JavaScript

Values, Types, and Operators

What is JavaScript?

- A high-level, dynamically typed, interpreted language
  - ECMA Script: scripting-language specification based on JavaScript
- Developed for Netscape to allow for more dynamic content
- Used in conjunction with HTML and CSS for web content, DOM manipulation on client side
- AJAX
- Used also for games, desktop and server-side applications
  - Node.js

What is JavaScript? (Cont')

- JavaScript vs Java
  - Java is *compiled*, JavaScript is *interpreted*
  - JavaScript is *dynamically typed*, Java is *strongly typed*
  - JavaScript lacks I/O (relies on environment)
  - Both support structured programming of C
  - JavaScript, as an object oriented language, is *prototype-based*
  - Functions are values of type 'Function'

Installing Node.js

- Node.js is a runtime environment. We will use it to do different things, the first of which is to run javascript code without a browser.
- Node.js is based on Chrome's V8 JavaScript engine
- See link on course webpage to download and install Node.js
  - Windows and Mac, just download the installer, click, go through wizard.
  - To get a Node prompt, open your Command Prompt or Terminal, type `node`
  - To exit, type `.exit`

Using JavaScript as a Calculator (learning about values, arithmetic, operators, etc.)

Below the surface of the machine, the program moves. Without effort, it expands and contracts. In great harmony, electrons scatter and regroup. The forms on the monitor are but ripples on the water. The essence stays invisibly below.

Master Yuan-Ma, The Book of Programming

Values

- For a computer, there is only data, expressed as long sequences of bits
- The bits must be separated into chunks to represent information: the chunks are *values*
- Each value plays a role, that is, has a type:
  - Numbers, strings, booleans, objects, functions, and undefined
- Invoke values and they are created automatically
- When values are used, its bits are recycled

Numbers

- Just type a number, and its bit pattern is in memory
- 64 bits are used to store a number: 2^64 different values
- Need to distinguish +/-, floating point, total range is 9 quadrillion
- For floating point numbers, can use dot: `9.7`
- Can use scientific notation: `2.33e10`
- Calculations with integers are precise, but not so with floats

Arithmetic and Special Numbers

- Add, subtract, multiply, divide, and remainder (modulo), are the same as they are in Java. Parentheses work the same way
  - Practice some using Node's interpreter
- Special numbers
  - `Infinity, -Infinity, NaN`

Strings (some very basic things)

- Represent text
- Escape special characters to include in a string (same as Java)
  - Some special chars: \t, \n,
  - To escape: `\\t, \\n, \", \',` etc. by adding another back-slash
- Concatenation: + (same as Java)

Unary Operators, Boolean Operators, Comparisons, Logical Operators, Grouping symbols (all same as Java)

- `typeof`
- `-`
- `true`
- `false`
- `>, <, >=, <=, ==, !=`
  - Can use comparators for characters, `a<b`
- `&&, ||, !`
- `()`
- Order of precedence: `(), !,` Comparison, `&&, ||`

Ternary Operators, Undefined Values

- Conditional Operator
  - `true ? 1 : 2`
- Undefined Values (lack of a meaningful value)
  - `null`
  - `undefined`

Automatic Type Conversion (Coercion)

FYI, consider:

```
console.log(8 * null)
// → 0
console.log("5" - 1)
// → 4
console.log("5" + 1)
// → 51
console.log("five" * 2)
// → NaN
console.log(false == 0)
// → true
```

Type Coercion Con't

- Using == when types differ involves a complex set of rules.
- Most of the time it just tries to convert one value's type to the other's
- An exception is `null`, which returns false unless compared with `null` or `undefined`
- `0==false, ""==false`
- If you don't want type coercion, use === or `!==`

Short-Circuiting Logical Operators

- `&&` returns the value on the left if it is false, right if it true
- `||` returns the value on its left if true, right if false
  - `5<4 || true`
  - `true && 2==2`
  - `false && 2==2`

Some more basics

Expressions vs Statements

- Expressions: a fragment of code that produces a value
- Expressions can nest: `(6>7)&&(1==1)`
- Statements:
- The smallest standalone element of an imperative programming language.
  - Statements *do something*
  - A program is a list of statements
  - Statements are usually followed by a semicolon:
    - `1; alert("text");`

Variables

Variables do not contain values; they grasp them

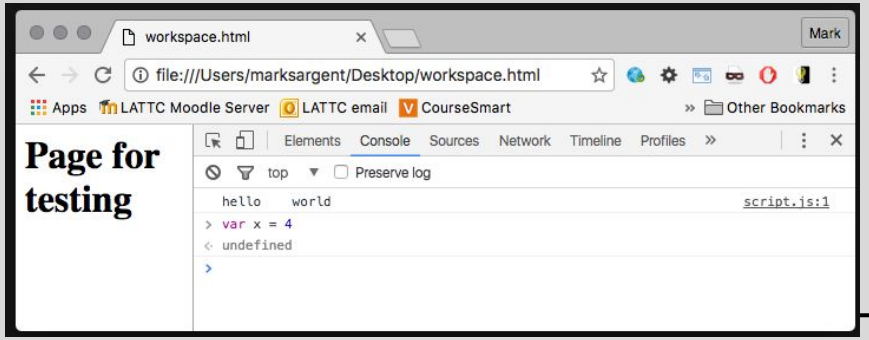
- Used to keep track of a program's internal state
- Use the `var` keyword to declare them
  - `var ten = 10;`
- Multiple variables can grasp, or *reference* the same value or object
- JavaScript is *pass by value*, or, *value by reference*, just like Java
  - We'll worry more about all this later
- Environment: the collection of variables and their values
  - It is never empty, even at startup it contains language standard variables

Keywords, Reserved Words

```
break case catch class const continue
debugger
default delete do else enum export extends
false finally for function if implements
import in instanceof interface let new null
package private protected public return
static super switch this throw true try
typeof var void while with yield
```

Using Chrome's Console

To run browser functions, we need a console attached to a browser. Press F12 (Windows) or Command-Option-I

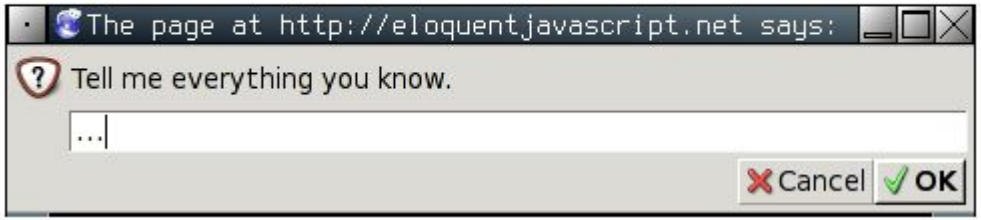


Calling Functions

- Type function: a piece of program wrapped in a value
  - *Apply* the value to run the wrapped program
    - E.g., `alert("Hello!");`
  - Executing a method is called *invoking*, *calling*, or *applying* it
  - Values given to functions are called *arguments*
  - Functions may take more than one value, or take no values
- console.log()
  - Writes the text argument to the console
- Return values: functions return values
- Side effects: things the function does to the state of the program

Prompt and Confirm

```
prompt("Tell me everything you know.", "...");
```




```
confirm("Shall we, then?");
```

Include a Script in an HTML Page

- Write a script:
  - Open a text editor, save it as type .js, and write your code in it
- In an HTML document, in the <head> section:

```
<script src="script.js"></script>
```