Why Discuss JavaScript?

- JavaScript is very widely used and growing
- Any AJAX application heavily relies on JavaScript
- JavaScript also has interesting language trade-offs
- You can think of JavaScript as a hybrid language with features from almost everywhere glued together

JavaScript Target

- Every language has a design target:
  - C: Systems programming
  - Java: Set-top box
  - JavaScript: Web scripting
- Every language modifies some abstract data structure
- In JavaScript, this is the document object model of an HTML web page

What’s a Scripting Language?

- Answer: One language embedded in another
- More specifically, a scripting language is used to write programs that produce inputs to another language processor
- Examples:
  - Embedded JavaScript produces HTML to be displayed by the browser
  - Shell Scripts compute commands executed by the shell
- Common characteristics of scripting languages:
  - Lots of string support
  - Simple structure with little/no declarations
  - Flexibility preferred over efficiency, safety, common sense

JavaScript History

- Developed by Brendan Eich at Netscape in 1995 as scripting language for Navigator 2
- Later standardized for browser compatibility, called ECMAScript
- Renamed to JavaScript in part of marketing deal with Sun - no relation to Java!
- Today: Many implementations available

Motivation for JavaScript

- Netscape, 1995
- Has >90% browser market share
- Opportunity to define the HTML scripting language
- Brendan Eich: “I hacked the JS prototype in 1 week in May, and it showed! Mistakes were frozen early. Rest of the yer spend embedding in browser and cursing my design”
- Initial uses of JavaScript: Form validation, page effects, dynamic content manipulation
- More recently: Web 2.0: Significant functionality implemented on web client
- Examples: Google Docs, Gmail, etc
### JavaScript Design Goals

- Make it easy to copy/paste code
- Tolerate minor errors (missing semicolon)
- Simplified even handling, e.g., `onClick`, `onMouseDown`, … inspired by HyperCard
- Full features that make it easy to write and modify code that does something from all other languages

### JavaScript Design

- Functions based on LISP/Scheme
- We have higher order functions, lambda, etc
- Objects in JavaScript are based on Smalltalk/Self
  ```javascript
  var pt = {x: 10, move:function(dx){this.x+=dx}}
  ```
- But lots of “issues”
- Douglas Crockford: “In JavaScript, there is a beautiful, elegant, highly expressive language that is buried under a steaming pile of good intentions and blunders”

### Language Syntax

- JavaScript is **case sensitive**
- But HTML is not case sensitive, so any HTML object in JavaScript is also not
- Example: `onClick` vs. `ONCLICK`
- Statements are terminated by returns or semi-colons
- JavaScript has blocks using `{ }`, but no separate scope!

### Variables

- You define variables using the `var` statement
- But no declarations; variables are implicitly defined by their first use, which must be an assignment.
- **Note:** Implicit definition has global scope, even if it occurs in nested scope
  ```javascript
  {  
    var x = "123"  
  }  
  return x;  
  ```
  //will return "123"

### Stand-alone JavaScript

- You can use the Rhino commend-line interpreter to play with JavaScript without a website
- Rhino has the same read-eval-print loop we have already seen in the LISP interpreter
- Play with it!

### JavaScript in the Browser

- Most of the time JavaScript is used in the browser to manipulate a web page
- **Main reason it is used:** Only kind of program that anyone can run in any browser and expect to function
- This is the main reason JavaScript is popular
Web Example: Page Manipulation

```html
<script type="text/JavaScript">
    function whichButton(event) {
        if (event.button==1) {
            alert("You clicked the left mouse button!");
        } else {
            alert("You clicked the right mouse button!");
        }
    }

    <body onmousedown="whichButton(event)">
        <script>
        ...
    </body>
</script>

Other events: onLoad, onMouseMove, onKeyPress, onUnload
```

Primitive Data Types

- **Boolean**: true and false
- **Numbers**:
  - 64-bit floating point
  - No integer type!
  - Special value NaN and Infinity
- **Strings using Unicode characters**
- **Special values null, undefined**

JavaScript Functions

- Declarations can appear in function body, allowing for local variables and inner functions
- **Parameter passing**:
  - Basic types by value
  - Objects by reference
- You can supply any number of arguments
  - `fun.length`: number of arguments in definition
  - `fun.arguments.length`: number of arguments in call
- Anonymous (lambda) functions: `(function (x,y) {return x+y}) (2,3);`

Function Examples

- **Curried function**
  ```javascript
  function CurriedAdd(x) {
    return function(y) { return x+y; };
  }
  g = CurriedAdd(2);
  g(3)
  ```
- **Variable number of arguments**
  ```javascript
  function sumAll() { 
    var total=0;
    for (var i=0; i<sumAll.arguments.length; i++)
      total+=sumAll.arguments[i];
    return(total);
  }
  sumAll(3,5,3,5,3,2,6)
  ```

Use of Anonymous Functions

- Anonymous functions very useful for callbacks
  ```javascript
  setTimeout(function(){ alert("done"); }, 10000)
  // putting alert("done") in function delays evaluation until call
  ```
- Simulate blocks by function definition and call
  ```javascript
  var u = {a:1, b:2}
  var v = {x:3, b:4}
  (function (x,y) { 
    var tempA = x.a; var tempB = x.b; 
    // local variables
    x.a=x.a; x.b=y.b;
    y.a=tempA; y.b=tempB
  })(u,v)
  ```
  // Side effects on u,v because objects are passed by reference

Objects

- In JavaScript, an object is nothing but a collection of named properties
- Can think of it almost like a hash table or associative array
- Defined by a set of name:value pairs:
  ```javascript
  objDuck = { name:"Quak", gender:"male" }
  ```
- New properties can be added **at any time**:
  ```javascript
  objDuck.species = "mallard"
  ```
- Can have methods, can refer to this
Basic Object Features

- Use a function to construct an object
  ```javascript
  function car(make, model, year) {
    this.make = make;
    this.model = model;
    this.year = year;
  }
  ```
- Objects have prototypes, can be changed
  ```javascript
  var c = new car("Tesla","5",2012);
  car.prototype.print = function () {
    return this.year + " " + this.make + " " + this.model;
  }
  c.print();
  ```

Objects and this

- The this variable is a property of the activation object for a function call
- In most cases, this points to the object which has the function as property (or method)
- Example:
  ```javascript
  var o = {x:10, f:function() { return this.x}}
  o.f()
  ```
  This will evaluate to 10

JavaScript Functions and this

- Local Variables stored in “Scope Object”
  ```javascript
  var x = 5; var y = 5;
  function f() { return this.x + y;}
  var o1 = {x : 10};
  var o2 = {x : 20};
  o1.f = f; o2.g = f;
  o1.g();
  15
  o2.g();
  25
  ```
  Both o1.g and o2.g refer to the same function.
  Why are the results for o1.g() and o2.g() different?

Concurrency

- Javascript is single-threaded
- However, AJAX model allows for some hacked-up asynchronous callback mechanism using XMLHttpRequest
- Widely used, but sad and pathetic hack
- Another form of concurrency: Use setTimeout for cooperative multitasking

Unusual features of JavaScript

- Built-in regular expressions
- Add, delete methods of objects dynamically
- Redefine native functions and objects
- Iterate over methods of an object:
  ```javascript
  for (variable in object) { statement }
  ```
JavaScript Overall

- JavaScript is in many ways the worst of all features combined in one language
- The language does everything possible to allow unreadable and buggy code
- Dynamic features make an performant interpreter extremely difficult
- And yet: JavaScript is one of the most widely-used languages!