React and React Native Basics

Data Flow di React, PropTypes, Atomic Design

5

PERHATIAN

Jangan lupa untuk isi form absensi. Bagi instruktur mohon mengisi form absensi yang telah diinfokan dan konfirmasi nomor urut peserta yang hadir.

Untuk student harap mengisi form absensi di sini sebelum kelas dimulai. Untuk kode peserta dapat ditanyakan kepada instruktur dan jangan lupa mencantumkan pertemuan ke 1

Data Flow di React

Data Flow di React

Intro

Seperti yang sudah kita bahas di bagian sebelumnya, aliran data React ini bersifat satu arah. Dari view ke action ke store dan dari store balik lagi ke view.

Untuk saling berinteraksi antar komponen, React mempunyai dua cara: state dan props. Masing-masing punya tugas dan tujuan yang berbeda. State digunakan untuk menyimpan kondisi saat ini untuk komponen, sementara props digunakan untuk mengirimkan informasi ke komponen lain.

Data Flow di React

State

Aplikasi web modern biasanya dibangun berdasarkan data atau istilah kerennya data-driven. Bayangkan aplikasi social media seperti Facebook, Twitter atau Instagram. Sebagai social media, data adalah aliran darah. Bayangkan sebuah news feed yang secara dinamis bertambah sesuai dengan aliran data yang masuk. Disinilah salah satu keunggulan utama React dan memang React dibuat untuk dapat menangani hal-hal seperti ini.

Mari kita lihat contoh yang lebih sederhana agar kita dapat melihat dan mengetahui apakah state itu sebenarnya. Percaya atau tidak, hampir semua aplikasi itu memiliki state. Apakah teman-teman mendevelop penggunakan View, jQuery, Angular, Ember dan lain sebagainya pasti semua library/framework tersebut memiliki cara untuk menangani state. Secara definisi, state adalah:

Seluruh informasi dari sebuah aplikasi yang dibutuhkan pada satu waktu.

Definisi diatas memang sudah disederhanakan, tapi cukup menjelaskan apa itu state dan tujuan dari state. Dengan kata lain state adalah seluruh informasi yang kita butuhkan untuk menggambarkan user interface pada satu waktu.

Data Flow di React

State dalam Class Component

Mari kita, perhatikan potongan kode di samping ini

MENDEFINISIKAN / MEMBUAT STATE

Perhatikan baris ke 6 - 8 dari kode di samping ini. Inilah contoh mendefinisikan sebuah state di dalam sebuah class component, yaitu di dalam constructor nya si component

MEMBACA STATE DI DALAM RENDER

Perhatikan baris ke 13 dari kode di samping ini. Inilah contoh cara membaca state di dalam render nya component

```
1   import React from 'react';
2
3   export default class Users extends React.Component {
4     constructor() {
5       super()
6       this.state = {
7         username: 'user01'
8       }
9     }
10
11    render() {
12      return (
13        <h1>{this.state.username}</h1>
14      )
15    }
16  }
17
```

Data Flow di React

State dalam Class Component

```
1   import React from 'react';
2
3   export default class Users extends React.Component {
4     constructor() {
5       super()
6       this.state = {
7         username: 'user01'
8       }
9     }
10
11    getUsername() {
12      return this.state.username
13    }
14
15    render() {
16      return (
17        <h1>{this.getUsername()}</h1>
18      )
19    }
20  }
21
```

MEMBACA STATE DI DALAM FUNCTION

Perhatikan baris ke 11 - 13 dari kode di samping ini. Inilah contoh cara membaca state di dalam function yang didefinisikan di dalam class component

Dan pada baris 17 adalah contoh kode yang memanggil function yang sudah kita buat di atas

Data Flow di React

State dalam Class Component

```
1   import React from 'react';
2
3   export default class Users extends React.Component {
4     constructor() {
5       super()
6       this.state = {
7         username: ''
8       }
9     }
10
11    setUsername = () => {
12      this.setState({
13        username: 'user01'
14      })
15    }
16
17    render() {
18      return (
19        <div>
20          <h1>{this.state.username}</h1>
21          <br />
22          <button onClick={this.setUsername}>Set Username</button>
23        </div>
24      )
25    }
26  }
27
```

MENULIS STATE DI DALAM SEBUAH FUNCTION

Perhatikan baris ke 12 - 14 dari kode di samping ini. Ini adalah contoh kode untuk melakukan penulisan terhadap sebuah state. SANGATLAH DILARANG untuk melakukan update langsung dengan syntax

✗ `this.state.username = 'user01'`

Function setUsername di trigger dengan klik button di baris 22

```
1  import React from 'react';
2
3  export default class Counter extends React.Component {
4    constructor() {
5      super()
6      this.state = {
7        counter: 0
8      }
9    }
10
11   increment = () => {
12     this.setState({
13       counter: this.state.counter + 1
14     })
15   }
16
17   decrement = () => {
18     this.setState({
19       counter: this.state.counter - 1
20     })
21   }
22
23   render() {
24     return (
25       <div>
26         <h3>{this.state.counter}</h3>
27         <br />
28         <button onClick={this.increment}>+</button>
29         <button onClick={this.decrement}>-</button>
30       </div>
31     )
32   }
33 }
34
```

Data Flow di React

State dalam Class Component

UPDATE STATE DI DALAM SEBUAH FUNCTION

Pada contoh ini, kita membuat 2 buah function : increment dan decrement untuk menambah dan mengurangi angka counter.

Kedua function tersebut di trigger ketika tombol pada baris 28 dan 29 di click

Perhatikan bahwa kita melakukan update state dengan setState, dan kita tetap memanggil state awalnya (this.state.counter) untuk di update dengan ditambah atau dikurangi 1

Data Flow di React

Props dalam Class Component

Props adalah salah satu fasilitas dari React untuk bisa menerima data yang dikirimkan dari component lain. Data ini sangatlah luas, bisa jadi berupa informasi statis, atau state dari component pengirim.

PROPS DARI INFORMASI STATIS

Pada contoh disamping, didemonstrasikan pengiriman props dari component Hero ke component HeroName dengan menggunakan nama props “name”, seperti terlihat pada baris ke 11

Component HeroName menerima props dengan cara yang ditunjukan pada baris ke 5

```
1   import React from 'react';
2
3   class HeroName extends React.Component {
4     render() {
5       return <h3>{this.props.name}</h3>
6     }
7   }
8
9   export default class Hero extends React.Component {
10    render() {
11      return <HeroName name="Hercules" />
12    }
13  }
14
```

Data Flow di React

Props dalam Class Component

Props adalah salah satu fasilitas dari React untuk bisa menerima data yang dikirimkan dari component lain. Data ini sangatlah luas, bisa jadi berupa informasi statis, atau state dari component pengirim.

```
1   import React from 'react';
2
3   class HeroName extends React.Component {
4     render() {
5       return <h3>{this.props.name}</h3>
6     }
7   }
8
9   export default class Hero extends React.Component {
10    constructor() {
11      super()
12      this.state = {
13        name: "Hercules"
14      }
15    }
16
17    render() {
18      return <HeroName name={this.state.name} />
19    }
20  }
21
```

PROPS DARI STATE YANG DIKIRIM

Pada contoh disamping, didemonstrasikan pengiriman props dari component Hero ke component HeroName dengan menggunakan nama props “name”, seperti terlihat pada baris ke 18. Dan value yang dikirim adalah value dari state yang bernama “name”

Component HeroName menerima props dengan cara yang ditunjukan pada baris ke 5

Data Flow di React

Props dalam Class Component

Props adalah salah satu fasilitas dari React untuk bisa menerima data yang dikirimkan dari component lain. Data ini sangatlah luas, bisa jadi berupa informasi statis, atau state dari component pengirim.

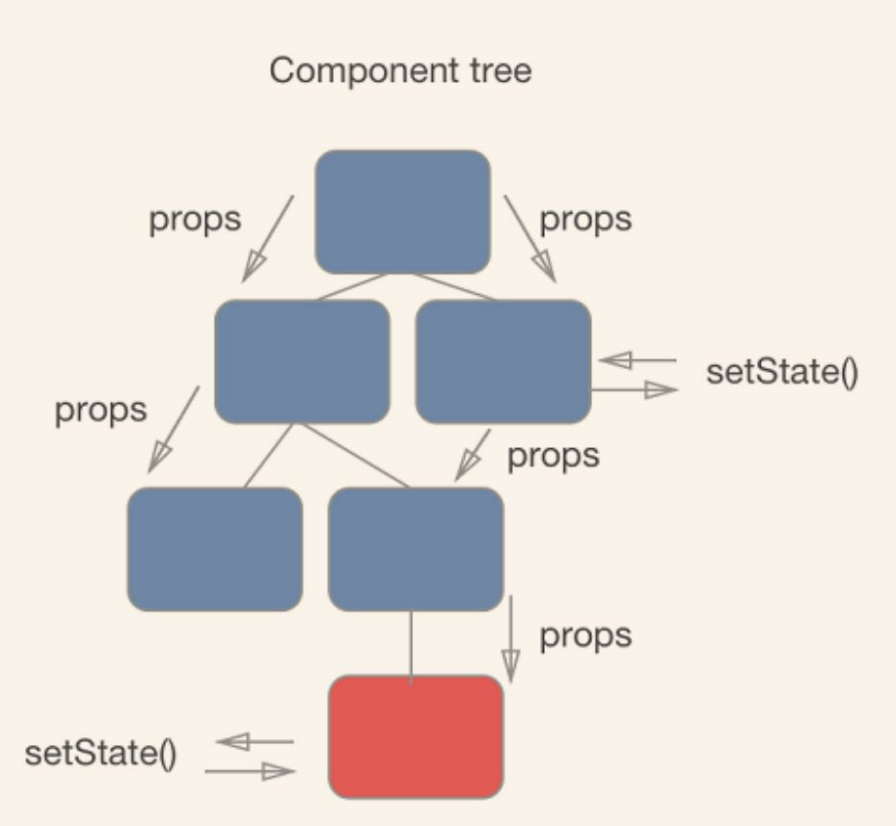
```
1   import React from 'react';
2
3   class HeroName extends React.Component {
4     render() {
5       return <h3>{this.props.name}</h3>
6     }
7   }
8
9   export default class Hero extends React.Component {
10    constructor() {
11      super()
12      this.state = {
13        name: "Hercules"
14      }
15    }
16
17    render() {
18      return <HeroName name={this.state.name} />
19    }
20  }
21
```

PROPS DARI STATE YANG DIKIRIM

Pada contoh disamping, didemonstrasikan pengiriman props dari component Hero ke component HeroName dengan menggunakan nama props “name”, seperti terlihat pada baris ke 18. Dan value yang dikirim adalah value dari state yang bernama “name”

Component HeroName menerima props dengan cara yang ditunjukan pada baris ke 5

Data Flow di React



Hubungan State dan Props

PropTypes +

PropTypes

PropTypes

PropTypes memungkinkan kita melakukan validasi terhadap props yang kita harapkan.

Pertama-tama kita harus melakukan instalasi library proptypes terlebih dahulu melalui terminal, pada root folder aplikasi kita, dengan perintah :

```
> npm install prop-types
```

…kemudian, lakukan import seperti pada baris 2

Misal pada contoh kali ini, kita akan melakukan pemeriksaan apakah **name** adalah sebuah string.

Kita akan membuat skenario yang akan trigger error, karena, state yang kita kirimkan (name) berisi boolean. Lihat pada baris 18

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3
4  class HeroName extends React.Component {
5    render() {
6      return <h3>{this.props.name}</h3>
7    }
8  }
9
10 HeroName.propTypes = {
11   name: PropTypes.string
12 }
13
14 export default class Hero extends React.Component {
15   constructor() {
16     super()
17     this.state = {
18       name: true
19     }
20   }
21
22   render() {
23     return <HeroName name={this.state.name} />
24   }
25 }
26
```

PropTypes

PropTypes

```
▸Warning: Failed prop type: Invalid prop `name` of type `boolean` supplied to `HeroName`, expected `string`.        index.js:1
    at HeroName (http://localhost:3000/main.a0afb83….hot-update.js:27:1)
    at Hero (http://localhost:3000/main.a0afb83….hot-update.js:46:5)
    at div
    at App (http://localhost:3000/static/js/main.chunk.js:228:1)
```

Di atas adalah contoh hasil error yang muncul pada console browser ketika aplikasi dijalankan

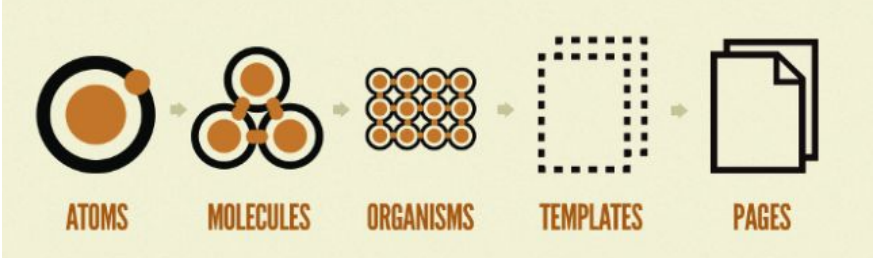
PropTypes ini bisa melakukan validasi untuk berbagai tipe data. Mulai dari tipe data scalar hingga ke struktur data seperti array, object dan kita juga bisa melakukan validasi sesuai dari bentuk dari sebuah object (shapeOf).

```jsx
import React from 'react';
import PropTypes from 'prop-types';

class HeroName extends React.Component {
  render() {
    return <h3>{this.props.name}</h3>
  }
}

HeroName.propTypes = {
  name: PropTypes.string
}

export default class Hero extends React.Component {
  constructor() {
    super()
    this.state = {
      name: true
    }
  }

  render() {
    return <HeroName name={this.state.name} />
  }
}

```

Atomic Design

Atomic Design

Intro



Atomic Design adalah pendekatan desain yang dipopulerkan oleh Brad Frost, yang memecah elemen aplikasi web menjadi bagian-bagian modular hingga yang paling kecil.

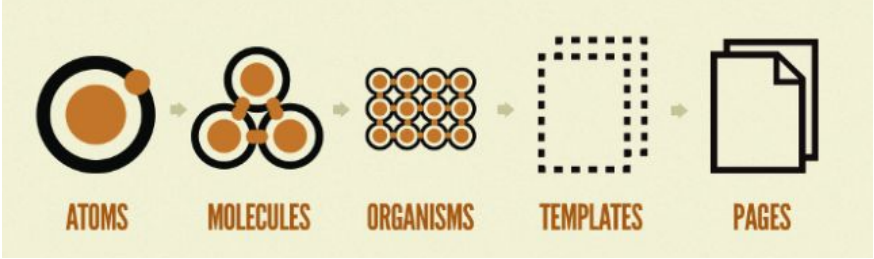
Elemen-elemen tadi dibagi menjadi beberapa level atau hirarki: atom, molekul, organisme, template, dan laman.Tujuannya adalah untuk mereduksi adanya kode berulang, mereduksi waktu set up, dan meningkatkan konsistensi antar komponen dan tampilan.

Sistem ini menggunakan pattern library sebagai repository dari komponen-komponen web yang bisa diimport (importable) dan dapat digunakan kembali (reusable). Hal ini juga sudah diterapkan, misalnya oleh Shopify yang memiliki Polaris, IBM yang memiliki Carbon, atau Lonely Planet yang memiiki Rizzo.

Dengan pattern library ini,biasanya waktu yang dibutuhkan memang lebih panjang daripada pendekatan desain laman (pages) biasa. Namun, metode ini lebih kolaboratif dan menjamin reusability dan maintainability untuk jangka panjang.

Atomic Design

Intro



Dalam prakteknya, komponen-komponen web dipecah menjadi bagian-bagian yang lebih kecil (modular) dan dikelompokkan sesuai level dan hirarki.

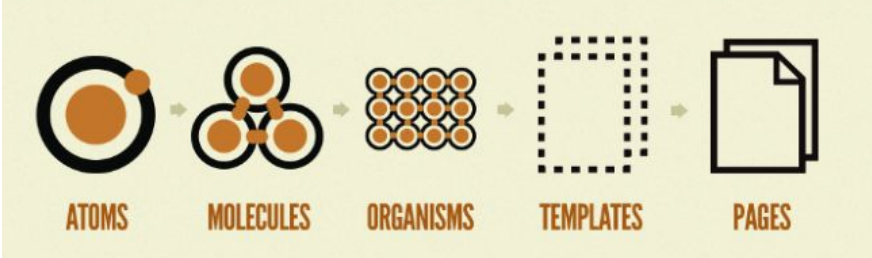
Misalnya pada level atom, telah ditentukan komponen-komponen kecil seperti tombol ‘Search’, input pencarian beserta placeholder, dan pelabelan ‘Search the Site’. Tak ketinggalan pula penentuan elemen-elemen palet warna, tipografi, ukuran teks, dan margin dilakukan pada level ini.

Di level molekul (molecule), elemen-elemen pada level atomis dikelompokkan dan diatur dalam modul yang lebih besar. Misalnya modul pencarian yang tersusun dari komponen-komponen: tombol Search, input pencarian (text field), dan label ‘Search the Site’.

Selanjutnya pada level organisme (organism), modul pencarian juga dikelompokkan dengan elemen-elemen lain seperti menu dan logo sebagai elemen header. Meningkat lagi ke hirarki di atasnya, modul header kemudian diatur bersama-sama dengan elemen-elemen lainnya menjadi sebuah template.

Atomic Design

Intro



Yang terakhir, template yang sudah ada dibuat menjadi sebuah halaman spesifik yang sudah dilengkapi dengan representasi konten yang nyata dan menunjukkan antarmuka yang sebenarnya dari sebuah website. Misalnya laman Home/ Beranda, atau laman blog yang dibangun dari sebuah template.

Kelebihan yang ditawarkan oleh metode Atomic Design, adalah kemampuan untuk pindah dari konsep yang abstrak kepada hal yang konkrit. Selain itu, ada pemisahan yang jelas antara struktur sebuah website dengan konten.

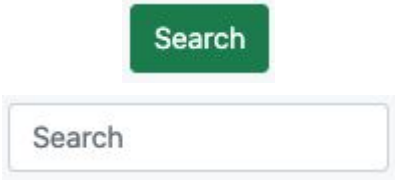
Adapun kekurangan dari sistem ini adalah tidak ada orang khusus yang merawat pattern library. Oleh karenanya, komponen-komponen di pattern library bisa menjadi usang, atau tidak terpantau sehingga menjadi penuh.

Atomic Design

Implementasi

Level Atom

Pada level ini, kita akan kumpulkan komponen-komponen yang akan kita gunakan untuk membuat tampilan yang kita inginkan



Level Molekul

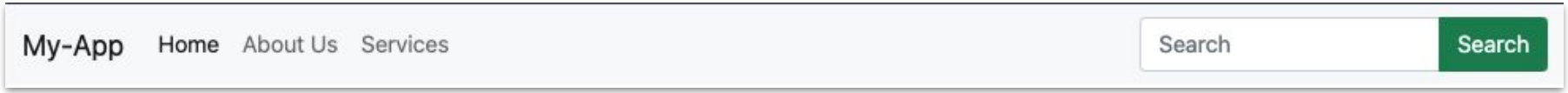
Pada level ini, komponen - komponen pada level atom, yang seirama, dikelompokkan dan diatur dalam modul yang lebih besar.

Atomic Design

Implementasi

Level Organisme

Pada level ini, modul pencarian juga dikelompokkan dengan elemen-elemen lain seperti menu dan logo sebagai elemen header. Meningkat lagi ke hirarki di atasnya, modul header kemudian diatur bersama-sama dengan elemen-elemen lainnya menjadi sebuah template.



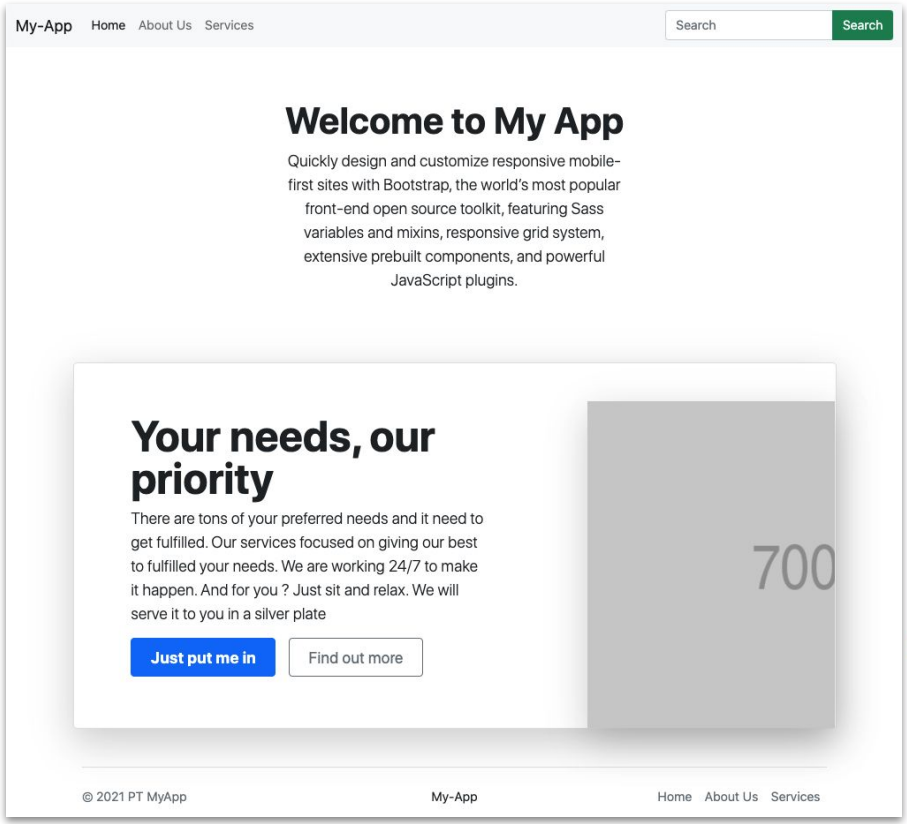
Atomic Design

Implementasi

Level Template

Pada level ini, kita akan gabungkan organisme-organisme yang sudah kita bangun, sehingga menjadi sebuah bentuk baku yang biasa kita sebut dengan template. Template ini lebih ke arah design kasar dari halaman web yang akan kita bangun, sehingga jelas penempatan-penempatan nya

Apakah ini sudah selesai ? Tentu saja belum. Selanjutnya, kita akan isikan materi-materi yang akan kita implementasikan pada template ini

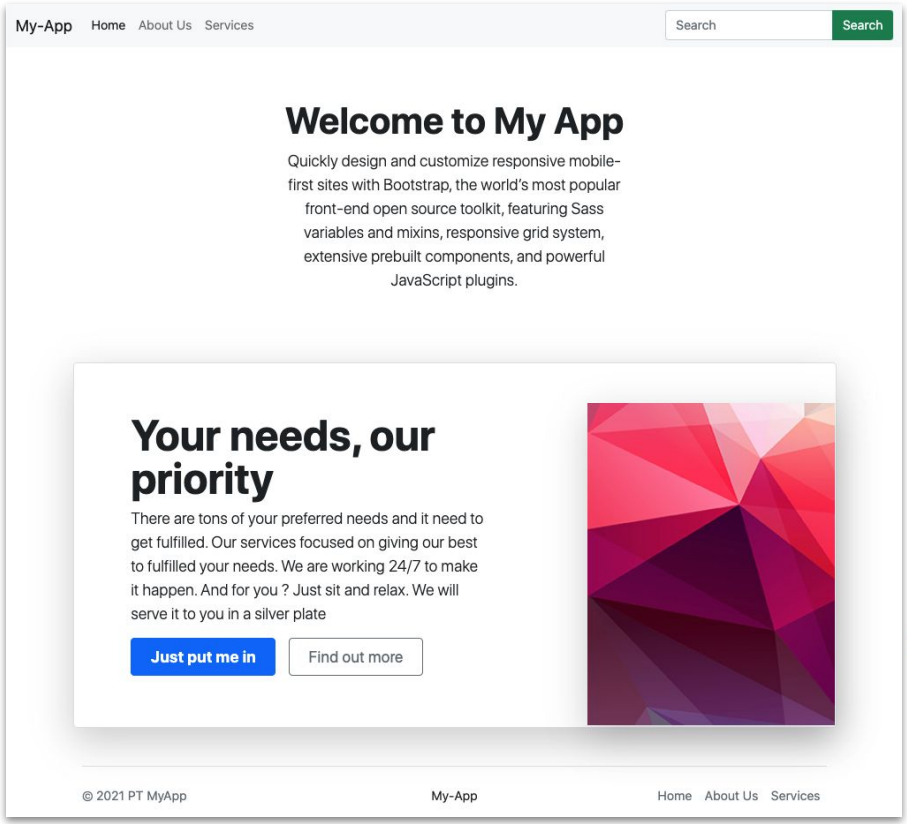


Atomic Design

Implementasi

Level Page

Inilah dia bentuk final dari Atomic Design kita. The real page. Inilah tujuan akhir kita, dimana kita sudah melengkapi template yang kita bangun dengan semua asset-asset yang cenderung kepada kelengkapan dari sisi estetis



Thank You

PT Hacktivate Teknologi Indonesia
Gedung Aquarius Pondok Indah
Jalan Sultan Iskandar Muda No.7
Kebayoran Lama, Jakarta Selatan

www.hacktiv8.com