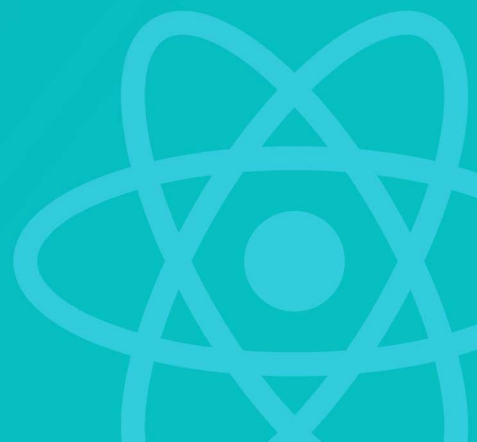


# Pengenalan ReactJS

Donny Stark



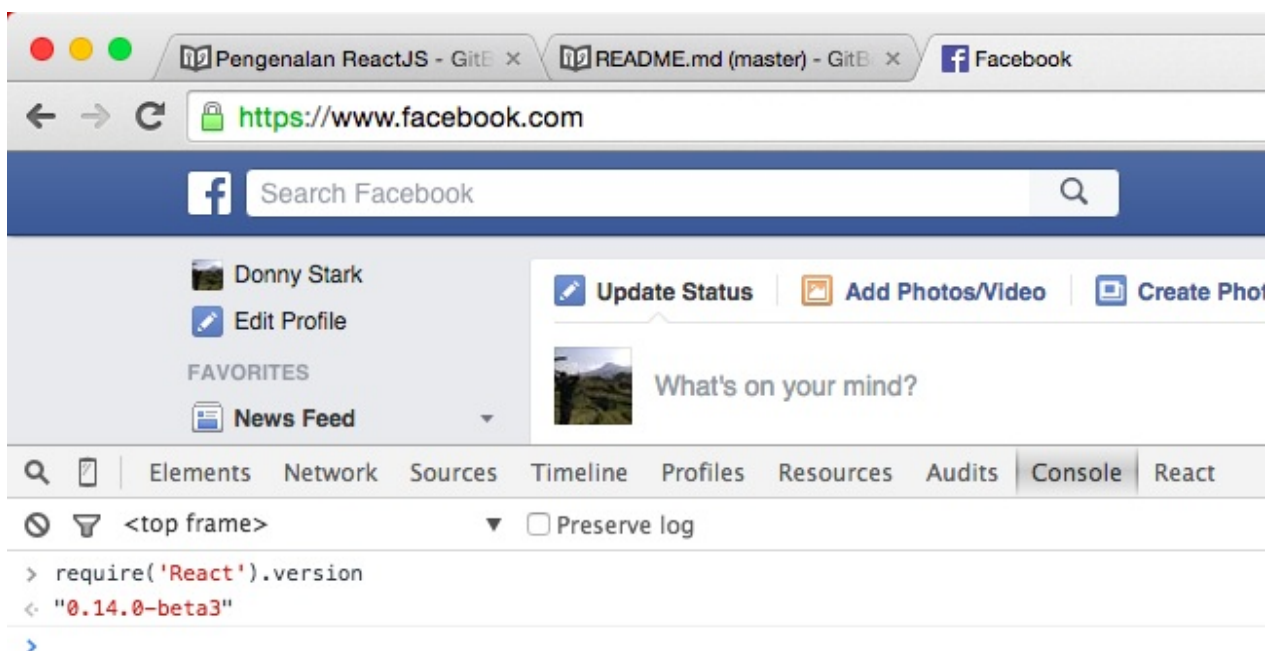
# Table of Contents

---

1. [Pendahuluan](#)
  - i. [Persiapan sebelum berkenalan](#)
  - ii. [Hal Baru](#)
2. [Instalasi ReactJS](#)
  - i. [Instalasi Perkakas](#)
  - ii. [Buat Projek Baru](#)
  - iii. [Build System dengan Gulp](#)
3. [Mengenal ReactJS](#)
  - i. [Spesifikasi Komponen](#)
4. [Learning by Doing](#)
  - i. [Cara Kerja](#)

# Pengenalan ReactJS

ReactJS adalah framework untuk membangun User Interface yang dibuat oleh Facebook. Akhir-akhir ini Facebook sedang gencar-gencarnya membagikan hasil hacking mereka selama membangun facebook.com, tak hanya dari ranah Pengembangan Web, bahkan hingga perangkat mobile iOS dan Android dengan React Native. Masih banyak lagi hasil hacking Facebook yang sedang mereka bagikan. Hal inilah yang saya suka dari Facebook, mereka membagikan sesuatu yang sudah mereka buktikan performancenya.



Yap.. mereka memakai versi beta di Facebook sendiri sebelum mempublikasikannya. Hal ini yang membuat saya kagum dengan Facebook. Sejak itu saya memfokuskan diri untuk berkiblat ke Facebook dalam hal Development. Sudah banyak yang dipublikasikan oleh facebook seperti yang akan kita bahas disini, ReactJS. Masih banyak yang sudah di open source kan oleh Facebook seperti React Native (Membuat aplikasi native dengan ReactJS, iOS dan Android), GraphQL, Relay, Flux, ImmutableJS dan perkiraan makin banyak lagi. Tapi saya akan membahasnya dibuku terpisah. Agar kita bisa mengikuti semua framework tersebut, sangat disarankan untuk memulainya dari ReactJS. Inilah kenapa saya menuliskan ReactJS terlebih dahulu, agar kita bisa mengikuti perkembangan yang sedang dipakai di Facebook.

# Persiapan

---

Sebelum memulai, ada beberapa persiapan yang harus terpenuhi terlebih dahulu.

## NodeJS

Sebenarnya tanpa NodeJS kita bisa membuat aplikasi web dengan ReactJS, namun kita disini tidak hanya mengejar "bisa jalan" tetapi juga mengejar "berjalan dengan optimal". Maka saya tekankan untuk mengikuti best practice yang sudah ada, yakni dengan bantuan nodejs untuk keperluan build system seperti uglify file javascript, penyatuan file, compile less ke css ataupun sass ke css dan masih banyak lagi kegunaan nodejs untuk memudahkan kita para pengembang web.

## NPM

NPM sudah jadi satu dengan NodeJS, bagi anda yang sudah terbiasa dengan npm anda bisa skip ke persiapan berikutnya. Bagi anda yang masih belum mengenal apa itu NPM, akan saya kupas dengan singkat. Sesuai dengan namanya Node Package Manager, tugasnya adalah sebagai package manager untuk ekosistem nodejs. Sama seperti composer kalau di php. Mendeklarasikan dan memuat semua dependensi modul suatu proyek, yang akan disimpan di package.json di root folder suatu proyek. Jika masih belum familiar anda masih bisa mengikuti buku ini, nanti pasti akan terbiasa.

## Browserify

Browserify membantu kita dalam menata susunan file javascript kita, bahkan hingga mengecilkan file javascript kita atau sering disebut dengan minification. Jika anda masih bingung anda tetap bisa mengikuti buku ini, seperti biasa.. pasti nanti akan terbiasa. Ikuti saja.

Sebenarnya ada alternatif lain seperti Webpack, tapi saya rasa Browserify lebih stabil untuk belajar.

## Gulp

Gulp membantu kita dalam task automation, melakukan beberapa task dengan satu kali

perintah. Seperti build process, compile less/sass dan lain sebagainya. Seperti biasa kita mengikuti best practice yang ada. Untuk saat ini task automation yang stabil untuk ekosistem nodejs adalah gulp.

### **Koneksi Internet Stabil**

Koneksi internet dibutuhkan untuk menginstal dependensi yang akan kita pakai dalam ekosistem pengembangan kita, seperti reactjs, browserify, gulp, compiler less dan lain sebagainya.

# Hal Baru

---

Banyak hal baru yang digunakan didalam ReactJS, seperti EcmaScript versi ke 6 atau sering disebut dengan es6. JSX untuk penyusunan komponen, state dan props. Untuk menghindari kesalah pahaman saat belajar, lebih baik mengenal hal hal baru ini lebih dahulu.

## ES6

---

Javascript yang sering kita gunakan adalah ES5, masih banyak sintaks jelek yang ada, bahkan class pun tidak ada disini. Untuk mengindahkan struktur code base kita, ReactJS menyarankan untuk menggunakan ES6. Seperti biasa, ikuti best practice yang sudah ada.

## JSX

---

JSX adalah XML dalam javascript. Pertama saya dengar hal ini.. Bukannya sangat buruk jika template kita dijadikan satu dengan logic? Tunggu.. disini, di ReactJS, tidak ada yang namanya template. Yang dikenal disini adalah komponen. seperti `<h1></h1>`, `<img src="" />`. Bayangkan betapa mudahnya jika anda bisa menyusun google map view hanya dengan `<GoogLeMap long="xxx" lat="yyy" />`. Di ReactJS anda bisa!!

Didalam JSX juga ada beberapa variabel yang bisa kita gunakan, yakni state dan props. Untuk apa mereka?

## Props

Props adalah properti yang disertakan dalam suatu komponen dari parentnya. Lebih bersifat inherit. Contohnya, `<GoogLeMap long="xxx" lat="yyy" />`, dalam class GoogLeMap, kita bisa menggunakan atribut yang disertakan didalam komponen tersebut dengan memanggil `this.props`, contohnya `this.props.lat`.

## State

State adalah variabel internal, seperti props tapi props adalah variabel yang datang dari

luar, sedangkan state adalah variabel yang sudah ada didalam suatu komponen. cara memanggilnya yakni dengan `this.state.xxx` .





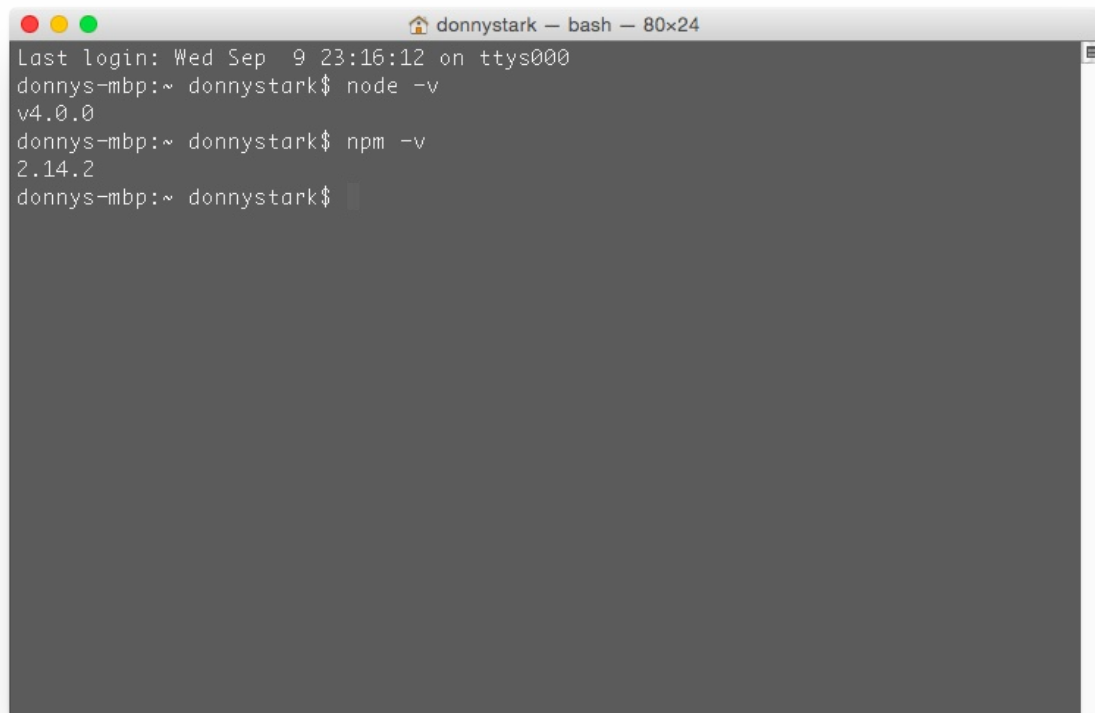
# Instalasi Perkakas

---

Seperti yang sudah dibahas diatas, perkakas yang kita perlukan adalah NodeJS, NPM, Browserify, dan Gulp.

## Install NodeJS dan NPM

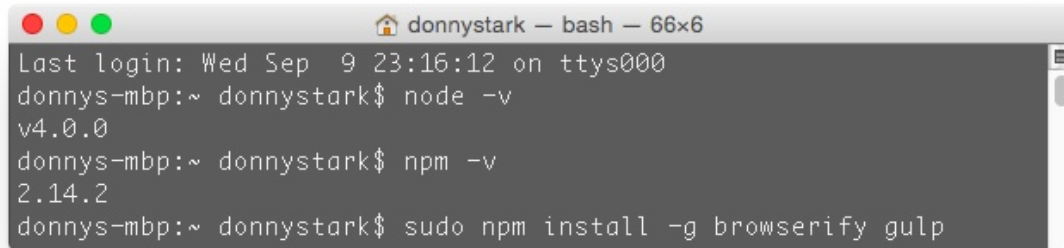
Untuk memasang NodeJS silakan menuju homepage NodeJS untuk menginstal, disarankan untuk menggunakan versi terbaru dari NodeJS. Saat buku ini ditulis, versinya adalah v4.0.0. Sedangkan NPM versi 2.14.2.

A terminal window titled 'donnystark — bash — 80x24' showing the installation of NodeJS and NPM. The terminal output is as follows:

```
Last login: Wed Sep 9 23:16:12 on ttys000
donnys-mbp:~ donnystark$ node -v
v4.0.0
donnys-mbp:~ donnystark$ npm -v
2.14.2
donnys-mbp:~ donnystark$
```

## Install Browserify dan Gulp

Browserify dan Gulp harus diinstall secara global, yakni dengan menambahkan `-g` saat menginstal. Setelah NodeJS dan NPM terinstal, instal browserify dan gulp dengan perintah `$ npm install -g browserify gulp`.



```
donnystark — bash — 66x6
Last login: Wed Sep  9 23:16:12 on ttys000
donnys-mbp:~ donnystark$ node -v
v4.0.0
donnys-mbp:~ donnystark$ npm -v
2.14.2
donnys-mbp:~ donnystark$ sudo npm install -g browserify gulp
```

## Projek Baru

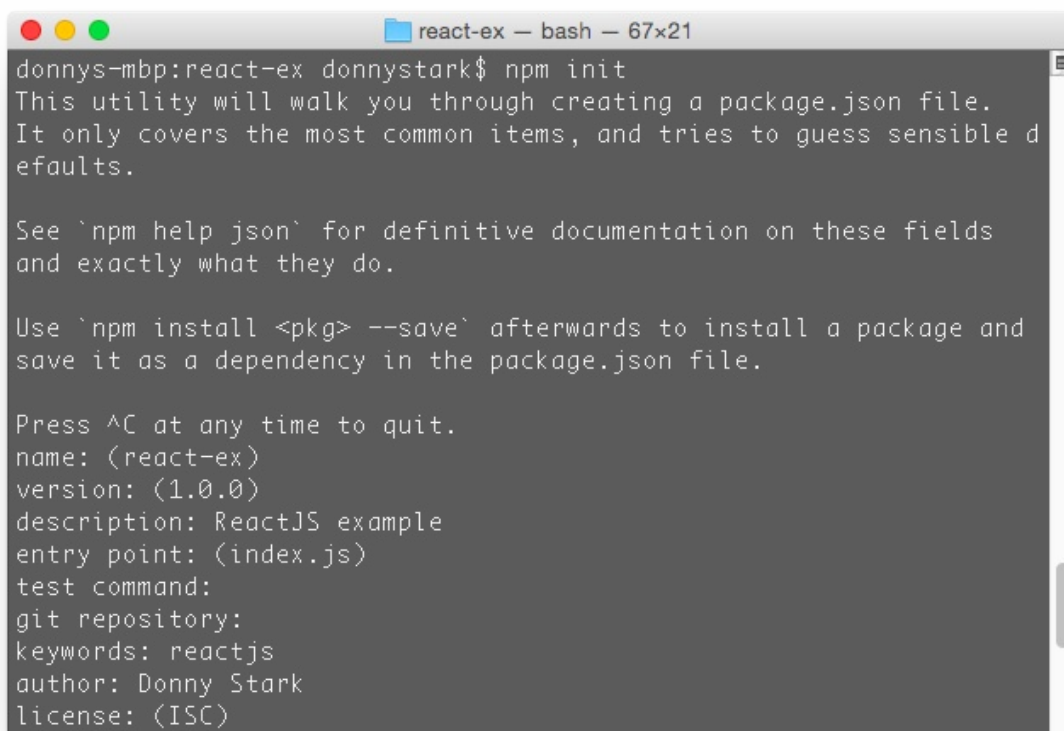
---

Semua proses dalam buku ini sudah tersedia di github, silakan clone repositori ini <https://github.com/rromadhoni/contoh-toko-online-reactjs> karena saya akan sangat jarang memasukkan kode dibuku ini agar tetap sederhana dan tidak panjang. Jika anda ingin menulis ulang silahkan, karena tulis ulang kode bisa membuat kita semakin mengerti alurnya. Tapi jika tidak ingin repot, bisa copy-paste saja direpositori ini.

Jika anda sudah terbiasa membuat project baru di ekosistem NodeJS, anda bisa melewati chapter ini. Tapi jika belum silakan ikuti sampai selesai.

### Buat Projek

Untuk mengawali projek, buat projek npm terlebih dahulu dengan perintah `npm init` lalu masukkan detail yang anda inginkan seperti gambar berikut.

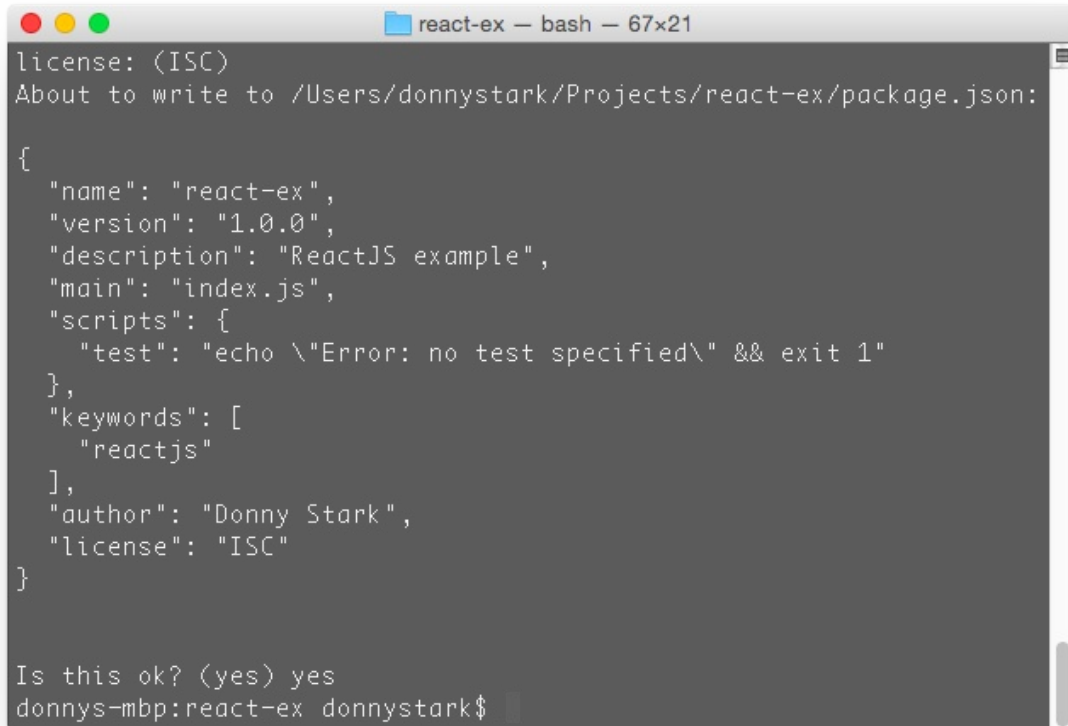


```
react-ex — bash — 67x21
donnys-mbp:react-ex donnystark$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (react-ex)
version: (1.0.0)
description: ReactJS example
entry point: (index.js)
test command:
git repository:
keywords: reactjs
author: Donny Stark
license: (ISC)
```



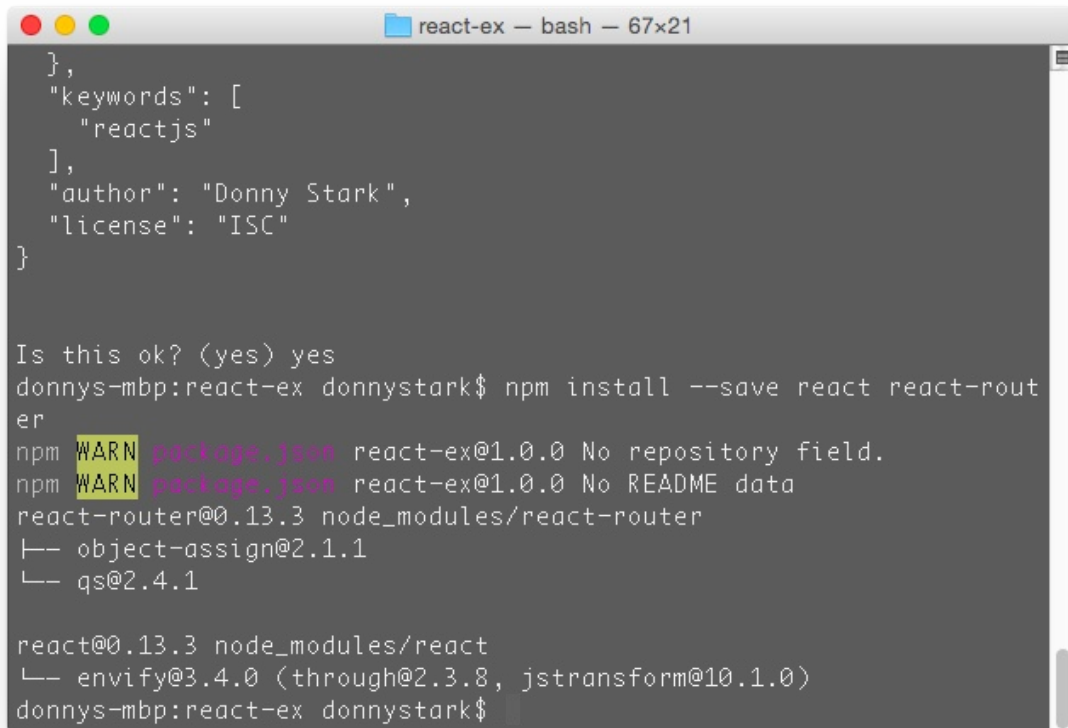
```
react-ex — bash — 67x21
license: (ISC)
About to write to /Users/donnystark/Projects/react-ex/package.json:

{
  "name": "react-ex",
  "version": "1.0.0",
  "description": "ReactJS example",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "reactjs"
  ],
  "author": "Donny Stark",
  "license": "ISC"
}

Is this ok? (yes) yes
donnys-mbp:react-ex donnystark$
```

## Install Dependency

Dependency adalah semua module yang bisa dicari di <https://www.npmjs.com/> Module yang kita butuhkan akan disediakan oleh npm. Dengan perintah: `npm install --save react react-router` react adalah module ReactJS yang akan kita pakai, react-router adalah router yang digunakan react dalam navigasi single web page. Sedangkan opsi `--save` adalah instruksi agar modul tersebut disimpan di package.json

A terminal window titled 'react-ex — bash — 67x21' showing the installation of React and React Router. The window has a dark background with light-colored text. The terminal output shows a JSON object for package.json, a confirmation prompt, and the command 'npm install --save react react-router'. The output of the command shows two warnings about missing repository and README data, and the installation of react-router@0.13.3 and its dependencies. Finally, it shows the installation of react@0.13.3 and its dependencies.

```
},
  "keywords": [
    "reactjs"
  ],
  "author": "Donny Stark",
  "license": "ISC"
}

Is this ok? (yes) yes
donnys-mbp:react-ex donnystark$ npm install --save react react-router
npm WARN package.json react-ex@1.0.0 No repository field.
npm WARN package.json react-ex@1.0.0 No README data
react-router@0.13.3 node_modules/react-router
├─ object-assign@2.1.1
└─ qs@2.4.1

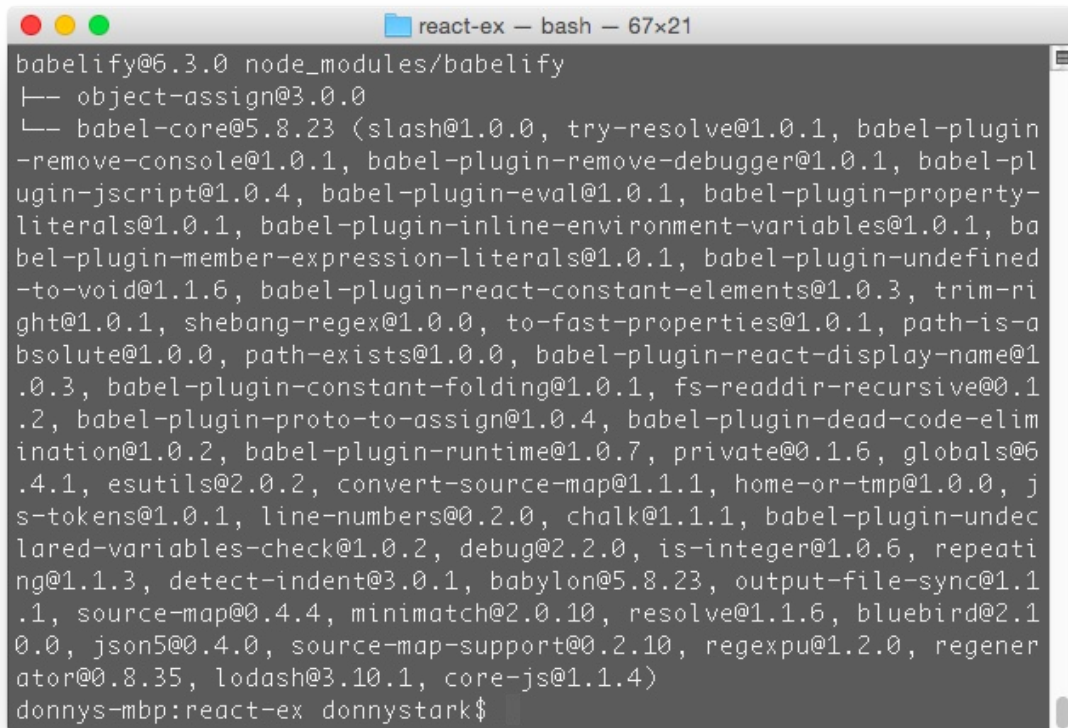
react@0.13.3 node_modules/react
└─ envify@3.4.0 (through@2.3.8, jstransform@10.1.0)
donnys-mbp:react-ex donnystark$
```

## Install Development Dependency

Disini kita akan mengunduh modul-modul untuk keperluan development seperti gulp, browserify, compiler less dan lain sebagainya. Sama seperti menginstall dependency, hanya saja dengan parameter `--save-dev`

```
npm install --save-dev browserify babelify gulp gulp-babel gulp-connect gulp-less
gulp-sourcemaps gulp-streamify gulp-uglify reactify vinyl-source-stream watchify
babelify
```

Jika berhasil akan muncul gambar seperti berikut:



```
react-ex — bash — 67x21
babelify@6.3.0 node_modules/babelify
├─┬ object-assign@3.0.0
│   └─┬ babel-core@5.8.23 (slash@1.0.0, try-resolve@1.0.1, babel-plugin-
│       remove-console@1.0.1, babel-plugin-remove-debugger@1.0.1, babel-pl
│       ugin-jscript@1.0.4, babel-plugin-eval@1.0.1, babel-plugin-property-
│       literals@1.0.1, babel-plugin-inline-environment-variables@1.0.1, ba
│       bel-plugin-member-expression-literals@1.0.1, babel-plugin-undefined
│       -to-void@1.1.6, babel-plugin-react-constant-elements@1.0.3, trim-ri
│       ght@1.0.1, shebang-regex@1.0.0, to-fast-properties@1.0.1, path-is-a
│       bsolute@1.0.0, path-exists@1.0.0, babel-plugin-react-display-name@1
│       .0.3, babel-plugin-constant-folding@1.0.1, fs-readdir-recursive@0.1
│       .2, babel-plugin-proto-to-assign@1.0.4, babel-plugin-dead-code-elim
│       ination@1.0.2, babel-plugin-runtime@1.0.7, private@0.1.6, globals@6
│       .4.1, esutils@2.0.2, convert-source-map@1.1.1, home-or-tmp@1.0.0, j
│       s-tokens@1.0.1, line-numbers@0.2.0, chalk@1.1.1, babel-plugin-undec
│       lared-variables-check@1.0.2, debug@2.2.0, is-integer@1.0.6, repeati
│       ng@1.1.3, detect-indent@3.0.1, babylon@5.8.23, output-file-sync@1.1
│       .1, source-map@0.4.4, minimatch@2.0.10, resolve@1.1.6, bluebird@2.1
│       0.0, json5@0.4.0, source-map-support@0.2.10, regexpu@1.2.0, regener
│       ator@0.8.35, lodash@3.10.1, core-js@1.1.4)
└─┬ donnys-mbp:react-ex donnystark$
```

Okay kita sudah berhasil membuat projek baru. Selanjutnya kita akan membuat build system dengan gulp.

# Build System dengan Gulp

---

Build System adalah sistem yang kita bangun agar semua task development seperti compile less/sass, uglify javascript, css minify dsb bisa menjadi satu kali perintah saja. Tentu saja mengikuti best practice yang ada.

React memiliki beberapa perlakuan khusus dalam build system, kita akan menggunakan Gulp sebagai task runnernya dan Browserify sebagai module bundlernya. Bagi yang belum tau Browserify, cara kerjanya seperti require.js tetapi menggunakan sintaks nodejs dalam memanggil modul. Namun tidak seperti require.js yang memuat semua modul secara asinkron, Browserify menyambung semua modul menjadi satu file. Anda tetap bisa mengikuti buku ini meski belum mengenal Browserify terlalu dalam.

Tetap di project yang sudah kita buat diatas, buat file baru di root folder dengan nama gulpfile.js dan sesuaikan dengan gulpfile.js yang ada direpositori.

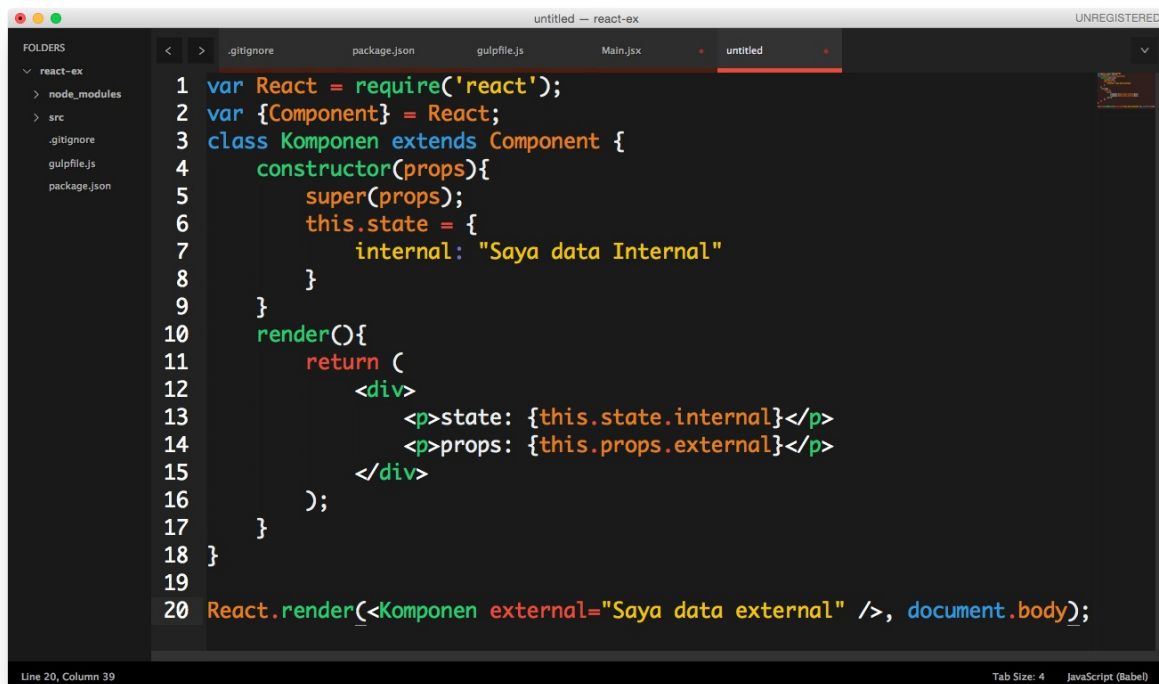
<https://github.com/rromadhoni/contoh-toko-online-reactjs/commit/d98e561745d444b9c2d9bed0f6e440e77ba8c1b8>

# Mengenal ReactJS

Tidak seperti framework pada umumnya, ReactJS memberi kita sedikit API. Yang mana sangat memudahkan pengembang dalam mempelajarinya. Jika mempelajari framework lain harus menghafal semua API, di ReactJS kita hanya butuh memahami konsep kerjanya saja. Jika sudah paham, waktu kita hanya tersita untuk menata logika aplikasi kita saja. ReactJS adalah framework berdasarkan komponen, bukan template. Karena ReactJS adalah framework untuk membangun User Interface.

Dalam suatu komponen ada dua variabel data, yakni state dan props. state adalah data internal dari suatu komponen, sedangkan props adalah data yang diwariskan dari komponen parentnya. Untuk lebih jelasnya kita akan buat komponen baru. Di chapter ini anda tidak diharuskan menulis kode, cukup lihat saja indahnya sintaks.

Berikut ini adalah dasar code base ReactJS dalam membuat suatu Komponen.



```

1 var React = require('react');
2 var {Component} = React;
3 class Komponen extends Component {
4   constructor(props){
5     super(props);
6     this.state = {
7       internal: "Saya data Internal"
8     }
9   }
10  render(){
11    return (
12      <div>
13        <p>state: {this.state.internal}</p>
14        <p>props: {this.props.external}</p>
15      </div>
16    );
17  }
18 }
19
20 React.render(<Komponen external="Saya data external" />, document.body);
  
```

Yap, sesingkat itu. Mari kita urai disini tiap barisnya.

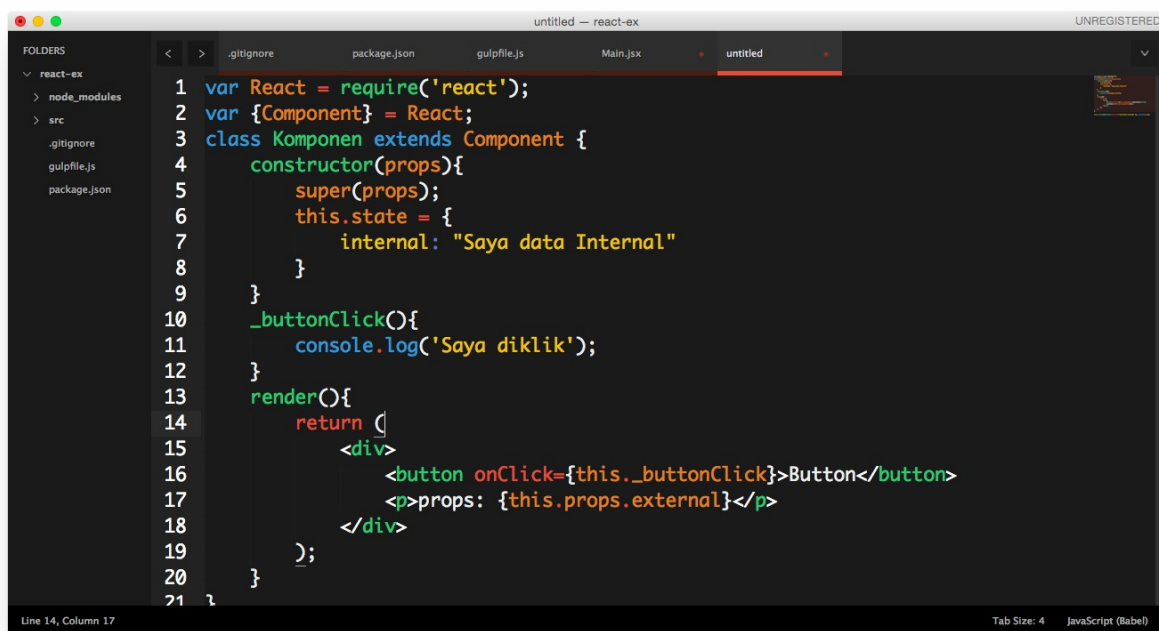
1. Muat ReactJS
2. Ambil Component dari React.Component (destrukturisasi dari ES6).



3. Buat komponen dengan deklarasi class baru bernama Komponen dengan meng-extends Component.
4. Konstruktorkan, dengan parameter props.
5. Sampaikan props ke parent.
6. Definisikan state, data internal.
7. Objek javascript biasa.
8. Render method, WAJIB ada.
9. render() method harus mengembalikan value berupa JSX. dikurung dengan ().
10. JSX, bisa berupa komponen lain ataupun komponen html seperti div, h1, p dan lain-lain, sedangkan untuk menampilkan data gunakan `{}`.
11. Render Komponen ke document.body dengan props external.

Selesai, sederhana bukan? Jika anda masih ragu silakan dipahami lagi. Karena pattern inilah yang nantinya akan dipakai disemua komponen yang akan kita buat. Saya tekankan jangan dipraktekkan terlebih dahulu, cukup mengerti saja.

Lalu untuk event?



```
1 var React = require('react');
2 var {Component} = React;
3 class Komponen extends Component {
4   constructor(props){
5     super(props);
6     this.state = {
7       internal: "Saya data Internal"
8     }
9   }
10  _buttonClick(){
11    console.log('Saya diklik');
12  }
13  render(){
14    return (
15      <div>
16        <button onClick={this._buttonClick}>Button</button>
17        <p>props: {this.props.external}</p>
18      </div>
19    );
20  }
21 }
```

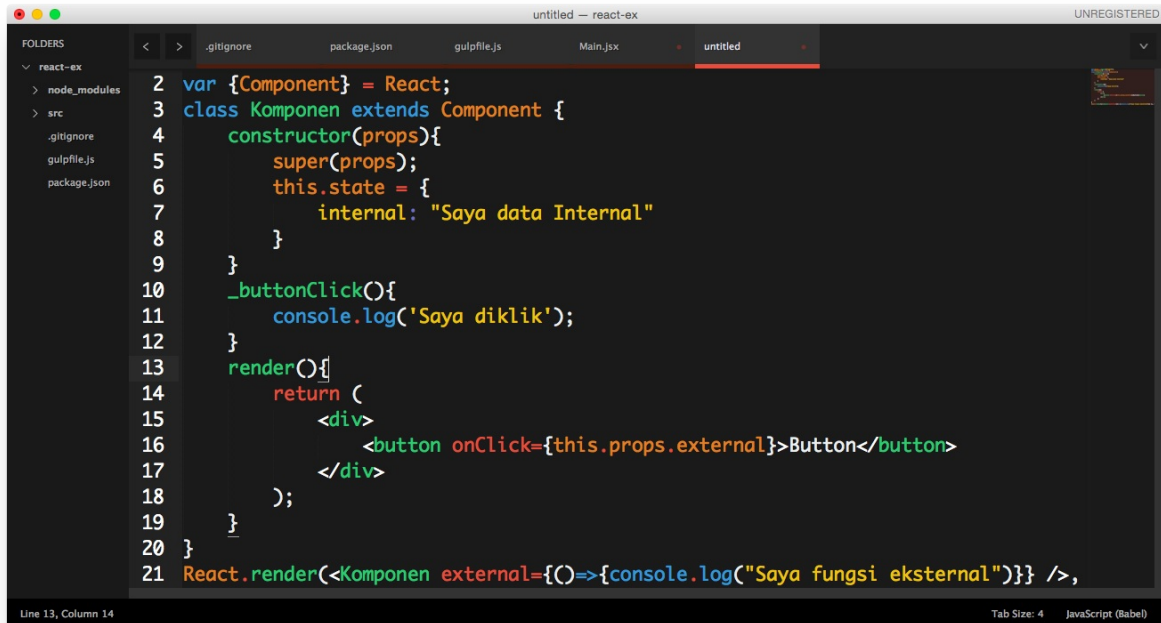
Pada baris 16, bisa dilihat bahwa button mempunyai listener onClick yang diisi dengan this.\_buttonClick yang mengarah ke method \_buttonClick() di baris 10. Itu saja. Ingat isi fungsi dengan menutupinya dengan `{}`.

Tapi bentar, jika props adalah data external, data apa saja yang bisa dimasukkan?

## Pengenalan ReactJS

Untuk props, terserah data apa saja, array bisa, string bisa, objek juga bisa, bahkan fungsi pun juga bisa.

Fungsi?



```
2 var {Component} = React;
3 class Komponen extends Component {
4   constructor(props){
5     super(props);
6     this.state = {
7       internal: "Saya data Internal"
8     }
9   }
10  _buttonClick(){
11    console.log('Saya diklik');
12  }
13  render(){
14    return (
15      <div>
16        <button onClick={this.props.external}>Button</button>
17      </div>
18    );
19  }
20 }
21 React.render(<Komponen external={()=>{console.log("Saya fungsi eksternal")}} />,
```

Pada baris ke 21, props external berisi fungsi yang kemudian oleh Komponen disematkan pada event onClick di button. Jadi jika button di klik maka fungsi yang dipanggil adalah `console.log("Saya fungsi eksternal")`.

Diatas adalah dasar dasar ReactJS, masih ada beberapa API yang nantinya sangat membantu dalam beberapa skenario tertentu.

# Spesifikasi Komponen

---

ReactJS memberikan beberapa kemudahan yang sangat dibutuhkan pada skenario tertentu. Baik dari segi performance maupun dari segi logic. Disamping itu, React juga memiliki spesifikasi tertentu dalam pembuatan komponennya.

## Spesifikasi

---

Dalam pembuatan komponen react, harus memenuhi kriteria tertentu, mulai dari buat komponen, lalu memberi implementasi `render()` dan sebagainya.

## Membuat Komponen

Dalam membuat komponen, ada 2 cara. Bisa dengan class di ES6, atau dengan `React.createClass()` di ES5. Dalam buku ini kita hanya fokus di ES6. Biar up-to-date gitu.

```
class Komponent extends React.Component {}
```

Yap itu saja, tapi akan terjadi error karena kita belum memberi satu method wajib, `render()`

## Method Render()

Method `render()` harus ada di setiap class komponen, dan wajib mengimplementasikan return JSX. JSX bisa berupa komponen lain maupun komponen native html seperti `div`, `p`, `b`, `button` dll. Lalu bagaimana kita memberi state awal?

## Method `getInitialState()`

Method `getInitialState()` dipanggil pertama kali komponen di `render()` dan harus return object yang nantinya dijadikan state awal. Mengingat kita menggunakan ES6, kita bisa melakukannya di constructor, tidak perlu method ini.

Masih banyak lagi, bisa dibaca disini:

<https://facebook.github.io/react/docs/component-specs.html>

Untuk mempercepat pemahaman lebih baik di skip dulu, next.

## Lifecycle Methods

---

Lifecycle methods digunakan untuk melakukan hal tertentu disela-sela waktu komponen di render. Mungkin terdengar asing, tapi jika dilanjutkan pasti tau maksudnya.

### Mounting: `componentDidMount()`

Dieksekusi sekali saja, yakni setelah pertama kali komponen di render. Dalam arsitektur flux, method ini berguna untuk me-listen store jika ada perubahan state.

### Mounting: `componentWillMount()`

Dieksekusi sekali saja yakni saat komponen akan dirender pertama kali.

### Updating: `componentWillUpdate(object nextProps, object nextState)`

Dieksekusi tiap komponen akan diupdate, sama seperti `render()` tapi terjadi sebelum `render()` dipanggil. Biasanya digunakan untuk persiapan render ulang atau filter apapun yang kita dapat dari `nextProps/nextState`.

### Updating: `componentDidUpdate(object prevProps, object prevState)`

Dieksekusi tiap komponen selesai di `render()` (tidak untuk render pertama kali, hanya saat update state). Disini kita bisa mendapatkan state dan props sebelum terjadi update, yakni di `prevProps` dan `prevState`.

### Updating: `componentWillRecieveProps(object nextProps)`

Dieksekusi tiap komponen mendapatkan props jika parent dari komponen ini dirender ulang. Kita bisa mengakses props baru di `nextProps`

### Updating: `shouldComponentUpdate(object nextProps,`

## **object nextState)**

Wajib return berupa boolean, jika return false maka komponen tidak akan di render(). Jika true maka komponen akan di render() dengan props dan state baru. Biasanya digunakan untuk tweak performance, yakni jika ada perubahan DATA di props atau state, komponen hanya akan diubah jika DATA antara state baru dan state lama ada perubahan. Jika tidak maka tidak di render(). Nanti akan sangat berguna saat kita membahas Immutable Data dengan ImmutableJS. Nanti.

## **Unmounting: componentWillUnmount()**

Terjadi sekali saat komponen akan dibuang dari DOM. Sering digunakan untuk meng-unlisten store dalam flux.

Selebihnya bisa dibaca disini: <https://facebook.github.io/react/docs/component-specs.html>

Oke kita sudah mengenal sifat-sifat ReactJS, waktunya kita bermain dengannya :).

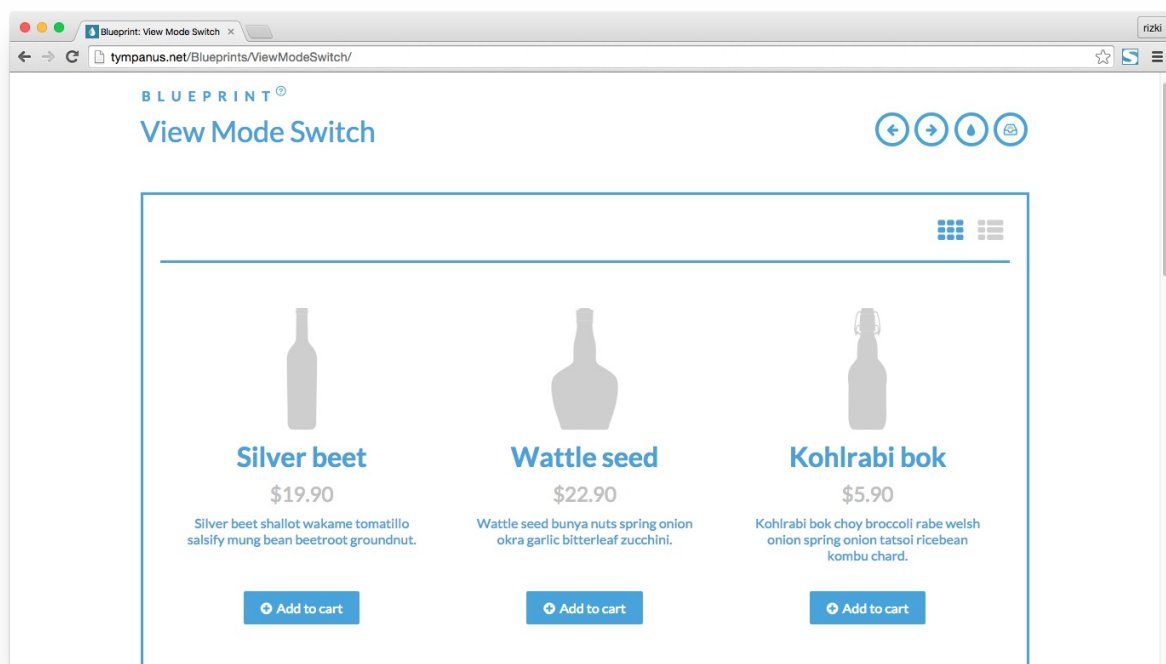
# Learning by Doing

Kita sudah tahu caranya membuat komponen, memberi data internal maupun external pada komponen, dan merender komponen. Tapi apa itu saja cukup? Jelas tidak.. Sangat sulit menjelaskan masalah kepada seseorang yang dia sendiri tidak tahu kalau dia punya masalah. Solusinya? Buat masalah, pecahkan.

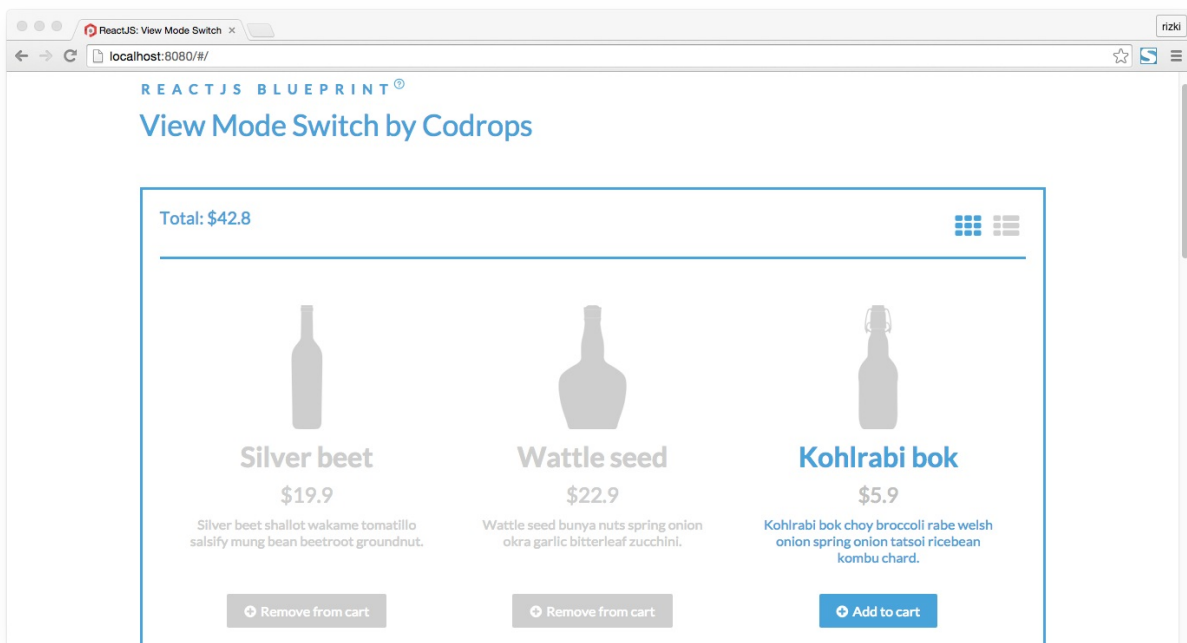
Di chapter ini kita akan belajar bagaimana menyusun sistem aplikasi dengan ReactJS. Untuk mempercepat pemahaman, cara belajar kita adalah dengan mencontoh project yang sudah ada, mencontoh artinya melihat dan melakukannya kembali. Jika hanya copy-paste namanya menyalin. Pengalaman pribadi saya, saya sangat cepat belajar ketika saya menulis ulang kode contoh. Tapi jika tidak suka ya terserah anda, bisa langsung copy.

Seperti biasa kode kodenya ada di github, tepatnya di commit <https://github.com/rromadhoni/contoh-toko-online-reactjs/commit/5f954f9a65442315c0b936e3011736a43444a742>

Skenarionya seperti ini, saya ingin ada interaksi add to cart dan remove from cart dari deretan produk saya di aplikasi ini. Tampilan ini saya dapat dari <http://tympanus.net/codrops/2013/07/01/view-mode-switch/>



Dan akan kita buat seperti ini



## Struktur direktori

Struktur direktorinya seperti ini

```
├─ .gitignore
├─ gulpfile.js
├─ package.json
├─ src
│   ├── Main.jsx
│   ├── build.js
│   ├── build.min.js
│   ├── contoh.jsx
│   ├── index.html
│   ├── js
│   │   ├── App.react.jsx
│   │   ├── components
│   │   │   ├── Header.react.jsx
│   │   │   ├── OptionSwitch.react.jsx
│   │   │   ├── ProductItem.react.jsx
│   │   │   └── ProductList.react.jsx
│   │   ├── pages
│   │   │   ├── Index.react.jsx
│   │   │   └── Product.react.jsx
│   │   ├── routes.jsx
│   │   └── utils
│   │       └── RouterContainer.jsx
│   └── stylesheets
│       └─ .DS_Store
```

```
├─ css
|   ├─ .DS_Store
|   └─ main.css
├─ fonts
|   ├─ .DS_Store
|   ├─ bpicons
|   |   ├─ bpicons.eot
|   |   ├─ bpicons.svg
|   |   ├─ bpicons.ttf
|   |   └─ bpicons.woff
|   └─ license.txt
|       └─ fontawesome
|           ├─ Read Me.txt
|           ├─ fontawesome.dev.svg
|           ├─ fontawesome.eot
|           ├─ fontawesome.svg
|           ├─ fontawesome.ttf
|           ├─ fontawesome.woff
|           └─ license.txt
├─ images
|   ├─ 1.png
|   ├─ 10.png
|   ├─ 2.png
|   ├─ 3.png
|   ├─ 4.png
|   ├─ 5.png
|   ├─ 6.png
|   ├─ 7.png
|   ├─ 8.png
|   └─ 9.png
└─ less
    ├─ component.less
    ├─ default.less
    └─ main.less
```



# Cara Kerja

---

Saya akan memberikan beberapa clue dalam project ini. Ada 2 module npm yang saya gunakan. Yakni `react` dan `react-router`.

## Module React

`react` adalah module utama dalam membangun aplikasi ReactJS, tidak perlu perlakuan khusus mengingat kita sudah membahasnya di atas.

## Module React Router

`react-router` adalah modul yang digunakan untuk menata routing aplikasi kita. Penggunaannya tetap menggunakan JSX.

```
1  "use strict";
2  var React = require('react');
3  var Router = require('react-router');
4  var {Link, Route, RouteHandler, DefaultRoute} = Router;
5
6  // pages
7  var App = require('./App.react');
8  var Index = require('./pages/Index.react');
9  var Product = require('./pages/Product.react');
10
11 // routes
12
13 var routes = (
14   <Route handler={App}>
15     <Route name="index" path="/" handler={Index} />
16     <Route name="product" path="/product/:product_id" handler={Product} />
17   </Route>
18 );
19 module.exports = routes;
```

Mulai dari baris ke 7, kita memuat semua pages. Dandi baris ke 13 kita membuat `routes` yang kemudian akan dipakai sebagai router kita. `Handler={}` artinya, apa yang terjadi jika route aplikasi berada di `#/nama_path`.

Untuk dokumentasinya bisa dibaca disini <https://github.com/rackt/react-router>

Sekarang tugas anda adalah, menyontoh project di atas. Untuk aset silakan disalin,

untuk file javascript ditekankan untuk ditulis ulang.

WTFFFFF... tulis ulang sebanyak itu???

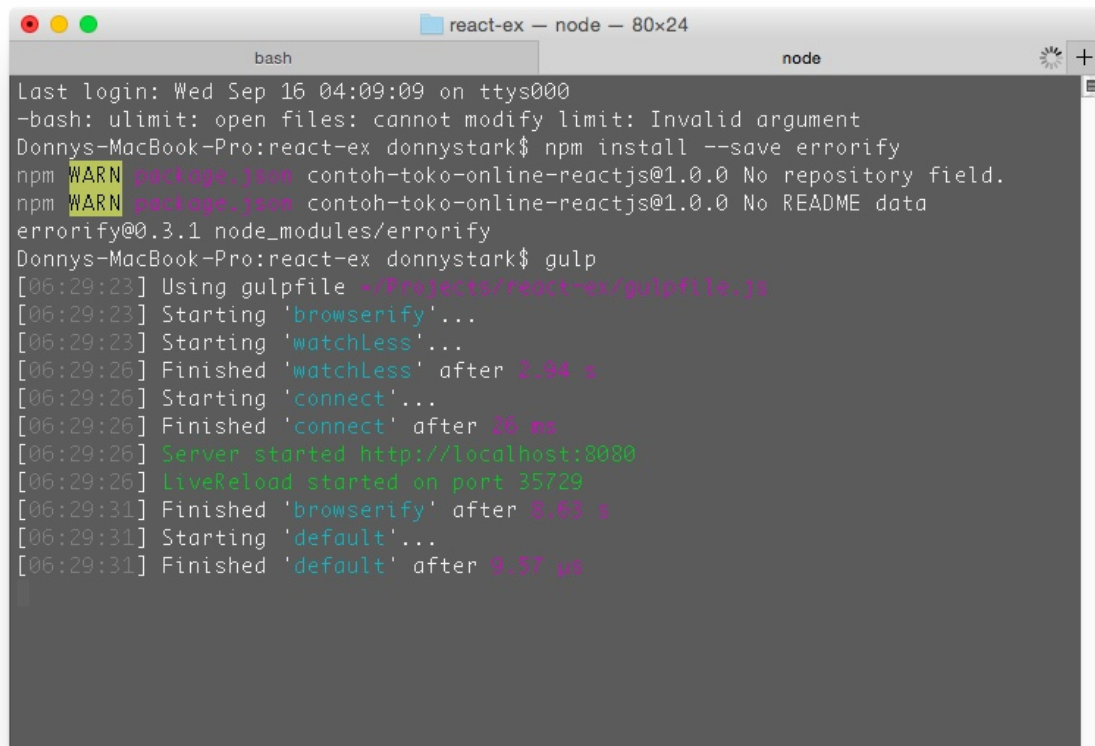
Heheheh.. Jadi gini ceritanya, anda ingin jadi ReactJS developer.. Jadi harus mengikuti jejaknya. Ya inilah jejaknya. Jejak kode. Paling tidak dengan metode ini, anda akan mengerti bagaimana pola pikir ReactJS Developer.

Saya menerapkan metode ini karena saya juga ingin memanfaatkan imajinasi anda dalam belajar. Saat anda menulis ulang, meski hanya menulis ulang, nantinya imajinasi anda yang akan berkeliaran, "ini apa sih? oh mungkin untuk yang ini, eh bukan, oohhh ternyata untuk ini ya". Saya rasa metode ini lebih menantang daripada hanya membaca buku buku macam buku dokumentasi. Lebih asyik untuk dipelajari dibandingkan yang bergaya dokumentasi. Kuncinya satu, believe your imagination.

## Menjalankan Contoh

Untuk menjalankan contoh, clone repositori diatas, lalu install dependency dengan menjalankan: `$ npm install` Setelah semua terinstall, jalankan gulp dengan: `$ gulp`

Lalu akan muncul seperti ini:



```
react-ex — node — 80x24
bash
Last login: Wed Sep 16 04:09:09 on ttys000
-bash: ulimit: open files: cannot modify limit: Invalid argument
Donnys-MacBook-Pro:react-ex donnystark$ npm install --save errorify
npm WARN package.json contoh-toko-online-reactjs@1.0.0 No repository field.
npm WARN package.json contoh-toko-online-reactjs@1.0.0 No README data
errorify@0.3.1 node_modules/errorify
Donnys-MacBook-Pro:react-ex donnystark$ gulp
[06:29:23] Using gulpfile ~/Projects/react-ex/gulpfile.js
[06:29:23] Starting 'browserify'...
[06:29:23] Starting 'watchLess'...
[06:29:26] Finished 'watchLess' after 2.94 s
[06:29:26] Starting 'connect'...
[06:29:26] Finished 'connect' after 28 ms
[06:29:26] Server started http://localhost:8080
[06:29:26] LiveReload started on port 35729
[06:29:31] Finished 'browserify' after 8.63 s
[06:29:31] Starting 'default'...
[06:29:31] Finished 'default' after 9.57 ms
```

Selanjutnya silakan buka <http://localhost:8080> untuk melihat hasilnya

Repo diatas juga berjalan di <http://contoh-reactjs.herokuapp.com/#/>