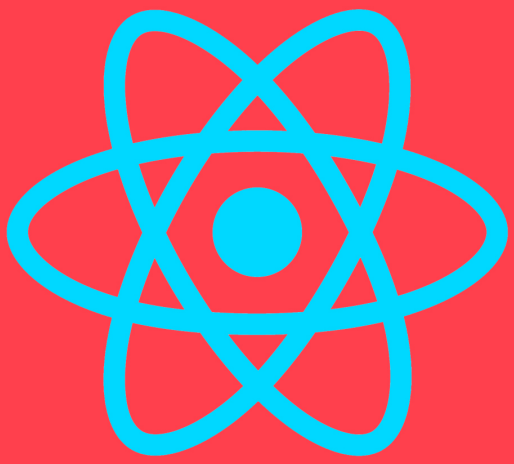
MUHAMMAD AZAMUDDIN



REACTJS

UNTUK JQUERY DEVELOPER

Table of Contents

Welcome

Halo, salam kenal, nama saya Muhammad Azamuddin, orang-orang biasa memanggil saya mas azam, kamu juga boleh menggunakan panggilan yang sama. Saya yang mengelola facebook page @arrowfunxtion, bukan karena saya master atau pakar javascript modern tapi karena saya memang sangat antusias sama si javascript dan segala perkembangannya. Tapi kalo kamu mau panggil saya Developer Kreatif juga boleh, namanya Developer Kreatif jadi harus banyak belajar dan banyak share untuk meningkatkan skill dan kebermanfaatan ^_^

Okay, lanjut lagi, dulu kita mengandalkan jQuery untuk membantu kita membuat tampilan yang ciamik, kini ada banyak library lain yang siap dipelajari, misal Angular, React, Backbone, Ember, Vue,dll. Kenapa? karena banyak perusahaan besar maupun startup sekarang mencari developer-developer yang menguasai framework dan library tersebut, bahkan tidak mensyaratkan pendidikan. Salah seorang kawan saya, kebetulan dia lebih muda dari saya, belum lulus kuliah, dia pernah bercerita bahwa dia mendapatkan tawaran dari salah satu startup e-commerce besar di Indonesia dan juga mendapatkan tawaran menjadi pengajar javascript modern dengan kisaran gaji 16-20jt, semua itu karena dia menguasai pengembangan javascript modern. Selain itu, menguasai framework-framework tersebut in sha Allah akan membuat hidup kita sebagai developer lebih mudah dan jalan bagi kita menjadi seorang developer profesional.

Nah kali ini saya ingin berbagi tentang salah satu library yang sering dipakai dalam pengembangan web app modern, yakni ReactJS. Saya berharap buku ini bisa menjadi pembuka bagi yang baru mulai belajar, terutama yang saat ini sudah bisa jQuery namun masih mengalami kesulitan untuk mempelajari dunia javascript modern. **Apakah kamu pernah berpikir seperti ini:**

- **Kenapa sih harus belajar library-library javascript modern, apakah jQuery saja tidak cukup?**
- **Kamu tidak mau belajar banyak konsep-konsep baru?**
- **Ingin belajar framework/library javascript modern tapi khawatir butuh waktu lama?**
- **Ingin bisa membuat web app, mobile app, desktop app tapi kok sepertinya terlalu banyak yang harus dipelajari ya.**

Meskipun hal-hal di atas berseliweran di pikiran kamu, **tapi kamu juga memikirkan hal-hal ini:**

- **Saya ingin membuat web app modern tapi tidak tahu harus mulai dari mana;**
- **Saya ingin memiliki skill seperti developer-developer profesional di luar sana;**
- **Saya ingin bisa membuat user interface yang menarik dan kekinian;**
- **Saya ingin mengembangkan web app secara cepat dan berkualitas;**
- **Saya ingin basis kode saya mudah dibaca dan mudah dipelihara;**
- **Saya ingin membuat aplikasi yang mudah diintegrasikan dengan layanan ataupun aplikasi lain;**

Maka dengan belajar React, kamu **sudah ke jalan yang benar** menuju keinginan-keinginan di atas. Kabar baiknya di ebook ini saya akan mengupas dasar-dasar dan fondasi yang harus kamu pahami dan penjelasan yang sejelas mungkin. Dan kita akan bandingkan implementasi dengan jQuery vs React. Bagaimana, asyik bukan?

INGAT! Lebih baik belajar mati-matian di awal daripada nanti mati beneran pas bikin web app. Bikin web app modern itu ada caranya, butuh persiapan yang matang, butuh tool dan library yang tepat, sudah siap belajar library React ini? Silahkan lanjutkan membaca...

Untuk memulai buku ini saya ingin sedikit bercerita tentang ide awal saya menulis buku ini. Ceritanya begini, saya melihat alur pembelajaran pembuatan website selalu dimulai dengan HTML, CSS & Javascript. Ya, tidak ada yang salah, karena sebelum belajar *server side* memang seorang web developer harus sudah nyaman dengan ketiga hal yang saya sebut itu. Tapi, biasanya yang dipelajari sebenarnya bukan Javascript, melainkan jQuery, sebuah *library* membantu memudahkan manipulasi DOM. Apalagi dengan tersedianya plugin-plugin yang sesuai kebutuhan. Betul?

Apakah jQuery jelek? Tidak juga, saya tidak berkata demikian. Dulu saya juga pengguna jQuery, kehadiran jQuery sangat membantu saya dalam mengembangkan tampilan website yang menarik. Untuk beberapa kasus, menggunakan jQuery sangatlah OK. Dan saya tahu kamu menginginkan buku ini artinya sudah belajar dan paham bahwa React penting untuk dipelajari. Dengan buku ini saya ingin membantu teman-teman yang sebelumnya nyaman dengan jQuery untuk memulai belajar ReactJS.

Seiring dengan perkembangan dalam dunia web app development, banyak bermunculan *framework-framework* javascript, misalnya Backbone, Ember, Angular, Vue, ReactJS. Umumnya kehadiran *framework-framework* tersebut untuk mengatasi permasalahan yang muncul ketika menggunakan jQuery. Apa permasalahan utama jika menggunakan jQuery? **Maintenance**. Pemeliharaan aplikasi kita menjadi sulit, apalagi jika basis kode kita cukup besar, akan timbul masalah-masalah seperti *code spaghetti, callback hell, software entropy*, dll yang semuanya itu mengakar ke sulitnya memelihara basis kode kita bila ingin melakukan perubahan.

Buku Ini Cocok Untuk Siapa?

- Buku ini ditujukan **untuk** kamu yang sudah terbiasa dengan HTML, CSS dan jQuery. Karena saya tidak akan menjelaskan dasar-dasar HTML, CSS ataupun Javascript. Saya anggap kamu sudah cukup familiar dengan ketiga hal itu.
- Buku ini **bukan untuk** yang sudah bisa ReactJS atau yang sudah pernah belajar Angular, Backbone, Vue dan f*ramework javascript* lainya. Kamu bakal "cape deh", karena memang buku yang ini bukan ditujukan untuk kamu! ^_^

Dengan buku ini diharapkan pembaca dapat

- Membuat React Komponen Pertama dengan ES5
- Memahami cara kerja State dan Props
- Anatomi Komponen React
- Dasar Composite Komponen
- Sedikit perbedaan syntax di React dengan HTML biasa

Hal yang perlu familiar

- JSON / OBJECT
- Method / Fungsi dalam Class

Mengapa Mempelajari React?

Ada orang yang memerlukan jawaban pasti, mengapa harus melakukan sesuatu termasuk mengapa harus belajar React? Dan inilah beberapa alasannya..

Apa itu ReactJS? React merupakan *library* javascript yang dibesut oleh Tim Facebook. Ketika mempelajari React, kita harus tahu ada ReactJS dan ReactNative. Apa bedanya? ReactJS digunakan untuk membuat UI web, sementara ReactNative digunakan untuk mengembangkan native mobile apps, NATIVE, bukan hybrid. Jadi dengan belajar React, kita menyelam sambil minum air (tenggelem donk), enggak. Maksudnya sekali belajar bisa dipakai untuk membuat Web dan Mobile Native. Perlu catatan, apa yang dibahas di buku ini adalah khusus ReactJS untuk Web.

React dipakai oleh internal Facebook baik untuk mengembangkan Facebook Timeline maupun untuk aplikasi lainnya seperti Facebook Business Manager, Facebook Page Mobile, Instagram, dll. Selain digunakan oleh Facebook, Whatsapp, React juga telah diadopsi oleh startup besar semisal Asana, AirBnB, Docker Site, Syncano, UI Wordpress.com terbaru, dan lain-lain. Mengapa begitu banyak yang tertarik menggunakan React untuk mengembangkan UI mereka? Ada beberapa alasan yang mendasarinya. Apa saja?

React hanyalah "V" dalam MVC

React tidak seperti Angular, Ember, Knockout. React bukanlah framework, dalam arti React tidak menyediakan fitur-fitur terbundle seperti routing, model, alur data, dll karena React memang hanyalah V dalam MVC. Namun bukan berarti react tidak powerful, justru React sangat powerful dengan kesederhaannya. Kurva belajar react jauh lebih mudah dibandingkan dengan kurva belajar *library atau framework js lainnya. Meskipun mudah dipelajari dan hanya "V" dalam MVC, namun React menjelma menjadi UI library* yang paling diminati.

Virtual DOM

React memanfaatkan *virtual DOM,* apa itu? virtual, yah artinya tidak nyata, atau semu. Maksudnya React tidaklah seperti jQuery yang memanipulasi DOM secara langsung. React bermain atau menggunakan *diffing algorithm* untuk mendeteksi perubahan

kemudian mencari cara terefisien dalam mengupdate UI pada virtual DOM. Sehingga performanya jauh lebih baik dibandingkan memanipulasi DOM secara langsung.

Berbasis Komponen

Dalam react kita berpikir komponen, kita akan membuat komponen. Komponen akan memberikan kemudaan bagi kita dalam megembangkan aplikasi web modern. Dengan komponen kita bisa bongkar pasang seperti bermain lego. Misal kita memiliki komponen yang sudah kita buat yang akan menampilkan 5 article terbaru, komponen tersebut kita beri nama `<LatestArticles/>` maka, selanjutnya kita bisa menggunakan komponen tersebut di manapun di project kita yang lain atau tim kita, cukup dengan memanggil `<LatestArticles/>` seperti menulis kode HTML biasa. Komponen kita menjadi reusable dan bisa kita komposisikan sesuai kebutuhan.

Selain itu, kita juga bisa menggunakan komponen-komponen yang sudah dibuat oleh komunitas react. Kita bisa cari komponen-komponen tersebut di https://react.rocks, http://npmjs.com, https://khan.github.io/react-components/, https://react.parts/web, dll. Banyak, tinggal pakai. *No need to reinvent the wheel.* Salah satu komponen bundle yang saya sukai adalah material design for React http://material-ui.com.

React Native

Seperti yang saya jelaskan di awal bab ini, keuntungan kita belajar React adalah kita bisa menerapkan pengetahuan kita untuk membuat Web dan Native Mobile Apps. Nah untuk mengembangkan mobile native apps ini kita pakai React Native.

Selain alasan-alasan di atas, saya juga teringat apa yang saya share di Microsoft Indonesia tahun 2016 yang lalu terkait React, antara lain:

- **Boost Productivity**
- **Easy Collaboration (Developer & Designer)**
- **Maintanable**
- **Selengkapnya silahkan lihat di https://slides.com/muhammadazamuddin/introduction-to-reactjs/live**

Membuat Komponen Pertama

Persiapan

Untuk memudahkan proses pembelajaran ini, kita akan bermain dengan JSBIN. Online editor yang bisa menjalankan kode HTML, CSS, Javascript termasuk React. Namun kita harus *setup* terlebih dahulu.

Langkah-langkah:

1. buka http://jsbin.com/
2. buat akun kalo belum punya (gratis).
3. buat BIN baru (file > new)
4. karena kita akan menggunakan React, maka kita tambahkan dulu *library* React dan ReactDOM.
    i. taruh kursor pada sebelum </body>
    ii. klik *Add Library* > pilih React + React DOM 15.1.0
5. kita juga memerlukan Bootstrap
    i. taruh kursor di antara tag *head*
    ii. klik *Add Library* > pilih Bootstrap Latest
6. persiapan selesai

Persiapan JSBIN telah selesai, kamu coba perhatikan ada beberapa Tab jendela antara lain HTML, CSS, Javascript, Console dan Output, fungsinya adalah untuk mengaktifkan dan menonaktifkan jendela.

Membuat Komponen

Buat render point

Elemen utama dimana kita akan *render* Aplikasi React kita, kita hanya melakukan ini sekali saja untuk Komponen paling atas. Pada jendela JSBIN yang baru saja kamu buat, deklarasikan div dengan id "app".

Listing 1: Buat elemen sebagai render point pada **Tab HTML**

```
<div id="app"></div>
```

JSX

Klik tab javascript pada jendela JSBIN. Lalu klik dropdown javascript, ganti ke JSX. Untuk saat ini ikuti saja step-stepnya, di bab berikutnya akan dijelaskan lebih detail.

Listing 2: Deklarasi Komponen React

```
var Komponen = React.createClass({
    render: function(){
      return (<div>
        Hello World
      </div>)
   }
})
```

Render komponen yang baru kita buat.

Listing 3: Render komponen tersebut ke element dengan ID "app"

```
ReactDOM.render(<Komponen/>, document.getElementById("app"))
```

Secara sederhana begitulah membuat komponen di React, mudah bukan. Coba lihat tab Output, seharusnya kamu melihat tulisan Hello World.

Penjelasan

Render Point

Membuat React app kita membutuhkan setidaknya satu elemen yang akan kita gunakan sebagai render point. Dalam hal ini kita gunakan `<div id="app"></div>` Gunanya adalah untuk menempelkan aplikasi React kita. Dan hanya perlu satu render point.

JSX

JSX atau Javascript Syntax Extension merupakan ekstensi yang memudahkan kita menulis komponen dalam bentuk kode yang mirip dengan HTML. Perhatikan kode di dalam render, itu contoh kode JSX. Agar kita bisa menggunakan JSX pada praktiknya

kita membutuhkan transformer.js atau jika di ES6 kita bisa menggunakan pre-compiler. Tapi itu diluar scope dari buku ini. Untuk sekarang kita menggunakan JSBIN yang sudah kita *setup.*

ReactDOM

ReactDOM merupakan module sendiri, kita memerlukannya agar kita bisa menempelkan aplikasi kita pada render point. Kita bisa menggunakannya karena kita sudah menambahkan library ReactDOM pada langkah setup. Untuk merender aplikasi kita.

```
var renderpoint = document.getElementById('app') // render point
ReactDOM.render(<KomponenAplikasiKita/>, renderpoint) // render aplikasi kita di render po
int.
```

Anatomi Dasar Komponen React

Dalam React, semuanya adalah komponen. Dan sebuah komponen memiliki anatomi atau susunan bagian. Ibarat tubuh kita memiliki kaki, kepala, tangan, dsb, Komponen React juga memiliki struktur serupa. Hal ini berbeda dengan jQuery yang tidak memiliki struktur baku. Dengan adanya struktur pada React kita bisa lebih mudah dalam mengelola kode kita. Mari kita lihat contoh pertama.

Anatomi dasar komponen react adalah sebagai berikut:

- render method
- JSX
- props
- state
- event handler

Semuanya hanyalah javascript, jadi kalo kamu familiar dengan jQuery, kamu kemungkinan akan familiar dengan Anatomi Komponen React. Baik, mari kita bahas satu per satu.

1. Render Method

```
var KomponenPertama = React.createClass({
    render: function(){
        return (<div>
            <b>Hello Komponen React<b/>
        </div>)
    }
})
```

Contoh di atas merupakan sebuah *Class* untuk membuat komponen yang kita beri nama KomponentPertama. *Class* tersebut hanya memiliki satu *method* yaitu *render*. *Render* merupakan method di mana kita mendeklarasikan apa yang akan muncul ketika kita menggunakan KomponenPertama tersebut dalam aplikasi kita. Pada contoh di atas, KomponenPertama jika direnderakan menghasilkan tulisan pada layar "Hello Komponen React" dengan cetak tebal. Sesederhana itu.

Perhatian: *Dalam membuat nama komponen kita harus membuatnya CamelCase, misal Komponen, atau KomponenPertama, bukan komponen, komponen-pertama, komponenPertama. Huruf pertama harus kapital. Hal ini merupakan konvensi dan untuk membedakan antara tag HTML dengan Komponen React.*

Untuk mencoba contoh di atas, buka JSBIN yang kita siapkan sebelumnya. Buka tab "JSX", lalu tambahkan kode di atas, pada bagian paling atas. Setelah itu pada render method "Komponen" component, tambahkan `<KomponenPertama/>`. Sehingga kode render "Komponen" Component pada JSBIN menjadi seperti ini:

```
var Komponen = React.createClass({
    render: function(){
      return (<div>
        <KomponenPertama/>
      </div>)
   }
})
```

Selamat! Kamu baru saja belajar dasar *composite* component. Mudah bukan? *composite* di sini maksudnya Kamu membuat komponen yang digunakan oleh komponen lainnya. Inilah salah satu cara berpikir react, dan *composite* ini merupakan pondasi dari *reusable component*.

2. JSX

Apakah kamu memerhatikan kode yang dikembalikan oleh method _render yang diapit oleh "( )" ? apakah itu HTML? Bukan, kode yang **MIRIP HTML** itu disebut dengan JSX (Javascript Syntax Extension). Merupakan Extensi untuk menulis javascript dengan syntax yang menyerupai HTML. Dalam menulis JSX pada render method. Kita harus memastikan beberapa hal, antara lain:

Nilai kembalian JSX pada method *render* harus terdiri dari satu tag paling luar (root tag), tidak boleh lebih.

Contoh Benar:

Lihat di contoh kode method render sebelumnya, JSX paling atas dari method render tersebut terdiri dari satu tag yaitu `<div>` .

```
/**
 * perhatikan bagian ini, root tag JSX adalah satu div.
 * ini BENAR
 */
return (<div>
    <b>Hello Komponen React</b>
</div)
```

Contoh Salah (Error)

```
/**
 * kode JSX ini kan menyebabkan error, karena tag paling atas ada lebih dari satu.
 * perhatikan baik-baik, contoh di bawah terdapat dua <div> yang menjadi tag paling luar.
 */
 return (<div>
  <b> Hello Komponen React</b>
 </div>

 <div>
   <b> Kamu Siap Belajar React?</b>
 </div>
 )
```

Contoh benar apabila ingin menampilkan nested tag

Mari kita ubah kode JSX pada contoh salah agar tidak error. Kita tahu tujuan dari kode JSX tersebut adalah agar Komponen menampilkan tulisan "Hello Komponen React" diikuti dengan "Kamu Siap Belajar React?". Tidak ada masalah sama sekali dengan tujuannya, hal itu sangat mungkin. Tinggal kita perbaiki agar hanya terdapat satu tag paling luar.

```
/**
 * kode JSX dibawah ini akan berjalan sebagaimana yang kita harapkan
 * satu tag <div> sebagai komponen paling luar.
 * diikuti dengan berapapun jumlah tag yang kita inginkan.
 */
return (<div>
    <div>
        <b>Hello Komponen React</b>
    </div>
    <div>
        <b> Kamu Siap Belajar React?</b>
    </div>
</div>)
```

Semua tag harus benar-benar ditutup

Semua tag harus benar-benar ditutup. Apabila tidak, maka akan muncul error yang menyebabkan Komponen tidak dapat direnderer. Berbeda dengan HTML, ketika ada tag yang tidak ditutup tetap akan jalan meskipun tampilannya rusak. React tidak akan merender apabila ada tag JSX yang tidak ditutup.

```
/**
 * kode JSX dibawah ini akan menyebabkan error, karena tag <b> tidak ditutup!
 * apabila kita perbaiki dengan menutup tag <b>, error masih muncul, kenapa?
 * karena tag <img> juga belum ditutup
 */
return (<div>
    <b> Hello Dunia!
    <img src="http://image.com/image.png">
</div>)

// tag <b> harus ditutup dengan </b> dan tag <img src="..."> seharusnya ditutup seperti in
i <img src="..." />
```

Gunakan className apabila ingin memberikan class ke sebuah tag.

Dalam tag HTML kita terbiasa menggunakan class, hal ini tidak berlaku untuk JSX. Untuk menambahkan class yang akan dibaca oleh kode CSS, kita gunakan className. Apabila menggunakan class, tidak akan terjadi kegagalan render, namun class tidak

dibaca sehingga tampilannya tidak sesuai dengan class pada kode CSS kita.

```
/**
 * Kita berikan class "alert aler-warning" dengan cara className="alert alert-warning"
 */
return (<div>
    <div className="alert alert-warning">
        Selamat malam, saatnya kita koding!
    </div>
</div>)
```

Inline-style pada JSX menggunakan object

Selain perbedaan sintak saat pemberian class pada sebuah tag, sintak pemberian inline-style juga sedikit berbeda dengan sintak pada HTML.

```
return (<div>
    <div style={{backgroundColor: 'yellow'}}> Background saya berwarna kuning </div>
</div>
```

Sintak untuk style object adalah:

```
{ propertyName: 'value', propertyName2: 'value', propertyNameDst: 'value'}
```

- Nama properti ditulis seperti pada kode CSS biasa;
- Nilai dari properti harus diapit tanda petik (" "), "red", "yellow", "left", "10px". Sementara pada CSS biasa, kita gunakan red, yellow, left, 10px, dst tanpa tanda petik;
- Apabila nama properti lebih dari satu kata, contoh background-color, padding-left, text-align, maka kita ubah menjadi snakeCase menjadi backgroundColor, paddingLeft, textAlign, dst.

Contoh

```
// Kode CSS
{
    background-color: red;
    padding-left: 10px;
    border-radius: 20px;
}

// Ubah ke inline JSX Css in JS
{
```

```
    backgroundColor: 'red',
    paddingLeft: '10px',
    borderRadius: '20px'
}
```

Kode *comment*

Kode komentar sedikit berbeda dengan komen HTML.

```
return (<div>
    <b> Ini akan direnderer oleh browser </b>
    {/* Ini tidak akan direnderer oleh browser karena hanya sebuah comment*/}
</div>)
```

3. Props

Mari kita membuat komponen baru yang dapat kita setting propertinya. Seperti pada HTML, misalkan tag `<form>` memiliki beberapa properti, seperti action, sehingga kita menulisnya `<form action="action/url"></form>`. Nah komponen react juga bisa kita setting propertinya.

Kita akan membuat komponen `<Sapa/>`, komponen ini akan menyapa nama orang tertentu sesuai dengan prop.

Kita buat komponen Sapa terlebih dahulu.

```
var Sapa = React.createClass({
    render: function(){
       return (<div>
           Halo, selamat belajar, {this.props.kepada}
        </div>)
    }
})
```

Pada deklarasi Class komponen di atas, perhatikan render methodnya. Ada kode `{this.props.kepada}`, apa maksudnya? maksudnya adalah, ganti kode tersebut dengan properti `kepada` yang dipassing ke Komponen `<Sapa/>`. Lihat kode di bawah ini:

```
<Sapa kepada="Azam"/>
// Komponen ini akan menampilkan
// "Halo, selamat belajar, Azam"

<Sapa kepada="Budi"/>
// "Halo, selamat belajar, Budi"
```

4. Event Handler

Event handler, merupakan method yang akan dipanggil ketika terjadi event yang disebabkan misalkan oleh interaksi user. Ketika menggunakan jQuery, sering kita memasang listener pada elemen di DOM. Event handler merupakan satu dari sekian bagian dari komponen. Mari kita lihat contohnya, buat komponen berikut pada JSBIN, tab "JSX", bagian paling atas.

```
var Sapa = React.createClass({
    render: function(){
        return (<div>
            Halo, selamat belajar, {this.props.kepada}
            <br/>
            <button
                onClick={this.katakanHalo}/>
                Katakan Halo
            </button>
         </div>)
    },
    onClick: function(e){
        alert("Haloooo, " + this.props.kepada)
    }
})
```

Coba kamu gunakan komponen <Sapa/> tersebut pada komponen <Komponen/>, jangan lupakan properti `kepada` ya, seperti berikut:

```
var Komponen = React.createClass({
    render: function(){
      return (<div>
        <Sapa kepada="Nama kamu"/>
      </div>)
   }
})
```

Seharusnya kamu bisa melihat hasilnya pada tab "Output". Apa hasilnya? hasilnya adalah sebuah tulisan "Halo, selamat belajar, Nama kamu" dan sebuah tombol dengan label "Katakan Halo". Apabila tombol tersebut diklik, maka akan muncul alert "Haloooo, Nama Kamu".

Selamat! kamu sudah belajar event handler pada react.

5. State

State adalah kondisi komponen. Segala sesuatu pasti memiliki kondisi, misal Buah, ada buah yang segar, busuk, agak busuk, nah sebenarnya itu adalah state dari buah itu, statenya masih baik, statenya udah busuk, dst. Contoh lain adalah mobil kita (yang belum punya, semoga disegerakan), mobil kita saat ini katakan memiliki lebih dari satu state, misal, state warnanya saat ini hitam, state pajaknya lunas, state ban masih baru, dst. Nah seiring dengan berjalannya waktu, pasti state itu bisa berubah, misal buah yang tadinya bagus menjadi busuk, mobil kita warna hitam, lalu kita cat menjadi putih, dst. Nah, state ini menentukan tampilan luar bukan? buah segar dan buah busuk tampilan luarnya beda bukan? Sama halnya dengan analogi-analogi tersebut, state pada komponen react juga bisa kita gunakan untuk menentukan tampilan dari komponen kita. Misal state warna merah, komponen kita berbackground merah, lalu kita ganti state menjadi kuning, background komponen kita berubah menjadi kuning. Masuk akal bukan?

State mirip dengan props, bedanya adalah state dibuat dan disimpan secara internal. Sementara props bisa diset dari luar Komponen. Selain itu, props bersifat static, artinya sekali dibuat maka props tidak dapat diubah, sementara state bersifat dapat diubah. Dan state ini sangat penting, karena bisa kita gunakan untuk beberapa fungsi, misal untuk menyimpan data, menentukan UI, dll.

Kita akan mengubah sedikit perilaku dari komponen `<Sapa/>` yang sudah kita buat. Kita ingin komponen Sapa tersebut akan menampilkan tiga buah tombol, masing-masing tombol tersebut dengan label nama orang yang berbeda. Bila tombol tersebut diklik, maka tulisan di atasnya, akan menyapa nama orang yang ada di label tombol.

```
var Sapa = React.createClass({
    getInitialState: function(){
        return {
            kepada: 'Andi'
        }
    },

    render: function(){
        return (<div>
            {/* perhatikan kita tidak lagi menggunakan this.props.kepada, namun this.state
.kepada */}
            {/* this.state.kepada merujuk pada getInitialState, kepada, maka defaultnya ad
alah "Andi" */}
            Halo, selamat belajar, {this.state.kepada}
            <br/>
```

```
            <button
                onClick={this.katakanHalo.bind(this, "Budi")}/>
                Budi
            </button>
            <button
                onClick={this.katakanHalo.bind(this, "Joni")}/>
                Joni
            </button>
            <button
                onClick={this.katakanHalo.bind(this, "Fadil")}/>
                Fadil
            </button>
         </div>)
    },
    onClick: function(namaOrang, e){
        this.setState({kepada: namaOrang})
    }
})
```

Apa hal baru yang kita temukan dari contoh kode di atas? Ada yang bisa menjawab?

- method getInitialState
- kode `{this.state.kepada}`
- kode `this.setState`

getInitialState

method ini digunakan untuk mendefinisikan state default atau kondisi awal komponen `<Sapa/>`. Method getInitialState menghasilkan sebuah state object. Pada contoh di atas, state object tersebut memiliki satu properti yaitu `kepada`. Inilah state yang dihasilkan pada contoh di atas:

```
// State Object
{
    kepada: "Andi"
}
```

kode `{this.state.kepada}`

Kode ini artinya, tampilkan state kepada dari komponen `<Sapa/>`. Kondisi awal state kepada adalah? Yup, betul "Andi". Artinya kode `{this.state.kepada}` akan direnderer menjadi "Andi". Sehingga komponen `<Sapa/>` pada saat direnderer pertama kali akan memunculkan tulisan "Halo, selamat belajar, Andi" diikuti oleh 3 tombol di bawahnya.

kode `this.setState`

Kode ini berfungsi untuk mengubah state komponen kita. Dalam contoh di atas, kita menggunakan `this.setState` untuk mengubah state kepada dari komponen `<Sapa/>` dengan nilai namaOrang sesuai dengan tombol yang diklik.

```
this.setState({kepada: namaOrang})
```

Apa yang terjadi ketika kita mengubah state komponen?

Ketika kita mengubah state komponen, maka secara otomatis react akan memanggil method render dari komponen yang berubah. Dengan artian, komponen tersebut direnderr ulang dengan state terbaru.

Komponen state kepada = "Andi" -> Klik tombol Joni -> state kepada berubah menjadi Joni -> Render ulang dengan state kepada = "Joni". Sehingga komponen direnderr browser menjadi "Halo, selamat belajar, Joni".

Overview

Kita akan membuat perbandingan antara implementasi jQuery dan React dalam membuat UI sederhana. Saya berusaha membuat contoh-contoh di buku ini sesederhana mungkin, mengabaikan faktor desain dan estetika. Tujuannya adalah agar kita bisa lebih fokus memahami fitur-fitur dasar React.

Kita akan membuat dua komponen masing-masing menggunakan jQuery dan React. Sehingga diharapkan kita akan mengetahui perbedaan jQuery dan React dari sisi

- *Event handling*
- *UI Update*
- Struktur

Pada bab sebelumnya saya sudah menjelaskan tentang Anatomi Dasar Komponen React sehingga kali ini saya akan menuliskan contoh kode membuat UI Sederhana dengan jQuery dan React, kamu tidak harus mengikutinya, tapi kamu silahkan perhatikan bagaimana dua implementasi kode bisa berjalan. Silahkan..

Komponen 1: Background Change

jQuery

HTML

```
  <div class="container">
  <input
     class="form-control"
     type="text"
     placeholder="Ketik kode warna atau nama warna dalam bahasa inggris ..."
   />
  <br>
  <div class="jumbotron" style="text-align: center; padding: 10px">
    GANTI WARNA
  </div>
  </div>
```

Javascript (jQuery)

```
$(document).ready(function(){
  $('input').on('input', function(){
    $('.jumbotron').css({background: $(this).val()})
  })
})
```

React

HTML

```
<div id="app"></div>
```

Javascript (JSX)

```
var App = React.createClass({
  getInitialState: function(){
    return {
      warna: 'lightyellow'
    }
```

```
  },

  gantiWarna: function(e){
    this.setState({warna: e.target.value})
  },

  render: function(){
    return (<div className="container">
      <input
        className="form-control"
        onChange={this.gantiWarna}
        placeholder="Ketik kode warna atau nama warna dalam bahasa inggris ..."
      />
      <br/>
      <div className="jumbotron"
        style={{
          padding: '10px',
          background: this.state.warna,
          textAlign: 'center'
          }}>
        GANTI WARNA
      </div>
    </div>)
  }
})

ReactDOM.render(<App/>, document.getElementById('app'))
```

JSBIN

http://jsbin.com/dumesa/edit?html,js,output REACT

http://jsbin.com/xorucoj/6/edit?html,js,output JQUERY

Komponen Validasi Email

jQuery

HTML

```
<div class="container">
<input
       name="email"
       type="text"
       class="form-control"
       placeholder="Email">
  <br>
<div class="email-warning alert alert-danger">
  Email tidak valid
</div>
<br>
<button class="btn btn-primary">Subscribe</button>
</div>
```

Javascript (jQuery)

```
$(document).ready(function(){
  $('.email-warning').hide()

  $('input[name="email"]').on('blur',function(){
    var email = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1
,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
    var valid = email.test($(this).val())
    if(!valid){
      $('.email-warning').show()
      $('button').prop('disabled', 'disabled')
    } else {
      $('.email-warning').hide()
      $('button').prop('disabled', '')
    }
  })
})
```

React

HTML

```
<div id="app"></div>
```

Javascript (JSX)

```
var EmailForm = React.createClass({
  getInitialState: function(){
    return {
      valid: true
    }
  },

  validateEmail: function(e){
    var email = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1
,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
    var valid = email.test(e.target.value)
    this.setState({valid: valid})
  },

  render: function(){
    return (<div className="container">
       <input
         name="email"
         className="form-control"
         placeholder="Email"
         onBlur={this.validateEmail}
       />
       <br/>
       <div className="alert alert-danger" style={this.state.valid ? {display: 'none'} : {
display: 'block'}}>
         Email tidak valid
       </div>
       <br/>
       <button disabled={!this.state.valid} className="btn btn-primary">Subscribe</button>
    </div>)
  }
})

ReactDOM.render(<EmailForm/>, document.getElementById('app'));
```

JSBIN

http://jsbin.com/juweqeb/1/edit?html,js,output JQUERY

http://jsbin.com/konohe/edit?js,output REACT

Hasil Observasi jQuery & React

Kamu sudah melihat dua implementasi pembuatan UI sederhana pada bagian sebelumnya. Sekarang kamu sudah mulai paham perbedaan dan kemiripan dalam:

- struktur komponen user interface
- event handling
- pengelolaan state

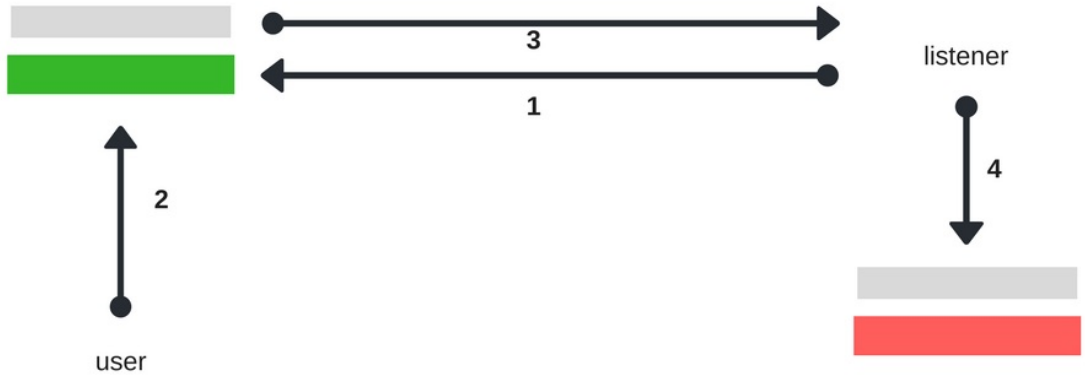
Struktur Komponen UI

Dalam jQuery kita menaruh kode kita dalam HTML. Lalu di tempat lain yang terpisah, kita menuliskan kode javascript kita yang akan memanipulasi DOM kode HTML kita. Biasanya dengan metode seperti ini, kita seringkali mencampur adukkan antara kode HTML dan Javascript, memang aplikasi kita berjalan tidak ada masalah tapi dalam skala yang lebih besar, hal ini bisa membingungkan untuk melakukan perubahan. Apalagi ketika kita bekerja sama dengan developer yang belum familiar dengan kode kita, butuh waktu yang lebih lama untuk mempelajari letak dan alur kode jQuery kita terlebih dahulu.

Event Handling

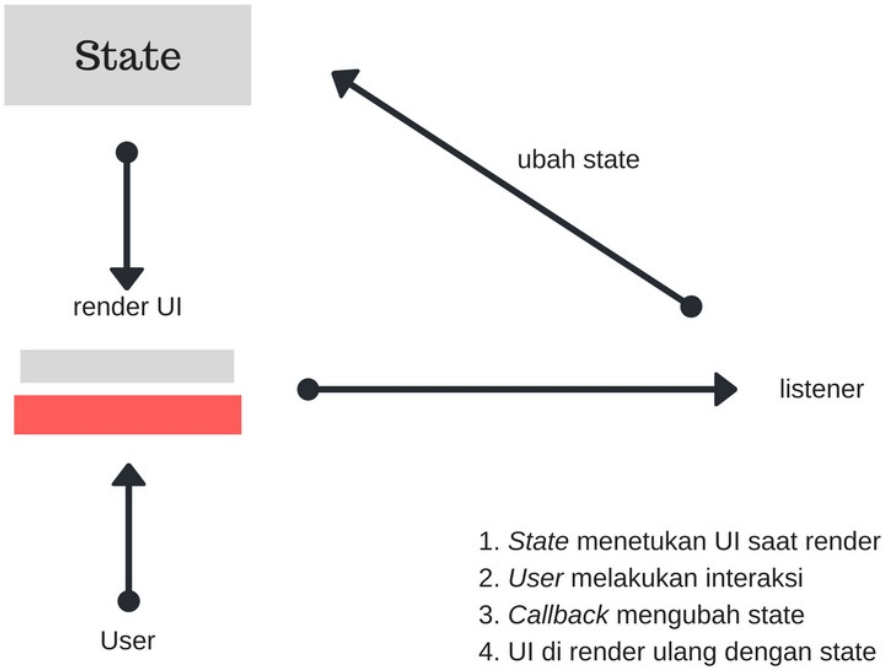
Sebenarnya mirip implementasi event handling antara jQuery dan React, pada jQuery kita meletakkan *listener* di suatu kode javascript, kita pilih dulu komponen yang akan kita dengarkan, lalu kita pasang callback jika terjadi *event.* Di react kita letakkan *handler* sebagai method dari class komponen tersebut, sehingga posisinya menjadi lebih jelas.

jQuery



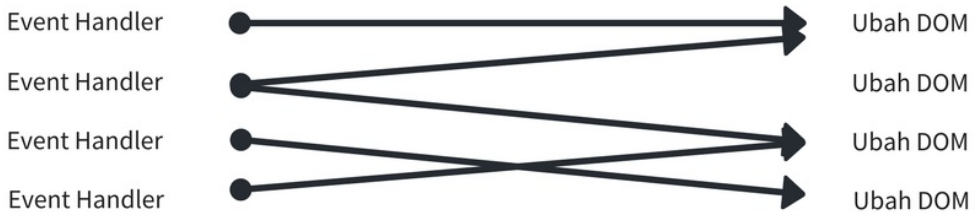
1. *Listener* bersiap-siap apa bila terjadi interaksi user
2. *User* melakukan interaksi dengan element
3. *Trigger* listener *callback* akibat interaksi user
4. *Callback* melakukan update DOM secara langsung

React

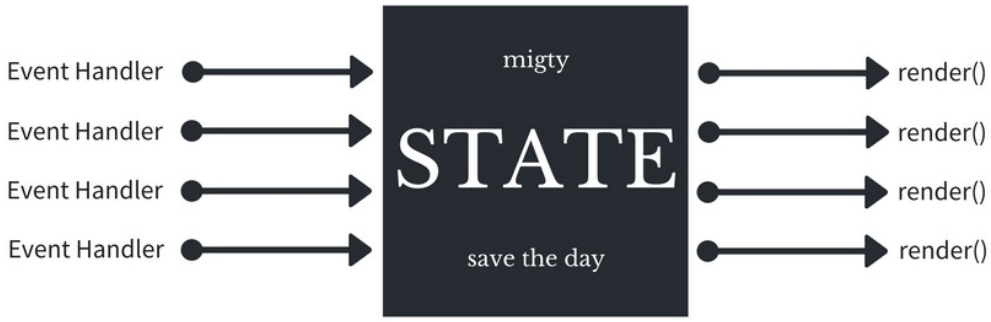


1. *State* menetukan UI saat render awal
2. *User* melakukan interaksi
3. *Callback* mengubah state
4. UI di render ulang dengan state baru (panggil render() method)

jQuery



React



Pengetahuan Lebih Lanjut

Latihan Komponen Welcome Screen

http://jsbin.com/gobadi/edit?html,js,output

Topik *Advanced*

Kita baru saja mempelajari dasar dari React, lalu apa selanjutnya? Banyak hal yang bisa memudahkan kamu ketika menggunakan React, antara lain menggunakan sintaks ES6, menggunakan state container seperti Redux atau menggunakan observer pattern seperti Mobx. Tapi kesemuanya adalah opsional, akan tetapi jika kamu serius ingin belajar React, saya sangat menyarankan untuk mempelajarinya. Tidak harus semuanya, minimal ES6, dan pilih antara Redux atau Mobx.

Kabar baiknya, @arrowfunxtion berkomitmen untuk share hal-hal di atas. Oleh karena itu like dan share @arrowfunxtion ya.

ES5 dan ES6

Apa yang sudah kita pelajari pada bab-bab sebelumnya menggunakan sintaks ES5, asumsi saya, teman-teman terbiasa menggunakan ES5 sesuai dengan target audience buku ini. Namun, pada praktiknya teman-teman mungkin akan tertarik untuk menggunakan ES6. Apa itu? ES6 atau EcmaScript 6, merupakan standar javascript terbaru dan belum semua browser mendukungnya.

Untuk menggunakan ES6, maka kita memerlukan bantuan pre-compiler semisal babel. Jadi apabila teman-teman ingin memanfaatkan fitur-fitur ES6, seperti Arrow Function, Spread Operator, dll, teman-teman bisa mempelajari Babel, Webpack, dan sintak ES6 itu sendiri. Yang jelas bukan nambah ribet ya, tapi banyak fitur-fitur powerful yang siap kamu pakai!

Apa contohnya?

Arrow Function

```
// ES5
$.ajax({
    ...
    success: function(response){
        console.log(this) // this merujuk ke jQuery
    }
})

//ES6
$.ajax({
    ...
    success: (response)=>{
        console.log(this) // this merefer ke scope yang memanggil $.ajax, bukan ke jQuery.
    }
})
```

Arrow function sangat berguna sekali dalam pemakaian sehari-hari terutama jika kita menggunakan class-class komponen.

Spread Properties

```
var properties = {
    color: 'red',
    height: '400px',
    }

// ES5
<Komponen color={properties.color} height={properties.height} />
// Bayangkan kalo propertiesnya ada banyak, bisa males kan ngetik satu-satu
// Bandingkand dengan ES6 di bawah ini

// ES6
<Komponen {...properties} />
// Cukup dengan {...properties} untuk menuliskan hasil yang sama dengan sampel sebelumnya.
```

Dan masih banyak lagi fitur-fitur ES6 lainnya. Jadi kamu sekarang semakin tertarik untuk mempelajari React dan ES6, like dan share http://fb.com/arrowfunxtion ya. Eh tunggu dulu, masih ada yang keren lagi untuk kamu padukan dengan React, apa itu? Redux dan Mobx, apa itu? Silahkan lanjutkan membaca ...

Redux

Redux merupakan state container yang akan membantu developer menulis kode yang clean, terstruktur rapi dan konsisten. React terkenal dengan alur data yang satu arah dan umumnya menggunakan implementasi dari arsitektur Flux. Namun, banyak orang menganggap Flux justru menambah kerumitan dalam workflow mereka sehingga lahirlah Redux. Dengan redux diharapkan developer mampu menerapkan Flux tanpa mengadopsi kerumitannya.

Seperti apa sih redux? silahkan bisa dibaca di http://redux.js.org/.

Namun sayangnya, sebagian orang menanggap Redux pun masih agak rumit, lalu beberapa orang seperti saya justru lebih nyaman menggunakan MobX, apa itu MobX? Baca halaman selanjutnya ya..

Mobx

MobX adalah tool favorit saya, sangat simpel dan tidak banyak aturan yang harus saya patuhi. Saya bisa implement sesuai dengan kebutuhan saya. MobX bukanlah state container seperti Redux. Justru karena itulah, Mobx lebih mudah untuk dipelajari dan diterapkan, setidaknya itu untuk saya, Developer Kemarin Sore.

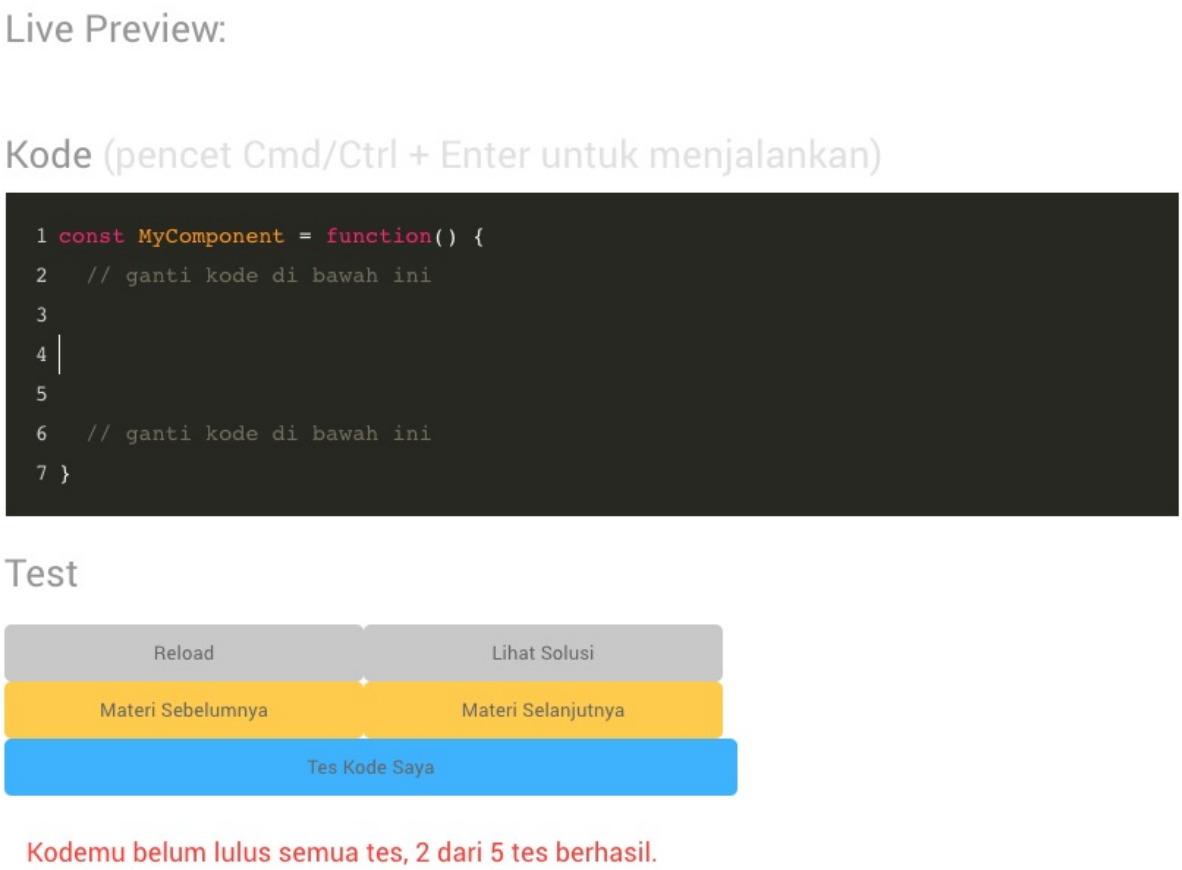
Menggunakan MobX ibarat menemukan tambatan hati buat si React. Kamu mungkin belum begitu memahami mengapa bisa begitu, untuk itu tetap stay tune dengan http://facebook.com/arrowfunxtion agar bisa terus update.

Selamat!

Kamu baru saja mempelajari dasar-dasar React sehingga kamu sekarang lebih paham keunggulan dibandingkan menggunakan jQuery saja. Apalagi setelah ini?

Terima Kasih, Semoga Bermanfaat

Eh, sebelum kamu tutup ebook ini, kamu perlu tahu ini, beberapa waktu lagi saya akan launching situs belajar React seacara interaktif. Kabar baiknya, semua boleh belajar di situ alias GRATIS untuk semua.



Oleh karena itu,

Mari tetap bersilaturahmi dengan saya, silaturahmi membuka rezeki ^_^

- **facebook**: Muhammad Azamuddin
- **linkedin**: https://www.linkedin.com/in/azamuddin/
- **facebook** page: @ArrowFunxtion

KEEP CODING, BE CREATIVE!