

MATA KULIAH : PEMROGRAMAN MOBILE
SESI PERTEMUAN : II (DUA)
MATERI : DASAR JAVASCRIPT
DOSEN : ALUN SUJJADA, S.KOM., M.T

A. PERKEMBANGAN JAVASCRIPT

Bahasa pemrograman Javascript berkembang sangat pesat pada beberapa tahun belakangan ini. Hal tersebut ditandai dengan munculnya bermacam-macam *library* dan *framework* berbasis Javascript. Saat ini Javascript tidak hanya sebagai bahasa pemrograman yang berjalan di sisi *client (browser)*, akan tetapi saat ini Javascript dapat berjalan di sisi *server* dengan munculnya sebuah Javascript *run time* yang dibangun oleh Google melalui Chrome V8 Engine yaitu Node JS. Javascript juga dapat digunakan untuk membangun aplikasi *desktop* melalui *framework* Electron. Bahkan saat ini Javascript sangat *powerfull* digunakan untuk pengembangan aplikasi *mobile* dengan *framework* Apache cordova, Ionic dan React.

Saat pertama kali diciptakan, Javascript hanya digunakan sebagai penunjang pengembangan aplikasi berbasis web, dimana fungsinya adalah untuk melakukan validasi *form input* dan *scripting* animasi saja. Adapun alasannya karena kurangnya kompatibilitas dengan beberapa browser. Setelah itu barulah muncul JQuery sebagai *library* berbasis Javascript yang menguasai perkembangan teknologi web, dimana JQuery memberikan kemudahan dalam proses manipulasi DOM (Documen Object Model) sehingga dapat menghasilkan aplikasi web yang sangat interaktif.

Kondisi ini kemudian berubah secara cepat setelah keluarnya Javascript ES5 pada sekitar tahun 2015 yang ditandai dengan munculnya teknologi NodeJS. Perubahan tersebut memicu berkembangnya pustaka-pustaka Javascript modern berbasis ES5 dan NodeJS.

Saat ini telah banyak berkembang pustaka Javascript modern seperti React, VueJS & Angular yang perlahan-lahan menggantikan kepopuleran JQuery.

B. VARIABEL DAN FUNGSI

Pada saat Javascript muncul, cara untuk mendeklarasikan variabel yaitu dengan menggunakan *keyword* **var** yang mempunyai 2(dua) *scope* yaitu *global* dan *fungsi*. Berikut ini adalah contoh kode programnya:

```
1  var firstName = 'James'
2  function printFullName(){
3      var lastName = 'Goosling'
4      return firstName + " " + lastName
5  }
6  console.log(printFullName()) // James Goosling
7  console.log(firstName) // James
8  console.log(lastName) // referenceError: lastName is not defined
```

Pada contoh kode program tersebut, maka variabel **firstName** bersifat global yang berarti dapat diakses didalam *function*, sehingga ketika *script* `console.log(printFullName())` dieksekusi akan tampil *output* “James Goosling” yang merupakan penggabungan antara `firstName` dan `lastName`. Akan tetapi ketika dieksekusi `console.log(lastName)` maka akan muncul pesan error, karena memang variabel `lastName` tidak dikenali diluar *function*.

Selain itu deklarasi menggunakan **var** rentan dengan adanya *error*, karena **var** dapat dilakukan deklarasi ulang seperti pada contoh kode program berikut ini:

```
1  var fruitName = 'Apple'
2  var fruitName = 'Banana'
3  console.log(fruitName) // Banana
```

Jika kode program tersebut dieksekusi maka tidak akan terjadi pesan *error* dan akan menghasilkan *output* “Banana”, hal tersebut sangatlah riskan bagi seorang programmer, jika terjadi secara tidak sengaja ketika program semakin kompleks. Masalah berikutnya dikenal dengan istilah **hoisting**, yaitu ketika sebuah variabel tidak dikenali, maka akan dicari dibaris berikutnya, jika ketemu maka nilai variabel tersebut akan dapat digunakan. Berikut ini adalah contoh kode programnya:

```
1  campus = "Universitas Nusa Putra"
2  console.log(campus)
3  var campus
```

Apakah hasil *output* dari kode program tersebut? Hasilnya adalah “Universitas Nusa Putra”, seharusnya jika berfikir secara logika maka akan terjadi *error* karena ketika perintah `console.log(campus)` dieksekusi tidak akan mengenali variabel **campus** (`ReferenceError: campus is not defined`), akan tetapi perlakuan Javascript akan mencari pada kode program selanjutnya apakah ada variabel **campus**, jika ada maka secara logika akan dipindahkan keatas.

```
1  var campus
2  campus = "Universitas Nusa Putra"
3  console.log(campus);
```

Untuk mengatasi permasalahan, maka sejak ES6 diperkenalkan cara baru dalam mendeklarasikan sebuah variabel. Yaitu dengan menggunakan keyword **let** dan **const**. Secara sederhana **let** digunakan untuk mendeklarasikan variabel yang bisa diubah-ubah nilainya saat *runtime*, sedangkan **const** untuk variabel yang bersifat konstanta atau tidak boleh diubah nilainya, berlaku untuk tipe data primitif dan tidak berlaku untuk tipe data *object*. Dengan menggunakan keyword **let** maka tidak akan terjadi *hoisting*, duplikasi variabel dan juga menyelesaikan masalah *scope*.

C. TEMPLATE LITERAL

Semenjak muncul Javascript ES6(ES 2015) banyak sekali fitur tambahan. Salah satu dari fitur tersebut adalah template literal, dimana kita bisa memasukkan ekspresi Javascript pada sebuah string atau teks. Selain itu template literal juga dapat digunakan untuk proses penggabungan string (concatenation). Aturan untuk cara penulisannya adalah menggunakan karakter *backtick* (```), bukan tanda petik tunggal. Jika untuk menuliskan ekspresi maka menggunakan karakter dollar (`$`). Berikut ini adalah contoh kode program:

```
1  let studentName = 'Santy'
2  let studentAddress = 'Medan'
3
4  //Sebelum ES6
5  console.log(studentName + " berasal dari " + studentAddress )
6
7  //Setelah ES6
8  console.log(`${studentName} berasal dari ${studentAddress}`)
```

```
1  let numOne = 100
2  let numTwo = 30
3  let result = `Hasil perkalian dari ${numOne} dan ${numTwo} adalah ${numOne * numTwo}`
4  console.log(result);
```

D. FUNCTION

Pada pemrograman Javascript terdapat fitur untuk membuat fungsi menggunakan *keyword* **function**, yaitu sebuah kode program yang diletakkan pada block scope dan dapat dipanggil ulang sesuai dengan nama fungsinya. Adapun sintaks untuk pembuatan fungsi adalah seperti berikut ini:

```
1  function functionName(parameter){  
2      //program code 1  
3      //program code 2  
4      //program code n  
5  }
```

Untuk contoh penggunaanya adalah sebagai berikut:

```
1  function getTotalNumber(params1, params2) {  
2      let result = params1 + params2  
3      return result  
4  }  
5  
6  let firstVal = 90  
7  let secondVal = 100  
8  let total = getTotalNumber(firstVal, secondVal)  
9  console.log(`Penjumlahan dari ${firstVal} dan ${secondVal} = ${total}`);
```

Namun setelah munculnya ES6, terdapat cara lain untuk menuliskan fungsi yaitu yang dikenal dengan istilah *arrow function*. Cara tersebut banyak digunakan nanti ketika mulai masuk ke pemrograman react. Sintaks dari *arrow function* adalah seperti berikut ini:

```
11  const nameFunction = (parameter) => {  
12      //program code 1  
13      //program code 2  
14      //program code n  
15  }
```

Keyword **const** bisa juga diganti dengan **let** sesuai dengan kebutuhan, kemudian diikuti dengan nama fungsi, parameter yang diperlukan dan **arrow function** (**=>**). Contoh implementasi penggunaanya adalah sebagai berikut:

```

1  const getMaxNumber = (params1,params2) => {
2      if (params1 > params2) {
3          return params1
4      }
5      return params2
6  }
7
8  console.log(getMaxNumber(90,100)) // 100
9  console.log(getMaxNumber(80,50)) // 80
10 console.log(getMaxNumber(40,40)) // 40

```

Contoh kode program tersebut bisa dipersingkat menggunakan operator ternary. Jika kode program hanya satu baris, maka *keyword* **return** dan blok *scope* kurang kurawal bisa dihilangkan menjadi seperti berikut ini:

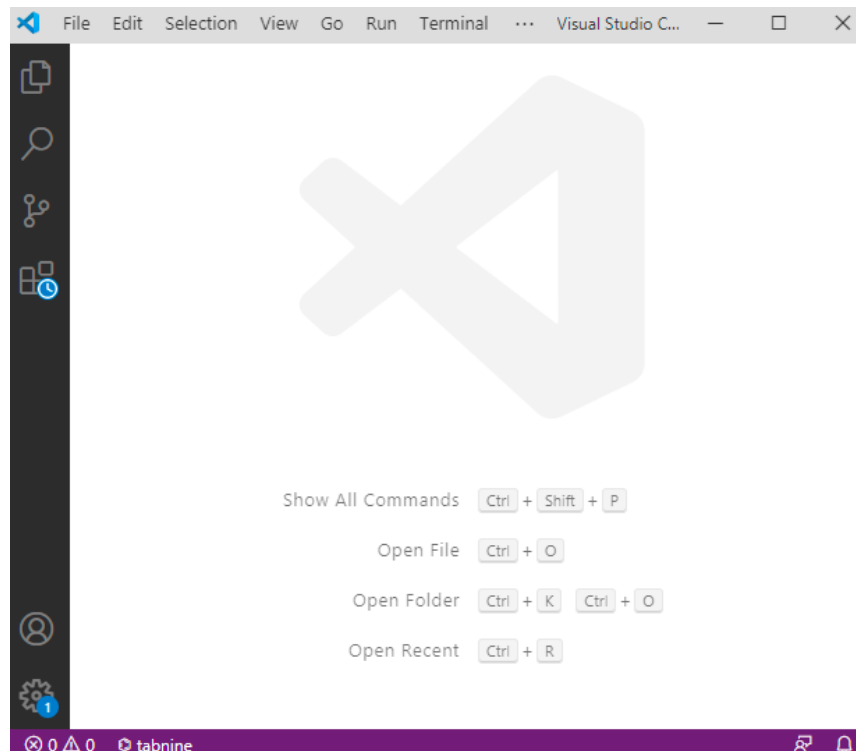
```

1  const getMaxNumber = (params1,params2) => params1 > params2?params1:params2
2
3  console.log(getMaxNumber(90,100)) // 100
4  console.log(getMaxNumber(80,50)) // 80
5  console.log(getMaxNumber(40,40)) // 40

```

E. PENGKODEAN PROGRAM

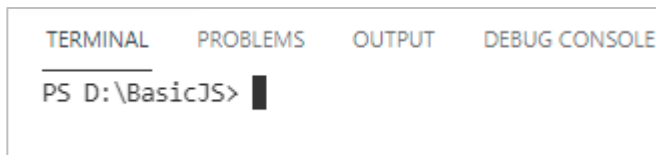
Adapun cara untuk pembuatan kode program Javascript adalah lakukan instalasi **Text Editor** seperti Visual Studio Code (disarankan), Notepad++, Sublime, Atom, VIM danlain-lain. Pada contoh materi *lecture note* ini menggunakan Visual Studio Code seperti berikut ini:



Kemudian buatlah *file* baru dan simpan dengan nama file berekstensi .js(dotjs), misal index.js. Setelah tulislah beberapa kode seperti contoh berikut ini:

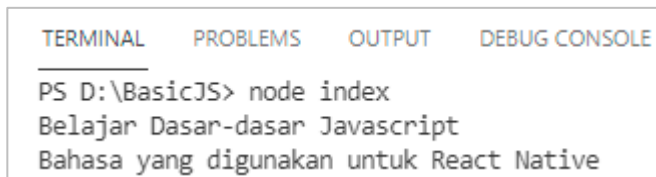
```
1 console.log(`Belajar Dasar-dasar Javascript`);  
2 console.log(`Bahasa yang digunakan untuk React Native`)
```

Lakukan eksekusi program yaitu dengan menampilkan *terminal window* terlebih dahulu dengan menekan tombol Ctrl + Backtick (``) seperti tampilan berikut ini:



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\BasicJS> █

Langkah berikutnya adalah menggunakan perintah node index.js, maka akan tampil hasil *output* seperti berikut ini:



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\BasicJS> node index
Belajar Dasar-dasar Javascript
Bahasa yang digunakan untuk React Native