# Table of Contents

# Setting up repository & installing development tools

## > create dedicated git-repo

## > clone it locally

```
git clone remote-repository-url/repository-name.git
```

## > create .gitignore (and add any .editor/ lines necessary)

*in minimum add the following lines to .gitignore:*

```
# npm modules
/node_modules/
# distributable (compiled) files
/distribution/
```

## > install development tools (bunch of software as npm modules)

### *Gulp*

*(our Task Runner of choice)*

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

### *Webpack*

*(for module loading in ES6 -style e.g. ``import 'something' from 'Somewhere';``)*
*(also does JSX & ES6 transpiling)*

```
npm install --save-dev webpack
```

### *ReactJS (+ react-dom)*

```
npm install --save-dev react react-dom
```

### *Babel (+ webpack-loader + specs for JSX & ES6)*

```
npm install --save-dev babel-core babel-loader babel-preset-es2015 babel-preset-react
```

### *Libsass (gulp plugin that contains node-sass + autoprefixer gulp plugin)*

```
npm install --save-dev gulp-sass gulp-autoprefixer
```

### *@font-face generation (automatically, using gulp plugin)*

```
npm install --save-dev gulp-font2css
```

### *Gulp's utilities*

```
npm install --save-dev gulp-concat merge-stream gulp-order browser-sync
```

## > configure gulp to combine all development tools as tasks

*create following gulpfile.js*

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

/* CSS building */
const sass = require('gulp-sass');
const autoprefixer = require('gulp-autoprefixer');
const font2css = require('gulp-font2css').default;

/* JS building */
const webpack = require('webpack');

/* Generic utils (keep the following to minimum) */
const concatenate = require('gulp-concat');
const mergestream = require('merge-stream');
const order = require('gulp-order');
const browsersync = require('browser-sync').create();
const path = require('path');
```

file continues...

task "html"

```
/*
 * Task to copy up-to-date index.html to distribution/ -folder, AND RELOAD BROWSER-SYNC
 */
gulp.task('html', function () {
  return gulp.src('sources/index.html')
    .pipe(gulp.dest('distribution'))
    .on('end', browsersync.reload);
});
```

file continues...

task "fonts-sass-css"

```
/*
 * Task to
 * 1) gather fontfiles and turn them into @font-face declarations
 * 2) gather .scss files,
 *    and compile them into CSS,
 *    then autoprefix those by browserlist (https://github.com/ai/browserslist#queries)
rules:
 *      last 2 versions of "each" browser
 *      & any browsers with > 1% global usage
 *      & Firefox's "Extended Support Release"
 *
 * 3) bundle fonts first, scss second, into bundle.min.css
 * 4) ("HOT")RELOAD BROWSER-SYNC
 */
gulp.task('fonts-sass-css', function(){
  let fontcss = gulp.src('sources/scss/fonts/**/*.{otf,ttf,woff,woff2}')
    .pipe(font2css())
    .pipe(concatenate('fonts.css'));

  let stylecss = gulp.src('sources/scss/**/*.scss')
    .pipe(sass({outputStyle: 'compressed'})) // Minify aswell
    .pipe(autoprefixer({
      add: true, // Add missing prefixes (d'oh)
      remove: true, // Remove if any outdated prefixes
      cascade: false, // SASS is already minified so do not cascade ( = would add
newlines as-per prefix )
      browsers: ['last 2 versions', '> 1%', 'Firefox ESR'] // http://browserl.ist/?
q=last+2+versions%2C+%3E+1%25%2C+Firefox+ESR
    }))
    .pipe(concatenate('styles.css'));

  return mergestream(fontcss, stylecss)
    .pipe(order(['fonts.css', 'styles.css']))
    .pipe(concatenate('bundle.min.css')) // Bundle css
    .pipe(gulp.dest('distribution'))
    .pipe(browsersync.reload({stream: true}));
});
```

file continues...

task "javascript" (and remaining tasks)

```
/*
 * Let webpack's babel-loader do transpiling (JSX & ES6),
 * and let webpack do module loading (ES6 style 'import x from X')
 */
gulp.task('javascript', function() {
  webpack({
    entry: {
      // This (object-format) enables us to have multiple final bundles, each property
name mapping to it's own [name].min.js
      bundle: "./sources/js/main.js" // -> bundle.min.js
    },
    output: {
      filename: "distribution/[name].min.js"
    },
    resolve: {
      extensions: ['.js', '.jsx'] // Enable "import Something from './path/to/somewhere'"
without '...somewhere.jsx'
    },
    module: {
      rules: [
        /* .jsx components & and entry files with ES6 => into ES5 compatible javascript
with ReactJS */
        {
          test: /\.(js|jsx)$/, // NOTE, our main file(s) are with .js extension.
Difficult to debug. jsx? works too but not recommended.
          include: path.resolve(__dirname, 'sources/js'), // NOTE, this REQUIRES an
absolute path. Difficult to debug.
          exclude: /node_modules/,
          loader: 'babel-loader', // NOTE, NOT JUST "babel" BUT "babel-loader". Difficult
to debug.
          query: {
            presets: ['react', 'es2015']
          }
        }
      ]
    }
  }, function(err, stats) {
    // NOP, it's just a required second argument, even if an empty lambda.
    // No, calling gulp callback here is not necessary, for the webpack to work.
  });
});

/*
 * Development (localhost) server
 */
gulp.task('browsersync', function() {
  browsersync.init({
    server: { baseDir: 'distribution' }
  })
});


gulp.task('watch', ['browsersync', 'html', 'fonts-sass-css', 'javascript'], function() {
  gulp.watch(['sources/scss/fonts/**/*.{otf,ttf,woff,woff2}', 'sources/scss/**/*.scss'],
['fonts-sass-css']);
  gulp.watch('sources/js/**/*.js', ['javascript']);
  gulp.watch('sources/index.html', ['html']);
});

gulp.task('just-build', ['html', 'fonts-sass-css', 'javascript']);
```

## > create directories and initial files

```
mkdir sources
mkdir distribution
cd sources
mkdir js
mkdir scss
cd scss
mkdir fonts
```

*generate sources/index.html*
*generate sources/**js**/main.js*
*generate sources/**scss**/main.scss*