JavaScript:
Introduction, Types

Lecture 21
History

- Developed by Netscape
  - "LiveScript", then renamed "JavaScript"
  - *Nothing* to do with Java!
- Interpreted
- Browser-based, client-side execution
- Standardized by ECMA ("ECMAScript")
  - But no one calls it that!
  - MIME type text/javascript
  - *cf.*, JScript (MS)
- Becoming popular outside of browsers
  - *e.g.* Node.js
- Translation target for other languages:
  - Syntax: CoffeeScript
  - Static types: Dart (Google), TypeScript (MS)
Client-Side Execution

GET /news/index.php HTTP/1.1
Host: www.osu.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu;...etc

<!DOCTYPE html>
<html lang="en">
  <head><title>My Page</title><meta charset="utf-8" />
  ...
Client-Side Execution

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Something Short and Sweet</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Hello <a href="planet.html">World</a>!
      <br />
      <img src="globe.png" alt="a globe"/>
    </p>
  </body>
</html>
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Something Short and Sweet</title>
        <meta charset="utf-8"/>
        <script>
            window.alert("Annoying!");
        </script>
    </head>
    <body>
        <p>
            Hello <a href="planet.html">World</a>!
        </p>
        <img src="globe.png" alt="a globe"/>
    </body>
</html>
Including Scripts

- Head: executed **before** body displays
  - Script (source) can be explicitly included
    ```html
    <script type="text/javascript">
      //default type in HTML 5!
      console.info("hi");
      ...
    </script>
    ```
  - Script can be linked in from external file
    ```html
    <script src="MyProgram.js"></script>
    ```
  - Recall: linking to CSS

- Inline: executed as body is displayed
- Browser blocks while downloading
  - Common advice: put scripts at *end of body*
  - Modern advice: use `<script src="..." async>`
Async/defer Downloading

defer
async

parser  fetch  execution
Demo

- Simple "hello world"
  - HTML file containing JavaScript
  - Body is empty, script writes HTML output
  - Browser displays result

- Examining result
  - "view source": see JavaScript program
  - Inspector tab: see rendered HTML
Some Objects Provided Implicitly

- Some objects are created implicitly by the execution environment (browser)
- **Document object** (`document`)
  - `document.writeln()` puts output in body
- **Window object** (`window`)
  - Refers to browser's display window
  - Alert method pops up a dialogue
    ```javascript
    window.alert("Say "\"cheese\"!");
    ```
  - Prompt method pops up a dialogue
    ```javascript
    name = window.prompt("Enter name");
    ```
Demo with Popups

- See: [codepen.io/cse3901/pen/BYqqPb](https://codepen.io/cse3901/pen/BYqqPb)
  - Alert window
  - Prompt window
  - Console output (info, warn, error)
- See example on class meetings page
Familiar (Java) Minor Syntax

- Statement separator ;
  - Wrinkle: ;'s are optional!
    - Implicitly automatically inserted
    - But clearer and safer to include explicitly
- Statement blocks {...}
- Parentheses in expressions (...)
- Comments // and /*...*/
Familiar (Java) Operators

- **Arithmetic (numbers are floats)**
  - + - * / %
  - Wrinkles:
    - No diff in / between ints and floats!
    - % works on floats!

- **Relational**
  - < > <= >=
  - == !=
  - Wrinkle: === !==

- **Logical**
  - && || !
Familiar (Java) Statements

- **Assignment**
  - =
  - += -= *= /= %=
  - ++ -- (pre and post)

- **Conditionals**
  - if (...), if (...) ... else
  - switch (c)
    - case 'a': ... case 'b': ... default;

- **Iteration**
  - while (...), do...while(...)
  - for (...;...;...)
  - break, continue
Primitive vs Reference Types

- Distinction is similar to Java
- A variable is a "slot" in memory
- A variable can be primitive
  - The slot holds the value itself
  - Boolean, number, string, (null, undefined)
  - New in ECMAScript 2015 (ES6): symbols
- A variable can be a reference
  - The slot holds a pointer to the value
  - Arrays and objects (including functions!)
Primitive vs Reference Types

- a: 34.2
- b: "hi"
- c: 4
- d: width: 12, height: 15, color: "blue"
Primitives: Checking Equality

```javascript
var a = 5;
var b = 5;
var c = 7;

if (a == b) //=>true, slots are equal
if (a == c) //=>false

var x = "hello";
var y = "hello";

if (x == y) //=>true! c.f. Java
```
Primitives: Assignment is Copy

```javascript
var a = 5;
var b = a; // copy contents of slot

b++;

if (a == 5) // => true, a is unchanged
```
Assignment is Copy (of Slot)

```javascript
var a = 5;
var b = a;

b++; if (a == 5)

if (a == 5)
```
Primitives: Argument Passing

```javascript
function inc (param) {
    param++;
}

var a = 5;
inc(a); //copies contents of slot
if (a == 5) //=>true
```
References: Equality/Assignment

```javascript
var a = {x:1, y:4}; //a new object
var b = {x:1, y:4}; //a new object

if (a == b) //=>false slots unequal

a = b; //copy contents of slot

if (a == b) //=>true
```
Assignment is Copy (of Slot)

\[
a = b;
\]

a != b

a == b
function inc (param) {
    param.x++;  
}

var a = {x: 1, y: 4};
inc(a);  //copy contents of slot
if (a.x == 2)  //=>true
References: Argument Passing

```javascript
function inc (param) {
    param = {x: 2, y: 7};
}

var a = {x: 1, y: 4};
inc(a); //copies contents of slot
if (a.x == 2) //=>false
```
Wrinkle: == vs ===

- Recall + operator in Java
  - Concatenation between strings
  - Addition between numbers
  - 3 + "4" also works! Results in "34"

- Similarly, JavaScript == (!=) tries to make types match
  - 3 == "3" is true!

- To prevent implicit type conversion, use === (!===)
  - 3 === "3" is false

- More on type conversion later...
Demo: Iteration

- See: codepen.io/cse3901/pen/Jpmejp
- Table generated by Javascript
  - Prompt for initial value
  - Calculate interest series
  - Print out a row of table for each year
Static vs Dynamic Types

- **Static**: known at compile time
  - *e.g.*, C, C++, Java, Ada
    - `int x`
    - `char[] a`
    - `FluffyCloud t`
    - `void* d`

- **Dynamic**: known only at run time
  - *e.g.*, Python, PHP, Ruby, JavaScript
    - `var x`
    - `var a`
    - `var t`
    - `var d`
Static Types

- a: 34.2 (number)
- b: "hi" (string)
- c: num[]
- d: Shape

Number:
- 4
- 0
- -300
- 3.14

Shape Properties:
- width: 12
- height: 15
- color: "blue"
Dynamic Types

- a: var, 34.2
- b: var, "hi"
- c: var
  - 4
  - 0
  - -300
  - 3.14
- d: var
  - width: 12
  - height: 15
  - color: "blue"

[]
Function Signatures

- **Statically typed**

  ```java
  String parse(char[] s, int i) { ... return e;}
  out = parse(t, x);
  ```

  - Parameter types (*i.e.* s and i) are declared
  - Return type (*i.e.* of parse) is declared
  - The *compiler* checks conformance of
    - (Declared) types of arguments (t, x)
    - (Declared) type of return expression (e)
    - (Declared) type of expression using parse (out)

- **Dynamically typed**

  ```java
  function parse(s, i) { ... }
  out = parse(t, x)
  ```

  - You are on your own!
Changing Types at Run-time

**Static Types**
```java
String a;
//a is undefined
a = "hi;";
//a is null string
a = "hi";
//compile-time err
a = 3;
//compile-time err
a.push();
//compile-time err
```

**Dynamic Types**
```javascript
var a;
//a is undeclared
a = "hi;";
//a is undefined
a = "hi";
//load-time error
a = "hi";
//a is a number
a = 3;
//a is a number
a.push();
//run-time error
```
Summary

- Executes at client-side, in browser
  - Interpreted (not compiled)
- Basic syntax: operators, statements
- Objects: document, window...
- Types
  - Primitives: boolean, number, string, null, undefined
  - References: arrays, objects (& functions)
- Working with primitives and references
  - Checking equality
  - Assignment
  - Parameter passing
- Dynamic types (vs static types)