Princeton University
COS 333: Advanced Programming Techniques
A Subset of JavaScript

Program Structure

```html
<html>
  ...
  <body>
    <script>
      function sqr(i)
      {
        var result;     // Otherwise result would be global.
        result = i * i; // Semicolons usually optional.
        return result;
      }
      document.write(sqr(5));
    </script>
    ...
  </body>
</html>
```

```html
<html>
  <head>
    <script>
      function sqr(i)
      {
        var result;      // Otherwise result would be global.
        result = i * i;  // Semicolons usually optional.
        return result;
      }
    </script>
  </head>
  <body onload="document.write(sqr(5)); ">
    ...
  </body>
</html>
```

```html
<html>
  <script src="http://someserver/somescript.js"></script>
  ...
</html>
```

Building and Running

Handled by browser.
Reserved Words (ECMAScript6)

break, case, catch, class, const, continue, debugger, default, delete, do, else, export, extends, finally, for, function, if, import, in, instanceof, new, return, super, switch, this, throw, try, typeof, var, void, while, with, yield

Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>1, 12345, 01, 012345, 0x1, 0x1DB5</td>
</tr>
<tr>
<td>Floating point</td>
<td>0.0, 1.23, 1.23e4</td>
</tr>
<tr>
<td>String</td>
<td>'hi', &quot;hi&quot;</td>
</tr>
<tr>
<td>Boolean</td>
<td>true, false</td>
</tr>
<tr>
<td>Null object</td>
<td>null</td>
</tr>
</tbody>
</table>

Explicit type conversions

- Integer: `parseInt(floatOrString)`
- Floating point: `parseFloat(intOrString)`
- Boolean: `Boolean(intOrFloatOrString)`
- String: `String(intOrFloatOrBoolean)`

Implicit type conversions

```
n = '123'; n *= 1; // Convert n to a number
n = 123; n += ''; // Convert n to a string
```

Operators

- Arithmetic: +, -, *, /, %, ++, --, unary -, unary +
- Assignment: =, *=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=
- Comparison: ==, !=, ===, !==, >, >=, <, <=,
- Logical: &&, ||, !
- String: +, +=
- Member: object.property, object["property"]
- Conditional: (condition ? IfTrue : ifFalse)
- Sequence: ,

Others: delete, function, get, in, instanceof, let, new, set, this, typeof, void, yield

Statements

Var
```
var variable;
```
Declares variable to be local to a function; otherwise it would be global

Compound
```
{
    statement
    statement; statement
}
```

Selection
```
if (expr)
    statement;
```
else
    statement;

switch (expr)
    case value1: statements; break;
    case value2: statements; break;
    ...
    default: statements;
}

Note: false == 0 == null == ' ' == ""; true == anything else

Iteration
    while (expr)
        statement;
    do
        statement;
    while expr;

    for (initialExpr; expr; finalExpr)
        statement;

    for (key in associativeArray)
        statement;

Note: false == 0 == null == ' ' == ""; true == anything else

break as in C

continue as in C

Exception Handling
    try
    {
        statements;
    }
    catch (exception)
    {
        document.write(exception.message);
    }
    throw 'someexception';

Function Definition
    function f(a, b, c) { statements; }

    function f()
    {
        for (i = 0; i < arguments.length; i++)
            ...arguments[i]...
    }

    return as in C
Function Call

\[ f(\text{expr}, \ldots); \]

Object references are passed by value

---

**Data Structures**

**Arrays**

colors1 = ['red', 'green', 'blue'];
colors2 = Array('orange', 'maroon', 'aqua');
for (var i = 0; i < colors2.length; i++)
    document.write(colors2[i]);

**Associative Arrays**
yanks = {'Ruth': 'RF', 'Gehrig': '1B', 'Mantle': 'CF', 'Jeter': 'SS'};
for (var yank in yanks)
    document.write(yank + ' = ' + yanks[yank]);

**Objects**

An object is an associative array

Each object created via MyClass has its own copy of each function:

```javascript
function MyClass(i)
{
    this.i = i
    this.get = function() { return this.i; }
    this.set = function(i) { this.i = i; }
}

obj1 = new MyClass(5);
i = obj1.get();
obj2.set(6);
```

All objects created via AnotherClass share copies of a single set of functions:

```javascript
function AnotherClass(i)
{
    this.i = i;
}
AnotherClass.prototype.get = function() { return this.i; }
AnotherClass.prototype.set = function(i) { this.i = i; }

obj2 = new AnotherClass(5);
i = obj2.get();
obj2.set(6);
```

---

**Standard Built-In Objects**

**Value properties:** Return a simple value

- Infinity, NaN, undefined, null literal

**Function properties:** Functions that are called globally rather than on an object

- eval(), uneval(), isFinite(), isNaN(), parseFloat(), parseInt(), decodeURI(),...
decodeURIComponent(), encodeURI(), encodeURIComponent()

**Fundamental objects**: Basic objects upon which all other objects are based

Object, Function, Boolean, Symbol, Error, EvalError, InternalError, RangeError, ReferenceError, SyntaxError, TypeError, URIError

**Numbers and dates**: Objects that represent numbers, dates, and mathematical calculations

Number, Math, Date

**Text processing**: Objects that represent strings and support manipulating them

String, RegExp

And others

---

**DOM Objects**

**window object**

Represents the browser window. Contains a reference to a `document` object.

Event handlers: onabort, onbeforeunload, onblur, onchange, oncontextmenu, ondragdrop, onerror, onfocus, onhashchange, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onresize, onscroll, onselect, onsubmit, onunload, onpageshow, onpagehide

**document object**

Represents the current document. Contains references to `element` objects.

Event handlers: ononline, onoffline, onreadystatechange

**element objects**

Each represents an HTML element. Each contains references to attributes and child elements.

Event handlers: onafterscriptexecute, onbeforescriptexecute, oncopy, oncut, onpaste, onbeforeunload, onblur, onchange, onclick, oncontextmenu, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onresize, onscroll

---

**Debugging**

In Firefox: Tools → Web Developer → Browser Console

In Chrome: More tools → Developer Tools

**Etc.**

We'll cover other features of JavaScript throughout the course as necessary.

---

Copyright © 2017 by Robert M. Dondero, Jr.