PEMROGRAMAN WEB LANJUTAN

WEBPACK & BABEL

Disusun oleh :

- 15.111.0091 – Erick Kwantan
- 15.111.0695 – Vinson Chandra
- 15.111.1011 – Silfi Langie

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER MIKROSKIL

MEDAN

Daftar Isi

1.9 Hot Module Replacement

Hot Module Replacement (HMR) bertujuan untuk bertukar, menambah, ataupun menghapus modul saat aplikasi sedang dijalankan tanpa melakukan reload secara berulang-ulang. Secara signifikan, dapat mempercepat developer dalam melakukan pengembangan seperti:

- Tetap mempertahankan aplikasi selama reload dilakukan
- Menghemat waktu pengembangan dengan memperbarui hanya pada bagian yang diubah
- Styling lebih cepat.

Langkah-langkah yang memungkinkan modul untuk bertukar masuk dan keluar dari aplikasi:

- Aplikasi akan meminta runtime HMR untuk memeriksa pembaruan.
- Runtime secara asynchronous mengambil pembaruan dan memberitahu aplikasi.
- Aplikasi kemudian meminta runtime untuk menerapkan pembaruan.
- Runtime secara synchronous menerapkan pembaruan.

Kita dapat mengatur HMR agar dapat melakukan proses secara otomatis, ataupun dapat memilih untuk meminta interaksi dari developer agar pembaruan terjadi.

Selain melakukan asset secara normal, compiler perlu diberitahu agar memungkinkan melakukan pembaruan dari versi sebelumnya ke versi yang baru. Pembaruan tersebut terdiri dari dua bagian:

- Manifest yang diperbarui (JSON)
- Satu atau beberapa modul yang diperbarui (JavaScript)

Manifest berisi kumpulan kompilasi baru dan daftar semua modul yang diperbarui. Masing-masing modul berisi kode baru untuk semua modul yang diperbarui (atau ditambah penanda/flag yang menunjukkan bahwa modul telah dihapus). Compiler memastikan bahwa ID modul sebelumnya dengan ID modul yang baru konsisten. ID modul akan disimpan di memori, tetapi juga bisa disimpan dalam sebuah file JSON.

HMR adalah fitur opt-in yang hanya mempengaruhi modul yang berisi kode HMR. Salah satu contohnya adalah mem-patch styling melalui style-loader. Agar patch berfungsi, style-loader mengimplementasikan antarmuka HMR. Ketika menerima pembaruan melalui HMR, maka modul yang lama akan digantikan dengan yang baru.

Ketika mengimplementasikan antarmuka HMR dalam modul, kita dapat menjelaskan apa yang seharusnya terjadi jika sebuah modul diperbarui. Namun, dalam kebanyakan kasus, tidak wajib untuk menulis kode HMR di setiap modul. Jika modul tidak memiliki handler HMR, maka pembaruan akan mengalami bubble up. Ini berarti bahwa satu handler dapat memperbarui module tree secara lengkap. Jika satu modul dari module tree diperbarui, seluruh rangkaian dependensi akan dimuat ulang.

HMR dapat digunakan dalam pengembangan aplikasi sebagai pengganti dari LiveReload. webpack-dev-server mendukung hot mode yang mencoba memperbarui dengan HMR sebelum mencoba untuk me-reload seluruh halaman. Yang perlu dilakukan hanyalah memperbarui konfigurasi webpack-dev-server dan menggunakan webpack yang dibangun di dalam plugin HMR. Sebelum memulai membuat aplikasi, install terlebih dahulu:

```
npm install webpack webpack-cli webpack-dev-server --save-dev
```

Struktur folder project kita akan menjadi seperti:

```
webpack-demo
| - package.json
| - webpack.config.js
| - dist
  | - index.html
| - src
  | - index.js
```

Package.json

```
{
  "name": "belajar-webpack",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "webpack",
+     "start:dev": "webpack-dev-server"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^4.6.0",
    "webpack-cli": "^2.0.15"
  },
  "dependencies": {
    "webpack-dev-server": "^3.1.3"
  }
}
```

webpack.config.js

```
  const path = require('path')
+ const webpack = require('webpack')

  module.exports = {
    mode: 'development',

    entry: {
      app: './src/index.js'
    },

    devServer: {
      contentBase: './dist',
+     hot: true
    },

    plugins: [
+     new webpack.NamedModulesPlugin(),
+     new webpack.HotModuleReplacementPlugin()
    ],

    output: {
      filename: 'bundle.js',
      path: path.resolve(__dirname, 'dist')
    }
  };
```

dist/index.html

```
<!doctype html>
<html>
  <head>
    <title>Webpack - Hot Module Replacement</title>
  </head>
  <body>
    <script src="bundle.js"></script>
  </body>
</html>
```

src/index.js

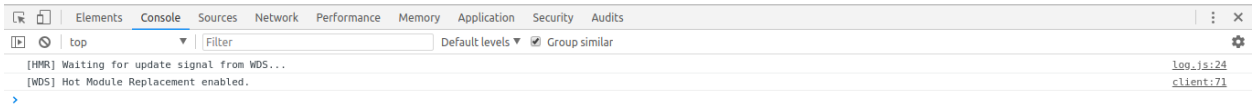
```
function component() {
  var element = document.createElement('div')

  element.innerHTML = 'Hello webpack'

  return element
}

document.body.appendChild(component())
```

Kemudian jalankan perintah `npm run start:dev`, sehingga hasilnya akan menjadi:

```
kwantz@kwantz:/var/www/mikroweb/web/belajar-webpack$ npm run start:dev

> belajar-webpack@1.0.0 start:dev /var/www/mikroweb/web/belajar-webpack
> webpack-dev-server

i ｢wds｣: Project is running at http://localhost:8080/
i ｢wds｣: webpack output is served from /
i ｢wds｣: Content not from webpack is served from ./dist
i ｢wdm｣: Hash: 28610e5bffd43145ef8b
Version: webpack 4.6.0
Time: 537ms
Built at: 2018-04-30 14:39:46
    Asset     Size  Chunks             Chunk Names
bundle.js  366 KiB     app  [emitted]  app
Entrypoint app = bundle.js
[./node_modules/events/events.js] 8.13 KiB {app} [built]
[./node_modules/loglevel/lib/loglevel.js] 7.68 KiB {app} [built]
[./node_modules/url/url.js] 22.8 KiB {app} [built]
[./node_modules/webpack/hot/log-apply-result.js] (webpack)/hot/log-apply-result.js 1.31 KiB {app} [built]
[0] multi (webpack)-dev-server/client?http://localhost:8080 webpack/hot/dev-server ./src/index.js 52 bytes {app} [built]
[./node_modules/url/util.js] 314 bytes {app} [built]
[./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 7.75 KiB {app} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3.58 KiB {app} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 1.05 KiB {app} [built]
[./node_modules/webpack-dev-server/node_modules/strip-ansi/index.js] (webpack)-dev-server/node_modules/strip-ansi/index.js 161 bytes {app} [built]
[./node_modules/webpack/hot sync ^\.\/log$] (webpack)/hot sync nonrecursive ^\.\/log$ 170 bytes {app} [built]
[./node_modules/webpack/hot/dev-server.js] (webpack)/hot/dev-server.js 1.66 KiB {app} [built]
[./node_modules/webpack/hot/emitter.js] (webpack)/hot/emitter.js 77 bytes {app} [built]
[./node_modules/webpack/hot/log.js] (webpack)/hot/log.js 1.03 KiB {app} [built]
[./src/index.js] 168 bytes {app} [built]
    + 13 hidden modules
i ｢wdm｣: Compiled successfully.
```

Aplikasi akan berjalan pada halaman browser dengan url http://localhost:8080/ dimana tampilannya akan menjadi seperti ini:

Hello webpack



Mari kita tambah sedikit kata pada `src/index.js`:

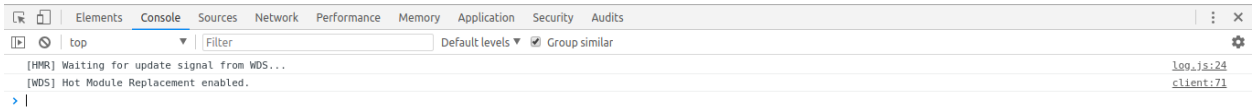
```
function component() {
  var element = document.createElement('div')

-  element.innerHTML = 'Hello webpack'
+  element.innerHTML = 'Hello webpack, my name is Erick'

  return element
}

document.body.appendChild(component())
```

Pada terminal / command prompt:

```
i ｢wdm｣: Compiling...
i ｢wdm｣: Hash: 08c018c75358d9652196
Version: webpack 4.6.0
Time: 41ms
Built at: 2018-04-30 14:45:22
                                Asset       Size  Chunks             Chunk Names
                            bundle.js     366 KiB     app  [emitted]  app
app.28610e5bffd43145ef8b.hot-update.js  462 bytes     app  [emitted]  app
  28610e5bffd43145ef8b.hot-update.json   45 bytes          [emitted]
Entrypoint app = bundle.js app.28610e5bffd43145ef8b.hot-update.js
[./src/index.js] 186 bytes {app} [built]
    + 27 hidden modules
i ｢wdm｣: Compiled successfully.
```

Kemudian kita lihat kembali di browser, tanpa kita melakukan refresh, maka tampilan nya akan berubah menjadi:

Hello webpack, my name is Erick



Kita juga dapat menggunakan HMR pada stylesheet css, dengan caranya adalah install terlebih dahulu `npm install --save-dev style-loader css-loader`, lalu edit file dengan beberapa tambahan berikut:

webpack.config.js

```
  const path = require('path')
  const webpack = require('webpack')

  module.exports = {
    mode: 'development',

    entry: {
      app: './src/index.js'
    },

    devServer: {
      contentBase: './dist',
      hot: true
    },

+   module: {
+     rules: [
+       {
+         test: /\.css$/,
+         use: ['style-loader', 'css-loader']
+       }
+     ]
+   },

    plugins: [
      new webpack.NamedModulesPlugin(),
      new webpack.HotModuleReplacementPlugin()
    ],

    output: {
      filename: 'bundle.js',
      path: path.resolve(__dirname, 'dist')
    }
  };
```

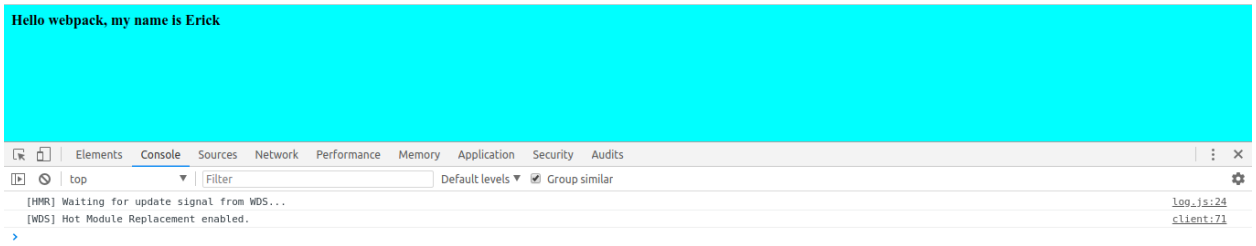
src/index.js

```
+ import './index.css'

  function component() {
    var element = document.createElement('div')

    element.innerHTML = 'Hello webpack, my name is Erick'

    return element
  }

  document.body.appendChild(component())
```

src/index.css

```
body {
    background: cyan;
    font-weight: bold;
}
```

Jalankan perintah npm run start:dev dan buka browser dan ketik http://localhost:8080



Lalu, kita edit file `src/index.css` dengan mengganti warna latarnya menjadi warna kuning: