JavaScript Alternatives and the Future of the Web

(with inputs from CS498RK at UIUC, used with permission, “TypeScript Deep Dive” Book, and Wikipedia

Outline

- What ails JavaScript ?
- JavaScript Alternatives
  - DART (Google)
  - TypeScript (MS)
  - Flow (Facebook)
- EcmaScript6, WebAssembly and the Future

What's good with JavaScript ?

- Easy to prototype
- Hybrid imperative/functional language
  - First-class functions
- Natural fit for real-time, event-driven applications
- Great open source community
- No need to install complex software, compiler

What ails JavaScript ?

- Types (or lack therof)
- Permissivity (e.g., lack of initialization)
- Prototype-based Object model (boon or bane)
- Ability to treat strings as code (e.g., eval)

Outline

- What ails JavaScript ?
- JavaScript Alternatives
  - DART (Google)
  - TypeScript (MS)
  - Flow (Facebook)
- EcmaScript 6 and WebAssembly

DART Language

- One of the early attempts to get around the eccentricities of JavaScript
- Prototyped by Google in the GOTO conference 2011 – standardized in 2014 by the ECMA
- DART code is compiled into JavaScript code using a source-to-source compiler

Features of DART

- High-level language with C-like syntax
- Object oriented, single inheritance based
- Supports interfaces, mixins, abstract classes, generics (to some extent)
- Cannot interoperate with native JS code without special wrappers

Four ways to run DART code

- Compiled to JavaScript
  - No special browser support
- In the Dartium browser
  - supported through a plugin in Chromium browser
- Standalone
  - Packaged as a VM with libraries
- Native
  - Compiled down to machine code

Pros and Cons of DART

- Pros
  - Not saddled with JavaScript legacy features
  - Strong type-checking possible (in checked mode)

- Cons
  - Difficult to debug the generated JavaScript
  - High learning curve; up-front investment
  - Cannot gradually migrate large JS code-base
  - Dart compilers didn’t support all features till recently

Outline

- What ails JavaScript ?
- JavaScript Alternatives
  - DART (Google)
  - TypeScript (MS)
  - Flow (Facebook)
- EcmaScript 6, WebAssembly and the Future

TypeScript

- Superset of JavaScript with static (optional) types
- Developed by Microsoft in 2012. Now at version 2.1
- Strict superset of JavaScript – all JavaScript code is valid TypeScript code
- Typescript is transpiled (i.e., source-to-source translated) to JavaScript

Type Annotations

var foo: number = 123;

foo = ‘123’; // This’ll result in type-error

NOTE: Even if you don’t explicitly annotate foo to be of type ‘number’, TypeScript will still indicate an error in the second line as it can infer types automatically (in many cases)

TypeScript: Support for Classes

```
class Point {
    x: number;
    y: number;
    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
    add(point: Point) {
        return new Point(this.x + point.x, this.y + point.y);
    }
}

var p1 = new Point(0, 10);
var p2 = new Point(10, 20);
var p3 = p1.add(p2); // {x:10,y:30}
```

TypeScript: Inheritance

```
class Point3D extends Point {
    z: number;
    constructor(x: number, y: number, z: number) {
        super(x, y);
        this.z = z;
    }
    add(point: Point3D) {
        var point2D = super.add(point);
        return new Point3D(point2D.x, point2D.y, this.z + point.z);
    }
}
```

TypeScript: TypeChecking

```
interface Point2D {
    x: number;
    y: number;
}
interface Point3D {
    x: number;
    y: number;
    z: number;
}
var point2D: Point2D = { x: 0, y: 10 }
var point3D: Point3D = { x: 0, y: 10, z: 20 }
function iTakePoint2D(point: Point2D) { /* do something */ }

iTakePoint2D(point2D); // exact match okay
iTakePoint2D(point3D); // extra information okay
iTakePoint2D({ x: 0 }); // Error: missing information `y`
```

TypeScript today

- Extensive tool and IDE support from MS
  - Supported by Visual Studio
- TypeScript compiler is written in JavaScript
  - Can run on web browsers without additional steps
  - Fully compatible with ES 6
- Standardized by ECMA in 2015
- Defacto language for Angular 2.0

TypeScript: Pros and Cons

- Pros
  - Lightweight compared to DART
  - Includes all of JavaScript – easy to learn for JS devs
  - Migration of legacy JS code is easier
- Cons
  - Inherits many of the problems of JavaScript
  - Type guarantees are not strong due to optional nature and the support for unannotated JS

Outline

- What ails JavaScript ?
- JavaScript Alternatives
  - DART (Google)
  - TypeScript (MS)
  - Flow (Facebook)
- EcmaScript 6, WebAssembly and the Future

Flow

- Uses type inference to infer types in unannotated/unmodified JavaScript code

- Annotations are simply comments in javaScript – no changes to JS syntax
  - Can be ignored by non-Flow aware tools
  - Same code is executed as what is checked

Flow: Pros and Cons

- Pros
  - Type inference is much more aggressive compared to TS, which defaults to *any* type
  - Has *non-nullable* types, though TS 2.0 has these
  - Does not require expensive compilation step
- Cons
  - Limited support for classes and encapsulation
  - Tool/IDE support is rather limited

Outline

- What ails JavaScript ?
- JavaScript Alternatives
  - DART (Google)
  - TypeScript (MS)
  - Flow (Facebook)
- EcmaScript 6, WebAssembly and the Future

ES6: Features

- Arrows
  - v => v + 1
- Classes
  - also constructors, getters and setters
- Iterators and Generators
  - Similar to languages like Python
- Modules and Module loading
  - Formalizes what node.js supported

WebAssembly: The Future ?

- Universal “assembly language” for the web
  - Successor of asm.js and other earlier attempts
  - Need special browser support for execution
  - Faster to parse and execute than JavaScript
  - Compile from different languages including C/C++, JavaScript, Java using Emscripten (LLVM-based)
  - Successor of NaCl in Chrome – execute native applications within the web browser
  - Introduced in 2015 by Google, formalized in 2016

WebAssembly: Goals

- Efficient and Fast
  - Use of stack machine semantics
- Safe
  - Memory safety and sandboxing
- Open and debuggable
  - Supposed to be easy when pretty printed as text
- Part of open-web platform
  - Backward compatible with JavaScript

WebAssembly: Example

```
(func $fac-iter (param $n i64) (result i64)
  (local $i i64)
  (local $res i64)
  (set_local $i (get_local $n))
  (set_local $res (i64.const 1))
  (loop $done $loop
    (if
      (i64.eq (get_local $i) (i64.const 0))
      (br $done)
      (block
        (set_local $res (i64.mul (get_local $i) (get_local $res)))
        (set_local $i (i64.sub (get_local $i) (i64.const 1)))
              )
          )
    (br $loop)
      )
  (get_local $res)
  )
```

WebAssembly: Pros and Cons

- Pros
  - Clean format; easy to parse and execute
  - Gets rid of thornier features of JavaScript
  - Can be compilation target for any language
- Cons
  - Not supported by many browsers today
  - Binary format as opposed to open source
  - Tool support is currently limited (but growing)

IoT and JavaScript: The future ?

- JavaScript is also becoming popular in the IoT space - Direct access to low level device APIs
  - Samsung: IoT.Js
  - Microsoft: NTVSIoT
  - Intel: Web of Things

- New set of challenges as IoT devices are typically resource constrained and interact extensively with the environment

IoT and JavaScript: Why ?

- JS is event-oriented. IoT is all about events
- Network code is distributed on multiple nodes
- Easy data sending and receiving thro' JSON
- Critical mass of programmers and technologies – no need for new language

ThingsJS

- Research project to run JavaScript on IoT devices
  - Edge computing
  - No VM or OS support
  - Self-Adaptable

- We are actively looking for student interns and honors thesis students to work on ThingsJS
  - Chance to get research/development experience
  - Come talk to us for more info (http://thingsjs.io)

Some final thoughts ….

- Prepare well for the exam – understand the concepts, solve in-class exercises. I will release a sample final exam about a week before the exam

- Post questions to Piazza – I will answer questions during exam period until the evening of Dec 14th .

- Please complete the online teaching evaluations. Also, send me individual feedback on Piazza.