Step up your game
&
bring your projects to the Next Level

bit.ly/goto-amstd

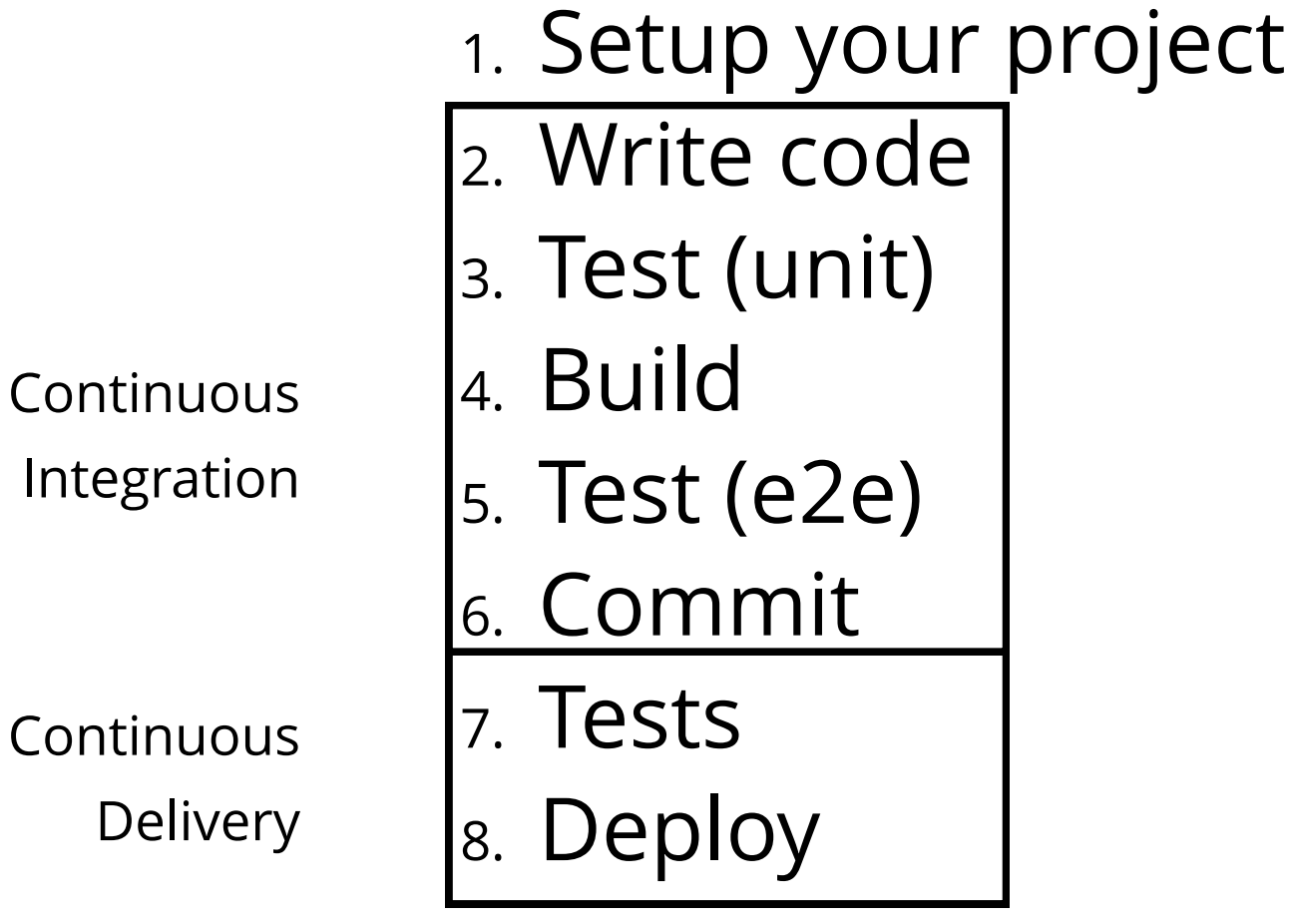
goto; conference

Please evaluate this talk via the mobile app!

Olivier Combe

@ocombe
github.com/ocombe

Front end dev @ HOLIMETRIX™


1. Setup your project
2. Write code
3. Test (unit)
4. Build
5. Test (e2e)
6. Commit
7. Tests
8. Deploy
Continuous Integration
Continuous Delivery

Setup your project

Opinions matter

A few things to decide:

- Application structure
  - group by type
  - group by feature/module
- Code conventions
  - how to name things
  - how to write code

→

- Style guide
- JSCS (code style linter)

Scaffolding Tools

- Use generators
- Let you decide what to include
- Quality depends on the generator used
- 2 main competitors:

Yeoman

Slush

- Alternative: use seeds/skeleton apps


E II R

Build tools

- One of the most important choice to make
- 4 main competitors:


Grunt

Gulp


Brunch

Broccoli

Grunt

- The largest ecosystem (~4400 plugins)
- Task runner, not really a build tool
- Configuration over code

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Load the plugin that provides the "uglify" t
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).
  grunt.registerTask('default', ['uglify']);

};
```

Gulp

- The challenger
- Large ecosystem (~1530 plugins)
- Stream-based build system
- Code over configuration
- Easy to customize (it's just js!)

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

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

Brunch

- One of the first ones
- In memory file-based build system
- Incremental rebuilds
- Simplification over customization
- Very opinionated

```
exports.config =
  files:
    javascripts:
      joinTo:
        'javascripts/app.js': /^app/
```

Broccoli

- New kid on the block (still in beta)
- Not a vegetable, more like a tree
- File-based build system with caching
- Customization over simplification
- Very flexible (too much?)

```
var uglifyJavaScript = require('broccoli-uglify-
var pickFiles = require('broccoli-static-compile

// create tree for files in the app folder
var app = 'app'
app = pickFiles(app, {
  srcDir: '/',
  destDir: 'appkit' // move under appkit namespa
})

appJs = uglifyJavaScript(app, {
  // mangle: false,
  // compress: false
});

// export the js tree
module.exports = appJs;
```

Assets managers

- Bundle your libs as static assets
- Support AMD, CommonJS & ES6 modules
- Extendable with plugins
- 3 main competitors:
  - Browserify
  - Webpack
  - JSPM + SystemJS

Browserify

- Built to bring node into the browser
- Opinionated (there is a browser-way-ify)
- Easy to use If you follow the conventions
- CLI tool
- Config goes into package.json
- Functionalities are split between plugins & transforms

```
browserify -t es6ify main.js | exorcist bundle.js.map > bundl
```

Webpack

- Built to help the user load all of his assets
- Pretty flexible
- More like a build tool (config file + CLI)
- Requires some work before you can start
- Add new functionalities with loaders & plugins

```
module.exports = {
  context: __dirname + '/app',
  entry: './index.js',
  output: {
    path: __dirname + '/app',
    filename: 'bundle.js'
  },

  module: {
    loaders: [{
      test: /\.js$/,
      loader: 'babel',
      exclude: /node_modules/
    }]
  }
};
```

JSPM + SystemJS

- Built to help the user load all of his assets
- Install assets from anywhere with JSPM
- Load them in the browser with SystemJS
- No build step (except for production)
- Add new functionalities with plugin loaders

config.js

```
System.config({
    // or 'traceur' or 'typescript'
    transpiler: 'babel'
    // or traceurOptions or typescriptOpti
    babelOptions: {}
});
```

index.html

```
<!doctype html>
<script src="jspm_packages/system.js"></scrip
<script src="config.js"></script>
<script>
  System.import('lib/main');
</script>
```

bootstrap.js

```
import _ from 'lodash-node/modern/lang/isEqua
import $ from 'jquery';

export function bootstrap() {
  // bootstrap code here
}
```

main.js

```
import {bootstrap} from './bootstrap
bootstrap();
```

</> Code

(Pre|Post) processors

- Take your code and transform it
- Give you access to new features
- Handle annoying processes for you
- Alert you if necessary

New language features

- With a simple pre-processor:
  - CoffeeScript
  - Babel (ES2015/JSX)
  - ~~Traceur (AtScript)~~
  - TypeScript

- Why?
  - Modularity (modules, imports)
  - Clarity (classes, let/const)
  - Economy (arrow functions, classes)
  - And more (types, annotations, ...)

Other tools

- Ng annotate:
  - Handles angular dependency injection annotations for you

```
var MyController = function($scope, greeter)
  // ...
}
MyController.$inject = ['$scope', 'greeter']
```

CSS Processors

- Languages improvements
  - SASS
  - LESS
  - Stylus

- Why?
  - Clarity (indentation, variables)
  - Economy (variables, mixins)
  - Modularity (includes, extends)
  - Optimizations
  - And more (plugins, syntax check...)

Other tools (2)

- Auto prefixer:
  - Handles css browser prefixes for you
- Remove unused css rules:
  - Uncss
  - PurifyCSS

Auto reload

- Watch your files
- Reload the browser when needed
- Inject files when possible
- Many tools:
  - Browser sync (node)
  - Livereload (app)
  - Fb flo (browser extension)

Debug

- Sourcemaps
- Chrome extensions:
  - Batarang
  - Ng-inspector

Test

“ When you fail,

fail fast,

and not silently

Linting tools

- Static code analysis tools
- Detect errors & potential problems
- Choose which rules to apply or write new ones
- 3 main competitors:
  - JSLint
  - JSHint
  - ESLint

“ Testing is not wasting time,
it's compensating your flaws
and betting on the long term

Unit tests

- Test individual units of code
- Run fast
- Focus on small/stable testable parts (services, ...)
- Test driven development

Unit tests (2)

- Test runners:
  - Karma
  - Mocha
- Test frameworks:
  - ~~QUnit~~
  - Jasmine
  - Mocha (+ Chai + Sinon)

Jasmine vs. Mocha, Chai, and Sinon

E2E Tests

- Test your app in a real browser
- Take time to run
- Focus on critical paths
- Test in multiple browsers (BrowserStack / Sauce labs)
- Test runner: Protractor (+ Jasmine)

CSS Testing

- Test visual regressions

- 2 libs based on resemble.js:
  - PhantomCSS (requires a full CasperJS setup)
  - BacktopJS

Deploy

Changelog

- Make it easier to see notable changes between versions
- Because your memory isn't unbeatable and you might not be around forever
- Because reading git commits doesn't tell everything
- 2 libs:
  - Conventional changelog
  - Github changelog generator

Keep a changelog

Documentation

- Comment your code with ngDoc

```
/**
 *
 * @ngdoc directive
 * @name awesomeElement
 * @module myModule
 * @restrict E
 * @description
 * This is a directive!
 *
**/
```

- Extract your doc with dgeni
- Or use readme.io

Continuous integration

- Automate tasks on commit:
  - tests
  - build
  - deploy

- Many solutions:
  - TravisCI (free for open source)
  - Jenkins (free & open source, host it yourself)
  - CircleCI (free for 1 concurrent build)
  - Codeship (free for 1 concurrent build)

Thank you!

goto; conference

Please evaluate this talk via the mobile app!