Program Structure

```html
<html>
...  
<body>
  <script>
    function sqr(i)
    {
      let result;
      result = i * i; // Semicolons usually are optional.
      return result;
    }
    document.write(sqr(5));
  </script>
...  
</body>
</html>
```

-----------------------------------------------------------------------------------

```html
<html>
...  
<body onload="document.write(sqr(5)); ">
  ... 
  <script>
    function sqr(i)
    {
      let result = i * i;
      return result;
    }
  </script>
</body>
</html>
```

-----------------------------------------------------------------------------------

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

Building and Running

Handled by browser.
Reserved Words

abstract, arguments, await, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, eval, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, let, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with, yield

Primitive Data Types

Number: 1, 12345, 01, 012345, 0x1, 0x1DB5, 0.0, 1.23, 1.23e4
All numbers are represented internally as floating point
String: 'hi', "hi"
Boolean: true, false
Null object: null

Explicit type conversions

Number: parseInt(someObject)
Number: parseFloat(someObject)
Number: Number(someObject)
Boolean: Boolean(someObject)
String: String(someObject)

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: =, +=, /=, %=, <<=, >>=, >>>=, &=, ^=, |=
Bitwise: &, |, ^, ~, <<, >>, >>>
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

Let

let somevar;
let somevar = someValue;
Declares variable to be local to a function; otherwise it would be global

Const

const SOMECONST = someValue;
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 == 0.0 == 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 == 0.0 == 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 (let i = 0; i < arguments.length; i++)
        ...arguments[i]...
}

return as in C

A function is a subtype of object; it can have properties

Function Call

\( f(expr, \ldots); \)

Object references are passed by value

Data Structures

Arrays

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

Associative Arrays

yankPositions =
    {'Ruth': 'RF', 'Gehrig': '1B', 'Mantle': 'CF', 'Jeter': 'SS'};
// yankPositions =
console.log(yankPositions['Ruth']);
console.log(yankPositions.Ruth);
for (var yank in yankPositions)
    console.log(yank + ' = ' + yankPositions[yank]);

Objects

An object is an associative array

class MyClass
{
    constructor(i) { this.i = i; }
    get() { return i; }
    set(i) { this.i = i; }
}

myObj = new MyClass(5);
console.log(myObj.get());
myObj.set(6);
console.log(myObj.i);
console.log(myObj['i']);

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, onclick, onclose, 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 children elements.

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

### Debugging

In Firefox: Tools → Web Developer → Web Console

In Chrome: More Tools → Developer Tools → Console

### Etc.

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

Copyright © 2019 by Robert M. Dondero, Jr.