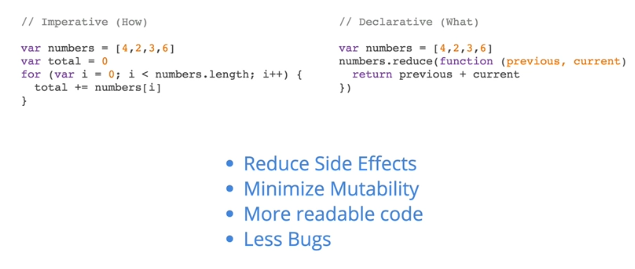
## Reactjs terminology

Imperative vs Declarative

Declarative codes (reduce is the api function)

- no need to keep track of the variable "total"

- has a layer of abstraction = "reduce()"



## Composition in Reactjs

- breaking everything down into components

E.g. breaking each element into functions



Normal JS vs ReactJs

Instead of returning business logic = string or whatever , reactjs makes elements (render())

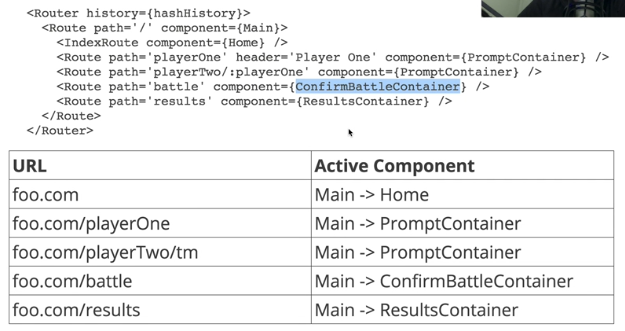
## Explicit Mutations

Change state of elements = use setState()



## React Router

Allows to map and route to “active components”, sorta like ng-route in angular but will see.



## Webpack

Bundle codes into one file,

E.g. codes below, notice there is “entry” and “output”

1) Webpack needs to know the starting point of your application, or your root JavaScript file.

2) Webpack needs to know which transformations to make on your code.

3) Webpack needs to know to which location it should save the new transformed code.



To make one webpack (from tutorial), create “webpack.config.js”

var HtmlWebpackPlugin = require('html-webpack-plugin')

var HTMLWebpackPluginConfig = new HtmlWebpackPlugin({

template: \_\_dirname + '/app/index.html',

filename:'index.html',

inject:'body'

});

Module.exports = {

Entry :[

‘./app/index.js’

],

Module:{

Loaders : [

{Test: /\.coffee$/, exclude: /node\_modules/, loader: “coffee-loader”}

]

},

Output:{

Filename: “index\_bundle.js”,

Path: \_\_dirname + ‘/dist’

},

Plugins:[HTMLWebpackPluginConfig]

};

## Babel

Do transfer information = code transformation from JSX into normal javascript so browser can understands



## Axios



HTTP requests = $http service in angularjs

## NPM scripts

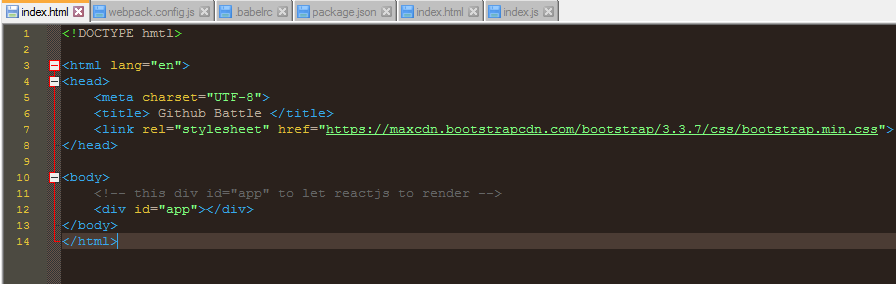
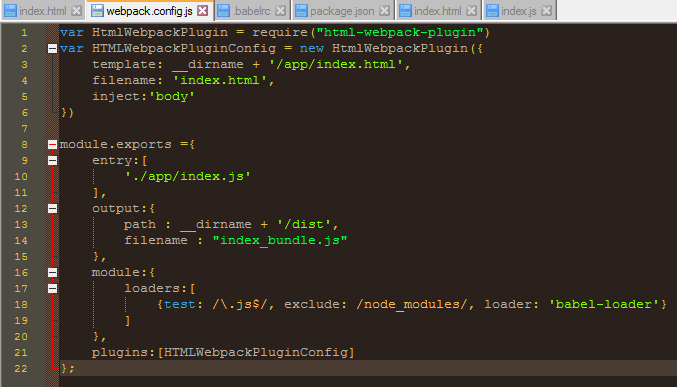
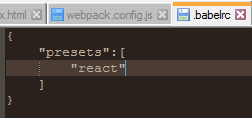
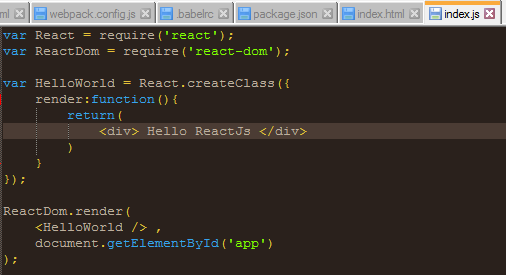
"scripts": {

"test": "ava 'app/\*\*/\*.test.js' --verbose --require ./other/setup-ava-tests.js"

}

To run the test in package.json , npm run test, then the script “ava” will run

## Setting up the environment for the first time (ReactJs)

1. Create folder “github-battle” or whatever
2. From terminal/cmd , type “npm init” to initiate the project which has package.json
3. Type “npm install --save react react-dom” to install react and react dom libraries for reactjs use. Check package.json has both dependencies
4. Type “npm install --save-dev html-webpack-plugin webpack webpack-dev-server -babel{core,loader} babel-preset-react” to install all the necessary transformation (babel-\*, html-webpack) , local server (webpack-dev-server). Check package.json if there are all those in devDependecies
5. Create folder “app” and cd into it, create index.html
6. Index.html
   1. 
7. Create “index.js” as an entry file in /app
8. Create “webpack.config.js” file in /github-battle
   1. 
   2. Module.exports {…} is to get the index.js as entry file, translate via “loaders” via babel-loader library with all the .js extension excluding all files in node\_modules folder and output to /dist named to index\_bundle.js
   3. Create .babelrc file in /github-battle
      1. 
      2. This file is to translate reactjs JSX into JS (from babel-preset-react) and load via babel-loader
   4. Then, htmlwebpackpluginconfig is to mimic from /app index.html to /dist index.html, so everytime we code into index.js, it translates via babel-loader from babel-preset-react to /dist index\_bundle.js and replace into /dist index.html from /app index.html
   5. In package.json, head over “scripts”
      1. 
      2. In cmd, type “npm run production”, to create /dist folder
9. NOW, START CODING IN INDEX.JS
   1. 
   2. Import react and react-dom libraries to create class and use reactdom to render it