CS320 Web and Internet Programming
JavaScript

Chengyu Sun
California State University, Los Angeles

"Hello World" in JavaScript

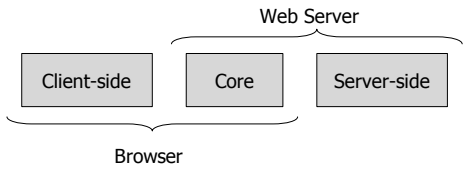
```
<html><title><head>hello world example</title></head>
<body>
        <script type="text/javascript">
                document.write("hello world!");
        </script>
</body>
</html>
```

<script> Tag

- `type="text/javascript"`, or `language="JavaScript"`
- `src="path_to_your_script"`

JavaScript

- Interpreted language
- Developed by Netscape
- Syntax is similar to Java



Core JavaScript

(For those who already know Java)

Lexical Elements

- Identifiers
  - i, _i, $i
- Literals
  - 123, 12.3, 1.2e3
  - "hello world", 'hello world'
  - true, false
  - /javascript/gi
  - null
- Statements
  - Semicolons are optional
- Comments
  - //, /* */

```
a = 1
b = 2

a = 1; b = 2

return
true;
```

Data Types

- Primitive types
  - Number
  - Boolean
  - String
- Object types
  - Array
  - Function
  - Some pre-defined object types
    - Wrapper objects, Date, RegExp

Variables

- Untyped, weak-typed, dynamically typed

```
i = 10;          // declaration
i = "ten";       var a;
```

Variable Scope

- Global and local
  - Shadowing
- No block scope

```
var scope="global";
function foo() {
    var scope="local";
    document.write(scope);
}
```

```
var scope="global";
function foo() {
    alert(scope);
    var scope="local";
    alert(scope);
}
```

```
var scope="global";
function foo() {
    scope="local";
    document.write(scope);
}
```

Operators

- Arithmetic
  - +, -, *, /, %
  - ++, --
- Relational
  - >, <, >=, <=, ==, !=
  - ===, !==
  - in
- Logical
  - &&, ||, !
- Bitwise
  - ~, >>, <<, &, !, ^
- Assignment
  - =
  - +=, -=, *=, /=, %=
- Conditional
  - ?:

Comparison

- Primitive types are compared by value
- Object types are compared by reference
- === returns true only if two operands are of the same type, and they have the same value (or reference)
- == may return true even if two operands are of different types

```
"12" == 12        1 == true
"1" == true       0 == false
```

Control Statements

- if/else
- switch, case, default
- for, for/in
- while, do/while
- break, continue
- try, catch, finally

```
var point = { x:20, y:10 };
for( var coord in point )
  document.write( coord + ": " + point[coord] + "<br>" );
```

Functions

- `function`
- No return type
- No argument type
- Functions as data

```
function factorial(n)
{
   var f = 1;
   for( var i=1 ; i <= n ; ++i ) f *= i;

   return f;
}
```

```
var f1 = factorial;
var f2 = function(msg) { document.write(msg+"<br>"); }
```

Function Arguments

```
function max()
{
   var m = arguments[0];

   for( var i=1 ; i < arguments.length ; ++i )
      if( arguments[i] > m ) m = arguments[i];

   return m;
}
```

Objects

- No class definition

```
var point = new Object(); // create an empty object

/* specify properties (fields) and their values */
point.x = 2;
point.y = 3;
```

Constructors

- A function with `this`

```
function Point( x, y)
{
   this.x = x;
   this.y = y;
}

var p1 = new Point( 1, 1 );
var p2 = new Point( 2, 2 );
```

Methods

```
function distance( p )
{
   return Math.sqrt(  (this.x-p.x)*(this.x-p.x) + (this.y-p.y)*(this.y-p.y) );
}

function Point(x, y)
{
   this.x = x;
   this.y = y;
   this.distance = distance;
}

                var p1 = new Point(1,1);
                var p2 = new Point(2,2);
                document.write( p1.distance(p2) )
```

Prototype and Inheritance

```
function distance( p )
{
   return Math.sqrt(  (this.x-p.x)*(this.x-p.x) + (this.y-p.y)*(this.y-p.y) );
}

function Point(x, y)
{
   this.x = x;
   this.y = y;
}

Point.prototype.distance = distance;

                var p1 = new Point(1,1);
                var p2 = new Point(2,2);
                document.write( p1.distance(p2) )
```

Objects as Associative Array

- p1.x, p1.y, p1.distance(p2)
- p1["x"], p1["y"], p1["distance"](p2)

Array

- var a = new Array();
  - a[0] = 1;
  - a[1] = 2;
  - a[3] = "x";
  - a[4] = true;
- var a =new Array(1, 2, "x", true);
- var a = [1, 2, "x", true];

Array Length

- `array_name.length`
- Length of an array may change

```
var a = new Array(); // a.length == 0

a[0] = 1; // a.length == 1
a[50] = 10000; // a.length == 51
```

Array Methods

- join(), concat()
- sort(), reverse()
- slice(), splice()
- push(), pop(), shift(), unshift()

Regular Expressions

- General patterns
  - abc
  - a|b|c
  - ab^*c
  - ab^+c
  - [a-z]
  - [a-zA-Z0-9]
  - $a[a-z]^*[0-9]^+c$

JavaScript Regular Expressions

- var p = /*pattern*/;
- var p = new RegExp("*pattern*");

Pattern Symbols

- a,b,C,1,2,3
- [a|b]
- [abc]
- [a-z], [A-Z], [0-9]
- [^a-zA-Z0-9]
- \w – word char
- \W – non-word char
- \s – white space char
- \S – non-white-space char
- \d – any digit
- \D – any non-digit

Repetition

- * -- 0 or more occurrences
- + -- 1 or more occurrences
- ? – 0 or 1 occurrence
- {n} – n occurrences
- {n,} – n or more occurrences
- {n,m} – at least n and no more than m occurrences

Positions and Flags

- ^ -- beginning of a string
- $ -- end of a string
- \b – word boundary
- \B – not word boundary
- */pattern/flag(s)*
  - i – case insensitive
  - g – global match
  - m – multiple line mode

String Methods for Pattern Matching

- search(*regexp*)
  - index of the 1st match
  - -1 if no match
- replace(*regexp*, *string*)
- match(*regexp*)
  - array of strings
  - null if no match

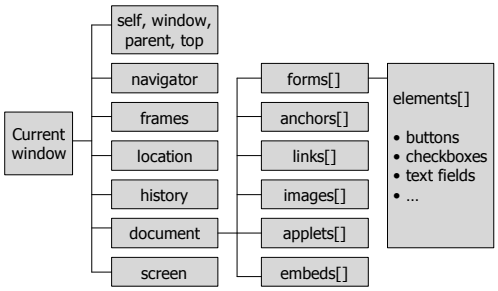
Client-side JavaScript

Window, Document, and Forms

Global Object

- Created by a JavaScript interpreter
- All the global methods and variables are properties of this object
- Client-side – `window`
  - All things related to a browser window

Window



Window Properties

- self, window, parent, top
- location
- alert(), confirm(), prompt()

Document

- open(), close(), write(), writeln()
- title, URL, forms[], links[], images[]

Forms and Form Object

```
<form name="sample" method="get">
        username:<input type=text name=user><br>
        password:<input type=password name=pass><br>
        <input type=submit name=register value="Register">
</form>
```

document.forms[].elements[]

Event Handlers

- `onclick`, `onchange`, `onfocus`, `onblur`
- `onsubmit`, `onreset`
- Attributes of a form element
- Properties of a form object

Form Element Attribute

Username: <input type=text name=user
onchange=some_javascript_funtion>

Form Object Properites

```
var e = document.forms[0].elements[0];
e.onchange = some_javascript_function;
```