INF5750

Introduction to JavaScript and Node.js

Outline

- Introduction to JavaScript
- Language basics
- Introduction to Node.js
- Tips and tools for working with JS and Node.js

What is JavaScript?

- Built as scripting language for adding dynamic features to Web pages
- Originally developed for Netscape Navigator in 1995
- Reverse engineered by MS in 1996 for IE as JScript
- Submitted to ECMA for standardisation - first language standard released in 1997 as ECMA-262 (ECMAScript)
- ECMAScript is the specification - JavaScript is an Oracle trademark

How is JS used?

- Client-side (e.g. browser)
  - Scripting language for web pages
  - 94,7% of web sites use JS (w3techs.com)
- Server-side (e.g. node.js)
  - Fastest growing (but still tiny) server-side language for web sites
- Other:
  - Browser extensions, mobile apps (Cordova), scripting applications (Adobe CS, OpenOffice++)

JavaScript Engines

- Support for (new) language features varies slightly
- Multiple javascript engines/interpreters exists:
  - V8 (Chrome and Node.js)
  - Chakra (Edge)
  - SpiderMonkey (Firefox)
  - JavaScriptCore (Safari)
  - +++

Key Features

- High-level
- Weakly typed
- Object-oriented
- Interpreted

High level

- High level of abstraction from the hardware
- No need to deal with memory etc
- Easier to develop and use across different platforms

Weakly typed

- Variable types are derived from the values they are assigned
- No type safety or type checking
- Built-in types still exist («String», «Number» etc)

Object oriented

- JS is object oriented, but prototype-based - no real *class* concept with inheritance etc
- All objects inherit from a prototype object
- More later (and next week)

Interpreted

- Not pre-compiled to byte or machine code
- Code is interpreted and executed by the JS engine at runtime

Java and Javascript are similar like Car and Carpet are similar

– Chris Heilmann

Basic language constructs

- Values and types
- Conditionals
- Functions
- Variables

- Prototypes [more next week]

Values and types

- JS uses weak/dynamic typing - flexible, but potentially more error-prone

- A variable can hold values of different types

Values and types

- Built-in types:
  - boolean
  - number
  - string
  - null and undefined
  - object
  - symbol
- Built-in types have properties and methods, e.g "hello world».toUpper()

Values and types

- The **object** type represent compound values

```
var anObject = {
    firstName: "John",
    lastName: "Doe"
}
```

- **Arrays** and **functions** are special forms of objects

Values and types

- Comparing different value types requires type conversion:
  - implicit conversion (coercion)
  - explicit conversion
- false values in JS: "", 0, -1, NaN, null, undefined false
- operators:
  - equality with coercion allowed: ==, !=
  - equality without coercion: ===, !==

Conditionals

- If… else if… if…
- Switches
- Conditional operator: [condition] ? [if true] : [if false]

Functions

- Defines a *scope* (more in a minute)
- Functions as named variables

```
var adder = function(a, b) {
   return a + b;
}
adder(1, 2); //= 3
```

- Immediately invoked function expressions

```
(function example() { console.log("Test"); }()
```

Functions

```
var a = "global scope";
scope1();

function scope1() {
    var b = "scope of scope1()";

    function scope2() {
        var c = "scope of scope2()";

    }

    console.log(c); //ReferenceError
}
```

Variables

- Variables and object properties must be valid identifiers (a-z, A-Z, 0-9, $, _)
- Declared variables belong to a function scope or global scope
- Scopes can be nested - variables are available to inner scopes
- Variable and function declarations are *hoisted*

Prototypes

- JS is object oriented, but does not have *classes** - uses Property-based inheritance

- Every JS object has a property [[Prototype]] referencing it’s prototype

- Prototype chain: when trying to access a property of an object, that property will also be searched in all consecutive prototypes of the object

* ES2015 introduces *class*, but only as syntic sugar

Asynchronous programming

- JS is single-threaded - I/O is asynchronous
- Requirement that web pages are not blocked e.g. while waiting for data to be returned over the network
- Rely on callbacks, promises, events
- More next week…

What is Node.js?

- Node.js is a JS runtime for running JS code server-side
- Based on V8 engine from Chrome
- Basic libraries included in Node.js itself
- npm package manager with 400 000+ libraries and frameworks

What is Node.js

- Most common use for node.js is development of web services
- Also be used for general scripting etc
- Demos:
  - "Hello World"
  - Using a library from npm

Node.js HTTP server

- Node.js comes with http library for http communication
- Example - receiving requests

- Frameworks would normally be used to abstract away details of e.g. the http library

Useful hints/tools

- CURL/Postman for HTTP requests
- Debugging node.js with Chrome
- Using "strict mode"
- Using a code checker like jshint, eslint, etc

CURL/Postman

- CURL - https://github.com/curl/curl
  - Command line tool for making HTTP requests
  - Use verbose mode to see details (-v or -vv)

- Postman - https://www.getpostman.com/
  - Available as chrome extension, app etc
  - GUI tool for interacting with API

Debugging node.js

- Chrome can be used for debugging node.js:

1. Run node with "-- inspect" argument
2. Open chrome://inspect in Chrome
3. Select "Open dedicated DevTools for node.js"

strict mode

- Enabled by adding "use strict;" to your .js file
- Strict mode disallows "bad" or "uncertain" behaviour

Code checker

- Different code checkers available to check code quality
- Useful to make sure your code is consistent
- Examples: JSlint, JSHint, ESlint

Obligatory assignment 1

- http://www.uio.no/studier/emner/matnat/ifi/INF5750/h17/Assignments/inf5750_assignment1.html