



JavaScript

- JavaScript is a lightweight interpreted Programming Language with some object-oriented capabilities
- Our interest in JavaScript is motivated by the fact that it has been embedded in the popular web browsers.
- Expelling Myths:

1. JavaScript is not related to Java (although there is a syntactic resemblance)

It is only a marketing ploy; sounds better than “ECMA Script”

(European Computer Manufacturers Assoc. Script)

2. JavaScript is not simple. It is a full-featured complex prog. Language that just happens to be easy for some tasks.
3. JavaScript is not just for browsers. There are sever-side versions as well as versions that can be embedded in larger blocks of code.



Client-side JavaScript

(executable content in www pages)

Use:

1. Dynamic content created by the client
2. Event handling by the client (eg. Responds to button pushes, mouse movement, keyboard, ...)
3. Access/Modify document components
4. Saving state information
5. Computations by the client

Limitation

1. No graphics
2. No file I/O
3. No network communications (other than browser control)

Examples of use:

<http://orfe.princeton.edu/courses/orf401/>

You name it.



Client-side JavaScript

(executable content in www pages)

Things to Note:

1. JS is case-sensitive (even though HTML isn't)
2. JS ignores spaces, tabs, newlines (ie. White spaces) except in strings
3. If JS statements are a simple line, the ";" can be omitted. **DON'T DO THIS!!!**
4. Comments: `/* */` or `//`
5. Identifiers **MUST** start with a letter or underscore or \$, then can contain digits.
6. Strings can be enclosed in either single or double quotes
7. The usual escape sequences can be used `\n` `\+` `\'` `\"`

Variables

1. Variables are untyped (as are functions):
This is legal: `j = 10;`
`j = "ten";` both in one program
2. Variables must be declared before they are use:
eg, `var j, sum;`
`var j = 7`
3. Variables defines in a function have local scope (and override/hide global variable with the same identifier)
4. Scope is not limited to blocks
So if you declare `var j`; in a for loop it is available elsewhere in the function



Client-side JavaScript

(executable content in www pages)

Functions:

1. Definition of Functions:

```
function fname (optional comma separated list of arguments)
{
    body of the function
    return return value (if there is something to be returned)
}
```

Example: function distance (x1,y1,x2,y2)

```
{
    var dx dy;
    dx = x2 - x1;
    dy = y2 - y1;
    return Math.sqrt (dx*dx +dy*dy);
}
```



Client-side JavaScript

(executable content in www pages)

Invoking Functions:

As you would expect:

e.g. `var d;`
 `d = distance (2,2,7,5);`

Functions can be passed / return either an array or a scalar

(e.g., useful when # of items purchased is one or more)

`typeof variable;` returns “number”, “string”, “boolean”, “object”, “function”,
“undefined”



Client-side JavaScript

(executable content in www pages)

Dynamic Functions:

One can also create un-named or anonymous function using the Function() constructor

```
e.g.      var f;  
          f = new Function ( "x1", "y1", "x2", "y2", "return  
          ..." );
```

variable # of arguments, last arg in the body of the function is returned

You can use this to create functions “on the fly” since the arguments of Function() are strings

Applications to eCommerce?

1. User enters a function in a form (e.g., discount schedule pricing rules)



Functions as Data:

Functions in JavaScript are data and can be treated that way

In particular, functions can be :

Assigned to variables

Stored in arrays

Passed to other functions

```
e.g.  var function  add (x,y){
      return  x + y;
      }
      function  doit (operator , operand1, operand2)
      return operator(operand1, operand2);
      }

      var a;
      var n;
      a = add;
      }
      n = doit (a, 7, 5);
```



Variable length argument lists:

Functions in JavaScript must be defined with a particular number of arguments (0 or more), but JavaScript doesn't check for agreement between the definition and the execution.

In addition, it keeps all of the arguments in an array called *arguments* so a function has access to them. This is very powerful!

```
e.g.      function max() {
            var i, m;
            m = Number.NEGATIVE_INFINITY;

            for (i=0, i,arguments.length; i++){
                if (arguments[i]>m) m=arguments[i];
            }

            return m;
        }
        function doit (operator , operand1, operand2)

        var largest = max(1, 13, 7, 22, 100, 5, 9);
```