Webpack and React

Ustun Ozgur

2015-06-13 Cumartesi

Webpack

- Bundler (birleştirici)
- pack (birleştirmek)
- weble ilgili şeyleri (js, css, html) tek bir dosyaya birleştirme
- browserify benzeri
- React ekosisteminde daha popüler
- ES6 modüllerini, commonjs modüllerini ve amd modüllerini destekler.

CommonJS modülleri

- Node.js'teki modüller
- Her modül tek bir eleman ya da nesne export eder (dışa aktarır)

Tek eleman dışa aktarma item exported

- foo.js dosyasında: `module.exports = foo`
- Kullanım: `foo = require('./foo')` aynı klasördeki bar.js'te

Birden fazla eleman dışa aktarma ve içe aktarma

- `module.exports.bar = bar; modüle.exports.baz = baz` foo.js
- Kullanım: second.js'te
  `foo = require('./foo'); foo.bar()`

Alternatif sözdizim

- `module.exports = {bar: bar, baz: baz};`
- Dışa aktarmak için ES6 kısayolu:
  `module.exports = {bar, baz};`
- İçe aktarmak için ES6 kısayolu:
  `{bar, baz} = require('./foo')` (destructuring) (parçalama)

React bileşenleri

- JSX'ten JS oluşturmak için dönüştürücü modül
- webpack'ın loader'ları var
- jsx-loader ya da babel-loader
- babel-loader daha iyi.
- `npm install babel-loader --save`

webpack.config.js

- webpack çalıştırmak için bir config dosyasına ihtiyaç var
- Bu girdi ve çıktıyı belirler ve kullanılacak loaderları gösterir

```
module.exports = {
  entry: './component.js',
  output: {path: 'build', filename: 'bundle.js'},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }

}
```

- react, underscore gibi kütüphaneleri kullanabiliriz ve bunlar da nodemodules içine kopyalanacaktır.

Production Ayarları

- `webpack -p` ile çalıştırın.
- Dosya adı varsayılan olarak sabit. Değiştirmek için `[chunkname]`

```
module.exports = {
  entry: './component.js',
  output: {path: 'build',
           filename: 'bundle.[chunkhash].js'},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }
}
```

Çıktı dosyasının hash'ini almak

- `-j` bayrağı ile bir json özet al
- Bunu `jq` üzerinden geçirerek json çıktısındaki belli bir veriyi al

```
webpack -j | jq ".assetsByChunkName.main"
```

- jq şuradan indirilebilir: `http://stedolan.github.io/jq/`

Production ve Dev Ayarları Tek Bir Dosyada

```
module.exports = {
  entry: './component.js',
  output: {path: 'build',
           filename: (
             process.env.PROD ?
               'bundle.[chunkhash].js'
               : 'bundle')},
  module: {
    loaders: [
      { test: /\.js$/, exclude: /node_modules/,
        loader: "babel-loader"}
    ]
  }
}
```

webpack'i dev ve prod için kullanmak

- dev için: `webpack`
- prod için:
  `PROD=true webpack -p | jq ".assetsByChunkName.main" >`
- HTM:'de bu `bundle_hash` içeriğini gömün

Geliştirme için İpuçları

- İzleme modu: -w bayrağı
- Geliştirme için statik sunucu
- webpack-dev-server yükleyin ve çalıştırın.
- Şu websocket bağlantısına link verin:
- `<script src="http://localhost:8080/webpack-dev-server.js`
- Çıktı şu adreste geçerli olacaktır `http://localhost:8080/bundle.js`
- HTML'de bu adrese referans verin.
- JS dosyanızda değişiklik yapın ve dosyanın otomatik olarak güncellendiğini görün.

Alıştırma:

- Şuradaki örneği inceleyin: `/solutions/01_hello_world_webpack`
- Örnek üzerinden giderek webpack'ı çalıştırın ve otomatik reload yapan dev server'i çalıştırın.
- Hello world uygulamasında bazı değişiklikler yapın, otomatik reload'ın çalıştığından emin olun.
- todo uygulamasını bileşenlere ayırın, bu bileşenleri farklı dosyalara atın, ve tek bir dosyada bu bileşenleri require edin.
- webpack ıle bu dosyaları tek bir dosyada birleştirin.