Урок 2

Настройка среды разработки

Разбираем методы установки webpack и bubel

Что такое Webpack и какие преимущества он даёт

- Конфигурируем webpack
  - Let welcome=require ("./welcome"); //функция webpack
- Подключаем модули

Определяем структуру проекта

Настройка всех загрузчиков

Создание заготовки главной страницы

Практика

- Задача 1. Hello world.

Домашнее задание

Дополнительные материалы

Используемая литература

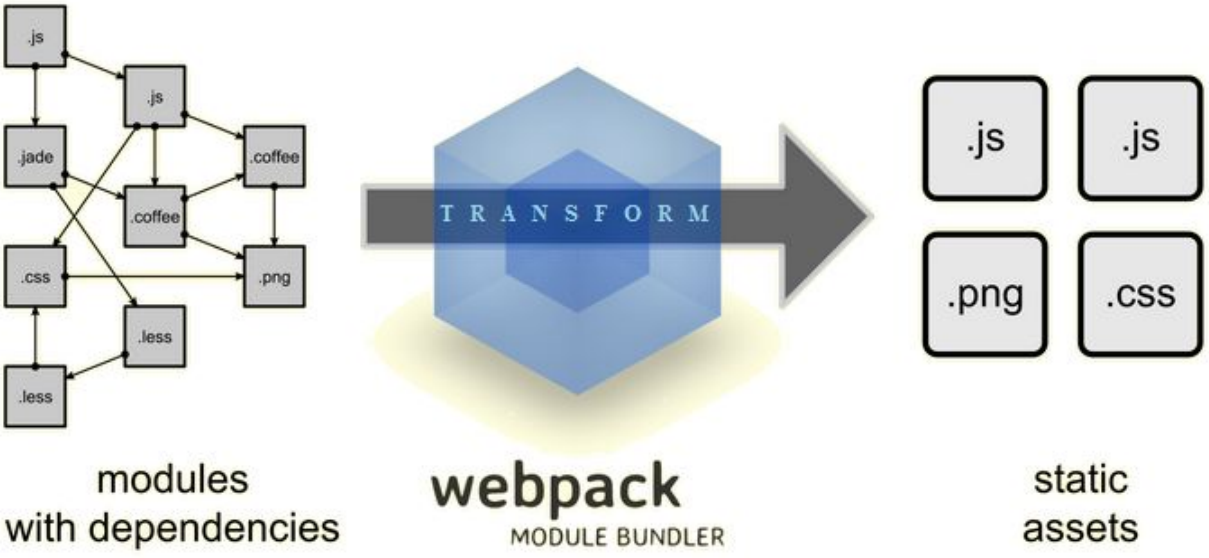
Что такое Webpack и какие преимущества он даёт

Веб-браузеры не понимают многие хорошо описанные модули. Например: веб-разработчики не включили модули стандарта ES6 в некоторые браузеры, таким образом хорошо прописанный код на react просто выдаст пустую страницу. Чтобы браузер понял написанную программу, можно или вручную переписать все команды в предыдущем стандарте, что займёт много времени, или использовать дополнительные инструменты. Одним из таких инструментов является бандлер модулей (module bundler). Он способен комбинировать разные модули и их зависимости в один файл в правильном порядке в один формат, понятном браузеру.

Два популярных бандлер-модулей это:

1. Browserify -пакет библиотек, для использования в браузере. Для его использования необходимо подключать дополнительные инструменты типа Grunt или Gulp, что может вызвать дополнительную трату времени.

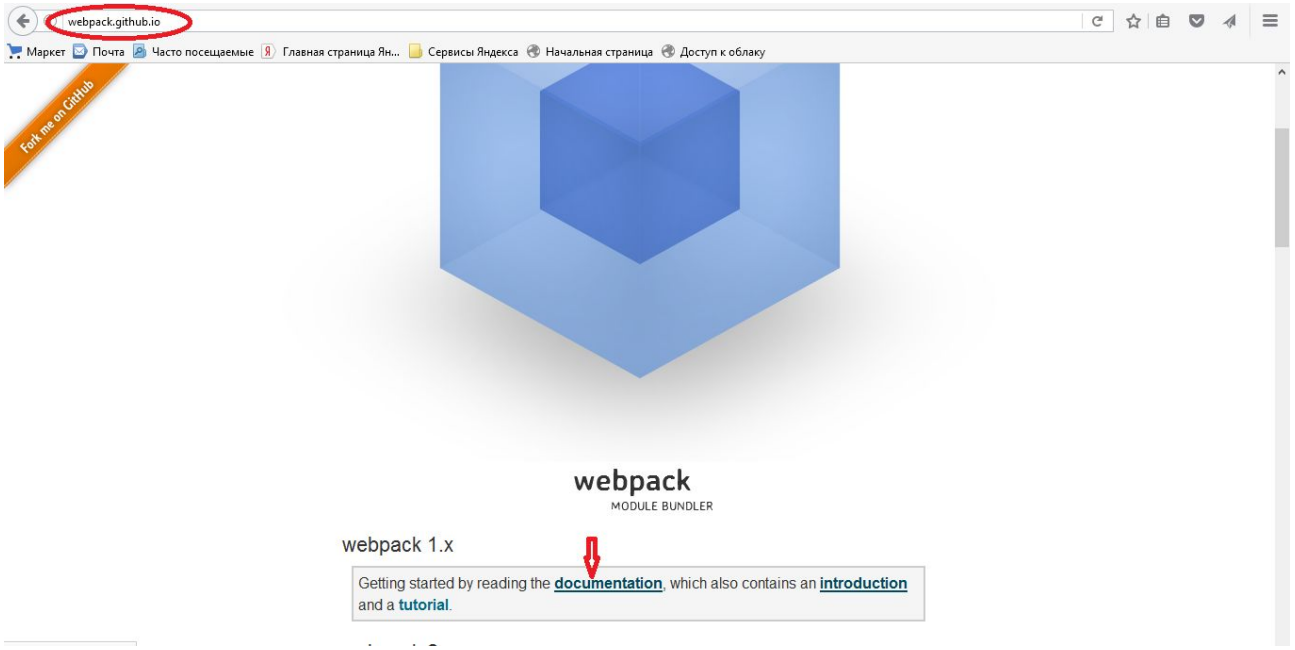
2. Webpack – утилита для сборки с широчайшими возможностями, позволяет компоновать модули, решать проблему взаимодействия различных компонентов в одном приложении и т.п., webpack берёт набор модулей с зависимостям и трансформирующих в наборы (бандлы), понятные для браузера. Трансформацией ресурсов занимаются загрузчики, которые являются сердцем webpack. Наглядно работа webpack выглядит так.



Вместо bower’a для установки и управления клиентскими зависимостями можно использовать стандартный Node Package Manager (npm) для установки и управления всеми фронтэнд-зависимостями

Конфигурируем webpack

Всю необходимую документацию про webpack можно найти на сайте http://webpack.github.io/



Webpack устанавливается глобально в систему и локально в проект как библиотека.

Устанавливаем глобально:

```
npm i -g webpack
```

Создаём простую сборку. Создаём два файла, содержащие наши модули:

Home.js

```
Let welcome=require ("./welcome"); //функция webpack
Welcome("home");
```

Welcome.js

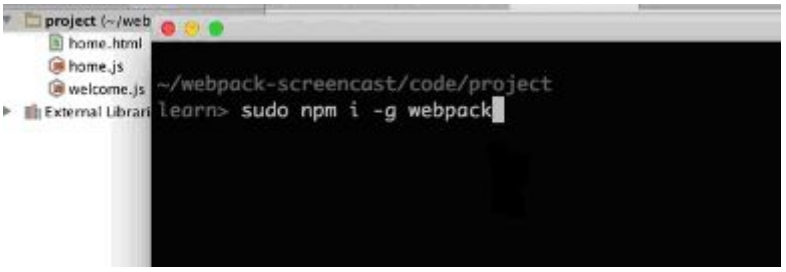
```
Module.export = function(messege){
Alert ('Welcome');
};
```

Модуль home.js должен запустить модуль welcome.js. Задача - собрать эти модули в единый файл, который будет подключаться в html.

Home.html

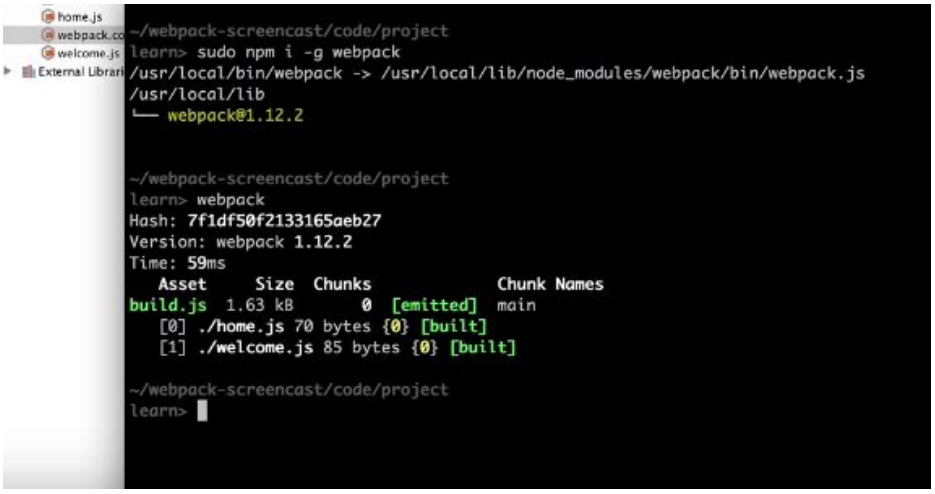
```
<!doctypehtml>
<html>
<head>
<scriptsrc="build.js"></script> //название файла для трансформации модулей
</head>
<body>
</body>
</html>
```

Для этого необходимо глобально установить webpackс помощью команды: npm i -g webpack



Внутри нашего проекта создаём новый файл webpack.config.js и создаём объект для импорта наших модулей

```
module.exports={
entry: "./home", // какой модуль собирать (home)
output: {          // куда выводить
filename: "bulding.js"}}}
```



Можно узнать информацию по сборке: какой файл, какие модули, время сборки. Webpack - большое количество инструментов. Вызываются командой weabpack--help

Устанавливаем статический сервер:

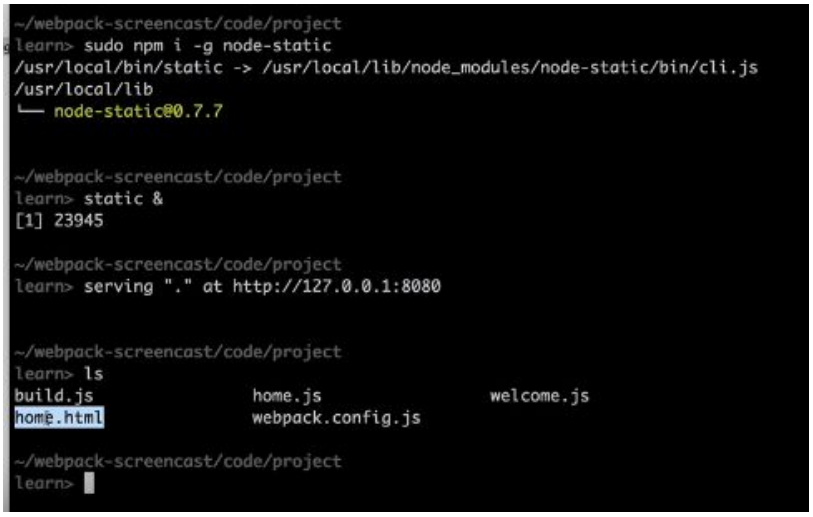
```
npm I –g node-static
```

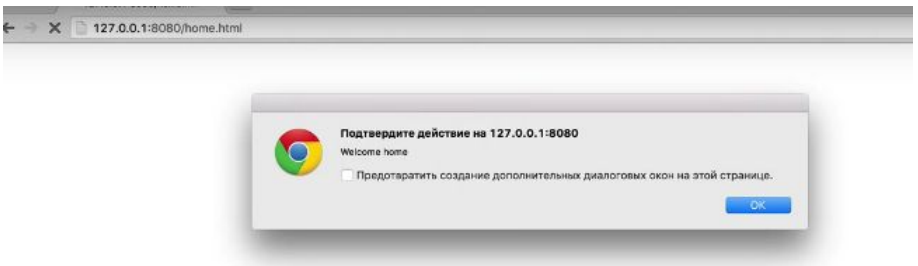
Запускаем его в нужной директории:

```
static
```

По выданному адресу будет показано содержимое директории.

Необходим для оптимизации кода, он будет следить за изменениями в коде и самостоятельно выполнять обновления. В данном случае он будет вызывать модуль home.



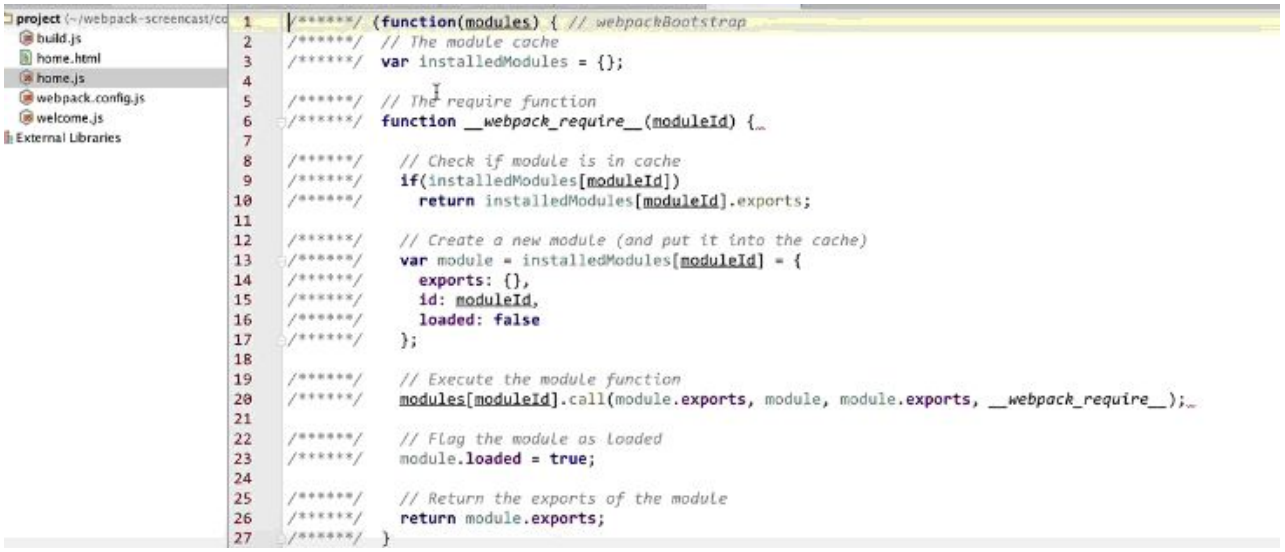


Сборка home имеет вид:

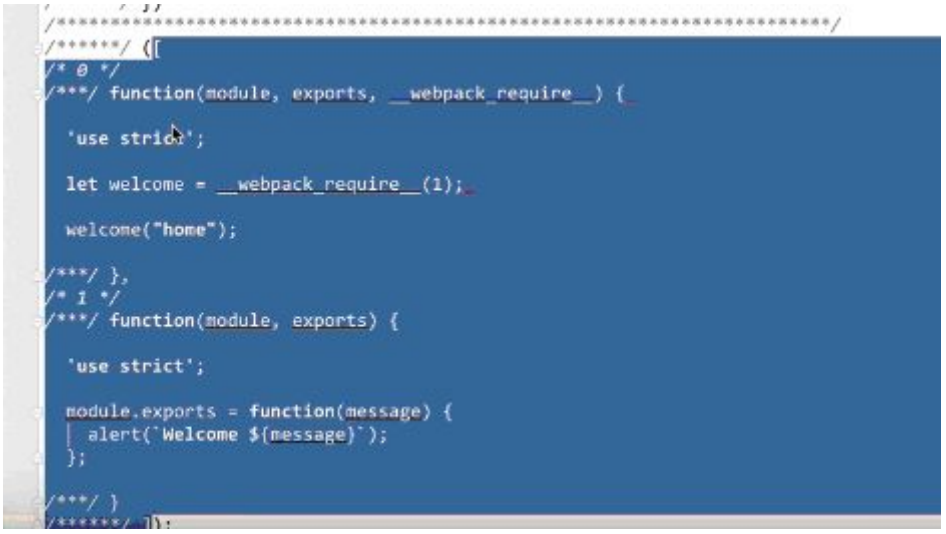
```
<!doctype html>
<html lang='ru'>
<head>
  <script src='build.js'></script>
</head>
<body>

</body>
</html>
```

Webpack-сборка будет состоять из внутренних функций с необходимыми инструментами:



В качестве аргумента функции будет массив, элементы массива-модули. Нулевой модуль –home.js. Следующему подключаемому модулю webpack присвоил значение 1(welcome.js)



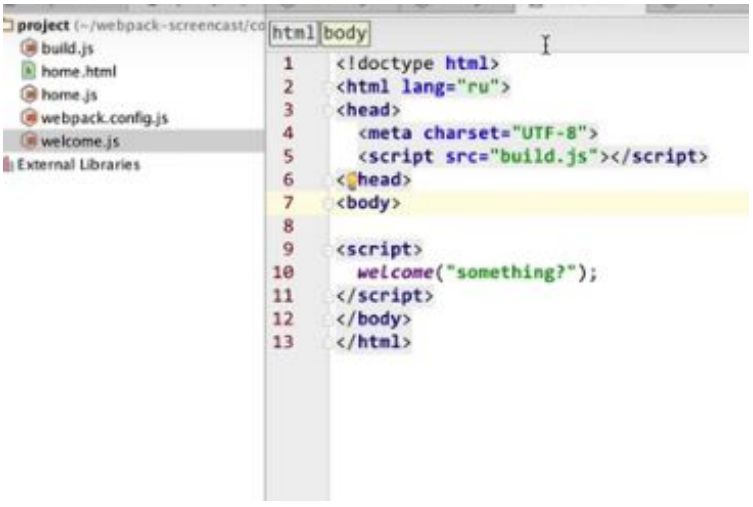
При запуске срабатывает внутренняя webpack-функция, поочередно подключающая модули.

Подключаем модули

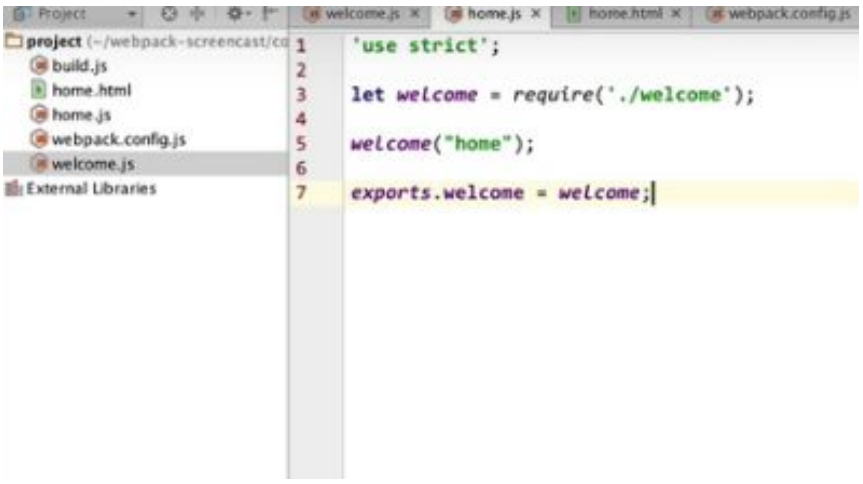
Особенностью webpack сборки является отсутствие глобальных переменных, т.е., если перейти на html-файл и запросить модуль, появится ошибка

Цель: вызвать модуль не отдельно от загрузочного файла.

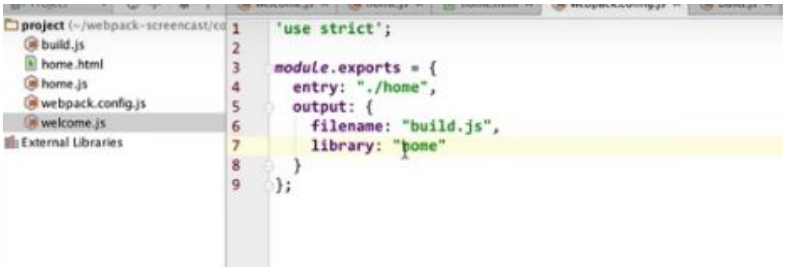
Загрузочный файл:



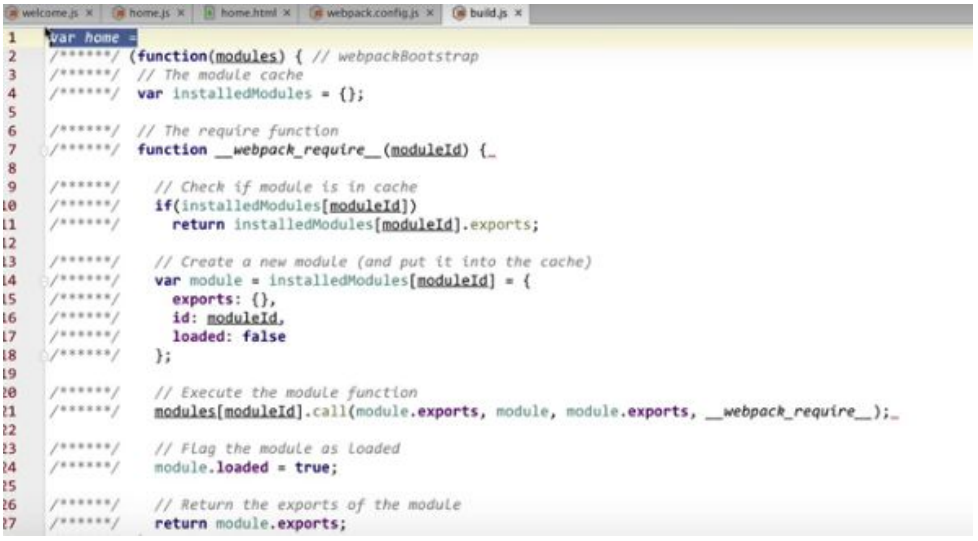
Для того, чтобы выполнить внешний вызов модуля, необходимо выполнить команду exports



В webpack.config добавляем новую переменную:



После сборки наш модуль будет помещён в эту переменную. Перезапускаем сборку командой webpack.



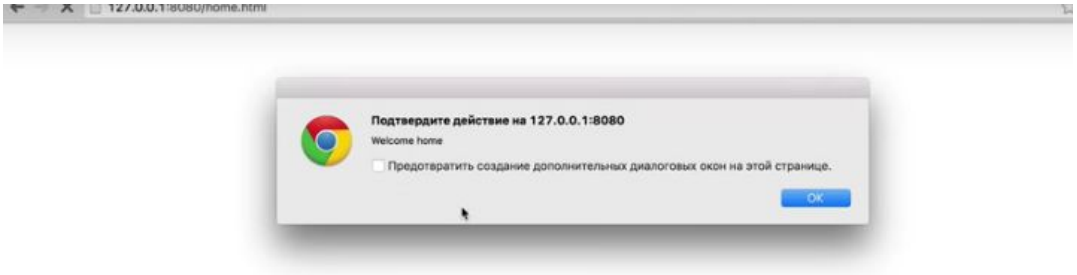
Появившаяся в сборке переменная будет вызывать нужный нам модуль.

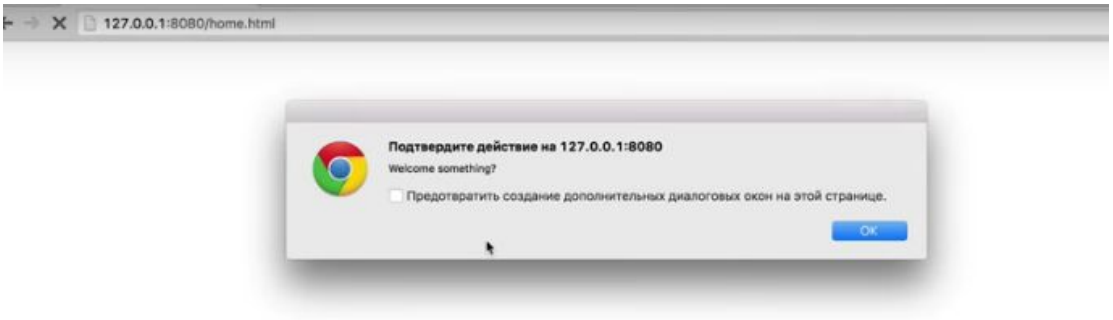
Редактируем файл home.html

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <script src="build.js"></script>
</head>
<body>

<script>
  home.welcome("something");
</script>
</body>
</html>
```

Результат:





В работе понадобятся свойства «dependencies», используются для упрощения установки модуля и «devDependencies». Модули из devDependencies и dependencies устанавливаются в одну и ту же папку node_modules/.

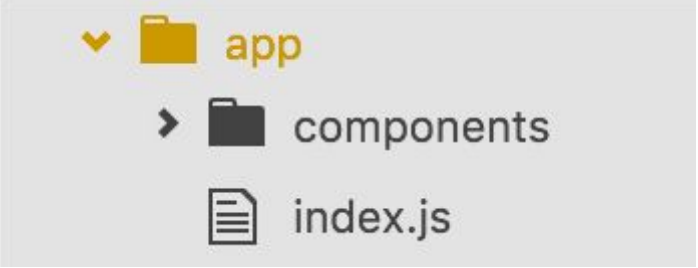
Определяем необходимые модули:

- babel-loader - используются для конвертации JS-файла, написанного на ES6, в совместимый с браузером E5;
- babel-preset-es2015 - используется для конвертации всех плагинов es2015;
- babel-preset-react - используется для конвертации react-плагинов;
- babel-preset-stage-0 - настраивается с начальными значениями на этапе “0-плагина”;

- browser-sync - дает возможность запустить сервер, на котором вы можете выполнять свои приложения. Отвечает за перезагрузку HTML/PHP-файлов. У него также есть возможность обновить/внедрить CSS и JavaScript-файлы в HTML;
- browser-sync-webpack-plugin - инструмент позволит обновить браузер, так как мы разрабатываем наши приложения и избежать обновления для css изменения;
- copy-webpack-plugin- отвечает за копирование файлы и каталоги в webpack;
- css-loader - webpack-плагин может загрузить файлы CSS, вставить небольшие изображения PNG в качестве URL-адреса данных и jpg файлы;
- file-loader - файловый загрузчик для webpack;
- html-loader - отвечает за экспорт в HTML;
- html-webpack-pligin - webpack плагин, который упрощает создание HTML-файлов. Это особенно полезно для webpack пакетов, которые включают в себя хэш в имени файла, который меняется при каждой компиляции;
- style-loader - Добавляет CSS к дом, вводя тег <Style>;
- url-loader - модуль-загрузчик URL для webpack Загрузчик URL-адрес работает как загрузчик файлов, но может возвращать URL-Адрес, если размер файла не превышает лимит.

Определяем структуру проекта

На данном этапе наш проект будет составлять из следующих компонентов.



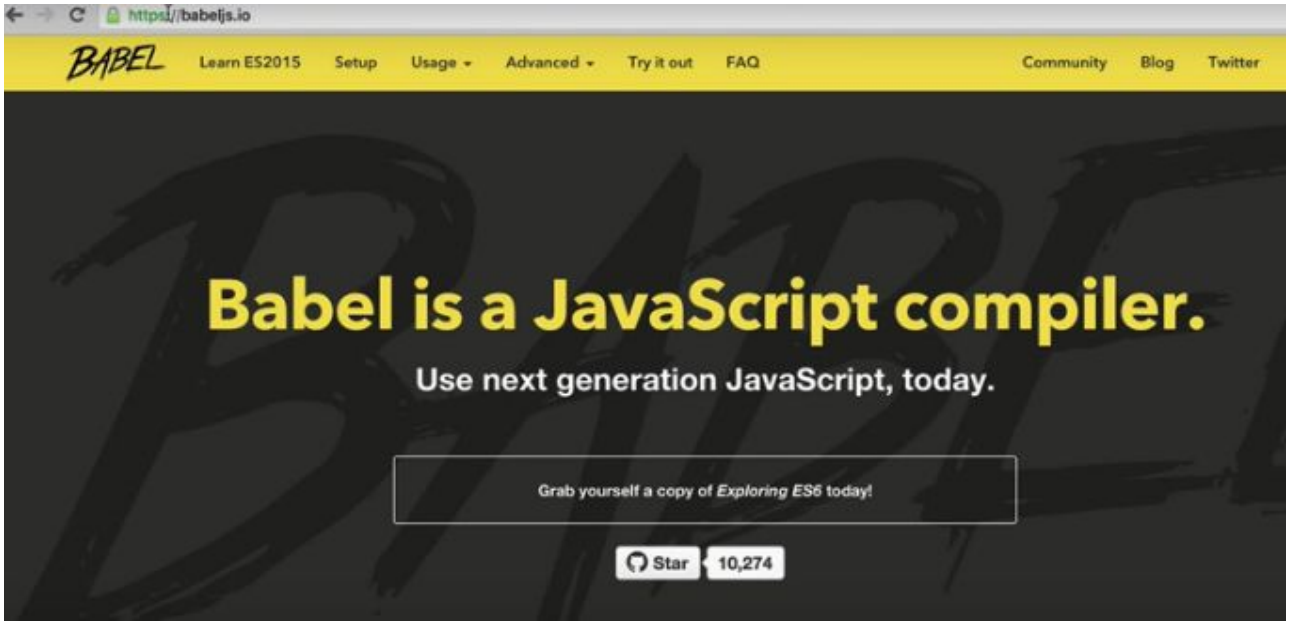
где components — все react компоненты;

Далее будем выбирать определенную модель приложения и продумать необходимые модули.

Настройка всех загрузчиков

Необходимо выполнить настройку загрузчика Babel для всех файлов JavaScript, чтобы транскомпилировать код на ES6 в код старой версии JavaScript, который смогут понять нынешние браузеры.

Общепринятый способ - использовать кроссбраузерно современный js - это babel. http://babeljs.io/ - получает современный js и превращает его в более старый/кроссбраузерный.



Для использования babel в нашем приложении воспользуемся концепцией loader'ов, которую предоставляет webpack.

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

К файлам, которые оканчиваются на js, нужно применять babel-loader. Loader - это модуль.

Создание заготовки главной страницы

Ранее был подключен webpack и buble. Выполним простейшую программу Hello world на основе выполненных подпрограмм. Создадим новый файла,:

1.js

```
LetHelloWorld=require ("./nachalo");
Welcome("first");
nachalo.js
Module.export = function(messege){
Document.word ('HelloWorld');
};
```

Модуль home.js должен запустить модуль welcome.js. Задача: собрать эти модули в единый файл, который будет подключаться в html.

Home.html

```
<!doctype html>
<html>
<head>
<scriptsrc="b1.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.6.15/browser.js"></script>
</head>
<body>// можно добавить стилей
</body>
</html>
```

Практика

Задача 1. Hello world.

Выполним простейшую программу Hello world на основе выполненных подпрограмм. Создадим новый файл:

1.js

```
LetHelloWorld=require ("./nachalo");
Welcome("first");
nachalo.js
Module.export = function(messege){
Document.word ('HelloWorld');
};
```

Модуль home.js должен запустить модуль welcome.js. Задача: собрать эти модули в единый файл, который будет подключаться в html.

Home.html

```
<!doctype html>
<html>
<head>
<scriptsrc="b1.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.6.15/browser.js"></script>
</head>
<body>//можно добавить стилей
</body>
</html>
```

Домашнее задание

1. Выполнить глобальное подключение webpack и необходимых загрузчиков.
2. Разработать структуру проекта. Продумать, какие компоненты будут входить в состав каждого js-элемента. Разработать компонент, выводящий ФИО разработчика.
3. Необходимо изменить один из файлов таким образом, чтобы в нём подключался новый компонент и выводился на главной странице.
4. * Для нового компонента необходимо прописать класс.
5. * Выполнить событие клик (onClick). При клике на компонент должна выводиться текущая дата.

Задачи со * являются заданиями повышенной сложности.

Дополнительные материалы

1. http://webpack.github.io/
2. http://babeljs.io/
3. https://webpack.github.io/
4. https://facebook.github.io/react/
5. https://www.codementor.io/tamizhvendan/tutorials/beginner-guide-setup-reactjs-environment-npm-babel-6-webpack-du107r9zr
6. https://facebook.github.io/react/docs/dom-elements.html
7. http://stackoverflow.com/questions/28511207/react-js-onclick-event-handler

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. habrahabr_full.complexdoc.ru
2. dnzl.ru
3. https://webpack.github.io/
4. https://facebook.github.io/react/
5. https://www.codementor.io/tamizhvendan/tutorials/beginner-guide-setup-reactjs-environment-npm-babel-6-webpack-du107r9zr
6. https://facebook.github.io/react/docs/dom-elements.html
7. http://stackoverflow.com/questions/28511207/react-js-onclick-event-handler