```

DOM Element Object	
Property	Value
tagName	"IMG"
<u>src</u>	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");  
icon.src = "kitty.gif";
```


Accessing elements:

`document.getElementById`

```
var name = document.getElementById("id");
```

JS

```
<button onclick="changeText();">Click me!</button>  
<span id="output">replace me</span>  
<input id="textbox" type="text" />
```

HTML

```
function changeText() {  
    var span = document.getElementById("output");  
    var textBox = document.getElementById("textbox");  
  
    textBox.style.color = "red";  
}
```

JS

Accessing elements:

`document.getElementById`

- ❑ `document.getElementById` returns the DOM object for an element with a given id
- ❑ can change the text inside most elements by setting the `innerHTML` property
- ❑ can change the text in form controls by setting the `value` property

Changing element style:

`element.style`

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Preetify

```
function changeText() {  
    //grab or initialize text here  
  
    // font styles added by JS:  
    text.style.fontSize = "13pt";  
    text.style.fontFamily = "Comic Sans MS";  
    text.style.color = "red"; // or pink?  
}
```

JS