

**IT 3204**  
**Introduction to Web Development**

JavaScript I  
September 20

**Syllabus Errors Corrected**

- The first exam is October 2. Please correct your paper syllabus.
- An error in the description of how term papers are graded has been corrected; the term paper is worth 10% of your grade as stated in the grading plan. (Applies to 5123 only.)
- Both errors have been corrected in the on-line syllabus.

**The Three Column Layout**

- Works as advertised in IE 6;
- Fails in Firefox. (The tops of the columns are not aligned.)
- There are two minor typos in the slide.
- I will out-stubborn this and post updated slides.

**Assignment 4, Part 2**

- Some of you got a freebie out of this because of my error. OK.
- Be sure you put a link on your index page so I know you're using your free shot.

**Client-Side Scripting**

- Programming statements (scripts) are included in HTML documents.
- The script is transferred (using HTTP) to the browser.
- The browser interprets the script.
- The script can manipulate the HTML document.
- The standard scripting language is JavaScript.

**JavaScript**

- Embedded in HTML documents
- Originally "Livescript"
- Developed by Netscape
- Partnership with Sun in 1995; new name: JavaScript
- Now an ISO Standard (ECMAScript)

### Parts of JavaScript

- Core Language
- Client-side extensions
  - Ability to manipulate Web page objects
  - Document Object Model
- Server-side extensions (rare)

### JavaScript is Not Java

- JavaScript is not (very) object-oriented; Java is all about objects
- JavaScript is dynamically-typed  
Java is strongly typed  
Java typing applies to variables and objects
- However, the two languages have similar syntax.

### Uses of JavaScript

Client-side programming:

- Standardized
- An alternative to server programming (but with conditions)
- An alternative to Java applets

### Advantages of JavaScript

- Easier to learn than Java
- May be faster than Java applets
- Provides for interaction with users through forms
- Allows dynamic Web pages by using the Document Object Model (DOM)

### What JavaScript Can't Do

- JavaScript is limited to the context of the Web browser
- No graphics capability (but can dynamically generate HTML)
- No file input/output
- Very limited networking capability:
  - URL manipulation
  - ability to use GET/POST with XMLHttpRequest.

### JavaScript is Event-Driven

- Web pages have "events"
  - A page is loaded
  - A page is unloaded
  - Mouse actions (hover, click)
  - Focus / blur actions
  - Keyboard actions
- Events can trigger JavaScript functions
- It is also possible to write JavaScript that is run when it is loaded, and not in response to an event

## JavaScript and HTML

- JavaScript is embedded in HTML
- JavaScript functions go in the **<head>** of the document.
- JavaScript not in a function definition is interpreted as it is loaded.
- JavaScript may also go in the body of a document.

## Event Processing

- Events are associated with objects
- Events have names

```
<input type="button" value="Button"
  onclick="alert('You clicked it');">
```

Button

## Event Processing

- Events are associated with objects
- Events have names

```
<input type="button" value="Button"
  onclick="alert('You clicked it');">
```

Button



## Event Processing

- Events are associated with objects
- Events have names

```
<input type="button" value="Button"
  onclick="alert('You clicked it');">
```

Button



## Using JavaScript in HTML

- Embedded in the document

```
<script type="text/javascript">
  ... statements
</script>
```

- From an external URL

```
<script type="text/javascript"
  src="myscripts.js">
</script>
```

## Using JavaScript in HTML

In an element's event declaration

```
<input type="button" value="Button"
  onclick="alert('You clicked it');">
```

## JavaScript is “Object-Based”

- Not really object-oriented
- Prototype-based inheritance
  - Instead of classes
  - So, no polymorphism
- Web documents and their elements are objects
- These objects can be manipulated using JavaScript.

## Properties and Methods

- Properties are the data; two types:
  - Primitives
  - References to other objects
- Primitives are scalar types like “number”
- Objects are accessed through variables  
`myCar.engine`  
Objects are lists of property/value pairs
- Methods are code bound to an object

## Comments

- Single-Line Comments  
`// This is a comment`
- Multi-Line Comments  
`/* So is...  
... this. */`


## Line Breaks

JavaScript statements may be broken across lines any place a space is allowed. Multiple spaces are OK.

```
if (a == b) c = d;  
if (a == b)  
    c = d;  
if (a ==  
b) c = d;
```

## Exception: String Literals

Line breaks are not allowed within string literals.

```
myStr = 'The rain in Spain';  
myStr = 'The rain in Spain stays  
mainly in the plain.';   
myStr = 'The rain in Spain stays';  
myStr += 'mainly in the plain.';
```

## Hiding JavaScript from Older Browsers

```
<script type="text/javascript">  
  <!--  
    ... statements  
  // -->  
</script>
```

## Hiding JavaScript from XML Parsers

```
<script type="text/javascript">
<!--
//
... JavaScript statements
//]]&gt;
// --&gt;
&lt;/script&gt;</pre></div><div data-bbox="135 247 310 262" data-label="Text"><p>(Just copy the example, OK?)</p></div><div data-bbox="195 274 458 311" data-label="Text"><p><i>&lt;INPUT ~~~~~<br/>ONCLICK = "ALERT( )" /&gt;</i></p></div><div data-bbox="629 130 788 147" data-label="Section-Header"><h2>JavaScript Syntax</h2></div><div data-bbox="554 146 845 255" data-label="List-Group"><ul><li>• JavaScript Identifiers:<ul><li>• Begin with a letter, underscore or dollar sign</li><li>• Are case-sensitive</li></ul></li><li>• Reserved words; about 60 (25 of them on p. 135)</li><li>• Predefined words (lots)<br/>alert, open, self, etc.</li><li>• End statements with a semicolon<br/>(instead of remembering the exceptions)</li></ul></div><div data-bbox="219 406 354 424" data-label="Section-Header"><h2>Primitive Types</h2></div><div data-bbox="132 423 375 438" data-label="Text"><p>The JavaScript primitive (scalar) types are:</p></div><div data-bbox="149 438 221 511" data-label="List-Group"><ul><li>• number</li><li>• string</li><li>• Boolean</li><li>• null</li><li>• undefined</li></ul></div><div data-bbox="664 406 750 422" data-label="Section-Header"><h2>Numbers</h2></div><div data-bbox="554 425 863 507" data-label="List-Group"><ul><li>• Internally, all numbers are double-precision floating point:<br/><i>72, 7.2, 7E2, 7.2E-2</i>, etc.</li><li>• Integer numeric literals can be expressed in hex<br/><i>0xff</i></li><li>• JavaScript coerces between numbers and strings</li></ul></div><div data-bbox="253 683 319 700" data-label="Section-Header"><h2>Strings</h2></div><div data-bbox="132 698 395 816" data-label="List-Group"><ul><li>• Zero or more characters</li><li>• Enclosed in quotes (") or apostrophes (')</li><li>• There is no difference.</li><li>• Quotes can be escaped<br/><i>'You\'re right!'</i></li><li>• Other escape sequences are available<br/><i>'You\'re right(\nWe should leave!'</i></li><li>• Backslash can be escaped<br/><i>\\d:\\bookfiles\\titles.txt</i></li></ul></div><div data-bbox="614 683 802 700" data-label="Section-Header"><h2>Other Primitive Types</h2></div><div data-bbox="554 699 787 743" data-label="List-Group"><ul><li>• Boolean: true or false</li><li>• Null: zero as number, false as Boolean</li><li>• Undefined: not assigned a value</li></ul></div><div data-bbox="943 951 968 969" data-label="Page-Footer"><p>5</p></div>
```

## Declaring Variables

- Implicit: just start using a variable name
- Explicit declaration:

```
var counter,  
    pi=3.14159,  
    flag=true;
```

Remember dynamic typing!
- Misspelled names act like new variable definitions.

## Numeric Operators

+ - \* / % ++ --  
If A is 7  
(++A)\* 3 is 24 (pre-increment)  
(A++)\* 3 is 21 (post-increment)

A is 8 after both

Don't worry (much) about precedence or associativity;  
things work like you expect

Parentheses can be used to force precedence.

## The Math and Number Objects

- Built-in objects; they are containers for useful methods and properties.

```
y=math.round(x);  
z=number.min_value;
```

- The toString method of number

```
strPrice=price.toString();
```

## Type Coercion (Implicit Conversion)

String concatenation operator: + (overloaded)

```
'August ' + 1997 == 'August 1997'
```

```
7 * '3' == 21  
7 * 'x' == NaN
```

## String Methods

```
length  
xyz = 'George';  
len=xyz.length;
```

charAt and indexOf

```
xyz.charAt(2) == 'o';  
xyz.indexOf('r') == 3
```

other string methods in textbook

## The typeof Operator

```
z=typeof x;  
z=typeof(x); } equivalent
```

Return values:

number  
string  
boolean  
object (including null variables)  
undefined

```
A = 'XYZ';  
X = typeof A;  
X will be 'STRING'
```

## Assignment

- Simple assignment: `=`  
`a=7;`
- Compound assignment  
`a+=1; // Now 8 if 7 before`
- Also `--`, `/=`, etc.

## Screen Output

- Document.write  
`document.write('This is XHTML<br />');`
- Output is HTML, rendered by browser
- Escaped characters like `\n` work; make source more readable but do not change display
- Multiple parameters are concatenated.

## Screen Output

- Alert: Displays an alert box  
`alert('Stop that!');`



Displays text, any markup is *not parsed!*

- Confirm: A dialog box with OK, Cancel
- Prompt: A text box with optional default

## Relational Operators

The usual suspects:

- `==` Equality (A single `=` is assignment; beware!)
- `!=` Inequality
- `<` Less than
- `>` Greater than
- `<=` Less than or equal
- `>=` Greater than or equal
- `===` Strictly equal (no type coercion)
- `!==` Not strictly equal

## Relational Operators

- `&&` and
- `||` or
- `!` not

These “short circuit” when evaluated;  
no expressions with side-effects!

## Control Flow

Böhm and Jacopini: sequence, selection and iteration

Selection:

```
if (a > b) x = y;
```

Compound statements: `{ }`

```
if (a > b) {  
    x=y;  
    b++;  
}
```

## Two Choices

Else used with if provides for selection among two choices.

```
if (a > b) {  
    // some stuff  
} else {  
    // other stuff  
}
```

## Several Choices: switch

Switch is a case statement

```
switch (creditCard) {  
    case 'mastercard':  
        // do mastercard stuff  
        break;  
    case 'visa':  
        // do visa stuff  
        break;  
    default:  
        alert('Unknown card!');  
}
```

## Iteration (Loops)

- Pre-test loops
- Post-test loops
- Counter-controlled loops

## Pre-test Loops

```
count = 0; // not part of loop  
while (count < 100) {  
    count++;  
}  
// next statement in sequence
```

Loop body is executed **zero or** more times.

## Post-test Loop

```
count = 0; // not part of loop  
do {  
    count++;  
} while (count < 100);
```

Loop body is executed **one** or more times.

## Counter-Controlled Loop

```
for (count=0; count <=10; count++){  
    // loop body  
}
```

Makes counter control explicit.

Can be simulated with “while” loop.