

TUGAS PERORANGAN

**BELAJAR MENGGUNAKAN JAVASCRIPT ES6
DENGAN BABEL DAN WEBPACK**

Disusun sebagai
MATA KULIAH: PBF

Oleh:

Rois Dwi Admaja/1741720193

TI – 3B / 23



PROGRAM STUDI: D-IV TEKNIK INFORMATIKA

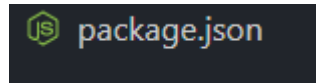
JURUSAN: TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2020

INSTALLASI BABEL

1. Sebelum installasi babel membuat sebuah file yaitu package.json



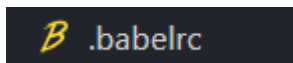
2. Setelah itu menjalankan `npm install --save-dev @babel/core @babel/cli` pada terminal

```
PS E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6> npm install --save-dev @babel/core @babel/cli
[.....] \ extract:is-data-descriptor: sill extract is-
```

3. Tambahkan scripts di atas dependencies pada file package.json seperti dibawah ini

```
1  {
2    "scripts": {
3      "build": "babel src -d lib"
4    },
5    "devDependencies": {
6      "@babel/cli": "^7.8.4",
7      "@babel/core": "^7.8.4",
8      "@babel/preset-env": "^7.8.4"
9    }
10 }
11
```

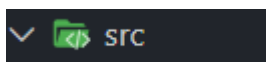
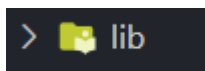
4. Lalu membuat file .babelrc



5. Lalu menambahkan script presets pada .babelrc

```
1  {
2    "presets": ["@babel/preset-env"]
3  }
```

6. Setelah itu buat folder src dan lib



7. Untuk mengecek apakah bisa dijalankan, buat index.html dan memasukkan template html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-
6    <title>Learn ES6</title>
7  </head>
8  <body>
9    <h1>Belajar ES6</h1>
10
11    <script src="lib/index.js"></script>
12  </body>
13 </html>
```

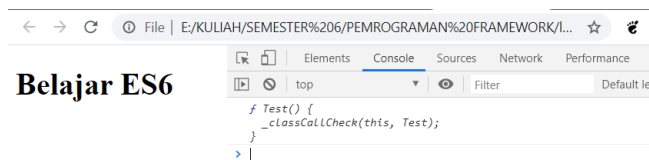
8. Lalu buat file index.js pada folder src dan mengisi class test lalu ditampilkan pada console.log()

```
1  class Test {  
2  
3  }  
4  
5  console.log(Test);
```

9. Lalu jalankan perintah npm run build pada terminal

```
PS E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6> npm run build  
@ build E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6  
> babel src -d lib  
  
Successfully compiled 1 file with Babel.
```

10. Buka browser dan cek di console

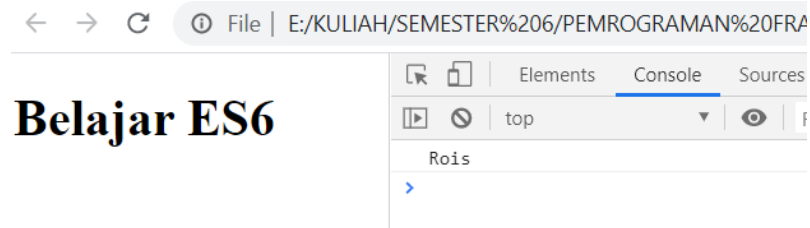


VAR, LET DAN CONST

A. VAR

Var merupakan variabel berjenis function scope berikut contoh dari variabel var

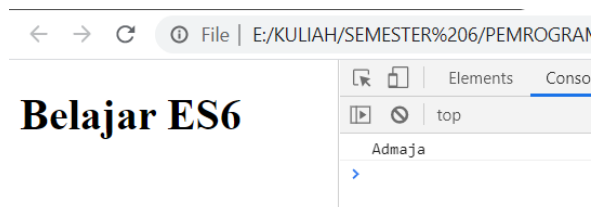
```
1  var nama = 'Rois';  
2  
3  console.log(nama);  
4
```



Pada variabel var jika kita mendeklarasikan ulang variabel tersebut tidak akan terjadi error berikut contoh kode

```
1  var nama = 'Rois';  
2  var nama = 'Admaja';  
3  console.log(nama);  
4
```

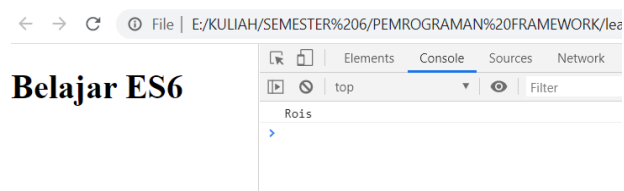
Maka hasilnya akan berganti dengan variabel yang var yang kedua



B. LET

Variabel let sama seperti variabel const yaitu berjenis block scope

```
5  let nama = 'Rois';  
6  
7  console.log(nama);
```



Pada variabel let jika kita mendeklarasikan ulang variabel tersebut maka akan terjadi error

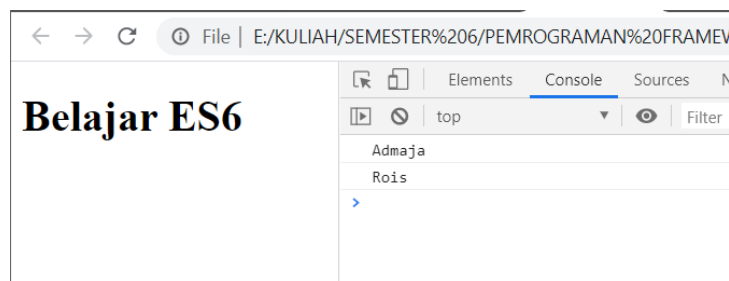
```
5 let nama = 'Rois';
6 let nama = 'Admaja';
7
8 console.log(nama);
```

```
SyntaxError: E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6\src
\index.js: Identifier 'nama' has already been declared (6:4)
4
5 let nama = 'Rois';
> 6 let nama = 'Admaja';
   ^
7
8 console.log(nama);
at Parser.raise (E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn
-es6\node_modules\@babel\parser\lib\index.js:7017:17)
at ScopeHandler.checkRedeclarationInScope (E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6\node_modules\@babel\parser\lib\index.js:7017:17)
```

Tetapi jika kita pisahkan salah satu variabel dan kita masukkan salah variabel didalam scope maka tidak akan terjadi error

```
5 let nama = 'Rois';
6 {
7   let nama = 'Admaja';
8   console.log(nama);
9 }
10
11 console.log(nama);
```

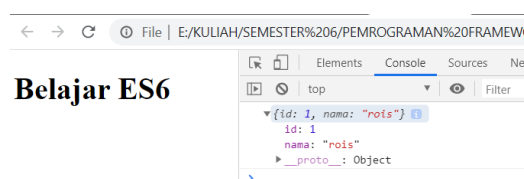
Disini akan menampilkan hasil nilai dari dua variabel diatas, tetapi lebih di prioritaskan pada variabel yang berada dalam scope



C. CONST

Const merupakan variabel konstanta dimana variabel tersebut tidak bisa di reassign ulang, tetapi kita masih bisa mengganti, atau menambahkan object. Contoh koding

```
5 const person = {
6   id : 1,
7   nama : 'rois'
8 };
9
10 console.log(person);
11
```



Jika kita mendeklarasikan ulang akan terjadi error

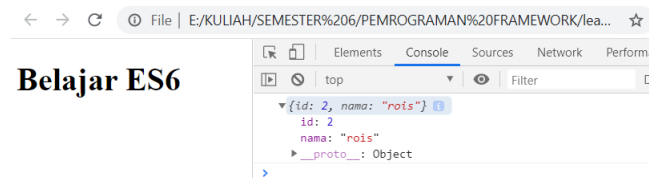
```
5  const person = {
6    id : 1,
7    nama : 'rois'
8  };
9
10 const person = {
11   id:2,
12   nama: john
13 }
14
15 console.log(person);
16
```

```
SyntaxError: E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6\src
\index.js: Identifier 'person' has already been declared (10:6)

8  |   };
9  |
> 10 |   const person = {
    |         ^
11  |     id:2,
12  |     nama: john
13  |   }
    at Parser.raise (E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn
```

Untuk menambahkan suatu objek baru akan seperti ini

```
5  const person = {
6    id : 1,
7    nama : 'rois'
8  };
9
10 person.id=2;
11
12 console.log(person);
13
```



Belajar ES6

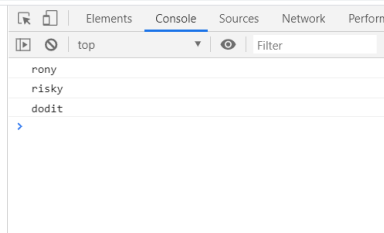
SYNTAX ARROW

Syntax arrow merupakan sitak yang digunakan untuk mempersingkat suatu fungsi ada beberapa penulisan syntax arrow, berikut ;

1. Contoh pertama

```
5 var members = ['rony', 'risky', 'dodit'];
6
7 members.forEach(member =>{
8   console.log(member);
9 })
10
11
```

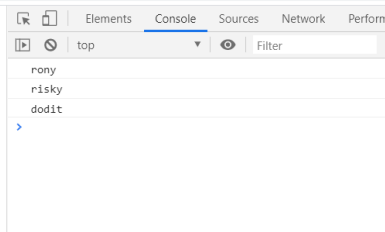
Belajar ES6



2. Contoh kedua

```
5 var members = ['rony', 'risky', 'dodit'];
6
7 members.forEach(member => console.log(member))
8
```

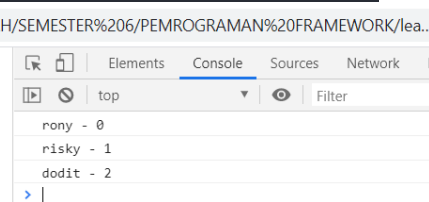
Belajar ES6



3. Contoh ketiga dengan lebih dari satu parameter

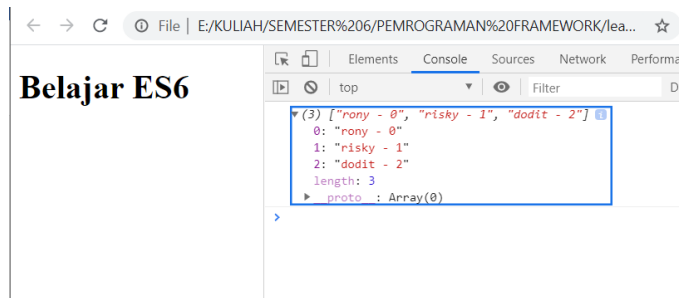
```
5 var members = ['rony', 'risky', 'dodit'];
6
7 members.forEach((member, index) => {
8   console.log(member + ' - ' + index);
9 })
10
```

Belajar ES6



4. Contoh keempat dengan menggunakan map

```
5 let members = ['rony', 'risky', 'dodit'];
6
7 let membersindex = members.map((member, index)=>{
8   return member + ' - ' + index;
9 })
10
11 console.log(membersindex);
12
13
```



5. Contoh kelima

```
5 let sekolah = {
6   members : ['risqi', 'dodit', 'punisher'],
7   getMembers() {
8     this.members.map((name) => {
9       console.log(this);
10     })
11   }
12 }
13
14
15 console.log(sekolah.getMembers());
16
```

The screenshot shows the same web browser window with 'Belajar ES6' on the page. The developer console displays the output of the `console.log(sekolah.getMembers());` statement. It shows three identical log entries, each representing an object with the following structure:

- `members`: Array(3) ["risqi", "dodit", "punisher"]
- `getMembers`: `f getMembers()`
- `__proto__`: Object

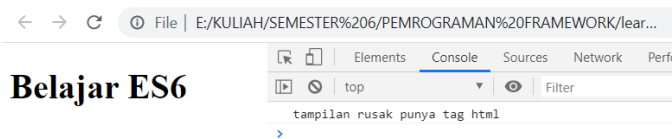
Below the log entries, the console shows `undefined`.

DEFAULT PARAMETER

Parameter fungsi default memungkinkan parameter bernama diinisialisasi dengan nilai default jika tidak ada nilai atau 'undefined' dilewatkan.

1. Contoh pertama penulisan default parameter

```
6 let createTag = (title, tag) => {
7   console.log(title + ' punya tag ' + tag);
8 }
9
10
11 createTag('tampilan rusak', 'html')
```

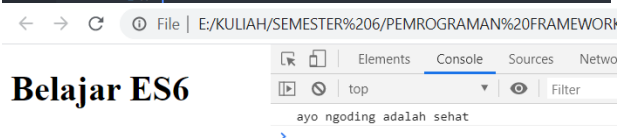


2. Contoh kedua

```
6 let createTag = ([title='ngoding', tag = 'sehat']) => {
7   console.log(title + ' adalah ' + tag);
8 }
9
10
11 createTag();
```

3. Contoh ke tiga

```
5 let generateTitel = ()=>{
6   return 'ayo ngoding'
7 }
8
9
10 let createTag = (title=generateTitel(), tag = 'sehat') => {
11   console.log(title + ' adalah ' + tag);
12 }
13
14
15 createTag();
```

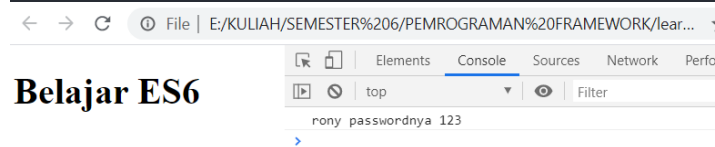


REST DAN SPREAD

1. Contoh 1

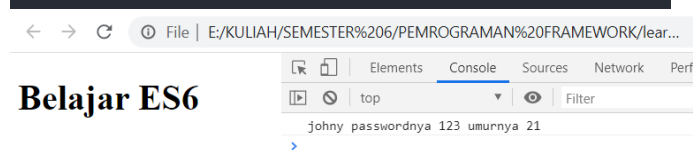
Titik 3(...) digunakan untuk memecah array

```
5 let signIn = (username, password) => {  
6   console.log(username + ' passwordnya '+password);  
7 }  
8  
9  
10 let data = ['rony', '123']  
11  
12 signIn(...data)
```



2. Contoh 2 membalik parameter

```
5 let signIn = (...member) => {  
6   console.log(username + ' passwordnya '+password  
7   + ' umurnya ' + umur);  
8 }  
9  
10  
11 let username = 'johny'  
12 let password = '123'  
13 let umur = '21'  
14  
15 signIn(username, password, umur)
```



TEMPLATE STRING

Template string berfungsi untuk memanipulasi html

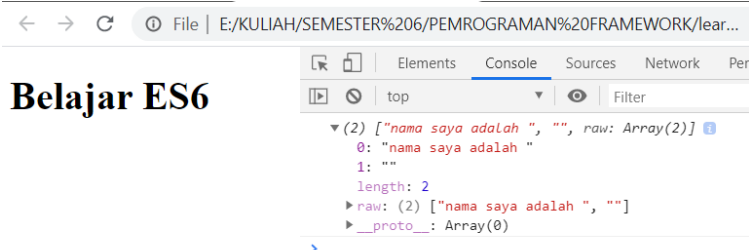
1. Contoh 1

```
5 let username = 'rois';
6 let umur = 20
7
8 let text = `Member ${username} umur ${20}`
9 let div = `
10   <div> ${username} </div>
11   <p> ${umur}</p>
12 `
13
14 document.write(div);
```



2. Contoh 2

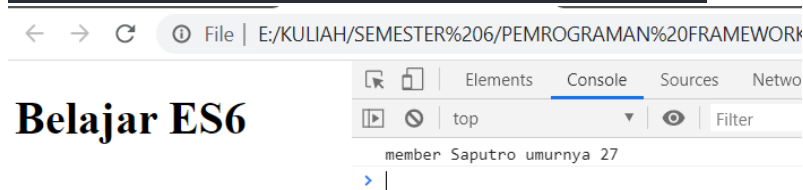
```
5 let username = 'rois';
6 let umur = 20
7
8 let test =(strings, username, umur)=>{
9   let strings1 = strings[0]
10  let strings2 = strings[1]
11
12  console.log([username]);
13
14 }
15
16 let output = test`nama saya adalah ${username}`
```



SHORTHAND AND DESTRUCTURING

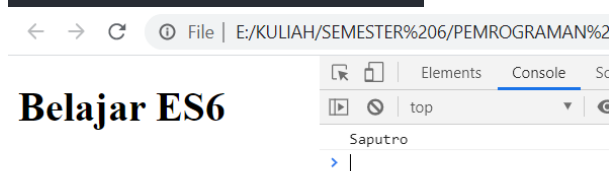
1. Contoh 1

```
1 let name = 'Saputro'
2 let umur = '27'
3
4 let getData = () => {
5   return `member ${name} umurnya ${umur}`
6 }
7
8 let member = {
9   name, umur, getData
10 }
11
12 console.log(member.getData());
13
```



2. Contoh 2

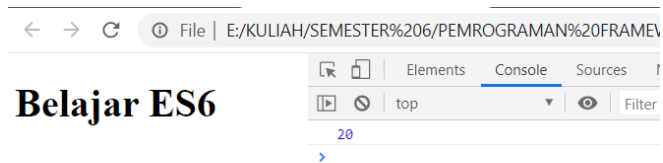
```
1 let member = {
2   name: 'Saputro',
3   umur: 20
4 }
5
6 let {umur, name} = member
7
8 console.log(name);
9
10
```



3. Pada contoh 3 kita dapat mengganti nama variabel , seperti contoh dibawah

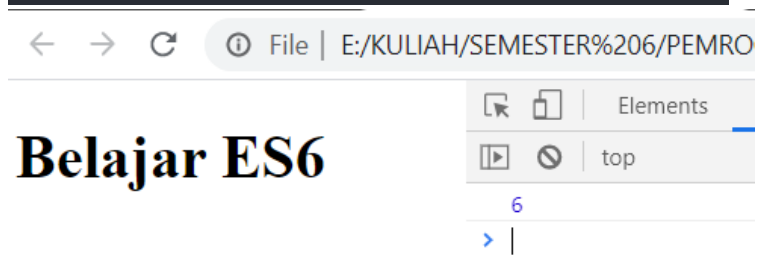
```
1 let member = {
2   name: 'Saputro',
3   umur: 20
4 }
5
6 let {umur:age, name} = member
7
8 console.log(age);
```

Disini kita mendefined umur dengan age setelah kita panggil age maka akan keluar sesuai nilai dari variabel umur yang telah kita deklarasikan diatas



4. Destructuring array

```
1 let umur = [1,2,3]
2 let [a,b,c] = umur
3 console.log(a+b+c);
4
```

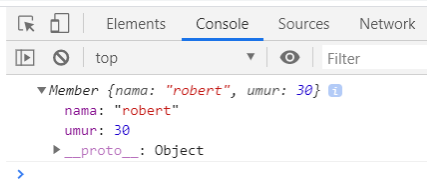


CLASS

1. Class dengan menambahkan konstraktor

```
1  class Member {
2      constructor(nama, umur) {
3          this.nama = nama
4          this.umur = umur
5      }
6  }
7
8  let member = new Member('robert', 30)
9  console.log(member);
10
```

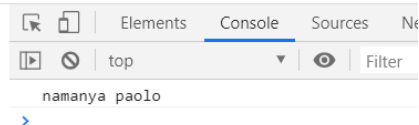
Belajar ES6



2. Class dengan fungsi

```
1  class Member {
2      // constructor(nama, umur) {
3      //     this.nama = nama
4      //     this.umur = umur
5      // }
6      getData(nama){
7          return `namanya ` + nama
8      }
9  }
10
11 let member = new Member()
12
13 console.log(member.getData('paolo'))
```

Belajar ES6



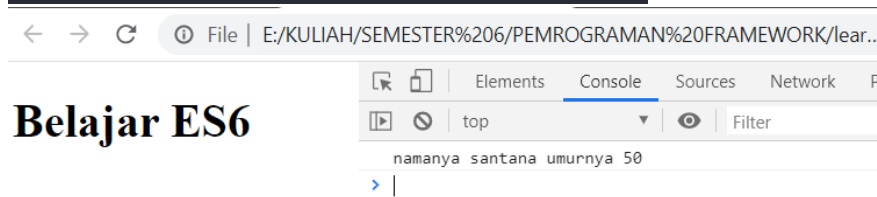
3. Kolaborasi constructor dan fungsi

```
1  class Member {
2      constructor(nama, umur) {
3          this.nama = nama
4          this.umur = umur
5      }
6      getData(nama){
7          return `namanya ${this.nama} umurnya ${this.umur}`
8      }
9  }
```

```

13 }
14
15 let member = new Member('santana', 50)
16
17 console.log(member.getData())

```

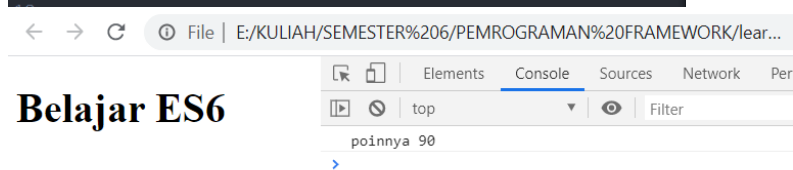


4. Menggunakan fungsi static

```

1 class Member {
2   constructor(nama, umur) {
3     this.nama = nama
4     this.umur = umur
5   }
6   getData(nama){
7     return `namanya ${this.nama} umurnya ${this.umur}`
8   }
9
10  static getPoint() {
11    return 'poinnya 90'
12  }
13 }
14
15 let member = new Member('jordi', 20)
16
17 console.log(Member.getPoint())

```



WEBPACK

1. Instalasi webpack

```
E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6>npm i -g webpack-cli
C:\Users\roisd\AppData\Roaming\npm\webpack-cli -> C:\Users\roisd\AppData\Roaming\npm\node_modules\webpack-cli\bin
\cli.jsnpm WARN webpack-cli@3.3.11 requires a peer of webpack@4.x.x but none is installed. You must install peer
dependencies yourself.
```

2. Konfigurasi file webpack.config.js

```
1  module.exports = {
2    entry: './src/index.js',
3    output: {
4      filename: 'dist/index.js'
5    }
6  }
```

3. Lalu jalankan webpack di terminal

```
PS E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6> webpack
Hash: 728d4b81b0dd58e20f07
Version: webpack 4.41.6
Time: 250ms
Built at: 02/18/2020 18:40:33
dist/index.js 944 bytes      0 [emitted] main
Entrypoint main = dist/index.js
[0] ./src/index.js 26 bytes {0} [built]
```

4. Jalankan program berikut

```
src > js index.js > ...
1  getNama = (nama) => {
2    console.log(nama);
3
4  }
5
6  getNama("Roisd")
7
```

← → ↻ ⓘ 127.0.0.1:8080

Belajar ES6

Elements Console Sources Network Pe

top Filter

Roisd

>

5. Setelah itu meyabungkan babel dengan webpack menggunakan babel-loader

```
PS E:\KULIAH\SEMESTER 6\PEMROGRAMAN FRAMEWORK\learn-es6> npm install b
abel-loader --save-dev
npm WARN learn-es6 No description
npm WARN learn-es6 No repository field.
npm WARN learn-es6 No license field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_
modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
```


FUNGSI EXPORT DAN IMPORT

1. Membuat folder app didalam folder src, lalu membuat file member.js

```
src > app > js member.js > [?] user
1 let user = {
2   nama: 'Ronal'
3 }
4
5 export {user}
```

2. Lalu import ke index.js

```
1 import {user} from './app/member'
2
3 console.log(user);
4
```

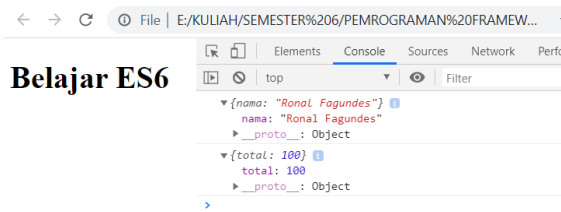
3. Buka browser lalu console



4. Menambahkan object baru

```
1 let user = {
2   nama: 'Ronal Fagundes'
3 }
4
5 let forum = {
6   total: 100
7 }
8
9 export {user, forum}
```

```
1 import {user, forum} from './app/member'
2
3 console.log(user);
4 console.log(forum);
```



Belajar ES6

5. Melakukan export import lebih dari 1 file
Dalam folder app buat file member.js

```
1 let user = {  
2   nama: 'Ronai Fagundes'  
3 }  
4  
5  
6  
7 export {user}
```

Lalu buat juga forum.js

```
1 export let forum = {  
2   total: 100  
3 }
```

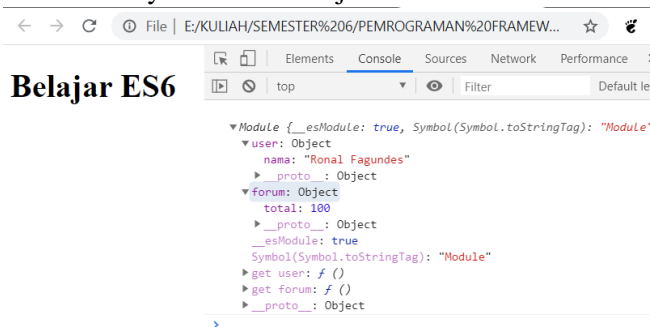
Lalu untuk memudahkan import pada index utama buat juga index.js di app

```
1 export * from './member'  
2 export * from './forum'
```

Setelah itu pada index.js di src hanya lakukan import di app saja

```
1 import * as app from './app/index'  
2  
3 console.log(app);  
4
```

Maka hasilnya akan sama saja



Belajar ES6

EXPORT DEFAULT

Melakukan export default

```
1  export default {  
2    total : 100  
3  }
```

Pada index pada folder src kita bisa import forum atau bisa dengan nama yang lain

```
1  import forum from './app/forum'  
2  
3  console.log(forum);  
4
```

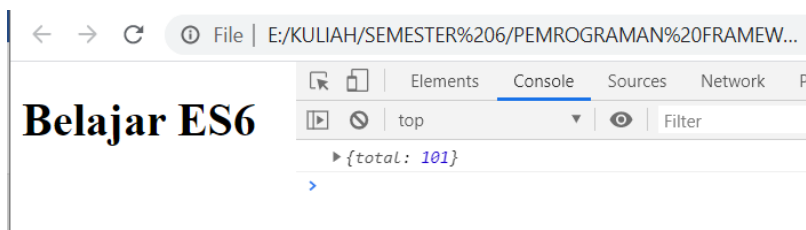
Saya ganti nilainya pada forum default

```
1  export default {  
2    total : 101  
3  }
```

Lalu saya ubah nama import pada index.js

```
1  import fo from './app/forum'  
2  
3  console.log(fo);  
4
```

Hasil akan tetap keluar

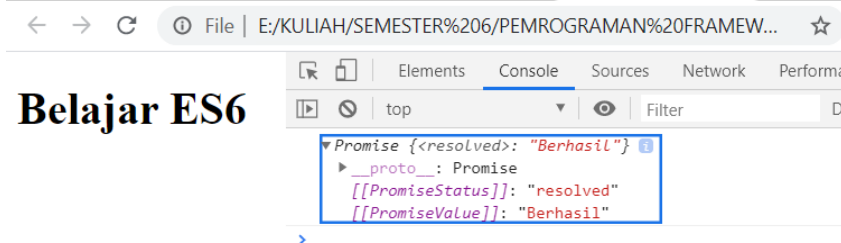


KONSEP DASAR PROMISE

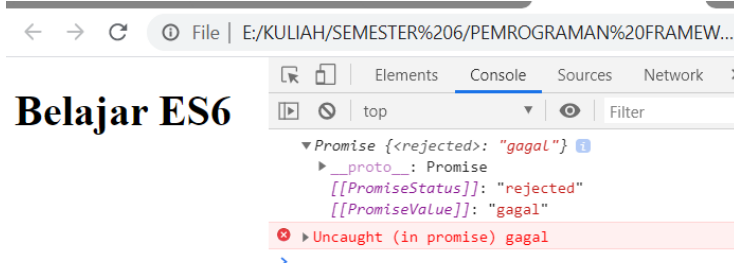
1. Melakukan promise, promise merupakan objek dimana jika berhasil akan mencetak yang berhasil dan gagal akan melakukan operasi yang ada dalam kondisi tersebut

```
1 let getMember = new Promise((resolve, reject)=>{
2   if (true) {
3     resolve('Berhasil')
4   }
5
6   reject('gagal')
7 })
8
9 console.log(getMember);
10
```

2. Dalam promise terdapat 2 key utama yaitu key status dan key value ketika tidak ada kesalahan maka statusnya adalah resolve dan kita bisa mengisi value apapun



Jika gagal maka akan rejected dan terjadi error yang tidak tertangkap



Untuk menangkap berhasil atau gagal kita bisa menggunakan `then` dan `catch`

```
1 let getMember = new Promise((resolve, reject)=>{
2   if (false) {
3     resolve('Berhasil')
4   }
5
6   reject('gagal')
7 }).then((msg)=>{
8   console.log('ini dalam ' + msg);
9 })
10 .catch((msg)=>{
11   console.log('ini dalam catch ' + msg);
12 })
13
```

Sebelumnya terjadi error tetapi kita menangkap error menggunakan `catch` lalu mengeluarkan pesan ini dalam `catch`

← → ↻

File | E:/KULIAH/SEMESTER%206/PEMROGRAMAN%20FRAMEW...

Belajar ES6

Elements Console Sources Network Perf

top Filter

▼ Promise {<pending>} 1

▼ __proto__: Promise

▶ constructor: f Promise()

▶ then: f then()

▶ catch: f catch()

▶ finally: f finally()

Symbol(Symbol.toStringTag): "Promise"

▶ __proto__: Object

[[PromiseStatus]]: "resolved"

[[PromiseValue]]: undefined

ini dalam catch gagal

> |

Tetapi jika berhasil maka akan masuk ke then

```
1 let getMember = new Promise((resolve, reject)=>{
2   if (true) {
3     resolve('Berhasil')
4   }
5
6   reject('gagal')
7 }).then((msg)=>{
8   console.log('ini dalam '+msg);
9
10  }).catch((msg)=>{
11    console.log('ini dalam catch ' + msg);
12  })
13 })
```

← → ↻

File | E:/KULIAH/SEMESTER%206/PEMROGRAMAN%20FR

Belajar ES6

Elements Console Sources Net

top Filter

ini dalam Berhasil

>