aurelia

aurelia.io

build your imagination

Who Am I?

TIBI COVACI

PROGRAMMER WITH OVER 20 YEARS OF EXPERIENCE

TRAINER FOR THE PAST 12 YEARS

MEMBER OF THE BOARD OF ADVISORS FOR AURELIA

tibi@covaci.se | @tibor19

Agenda

- Introduction
- ECMASCript Past, Present and Future
- Interlude - Using Transpilers and Polyfills
- Introduction
- Custom Attributes and Elements
- Architecture of Aurelia
- Workshop files at http://bit.ly/1OOvZuK
- https://github.com/tibor19/aurelia-workshop-gotocon-lon-2015

ECMAScript Past, Present and Future

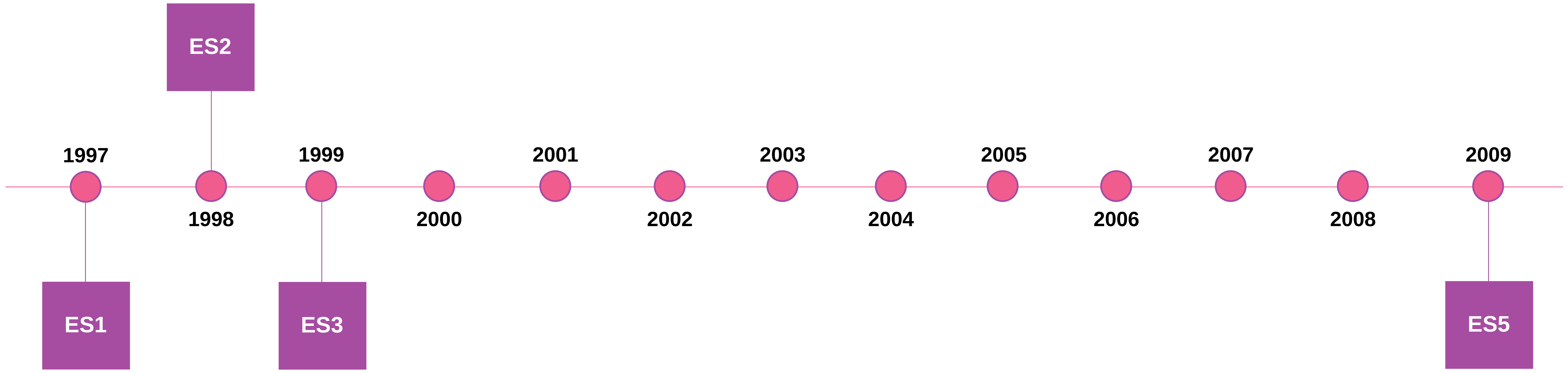
ECMAScript Past

ECMAScript Present (ES6)

ECMAScript Future

ECMAScript Past

ECMAScript Past



ECMAScript Present - ES6 (2015)

ES6 Feature Overview

- arrows
- classes
- enhanced object literals
- template strings
- destructuring
- default + rest + spread
- let + const
- iterators + for..of
- generators
- ~~comprehensions~~
- unicode
- modules
- module loaders
- map + set + weakmap + weakset
- proxies
- symbols
- subclassable built-ins
- promises
- math + number + string + object APIs
- binary and octal literals
- reflect api
- tail calls

ECMAScript Goals

- Fix Language Problems
- Codify Standard Usage Patterns
- A Better Target for Transpilers
- Move Towards a JavaScript DOM
- Security

ES6 Feature Categories

- Syntax
- Libraries
- Runtime

Variable Declarations

let

```
for(let i = 0; i < 10; ++i){
  console.log(i); // 0, 1, 2, 3 ... 9
}

console.log(i); //i is not defined
```

const

```
const PI = 3.14159265359;
PI = 0;
console.log(PI); // 3.14159265359
```

Modules

export

```
//lib/math.js

export function sum(x, y){
  return x + y;
}

export const PI = 3.14159265359;
```

import

```
//app.js

import {sum, PI} from 'lib/math';

alert('2π = ' + sum(PI, PI));
```

dynamic import

```
System.import('lib/math').then(function(m) {
  alert('2π = ' + m.sum(m.PI, m.PI));
});
```

Classes

```
export class Employee {
  constructor(firstName, lastName){
    this.firstName = firstName;
    this.lastName = lastName;
  }

  get fullName(){
    return `${this.firstName} ${this.lastName}`;
  }

  calculateSalary(){
    return 10000;
  }
}
```

```
export class Manager extends Employee {
  constructor(firstName, lastName, bonus){
    super(firstName, lastName);

    this.bonus = bonus;
    this.directReports = [];
  }

  addDirectReport(employee){
    this.directReports.push(employee);
  }

  calculateSalary(){
    return super.calculateSalary() + this.bonus;
  }

  static seniorManager(firstName, lastName){
    return new Manager(firstName, lastName, 1000);
  }

  static juniorManager(firstName, lastName){
    return new Manager(firstName, lastName, 100);
  }
}
```

Also showing template strings (string interpolation).

Arrow Functions

```
// Expression bodies
let odds = evens.map(v => v + 1);
let nums = evens.map((v, i) => v + i);

// Statement bodies
let fives = [];

nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v);
});

// Lexical this
let bob = {
  name: 'Bob',
  friends: [],
  logFriends() {
    this.friends.forEach(x => console.log(this.name + ' knows ' + x));
  }
}
```

Also showing enhanced object literals.

Promises

```
function timeout(duration = 0) {
  return new Promise((resolve, reject) => {
    setTimeout(resolve, duration);
  })
}


let p = timeout(500).then(() => {
  return timeout(1000);
}).then(() => {
  throw new Error('something got broked');
}).catch(err => {
  return Promise.all([timeout(100), timeout(200)]);
});
```

Also showing default parameter values.

Iterators and for..of

```
let powersOfTwo = {
  [Symbol.iterator]() {
    let power = -1;
    return {
      next(){
        return { done: false, value: Math.pow(2, ++power) };
      }
    };
  }
};

for(let n of powersOfTwo){
  if(n > 1024)
    break;

  console.log(n);
}
```

Also showing enhanced object literals, symbols and Math.pow.

Generators

```
let powersOfTwo = {
  *[Symbol.iterator]() {
    let power = -1;
    while(true){
      yield Math.pow(2, ++power);
    }
  }
};

for(let n of powersOfTwo){
  if(n > 1024)
    break;

  console.log(n);
}
```

Also showing enhanced object literals, symbols and Math.pow.

ES6 Feature Overview

- arrows
- classes
- enhanced object literals
- template strings
- destructuring
- default + rest + spread
- let + const
- iterators + for..of
- generators
- ~~comprehensions~~
- unicode
- modules
- module loaders
- map + set + weakmap + weakset
- proxies
- symbols
- subclassable built-ins
- promises
- math + number + string + object APIs
- binary and octal literals
- reflect api
- tail calls

ES6 Feature Categories

- Syntax
- Libraries
- ~~Runtime~~

Transpilers

- http://babeljs.io
- http://www.typescriptlang.org
- http://github.com/google/traceur-compiler

Polyfills

- https://github.com/zloirock/core-js
- https://github.com/paulmillr/es6-shim
- https://github.com/ModuleLoader/es6-module-loader
- https://github.com/systemjs/systemjs

ECMAScript Future

Overview

- **New Release Cadence**
- **Parallel Development**
- **New Naming Scheme**
  - ES 2015 (formerly ES6)
  - ES 2016 (formerly ES7)

- **New Features**
  - Property Initializers
  - Decorators
  - Object.observe
  - async/await
  - more…

ES 2016 Feature Mashup

```
import {CustomerService} from 'server/customer-service';
import {LoadingIndicator} from 'ux/loading-indicator';
import {inject} from 'aurelia-framework';

@inject(CustomerService, LoadingIndicator)
export class CustomerEditScreen {
  customer = null;

  constructor(customerService, loadingIndicator){
    this.customerService = customerService;
    this.loadingIndicator = loadingIndicator;
  }

  async activate(params){
    try{
      this.loadingIndicator.show();
      this.customer = await this.customerService.get(params.id);
    } finally{
      this.loadingIndicator.hide();
    }
  }

  async save(){
    await this.customerService.save(this.customer);
  }
}
```

```
//from aurelia-framework

export function inject(...rest){
  return function(target){
    target.inject = rest;
  }
}
```

Showing decorators, property initializers and async/await.

End to End Gulp

Getting Started

Getting Started

toolkit that will help you automate painful or time-consuming tasks in your development

Setup

- Install NodeJS

  https://nodejs.org/
- Install Gulp Globally

  ```
  npm install -g gulp
  ```
- Install Gulp Locally in Your Project

  ```
  npm install --save-dev gulp
  ```

gulpfile.js

```
var gulp = require('gulp');

gulp.task('task-name', function() {
  //task implementation goes here
});
```

Writing Tasks

Basic Tasks

gulp.src(globs)

Pull files in.

stream.pipe()

Transform files.

gulp.dest(path)

Push files out.

```
gulp.task('build', function () {
  return gulp.src('**/*.js')
    .pipe(babel())
    .pipe(gulp.dest('dist/'));
});
```

gulp.watch(glob, tasks)

Watch files for changes.

Asynchronous Tasks

Use a Callback, Stream or Promise

```
gulp.task('serve', ['build'], function(done) {
  browserSync({
    open: false,
    port: 9000,
    server: {
      baseDir: ['.'],
      middleware: function (req, res, next) {
        res.setHeader('Access-Control-Allow-Origin', '*');
        next();
      }
    }
  }, done);
});
```

Common Plugins

Transpilers

Babel

```
var gulp = require('gulp');
var changed = require('gulp-changed');
var plumber = require('gulp-plumber');
var babel = require('gulp-babel');
var sourcemaps = require('gulp-sourcemaps');

gulp.task('build-system', function () {
  return gulp.src('**/*.js')
    .pipe(plumber())
    .pipe(changed('dist/', {extension: '.js'}))
    .pipe(sourcemaps.init({loadMaps: true}))
    .pipe(babel())
    .pipe(sourcemaps.write({includeContent: true}))
    .pipe(gulp.dest('dist/'));
});
```

TypeScript

```
var gulp = require('gulp');
var ts = require('gulp-typescript');

var tsProject = ts.createProject({
  declarationFiles: false,
  noExternalResolve: true,
  target: "es5",
  module: "amd",
  outDir: paths.out,
  emitDecoratorMetadata: true,
  experimentalDecorators: true
});

gulp.task('build-ts', function() {
    var tsResult = gulp.src([
      paths.tsSource,
      paths.jspmDefinitions,
      paths.typings
    ]).pipe(ts(tsProject));

    return tsResult.js.pipe(gulp.dest(paths.root));
});
```

CSS

LESS

```
var gulp = require('gulp');
var less = require('gulp-less');
var path = require('path');

gulp.task('build-less', function () {
  return gulp.src('**/*.less')
    .pipe(less({
      paths: [ path.join(__dirname, 'less', 'includes') ]
    }))
    .pipe(gulp.dest('dist/'));
});
```

Developer Ergonomics

BrowserSync

```
var gulp = require('gulp');
var browserSync = require('browser-sync');

function reportChange(event){
  console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');
}

gulp.task('watch', ['serve'], function() {
  gulp.watch(paths.source, ['build', browserSync.reload]).on('change', reportChange);
});
```

Release Management

Changelog Generation

```
var gulp = require('gulp');
var changelog = require('conventional-changelog');
var fs = require('fs');

gulp.task('changelog', function(callback) {
  var pkg = JSON.parse(
    fs.readFileSync('./package.json', 'utf-8')
  );

  return changelog({
    repository: pkg.repository.url,
    version: pkg.version,
    file: paths.doc + '/CHANGELOG.md'
  }, function(err, log) {
    fs.writeFileSync(paths.doc + '/CHANGELOG.md', log);
  });
});
```

Versioning

```
var gulp = require('gulp');
var bump = require('gulp-bump');

gulp.task('bump-version', function(){
  return gulp.src(['./package.json'])
    .pipe(bump({type:'patch' }))
    .pipe(gulp.dest('./'));
});
```

Code Quality

Linting

```
var gulp = require('gulp');
var eslint = require('gulp-eslint');

gulp.task('lint', function() {
  return gulp.src(‘src/**/*.js')
    .pipe(eslint())
    .pipe(eslint.format())
    .pipe(eslint.failOnError());
});
```

.eslintrc

```
{
  "extends": "./node_modules/aurelia-tools/.eslintrc",
  "rules": {
    "no-new-func": 0
  }
}
```

Much More…

- Unit Testing
- End-to-End Testing
- Code Coverage Reports
- Bundling
- Deploy
- Documentation Generation
- etc.

Patterns and Practices

Organize Tasks

- Break Up Your Gulpfile
  - One file per task or task category
- Place all task files in a ‘tasks’ folder
- Use the “require-dir” library

gulpfile.js

```
require('require-dir')('tasks');
```

Centralize Configuration

- File Paths
- Transpiler Options
- Command line Arguments
- etc.

Compose Tasks

```
var gulp = require('gulp');
var runSequence = require('run-sequence');

// this task calls the clean task (located
// in ./clean.js), then runs the build-system
// and build-html tasks in parallel
// https://www.npmjs.com/package/gulp-run-sequence
gulp.task('build', function(callback) {
  return runSequence(
    'clean',
    ['build-js', 'build-html'],
    callback
  );
});
```

Recap

Getting Started

Writing Tasks

Common Plugins

Patterns and Practices

Agenda

What & Why?

Building An App

What & Why?

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

What?

JavaScript (ECMAScript 6/7)

- Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

What?

- JavaScript (ECMAScript 6/7)
  - Collection of Collaborating Libraries
- Build JavaScript Clients (SPA)
- Philosophy

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

Philosophy

- Open Source (MIT)

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

Philosophy

- Open Source (MIT)
- Clean Code

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

Philosophy

- Open Source (MIT)
- Clean Code
- Simple Conventions

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

Philosophy

- Open Source (MIT)
- Clean Code
- Simple Conventions
- Testable

What?

JavaScript (ECMAScript 6/7)

Collection of Collaborating Libraries

Build JavaScript Clients (SPA)

Philosophy

- Open Source (MIT)
- Clean Code
- Simple Conventions
- Testable

Rapid Adoption & Large Active Community

Commercially Backed by Durandal Inc. and Partners

Why?

Why?

- ES6, Modern DOM, Web Components

Why?

- ES6, Modern DOM, Web Components
- No External Dependencies (except polyfills)

Why?

- ES6, Modern DOM, Web Components
- No External Dependencies (except polyfills)
- Simple, Powerful Programming Model

Why?

- ES6, Modern DOM, Web Components
- No External Dependencies (except polyfills)
- Simple, Powerful Programming Model
- Modern/Future Compatible 2-Way Databinding

Why?

- ES6, Modern DOM, Web Components
- No External Dependencies (except polyfills)
- Simple, Powerful Programming Model
- Modern/Future Compatible 2-Way Databinding
- Commercial Support

Why?

- ES6, Modern DOM, Web Components
- No External Dependencies (except polyfills)
- Simple, Powerful Programming Model
- Modern/Future Compatible 2-Way Databinding
- Commercial Support
- Pathway Forward for Durandal and Angular

Building An App

Setup

Download Skeleton and Unzip

https://github.com/aurelia/skeleton-navigation

Install Build Dependencies

Install NodeJS

- npm install -g gulp
- npm install
- npm install -g jspm
- jspm install

Start Developing

gulp watch
browse to http://localhost:9000

Questions?