[Lesson 4]

Roi Yehoshua 2018

[What we learnt last time?]

- What is Pixel perfect
- Why is it important
- Tools for implementing Pixel perfect
- Adaptive and responsive layout
- Cross-Browser
- Graceful Degradation and Progressive Enhancement

[Our targets for today]

- Gulp
- Using Gulp for automation
- Gulp plugins

Modern workflow

- ➔ Modern frontend development envisages many tasks related to the processing of the source code before it could be served to the browser
- ➔ This may be required both for development and delivering to the production

  Example of such tasks:
  - → Concatenation of files
  - → Compilation of scripts (CoffeeScript, Typescript, Babel)
  - → Compilation of styles (SASS, Less, Stylus)
  - → Code linting
  - → Unit testing
  - → Minification

- ➔ These are only some of the most common examples, but there could be much more
- ➔ Running all of these tasks might be required every time a source code is changed

[Gulp]

- ➔ Gulp is a toolkit for automating development tasks
- ➔ It has simple API, many plugins, and proven to be powerful tool for development, that’s why it is very popular
- ➔ Gulp is using node streams, so many tasks could be done as one without creation of intermediary files on the disk
- ➔ It supports lots of plugins which significantly improves its usability

[gulpfile.js]

- ➔ After installing Gulp you will be expected to create a gulp file
- ➔ It could be one of the following:
  - → gulpfile.js file in the project root. This is useful if gulp file is small
  - → gulpfile.js folder in the project root. In this case your gulp file will be placed in this directory and names index.js. This way you can place different tasks in their own files. This is usually done for bigger gulp files

- ➔ Optionally, gulp file or folder could be called
  - → gulpfile.ts - if written in Typescript. It requires ts-node module installed
  - → gulpfile.babel.js - if used with babel. It requires @babel/register module installed

[gulp.task]

➔ gulp.task - creates tasks that gulp will run

This is an example of a task named “styles”

```
gulp.task('styles', function() {
   //code for processing styles
});
```

Combined task that calls two other tasks - “styles” and “scripts”

```
gulp.task('build', ['styles', 'scripts']);
```

We can call it from the console with the following command

```
gulp build
```

[gulp.src and .pipe]

- ➔ gulp.src - sets source of files and starts pipeline of operations
- ➔ .pipe - streaming method that receives operations with source files

In the following script we set as source all files in the directory src/scss/ that have scss extension. Using .pipe() method we then call sass() plugin to compile these scss files

```
gulp.src('src/scss/*.scss')
   .pipe(sass())
```

[gulp.dest]

➔ gulp.dest - writes result of the stream to the final file

After we processed scss files, we are writing compiled css-files to the ./build/css/ directory.

```
gulp.src('src/scss/*.scss')
   .pipe(sass())
   .pipe(gulp.dest('./build/css/'));
```

[gulp.watch]

➔ gulp.watch - watches for any changes in the traced files and calls corresponding tasks for them

Gulp will watch all javascript files in the folder src and will call tasks scripts and reload if any of them will be changed.

```
gulp.watch('src/*.js', ['scripts','reload']);
```

[Example]

Example of a task. It will lint, concat, uglify and finally write app.min.js bundle to the dist/app directory

```
gulp.task('scripts', function () {
   return gulp.src('src/*.js')
       .pipe(jshint('.jshintrc'))
       .pipe(concat('app.min.js'))
       .pipe(uglify())
       .pipe(gulp.dest('dist/app'));
});
```

Another task that will watch javascript files in the src directory and will call task scripts on any changes

```
gulp.task('watch', function(){
   watch('src/*.js', ['scripts']);
});
```

[DAN.IT]
EDUCATION

[Plugins]

- ➔ In previous examples you could have noticed that in pipe() we call functions like jshint, concat or uglify - each of them is a different gulp plugin
- ➔ There are lots of such plugins created for automation of most usual tasks you will have to do with your source code
- ➔ The usage could vary from plugin to plugin, but each of them has a provided documentation with an explanation on how to configure and use them

Here are some of useful plugins:

- → jshint - a linting plugin for gulp. Checks your code depending on chosen guidelines and provides you hints on the code improvement
- → concat - capable of unifying several files in one
- → uglify - minimizes your code and makes it harder to read
- → babel - provides many polyfills allowing you to write code in chosen ECMAScript standard without worrying about backward compatibility
- → sass - allows you writing your styles in sass language that will later be compiled to CSS

[Control questions]

1. What is Gulp?
2. What are the practical usages of Gulp?
3. What is a Gulp task?
4. How can we build our bundle on the fly?
5. Name some useful Gulp plugins