



MODUL WORKSHOP JAVASCRIPT PROGRAMMING LANGUAGE FUNDAMENTALS



Laboratorium Pengembangan
Aplikasi dan Pemrograman Komputer
LEMBAGA PENGEMBANGAN KOMPUTERISASI
UNIVERSITAS GUNADARMA



KATA PENGANTAR

Modul ini merupakan panduan bagi peserta *workshop Javascript Programming Language Fundamentals* yang diselenggarakan oleh Laboratorium Pengembangan Aplikasi dan Pemrograman LePKom Universitas Gunadarma. Modul ini dibagi menjadi empat bagian utama yaitu : Pengenalan Javascript, struktur pemrograman, statement pengambilan keputusan, statement pengulangan serta pengenalan OOP (*Object Oriented Programming*) pada *Javascript*. Setiap bagian dari modul ini menyajikan materi yang dikemas secara ringkas dan praktis. Setiap materi disertai latihan program yang mewakili tiap bagian.

Penyusun berharap modul ini dapat digunakan sebaik-baiknya oleh seluruh peserta *workshop* dan dapat membantu peserta dalam mempelajari dan memahami *Java* selama mengikuti *workshop*.



Daftar Isi

Kata Pengantar.....	i
Daftar Isi.....	ii
Bab 1 Pendahuluan	1
1.1. Pengertian JavaScript.....	1
1.2. Sejarah JavaScript.....	1
1.3. Kelebihan JavaScript.....	2
Bab 2 Pemrograman JavaScript.....	3
2.1. Membuat Program Pertama, Hello World!.....	3
2.2. Struktur Program JavaScript.....	5
2.3. Variabel dan Tipe Data.....	6
2.3.1 Penamaan Variabel.....	7
2.3.2 Scope Variabel.....	8
2.3.3 Tipe Variabel.....	10
2.3.4 Escape Characters dan Unicodes.....	11
2.4 Operator.....	12
2.5 Array.....	15
2.6 Statement Pengambilan Keputusan.....	21
2.7 Statement Perulangan.....	23
2.8 Message Box.....	25
Bab 3 OOP Pada JavaScript.....	30
3.1. Memahami Object.....	30
3.2 Memahami Property dan Method.....	33
3.3 Membuat dan Memanggil Object	35
3.4 Memahami This.....	42
Lampiran	

1

Pendahuluan

Objektif :

- Mengetahui Bahasa Pemrograman JavaScript
 - Mengetahui Sejarah Perkembangan JavaScript
-

1.1 Pengertian *JavaScript*

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat interaksi atau menambah fitur web dinamis kedalam sebuah web. Sebuah halaman web tidak wajib menggunakan *JavaScript* namun *JavaScript* hampir ada di setiap halaman web modern.

Banyak hal yang dapat dilakukan dengan *JavaScript* seperti memulai fitur sederhana menentukan *layout*, membuat respon ketika memilih menu *button*, carousels, dan *gallery* gambar. Tidak hanya itu, dengan *JavaScript* juga bisa membuat *game*, animasi 2D dan 3D, aplikasi yang berhubungan dengan *database* dan masih banyak lagi.

JavaScript merupakan bagian dari 3 teknologi penting yang harus dikuasai programmer web dan salah satu teknologi inti *World Wide Web*, yakni HTML untuk konten, CSS untuk tampilan, dan *JavaScript* untuk interaksi. Kini *JavaScript* dapat disisipkan ke dalam perangkat lunak lain seperti *server-side* dalam server web dan basis data, dalam program non web seperti *software* pengolah kata dan pembaca PDF, dan sebagai *runtime environment* yang memungkinkan penggunaan *JavaScript* untuk membuat aplikasi *desktop* maupun *mobile*.

1.2 Sejarah *JavaScript*

JavaScript pertama kali dikembangkan oleh Brendan Eich dari Netscape di bawah nama Mocha, yang nantinya namanya diganti menjadi *LiveScript*, dan akhirnya menjadi *JavaScript*. Fitur *JavaScript* sebelumnya telah mendukung *Java* untuk lebih bisa dimanfaatkan para pemrogram *non-Java*.

Namun, seiring berjalannya waktu pengembangan bahasa pemrograman bernama *LiveScript* mampu untuk mengakomodasi beberapa proses pembuatan website. Bahasa pemrograman inilah yang akhirnya berkembang dan diberi nama *JavaScript*, walaupun tidak ada hubungan bahasa antara *Java* dengan *JavaScript*.

Terinspirasi dari kesuksesan *JavaScript*, Microsoft mengadopsi teknologi serupa. Microsoft membuat '*JavaScript*' versi mereka sendiri bernama Jscript lalu ditanam pada Internet Explorer 3.0 dan menjadi perang browser karena Jscript milik Microsoft berbeda dengan *JavaScript* racikan Netscape. Akhirnya pada tahun 1996, Netscape mengirimkan standarisasi ECMA-262 ke Ecma International sehingga lahirlah standarisasi kode *JavaScript* bernama ECMAScript atau ES yang saat ini mencapai versi 8.

JavaScript bisa digunakan untuk banyak tujuan, misalnya untuk membuat efek *rollover* baik di gambar maupun teks, dan yang penting juga adalah untuk membuat AJAX

(*Asynchronous JavaScript And XML*). *JavaScript* adalah bahasa yang digunakan untuk AJAX.

1.3 Kelebihan *JavaScript*

Selanjutnya kita akan membahas tentang kelebihan dari *JavaScript*. Kelebihan yang dimaksud antara lain, lebih mudah dipelajari jika dibandingkan dengan bahasa pemrograman yang lainnya. Penanganan dan pencarian kesalahan ataupun error juga lebih mudah. Anda juga tidak membutuhkan compiler sebab *web browser* bisa menginterpretasikannya dengan HTML.

JavaScript juga dapat ditugaskan ke *even* tertentu ataupun halaman web tertentu, misalnya yaitu klik ataupun *mouseover*. *File .js javascript* juga dapat digunakan di berbagai web browser dan platform sehingga sangat merakyat serta penggunaannya hampir didukung sepenuhnya.

Selain itu, *JavaScript* bisa juga digunakan untuk memvalidasi *input* tertentu sehingga keinginan untuk mengecek data secara manual menjadi berkurang. Mengenai memori yang digunakan, *javascript* tergolong kecil, ringan, dan lebih cepat dibanding bahasa pemrograman yang lainnya.

Apa yang dapat *JavaScript* lakukan?

- ***JavaScript* memberikan alat pemrograman desainer HTML** - penulis HTML biasanya bukan programmer, tapi *JavaScript* adalah bahasa *scripting* dengan sintaks yang sangat sederhana! Hampir setiap orang dapat menempatkan "potongan" kecil dari kode ke halaman HTML mereka
- ***JavaScript* dapat bereaksi terhadap peristiwa** - Sebuah *JavaScript* dapat diatur untuk mengeksekusi ketika sesuatu terjadi, seperti ketika sebuah halaman telah selesai loading atau ketika pengguna mengklik elemen HTML
- ***JavaScript* dapat membaca dan menulis elemen HTML** - *JavaScript* dapat membaca dan mengubah isi dari elemen HTML
- ***JavaScript* dapat digunakan untuk memvalidasi data** - *JavaScript* dapat digunakan untuk memvalidasi data formulir sebelum diserahkan ke server. Ini menghemat server dari pemrosesan tambahan
- ***JavaScript* dapat digunakan untuk mendeteksi browser pengunjung** - Sebuah *JavaScript* dapat digunakan untuk mendeteksi browser pengunjung, dan - tergantung pada browser - load halaman lain yang khusus dirancang untuk browser yang
- ***JavaScript* dapat digunakan untuk membuat cookies** - *JavaScript* dapat digunakan untuk menyimpan dan mengambil informasi di komputer pengunjung

Setidaknya terdapat *tiga hal* yang menarik tentang *JavaScript*:

- Dapat diintegrasikan penuh dengan HTML/CSS.
- Hal simple dapat dilakukan dengan cara yang simple.
- Didukung oleh semua jenis Browser dan sudah diaktifkan secara default.

JavaScript adalah satu-satunya teknologi browser yang dapat menggabungkan ketiga hal diatas dan itulah yang dapat membuat *JavaScript* menjadi semakin unik.

2

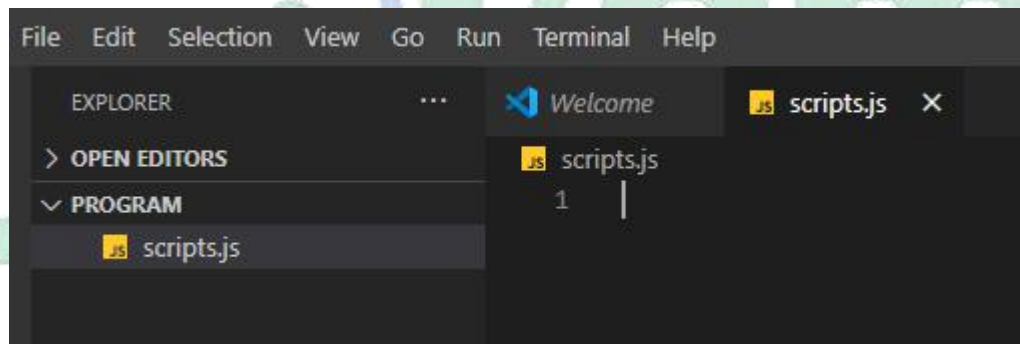
Pemrograman Javascript

Objektif :

- Mengetahui struktur pemrograman pada Javascripts
 - Mengetahui statement pengambilan keputusan dalam Javascript
 - Mengetahui statement pengulangan dalam Javascript
 - Mengetahui penggunaan *message box* pada Javascript
-

2.1 Membuat Program Pertama, *Hello World!*

Program pertama yang akan dibahas adalah program *console* yang akan menampilkan tulisan “*Hello World*”. Langkah pertama yang dilakukan yaitu membuka *Editor* misalkan *VSCode/Notepad++/Sublime*. Kemudian buat *file* dengan nama bebas berekstensi *.js*. Pada program ini akan dibuat file dengan nama *scripts.js*



Gambar 1. Latihan Script.js

Kemudian, ketikkan perintah berikut ini pada *file* yang telah dibuat sebelumnya

```
alert("Hello World!");
```

Lalu simpan *scripts.js* tersebut pada direktori yang anda kehendaki. Langkah selanjutnya adalah membuat *interface* tampilan *javascript* tersebut. Selanjutnya silahkan buat *file* *html* untuk menampilkan pesan tulisan. *File* *html* tersebut diberi nama bebas berekstensi *.html* pada direktori yang sama dengan *.js* yang sudah dibuat. Pada contoh kasus ini, silahkan buat *file* dengan nama *index.html*. Kemudian ketik-kkan program di bawah ini:

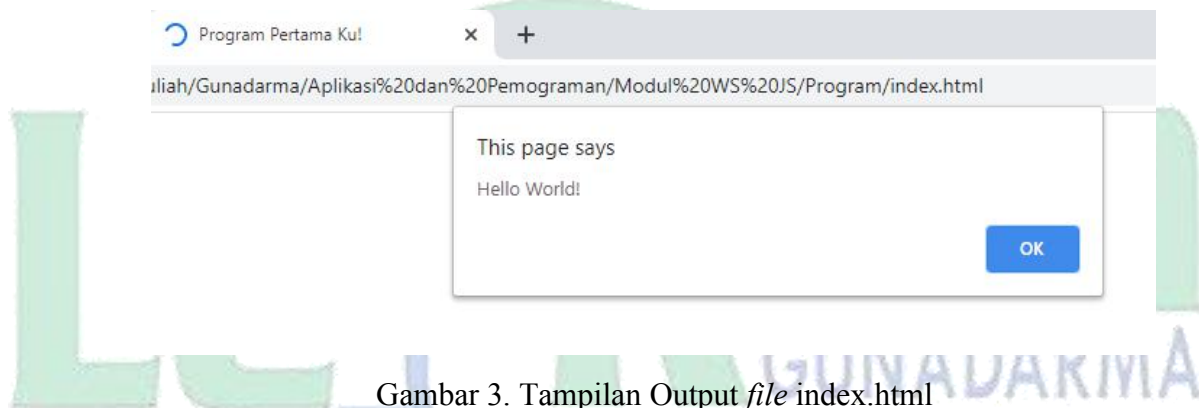
```

index.html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <title>Program Pertama Ku!</title>
5  </head>
6  <body>
7  |   <script type="text/javascript" src="scripts.js"></script>
8  </body>
9  </html>

```

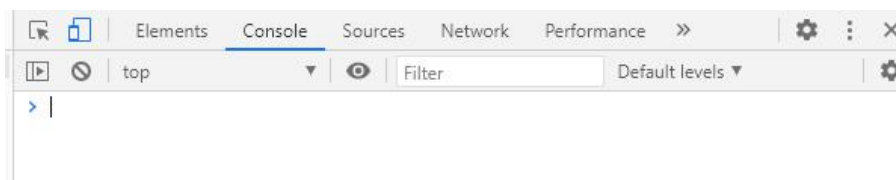
Gambar 2. Tampilan index.html

Simpan *file html* tersebut lalu jalankan program. Langkah yang diperlukan untuk dilakukan untuk menjalankan program adalah pertama buka *browser* yang akan digunakan (*Chrome, Firefox, Edge, dan lainnya*). Kemudian buka direktori penyimpanan *scripts.js* dan *index.html* yang sudah dibuat. Lalu *drag and drop index.html* menuju *browser*.



Gambar 3. Tampilan Output *file index.html*

Ada cara lain untuk menuliskan program yang mirip seperti *file* di atas yaitu melalui *console* pada *browser*. Caranya adalah sebagai berikut, pertama silahkan buka *browser* yang dipilih seperti *Chrome*. Lalu, tekan tombol *F12* pada *keyboard* dan akan muncul menu *developer tools* pada *browser*. Selanjutnya adalah silahkan pilih *tab* bernama *console* seperti yang ditunjukkan pada Gambar 4.



Gambar 4. Tampilan *Console*

Kemudian tuliskan program yang sama persis dengan apa yang telah diketik sebelumnya pada *scripts.js*. Maka tampilan yang keluar akan sama persis dengan apa yang telah dilakukan sebelumnya.

2.2 Struktur Program Javascript

Program *javascript* terdiri dari susunan *statement*. *Statement* adalah sarana untuk melaksanakan semua tugas program. Perintah yang diberikan untuk program ditulis dalam *statement*. Terdapat tiga jenis *statement* dalam javascript yaitu *simple statement* (statement sederhana), *compound statement* (statement blok/blok) dan komentar.

Simple statement dapat berupa perintah untuk melaksanakan tugas, memasukkan nilai dalam variabel atau memanggil *function*. Contoh dari *simple statement* dapat dilihat pada Gambar 5.

```
scripts.js > ...  
1 console.log("Hello World!");  
2 var j = 12;  
3 if (j>=12) validate();  
4
```

Gambar 5. Contoh *Simple Statement*

Compound statement biasa disebut blok program atau blok. *Compound statement* di pemrograman *Pascal* biasa dimulai dengan *statement begin* dan diakhiri dengan *end* sedangkan dalam *javascript* sama seperti penggunaan blok dalam C/C++, yaitu dimulai dengan { dan diakhiri dengan }. Contoh dari *compound statement* dapat dilihat pada Gambar 6.

```
if (x==true) {  
    console.log("Olivia has reached the save zone!");  
    alert("Mission Success");  
} else {  
    console.log("Olivia has been arrested!");  
    alert("Mission Failed");  
}
```

Gambar 6. Kodingan *Compound Statement*

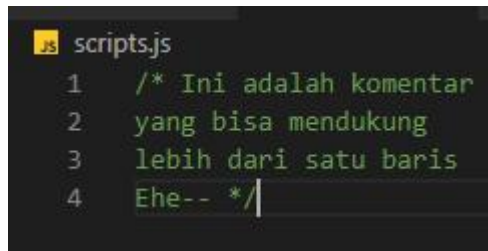
Javascript mempunyai tiga macam cara untuk memberikan komentar pada kode program. Komentar dipakai untuk menjelaskan mengenai program atau algoritma dan umumnya digunakan sebagai alat bantu dokumentasi *programmer*.

Cara pertama adalah komentar model bahasa C++ yaitu diawali dengan // dan diakhiri dengan akhir baris tersebut, contoh :

```
scripts.js  
1 // Ini adalah komentar satu baris
```

Gambar 7. Contoh Komentar Gaya Bahasa C++

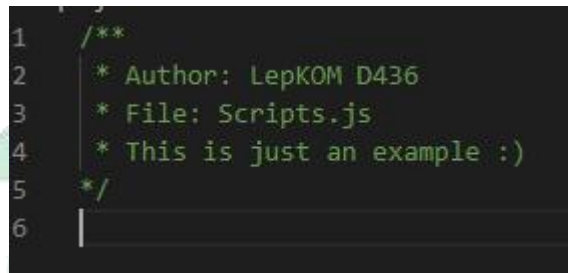
Cara kedua adalah gaya bahasa C yaitu komentar dimulai dengan `/*` dan diakhiri dengan `*/`, contoh:



```
1 /* Ini adalah komentar
2 yang bisa mendukung
3 lebih dari satu baris
4 Ehe-- */
```

Gambar 8. Contoh Komentar Gaya Bahasa C

Cara ketiga adalah komentar yang dimulai dengan `/**` dan di akhiri dengan `*/`. Umumnya komentar ini digunakan pada awal *file* sebagai pendeklarasian, pernyataan, atau penjelasan suatu blok program dalam bentuk tulisan sebagai bahan dokumentasi.



```
1 /**
2  * Author: LepKOM D436
3  * File: Scripts.js
4  * This is just an example :)
5  */
6
```

Gambar 9. Contoh Komentar Dalam Javascript

Tidak peduli berapa banyak `*` yang ada, selama komentar diawali dengan `/**` dan diakhiri dengan `*/` maka isi dari blok tersebut didefinisikan sebagai komentar.

2.3 Variabel dan Tipe Data

Variabel adalah sebuah lokasi (*address*) dalam memori komputer yang diwakili dengan nama yang diberikan untuk menyimpan nilai tertentu. Nilai dalam variabel ini dapat digunakan berulang - ulang dalam program maupun diubah nilainya. Nilai yang disimpan dalam variabel bermacam - macam. Tipe-tipe nilai ini yang disebut dengan tipe data.

Secara umum ada beberapa jenis tipe data seperti *integer*, *float* atau *real*, *char*, *string*, *Boolean*, dan sebagainya. Lalu setiap jenis tipe data tersebut mempunyai ciri khas dan fungsinya masing – masing dalam program. Contoh tipe data pada bahasa pemrograman *Java*.

Namun, perlu diketahui bahwa pada *javascript*, paradigma tersebut tidak berlaku sama sekali. Hal tersebut dikarenakan ciri khas tipe data pada *javascript* mempunyai sifat yang dinamis. Sifat dinamis yang dimaksud adalah suatu konsep di mana sebuah wadah (dalam kasus ini adalah variabel) dapat mengikuti isi (dalam hal ini merupakan

nilai atau *value*) yang akan dimasukkan ke dalamnya walau berbentuk sebagaimana rupanya (dalam kasus ini adalah tipe datanya). Contoh penggunaan sifat dinamis pada pemrograman *javascript* dapat dilihat pada Gambar 10.

```
var x; //tipe data undefined (tidak didefinisikan)
x = 10; //sekarang x menjadi integer
x = 3.14; // kemudian x menjadi float
x = "Dea Anchor"; // Ia pun menjadi String
```

Gambar 10. Penggunaan Variabel Pada Javascript

Sehingga seorang *programmer* dapat menggunakan satu jenis *variabel* untuk melakukan berbagai operasi dengan tipe data yang diinginkannya.

2.3.1 Penamaan Variabel

Variabel adalah sebuah tempat/wadah yang memiliki nama, yang digunakan untuk menyimpan nilai. Ada beberapa cara tahap yang perlu dilakukan untuk membuat variabel pada *javascript*, yaitu *deklarasi*, *inisiasi*, dan penugasan (*assignment*). Tahap deklarasi (*declaration*) merupakan fase di mana variabel harus didaftarkan atau dibuat ke dalam lingkup yang sesuai. Kemudian, tahap inisiasi (*initiation*) merupakan fase di mana variabel harus menyediakan memori ke dalam lingkup yang sesuai. Terakhir, tahap penugasan (*assignment*) adalah fase di mana anda menentukan nilai spesifik yang diberikan untuk variabel yang anda telah buat sebelumnya.

```
scripts.js
1  var x; //deklarasi & inisiasi
2  x = 20; //assignment
```

Gambar 11. Contoh Penggunaan Assignment Pada Javascript

Deklarasi variabel ada 3, yaitu *var*, *let*, dan *const* yang masing masing mempunyai perilaku yang berbeda. Deklarasi *var* adalah deklarasi umum ketika kita membuat suatu variabel. Contoh penamaan variabel dapat dilihat pada Gambar 12.

```
scripts.js > ...
1  var no_spasi_please; //ditolak
2  var 1sayanaroangkadiawal; //ditolak
3  var variabelDiterima1; // diterima
4  var ini_juga_diterima; //diterima
5  var dollar$diterima //diterima
6  var $_$diterima //ditemari
7  var suatuVariabelDiterima; //diterima
```

Gambar 12. Contoh Penamaan Variabel Dalam Javascript

Deklarasi *let* merupakan deklarasi yang umum digunakan jika kita ingin meng-*assign* suatu variabel ketika berada di suatu block program namun variabel tersebut hanya ingin digunakan pada block program tersebut saja. Contoh dari pendeklarasian *let* dapat dilihat pada Gambar 13.

```
scripts.js > ...
1  var x;
2  {
3      x = 10
4      console.log(x) //x = 10
5  }
6  console.log(x) //x tetap 10
7  {
8      let x = 2;
9      console.log(x); //x adalah 2
10 }
11 console.log //namun x tetaplah 10
```

Gambar 13. Contoh Pendeklarasian Let

Deklarasi *const* merupakan deklarasi di mana nilai variabel tersebut akan konstan yang berarti nilai tersebut tidak akan berubah sama sekali selama program berjalan. Jika program tersebut dipaksakan maka program akan *error*. Contoh *error* pada *console javascript* dapat dilihat pada Gambar 14.

```
> const x = 3.14;
< undefined
> console.log(x);
3.14
< undefined
> x = 1;
✖ ▶ Uncaught TypeError: Assignment to constant variable.
   at <anonymous>:1:3
> |
```

Gambar 14. Contoh Error Pada Javascript

2.3.2 Scope Variabel

Scope (cakupan) variabel adalah batasan daerah dalam program tempat variabel dapat diakses dengan memanggil nama dari variabel secara langsung. *Scope* juga berguna sebagai batasan bagi program untuk menyediakan ruang memori untuk

variabel atau untuk menghilangkannya.

Secara umum, terdapat dua jenis *scope* variabel yaitu *global scope* dan *function scope*. *Global scope* adalah variabel yang dapat digunakan di mana saja pada suatu program javascript. Contoh penggunaan dari *Global Scope* dapat dilihat pada Gambar 15.



```
Scripts.js > functionScope
1  var heroineName = "Kiana Kaslana";
2
3  // heroneName dapat langsung digunakan di block ini
4
5  function functionScope() {
6      console.log(heroineName);
7      //berikut pula di sini
8  }
```

Gambar 15. Contoh penggunaan *Global Scope*.

Variabel *heroineName* pada Gambar 15 adalah *global scope* sehingga variabel tersebut bisa diakses baik pada bagian manapun pada program *javascript*. *Function* yang ada pada program pun juga dapat menggunakan variabel tersebut.

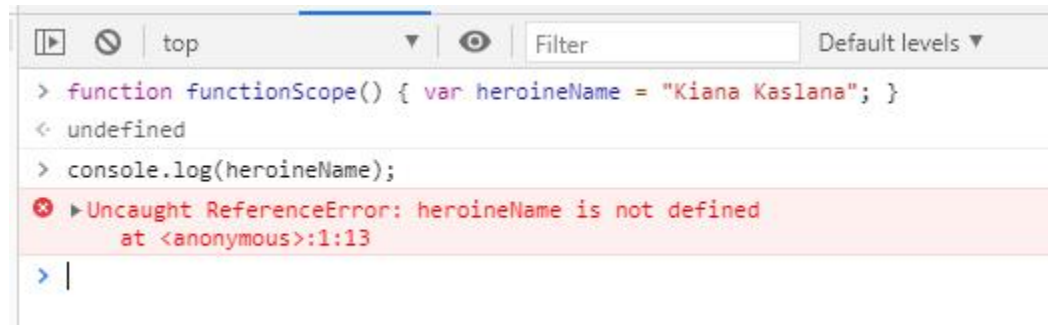
Berbeda dengan *functional scope* di mana fungsi tersebut merupakan variabel yang hanya dapat digunakan pada *function* yang bersangkutan. Nilai dari variabel dalam *function* tidak dapat digunakan keluar walau nama variabelnya dibuat sama sekalipun, *Scope* ini sering disebut *local*. Contoh dari penggunaan *local scope* dapat dilihat pada Gambar 16.



```
Scripts.js > ...
1
2  function functionScope() {
3      var heroineName = "Kiana Kaslana";
4  }
5
6  console.log(heroineName);
```

Gambar 16. Contoh Penggunaan *Local Scope*

Jika kita menggunakan program pada Gambar 16 maka akan terjadi *error* seperti pada Gambar 17.



Gambar 17. Error Dalam Penggunaan Javascript

2.3.3 Tipe Variabel

Pada pembahasan sebelum anda sudah mengetahui bahwa tipe variabel pada *javascript* bersifat *dynamic*. Namun, secara garis besar tipe data tersebut dapat dibagi menjadi 2 golongan. Golongan pertama disebut *primitive data* yang berisikan tipe data *string*, *number*, *Boolean*, dan *undefined* serta golongan kedua yang bisa disebut *complex data* yang berisikan tipe data *object* dan *function*. Contoh pembagian golongan dapat dilihat pada Gambar 18.

```

1  /**
2   * Primitive Data
3   */
4  var x; //undefined
5  var x = 10 //number
6  var name = "Klee"; // String
7  var visible = false; //boolean
8
9  /**
10   * Complex Data
11   */
12  var f_name = {firstName: "Bianka", lastName: "Ataegina"}; // Object
13  var carName = ["Volvo", "BMW", "Aston Martin"]; // Array as Object
14  function myFunc(){} // Function
15

```

Gambar 18. Contoh Pembagian Golongan Tipe Data

Kodingan pada Gambar 18 dapat diperiksa valid tidaknya menggunakan perintah *typeof()* operator. Contoh implementasi *typeof()* dapat dilihat Pada Gambar 19.


```

> x
< ▶ (3) ["BMW", "Volvo", "Nissan"]
> typeof(x);
< "object"
> typeof function myFunc(){};
< "function"
> typeof(10);
< "number"
> typeof(false);
< "boolean"
> |

```

Gambar 19. Contoh implementasi *typeof()*

Pada gambar 19 dapat dilihat perbedaan tipe data yang bisa dicoba secara langsung menggunakan operator *typeof()*.

2.3.4 *Escape Characters* dan *Unicode*.

Pada salah satu tipe data yaitu *String*, ada beberapa hal khusus yang dapat dilakukan oleh tipe data tersebut. Hal khusus ini perlu diketahui untuk melakukan operasi *String*. Pertama adalah mengenai *escape characters*, yaitu simbol – simbol khusus yang mempunyai makna jika digunakan pada operasi *string*. Kumpulan simbol *escape string* dapat dilihat pada Gambar 20.

escape character	hasil
\0	karakter NULL
\'	'
\"	"
\\	\
\n	new line / baris baru
\t	tab
\b	backspace
\uXXXX	unicode

Gambar 20. Kumpulan Simbol *Escape String*

Kedua adalah mengenai *unicode*s. *Unicode*s merupakan notasi kode khusus yang ditujukan untuk menampilkan output berupa simbol – simbol unik seperti *copyright* dan lainnya. Kumpulan simbol *unicode string* dapat dilihat pada Gambar

21.

unicode	hasil
\u00A9	©
\u00AE	®
\u00B1	±
\u00B5	µ
\u2122	™
...	...

https://en.wikipedia.org/wiki/List_of_Unicode_characters

Gambar 21. Kumpulan *Unicode String* pada *Javascript*

2.4 Operator

Pada pemrograman *javascript*, operator memerlukan hal krusial yang wajib diketahui. Ada beberapa operator umum dalam *javascript* yaitu operator aritmatika, operator penugasan, operator perbandingan, operator logika, operator *string*, operator kondisional, dan operator *typeof*.

Operator aritmatika adalah notasi yang digunakan untuk melakukan operasi terhadap segala hal yang melibatkan perhitungan matematis dalam lajunya program *javascript*. Contoh simbol operator dapat dilihat pada Gambar 22.

Arithmetic Operators			
Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1
++	Increment	A = 10; A++	11
--	Decrement	A = 10; A--	9

Gambar 22. Simbol Operator Pada *Javascript*

Operator penugasan adalah operator yang digunakan untuk memberikan perantaraan pemberian value terhadap suatu variabel. Contoh dari operator penugasan dapat dilihat pada Gambar 23.

Assignment Operators

Operator	Example	Equivalent Expression
=	$m = 10$	$m = 10$
+=	$m += 10$	$m = m + 10$
-=	$m -= 10$	$m = m - 10$
*=	$m *= 10$	$m = m * 10$
/=	$m /= 10$	$m = m / 10$
%=	$m \% = 10$	$m = m \% 10$
<<=	$a <<= b$	$a = a << b$
>>=	$a >>= b$	$a = a >> b$
>>>=	$a >>>= b$	$a = a >>> b$
&=	$a \&= b$	$a = a \& b$
^=	$a \wedge= b$	$a = a \wedge b$
=	$a = b$	$a = a b$

www.geekyshows.com

Gambar 23. Contoh Operator Penugasan

Operator perbandingan adalah operator yang bertugas untuk melakukan komparasi antara dua *operand* (objek dari operasi). Adapun contoh dari operator perbandingan yang dapat dilihat pada Gambar 24.

operator perbandingan	keterangan
==	sama dengan
!=	tidak sama dengan
===	strict sama dengan
!==	strict tidak sama dengan
>	lebih besar dari
<	lebih kecil dari
>=	lebih besar sama dengan
<=	lebih kecil sama dengan


Gambar 24. Operator Perbandingan Pada *Javascript*

Operator logika adalah operator yang pada umumnya digunakan pada *statement if-condition*. Operator ini akan melakukan komparasi antara dua operand berbentuk logika (artinya melakukan pertimbangan logika). Adapun contoh penerapan dari operator logika yang dapat dilihat pada Gambar 25.

operator logika	keterangan
&&	AND
 	OR
!	NOT

Gambar 25. Operator Logika Pada *Javascript*.

Operator *string* merupakan operator yang lahir jika ada 2 *operand* bertipe *string* atau satu *string* - satu number dilakukan operasi layaknya operator aritmatika (umumnya menggunakan operasi penambagan (*addition*)). Hasil akhir operator ini kadang disebut *konkatinasi* (penggabungan kata). Contoh penerapan *string* pada *javascript* dapat dilihat pada Gambar 26.



```

> var str = "Seven" + "Eleven";
< undefined
> console.log(str);
SevenEleven
< undefined
> var str = "Seven " + 7;
< undefined
> console.log(str);
Seven 7
< undefined
> |

```

Gambar 26. Contoh Penerapan *String* Pada *Javascript*

Operator kondisional adalah operator yang membandingkan 2 operand yang bersifat *Boolean*. Contoh penerapan operator kondisional dapat dilihat pada gambar 27.



```
> x = 10;
< 10
> var cond = (x % 2 == 0) ? "Genap" : "Ganjil";
< undefined
> console.log(cond);
Genap
< undefined
> |
```

Gambar 27. Contoh Penerapat Operator Kondisional Pada *Javascript*

Pertama berikan angka pada variabel *x* bernilai 10 lalu buat variabel *cond* untuk melihat hasil *Boolean* dari operator kondisional. Setelah itu silahkan buat tulisan $(x \% 2 == 0)$ yaitu *statement* yang mengatakan “*x* module 2 sama dengan 0) “di mana *x* adalah 10. Setelah itu ada operator kondisional yaitu *?* di mana jika kondisi bernilai *true* maka akan mengambil value kiri (genap) dan jika kondisi bernilai *false* akan mengambil value kanan (ganjil).

Operator *typeof()* adalah operator yang dikhususkan untuk memeriksa tipe dari sebuah data (*primitive* atau *complex*). Seperti contoh pembuktian pada sub-bab 2.3.3 operator ini akan memeriksa apakah tipe ini merupakan *number*, *string*, *object*, atau *function*.

2.5 Array

Struktur data merupakan cara-cara atau metode yang digunakan untuk menyimpan data di dalam memori komputer. Salah satu struktur data yang sering digunakan dalam pemrograman adalah *Array*. *Array* merupakan struktur data yang digunakan untuk menyimpan sekumpulan data dalam satu tempat. *Array* adalah tipe data yang berisi kumpulan dari nilai atau tipe data lain. Nilai di dalam *array* disebut dengan elemen, dan setiap elemen memiliki ‘nomor urut’ yang dikenal dengan istilah *index*.

Setiap data dalam *Array* memiliki indeks, sehingga anda akan mudah memprosesnya. Penomoran index di dalam *array* dimulai dari angka 0, sehingga elemen pertama berada di index 0, elemen kedua berada di index 1, dst. *Index* maksimum yang bisa ditampung *array* dalam *Javascript* adalah 4.294.967.294 ($2^{23} - 2$), dengan jumlah elemen maksimum adalah 4.294.967.295.

Array di dalam *Javascript* tidak bertipe (*untyped array*). Elemen dari *array* bisa bertipe data *string*, *number* dan *boolean* dalam sebuah *array* yang sama, bahkan elemen

dari *array* bisa berupa objek atau *array* yang lain.

Array di dalam Javascript bersifat dinamis, dan tidak perlu didefinisikan berapa ukuran *array* pada saat membuat variabel. Jumlah elemen dapat ditambah dan dikurang setiap saat.

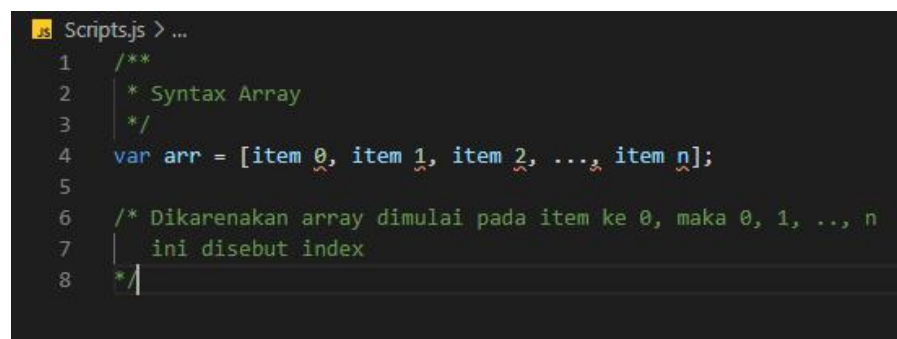
Array Index di dalam Javascript juga tidak harus berurutan, *Javascript* membolehkan elemen dari *array* 'tidak teratur'. Anda bisa mengisi hanya *index* 0, 5, dan 10 saja di dalam *array*. Contoh penerapan *array* pada *javascript* dapat dilihat pada Gambar 28.



```
Scripts.js > ...
1  /**
2   * Syntax Array
3   */
4  var arrEmpty = []; // Merupakan Array kosong
5
6  // Misalkan kita ingin memasukkan 3 nama merk
7  // kendaraan dengan tipe data undefined maka
8  // Gunakan Array
9
10 //penulisan model satu
11 var arr = ["Yamaha", "Toyota", "Honda"];
12 // Penulisan model dua
13 var car = [
14     "Hyundai",
15     "Subaru",
16     "Convert"
17 ];
```

Gambar 28. Penerapan *Array* pada *Java*

Seperti yang sudah dijelaskan sebelumnya, *array* terdiri dari beberapa *item* yang mempunyai tipe data yang sama semua. Oleh karena itu didapatkan rumus umum *array* yang dapat dilihat pada Gambar 29.



```
Scripts.js > ...
1  /**
2   * Syntax Array
3   */
4  var arr = [item 0, item 1, item 2, ..., item n];
5
6  /* Dikarenakan array dimulai pada item ke 0, maka 0, 1, .., n
7   | ini disebut index
8   */
```

Gambar 29. Rumus Umum *Array* pada *Javascript*

Dikatakan bahwa *array* tidak memiliki konsep teratur, hal ini dikarenakan

array bebas ditentukan urutan oleh data pada *array* yang telah dibuat dengan cara mempergunakan *index* sebagai parameter posisinya yang dapat dilihat pada gambar 30.

```
> var arr = [];  
< undefined  
> arr[0] = 1;  
< 1  
> arr[3] = 5;  
< 5  
> arr[4] = 7;  
< 7  
> console.log(arr);  
▶ (5) [1, empty x 2, 5, 7]  
< undefined  
> |
```

Gambar 30. Parameter *Array* dalam *Javascript*

Salah satu dari kumpulan *array* tersebut dapat diakses menggunakan konsep yang sama yaitu *indez* yang dapat dilihat pada Gambar 31.

```
> console.log(arr[3]);  
5  
< undefined
```

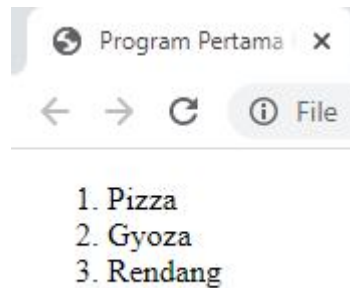
Gambar 31. Penerapan Konsep *indez*

Contoh kasus: Jika praktikan diminta untuk mencetak suatu *array* makanan namun dalam bentuk *website* (bukan `console.log()`). Apa yang akan praktikan lakukan? Maka yang praktikan harus lakukan akan dilakukan dapat dilihat pada Gambar 32.

```
1  /**  
2   * Menggunakan LOOP Untuk memasukkan semua  
3   * Data  
4   */  
5  
6  // Pertama buat array berisikan makanan yang dihidangkan  
7  var food = ["Pizza", "Gyoza", "Rendang"];  
8  
9  // membuat outputnya untuk dituliskan ke file html  
10 document.write("<ol>");  
11 //document.write() mirip console.log, namun output dilempar ke halaman website bukan console  
12 // dan <ol> di sini merupakan syntax html = orderedlist <ol></ol>  
13  
14 for (let index = 0; index < food.length; index++) {  
15   document.write("<li>" + food[index] + "</li>"); //mirip ol, li merupakan syntax html  
16 }  
17 document.write("</ol>");
```

Gambar 32. Contoh Kasus 1

Loop dapat digunakan untuk mencetak seluruh isi dari *array* yang telah dibuat. Lalu, dikarenakan perintahnya adalah tampilkan *output* ke dalam *website*. Maka diperlukan cara untuk mencetak *output* tersebut dengan *syntax* seperti *document.write()* serta meng-embed *syntax javascript* tersebut menuju html yang dapat dilihat pada Gambar 33.



Gambar 33. Penerapan *Loop* pada *Javascript*

Jika kodingan dalam gambar 33 diterjemahkan ke dalam format *file html* ini yang sebenarnya terjadi dapat dilihat pada Gambar 34.

```
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Program Pertama Ku!</title>
5  </head>
6  <body>
7
8    <ol>
9      <!-- hasil looping ada tadi pada .js -->
10     <li>
11       1. Pizza
12     </li>
13     <li>
14       2. Gyoza
15     </li>
16     <li>
17       3. Rendang
18     </li>
19   </ol>
20
21   <!-- script di bawah ini adalah penghubung antara .html dan .js -->
22   <script type="text/javascript" src="scripts.js"></script>
23 </body>
24 </html>
```

Gambar 34. Format *File HTML*

Berikutnya akan dipelajari mengenai beberapa *method* yang bisa diimplementasikan ke dalam *array* yang sudah dibuat. Kenapa ada kata “beberapa” karena ada banyak sekali *method* yang ada pada *javascript* ini. Pada contoh kali ini akan ada pembahasan mengenai *toString()*, *join()*, *pop()*, dan *push()*.

toString() adalah *method* yang digunakan untuk mengonversikan *array* anda ke dalam bentuk *string* dan dipisahkan dengan *comma*. Penggunaan *toString()* dapat dilihat pada Gambar 35.


```

> var arr = ["Apple", "Banana", "Mango", "Kiwi"];
< undefined
> arr.toString();
< "Apple,Banana,Mango,Kiwi"

```

Gambar 35. Penggunaan *toString()* pada *Javascript*

Dapat juga dilakukan hal yang sama dengan menggunakan *method join()* dan mengganti *comma* dengan apapun *statement* yang ada yang dapat dilihat pada Gambar 36.

```

> arr.toString();
< "Apple,Banana,Mango,Kiwi"
> arr.join(" ");
< "Apple Banana Mango Kiwi"
> |

```

Gambar 36. Penerapan *Method Join()*

pop() digunakan untuk menghapus isi *array* yang berada di *index* terakhir pada *array* anda. Contoh *pop()* dapat dilihat pada Gambar 37.

```

> var result = arr.pop();
< undefined
> console.log(arr);
▶ (3) ["Apple", "Banana", "Mango"]
< undefined
> console.log(result);
Kiwi
< undefined
> |

```

Gambar 37. Penerapan *Method Pop()*

push() digunakan untuk menambahkan isi *array* yang berada di *index* terakhir *array* anda. Contoh dari penggunaan *push()* dapat dilihat pada Gambar 38.


```

> var add = arr.push("Guava");
< undefined
> console.log(arr);
  ▶ (4) ["Apple", "Banana", "Mango", "Guava"]
< undefined
> console.log(add);
  4
< undefined
> |

```

Gambar 38. Penggunaan *Push()* pada *Javascript*

Metode selanjutnya adalah *sort* yang berarti cara *array* untuk mengatur dirinya sesuai dengan ekspektasi *programmer*. *Sort* dapat dibagi menjadi 2 jenis yang umum dipakai yaitu yaitu *sort()* dan *reverse()*

sort() merupakan perintah untuk menyusun *array* ke dalam urutan index yang sesuai. Kemudian, ***reverse()*** adalah kebalikan dari *sort()*. Penerapan *sort()* dan *reverse()* dapat dilihat pada Gambar 39.

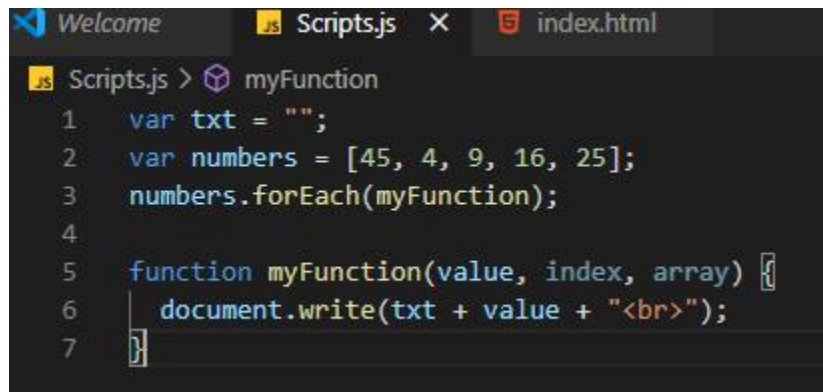
```

> console.log(arr);
  ▶ (4) ["Apple", "Banana", "MeLon", "Guava"]
< undefined
> console.log(arr.sort());
  ▶ (4) ["Apple", "Banana", "Guava", "MeLon"]
< undefined
> console.log(arr.reverse());
  ▶ (4) ["MeLon", "Guava", "Banana", "Apple"]
< undefined
> |

```

Gambar 39. Penerapan *Sort()* dan *Reverse()* pada *Javascript*

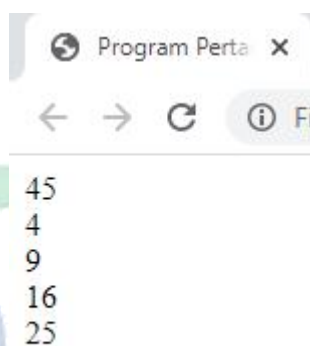
Terakhir, *javascript* memiliki *array iteration* yang paling sederhana yaitu *forEach()*. *ForEach()* adalah perintah untuk melakukan perulangan sebanyak isi data yang ada di dalam suatu objek. Objek yang dimaksud adalah variabel yang dituliskan sebelum *forEach()* dan isi () adalah parameter jika memang variabel tersebut dimasukkan ke dalam *function*. Penerapan *foreach()* dapat dilihat pada Gambar 40.

A screenshot of a code editor with three tabs: 'Welcome', 'Scripts.js', and 'index.html'. The 'Scripts.js' tab is active, showing a JavaScript file named 'myFunction'. The code consists of seven lines: 1. 'var txt = "";', 2. 'var numbers = [45, 4, 9, 16, 25];', 3. 'numbers.forEach(myFunction);', 4. (empty line), 5. 'function myFunction(value, index, array) {', 6. ' document.write(txt + value + "
");', 7. '}'

```
1  var txt = "";
2  var numbers = [45, 4, 9, 16, 25];
3  numbers.forEach(myFunction);
4
5  function myFunction(value, index, array) {
6      document.write(txt + value + "<br>");
7  }
```

Gambar 40. Penerapan *Foreach* Pada *Javascript*

Output kodingan dari Gambar 40 bisa dilihat pada gambar 41.

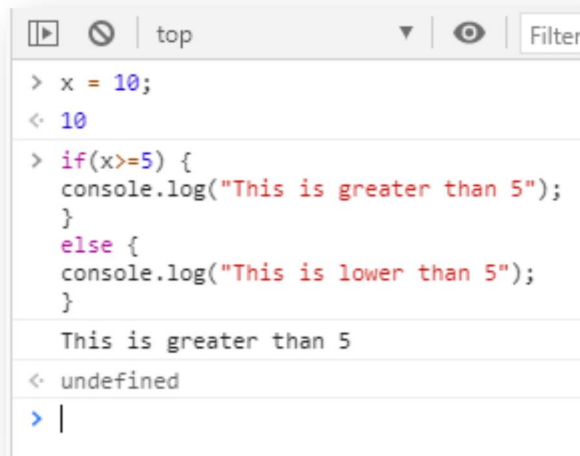


Gambar 41. *Output Kodingan Foreach*

2.6 *Statement Pengambilan Keputusan*

Dalam program *javascript* ataupun program dengan bahasa pemrograman lain, praktikan dituntut untuk membuat program dimana program tersebut harus memenuhi kondisi tertentu untuk dapat menjalankan programnya. Hal tersebut dikatakan bahwa program harus dapat mengambil keputusan.

Dalam *javascript*, ada dua jenis cara untuk menuliskan kondisi pada suatu program. Pertama adalah kondisi *if-else* dan *switch*. *if-else* adalah perintah menulis kondisi dengan ketentuan menggunakan parameter. Jika nilai parameter pada *if* bernilai *true* maka jalankan *block if* jika *false* jalankan *block program else*. Penerapan *if-else* dapat dilihat pada Gambar 42.



```
> x = 10;
< 10
> if(x>=5) {
  console.log("This is greater than 5");
}
else {
  console.log("This is lower than 5");
}
This is greater than 5
< undefined
> |
```

Gambar 42. Penerapan *if-else* Pada *Javascript*

Switch() merupakan kondisi di mana *statement* dapat menerima sebuah parameter dan jika dari *case* yang *statement* tersebut mengandung parameter tersebut, maka program akan menjalankan parameter yang dimaksud jika tidak *statement* akan menjalankan *default* program nya. Penerapan *switch()* dapat dilihat pada Gambar 43.



```
Scripts.js
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Gambar 43. Penerapan *Switch* Pada *Javascript*

Contoh implementasi dari metode *switch* dapat digunakan menggunakan *function Date()* yang dapat dilihat pada Gambar 44.



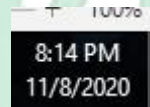
```

1  switch (new Date().getDay()) {
2      case 0:
3          console.log("Sunday");
4          break;
5      case 1:
6          console.log("Monday");
7          break;
8      case 2:
9          console.log("Tuesday");
10         break;
11         case 3:
12             console.log("Wednesday");
13             break;
14         case 4:
15             console.log("Thursday");
16             break;
17         case 5:
18             console.log("Friday");
19             break;
20         case 6:
21             console.log("Saturday");
22             break;
23         default:
24             console.log("WHERE ARE YOU LIVING!?");
25     }
26

```

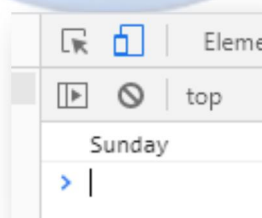
Gambar 44. Penerapan *Function Date* Pada *Javascript*

Contoh waktu dalam yang terdapat komputer dapat dilihat pada Gambar 45.



Gambar 45. Contoh Waktu Dalam Komputer

Pada Gambar 45 terdapat tanggal yaitu 08 November 2020. Di tanggal tersebut adalah hari Minggu maka hasil *output javascript* dari *date* adalah:



Gambar 46. *Output Date* Pada *Console Javascript*

2.7 *Statement* Perulangan

Perulangan adalah suatu fase di mana anda diminta untuk menjalankan atau melakukan suatu perintah program yang dilakukan sebanyak *n* kali. Terdapat beberapa varian dalam *statement* perulangan, secara umum dapat dibagi terhadap dua yaitu *for*

loop dan *while loop*.

for loop adalah perulangan yang biasanya digunakan untuk menjalankan iterasi yang terhitung. Tetapi bagaimanapun juga, *for loop* dapat digunakan untuk berbagai macam pengulangan yang lain. Sintaks umum *for loop* dapat dilihat pada Gambar 47.

```
Scripts.js
1  for (statement 1; statement 2; statement 3) {
2      // code block to be executed
3  }
```

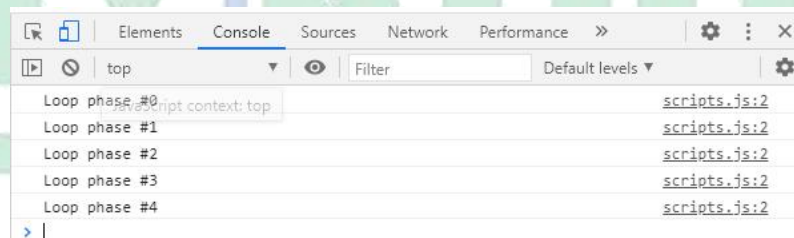
Gambar 47. Sintaks Umum *For Loop*

Implementasi dari *statement for loop* dapat dilihat pada Gambar 48.

```
Scripts.js > ...
1  for (let index = 0; index < 5; index++) {
2      console.log("Loop phase #" + index + "\n");
3  }
```

Gambar 48. Implementasi *For Loop* Pada Javascript

Output dari kodingan pada Gambar 48 dapat dilihat pada Gambar 49.



Message	Source
Loop phase #0	scripts.js:2
Loop phase #1	scripts.js:2
Loop phase #2	scripts.js:2
Loop phase #3	scripts.js:2
Loop phase #4	scripts.js:2

Gambar 49. *Output Statement For Loop*

while loop pengulangan adalah *statement* yang melakukan perulangan selama kondisi yang diuji bernilai *true*. Bentuk umum *While Loop* dapat dilihat pada Gambar 50.

```
while (condition) {
    // statements;
}
```

Gambar 50. Bentuk Umum *While Loop*

condition merupakan ekspresi yang menghasilkan nilai *boolean*. Cara kerjanya yaitu, pertama kali *condition* diperiksa, apabila bernilai *true*, maka *statement* dijalankan. Setelah *statement* dijalankan, *condition* diperiksa lagi, apabila masih bernilai *true* maka *statement* dijalankan lagi. Kemudian *condition* diperiksa

lagi, bila bernilai *false* maka *loop* berhenti. *Statement* tidak dijalankan sama sekali apabila kondisi bernilai *false* sejak awal.

Bentuk lain dari *while loop* adalah *do – while* yang cara kerjanya sama namun yang membedakan adalah *do* di sini menjadi indikator bahwa block program minimal satu kali harus dijalankan sebelum dilakukan *conditioning*. Sintaks *whileloop* dapat dilihat pada Gambar 51.

```
do {  
    //statement  
} while (condition);
```

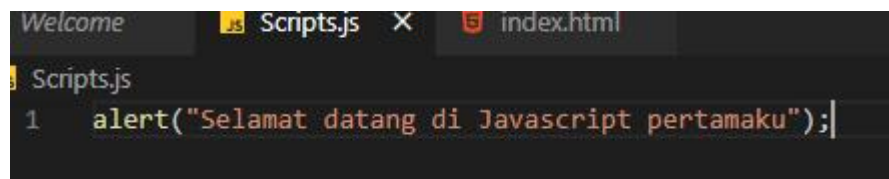
Gambar 51. Sintaks *While Loop*

2.8 Message Box

Pada materi – materi sebelumnya sudah digunakan perintah seperti *console.log()* dan *document.write()* untuk menuliskan *output* program yang ingin diketahui. Namun, ada cara lain yang dapat dilakukan untuk melihat *output* tersebut. Cara tersebut adalah dengan menggunakan *message box* (jendela dialog).

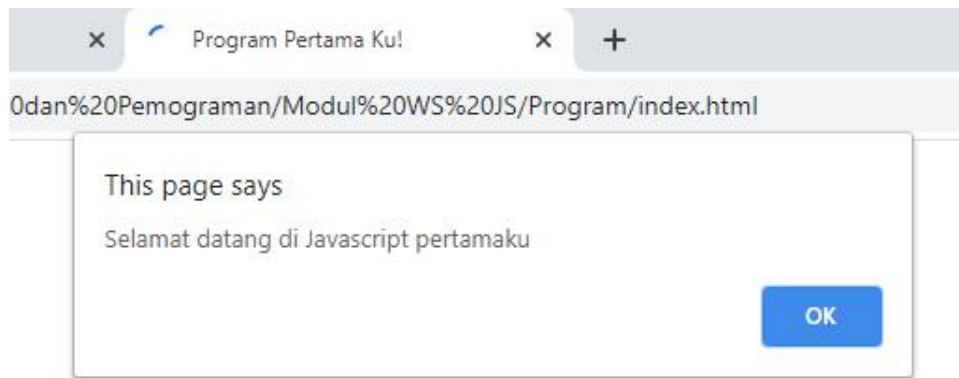
Ada beberapa jenis dari *message box* yaitu *alert()*, *confirm()*, dan *prompt()*. Setiap *message box* mempunyai fungsi masing – masing, namun inti dari semuanya sama yaitu sebagai media berinteraksi dengan *user* secara *objek visual*.

alert() merupakan *message box* yang digunakan sebagai *notice* suatu berita atau pemberitahuan. Misal ketika pengguna salah menginput data dan akan diberi pesan peringatan terhadap kesalahan tersebut yang bisa menggunakan *alert()*. Sintaks *alert* dapat dilihat pada Gambar 52.

A screenshot of a code editor with a dark background. At the top, there are tabs for 'Welcome', 'Scripts.js', and 'index.html'. The 'Scripts.js' tab is active. Below the tabs, the code '1 alert("Selamat datang di Javascript pertamaku");' is written in a light-colored font. The line number '1' is on the left, and the code is on the right.

Gambar 52. Sintaks *Alert()* pada *Javascript*

Output kodingan pada gambar 52 dapat dilihat pada Gambar 53.



Gambar 53. *Output Alert Pada Javascript*

confirm() adalah *message box* yang ditujukan untuk memberikan opsi kepada pengguna apakah pengguna tersebut memilih *true* or *false* di dalam menjawab pertanyaan sebuah program. Contoh pemrograman dari *confirm* dapat dilihat pada Gambar 54.

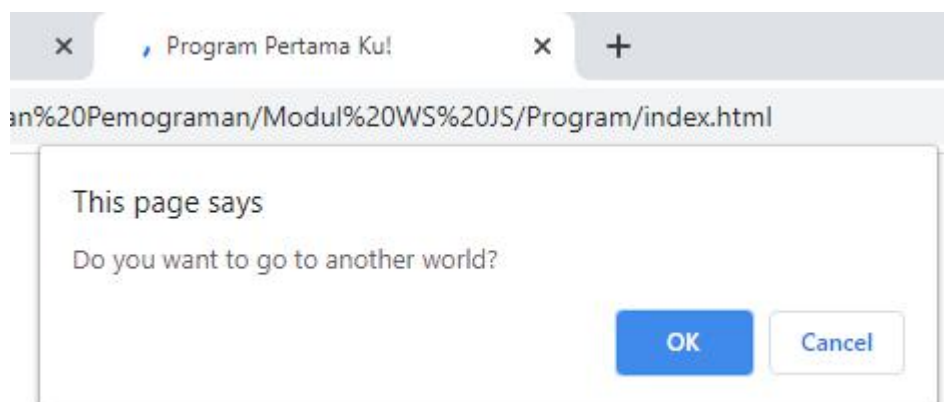
```

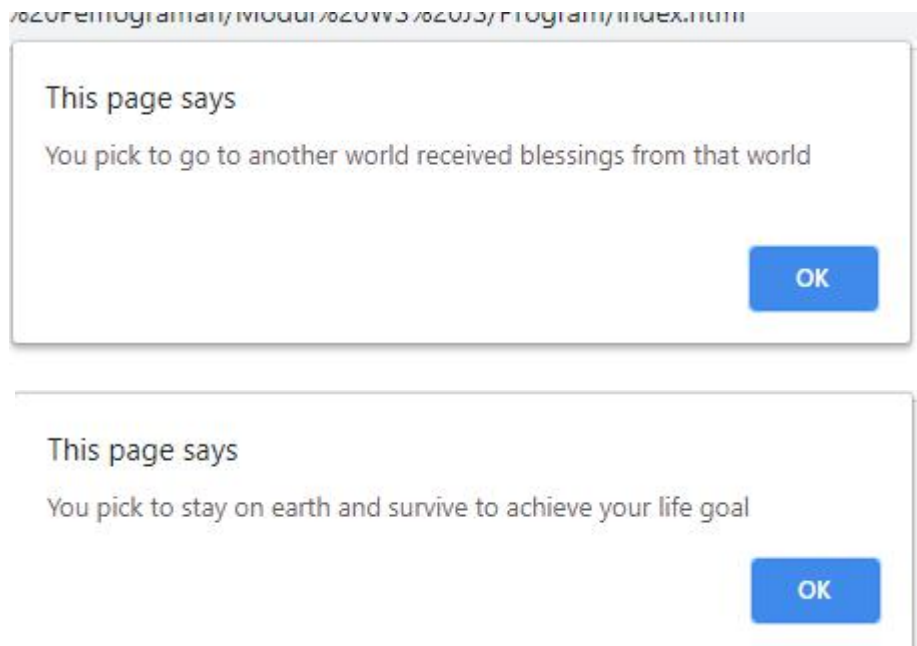
1  var confirmation = confirm("Do you want to go to another world?");
2  if (confirmation) {
3      alert("You pick to go to another world received blessings from that world");
4  } else {
5      alert("You pick to stay on earth and survive to achieve your life goal");
6  }

```

Gambar 54. Contoh Pemrograman *Message*

Output dari kodingan yang terdapat pada Gambar 54 dapat dilihat pada Gambar 55.





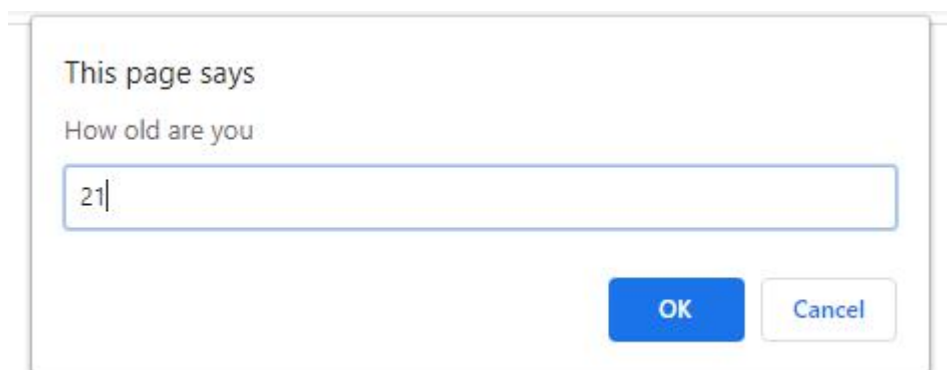
Gambar 55. *Output Program Confirm*

prompt() adalah *message box* yang ditujukan untuk menerima masukan dari *user*. Hasil masukan tersebut akan disimpan pada suatu variabel untuk dapat digunakan sesuai kebutuhan program. Contoh pemrograman *prompt()* dapat dilihat pada Gambar 56.

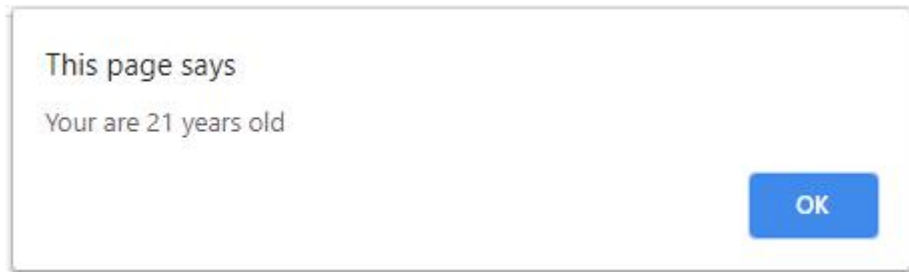
```
Scripts.js > ...  
1  var age = prompt("How old are you");  
2  alert("Your are " + age + " years old");
```

Gambar 56. Contoh Kodingan *Prompt*

Output dari kodingan gambar 56 dapat dilihat pada Gambar 57.



Gambar 57. *Output Kodingan Prompt* akan meminta *input* dari *user*



Gambar 58. *Output Kodingan Prompt setelah user memberikan input*



3

OOP Pada JavaScript

Objektif :

- Memahami Object
 - Memahami Property dan Method
 - Membuat dan Memanggil Object
 - Memahami This
-

OOP (*Object Oriented Programming*) adalah proses pembuatan program / aplikasi dengan cara menyusunnya dari beberapa *object* (komponen yang lebih kecil). Sederhananya, pemrograman OOP adalah paradigma atau teknik atau cara dalam menulis program. *Object* atau komponen itu sendiri juga bisa tersusun dari *object* yang lain. Pemrograman dengan konsep tersebut memiliki berbagai kelebihan dibandingkan dengan konsep pemrograman lain seperti dapat membangun kelasnya sendiri kemudian digabung menjadi satu kesatuan sehingga dapat menghemat waktu dibandingkan dengan membangunnya satu per satu (*Parallel Development*), kelas-kelas yang sudah ada dapat digunakan kembali (*reusable*), dan konsep pemrograman ini memiliki basis *coding* yang terpusat dan mudah diatur serta data-datanya dapat lebih mudah diakses ketika dibutuhkan dan dapat meningkatkan keamanan program kalian (*secure*).

3.1 Memahami Object

Topik *object* pada materi ini merupakan pembahasan yang cukup penting karena didalam *Javascript* hampir semua elemen terbuat dari *object*. Salah satu contoh dari *object* adalah *Array*. *Array* adalah kumpulan nilai yang memiliki *index*. Pengertian lain dari *array* adalah sebuah variabel yang dapat menampung banyak nilai dan memiliki *index* yang dimulai dari nol. Sedangkan hampir mirip dengan *array* yaitu *object* adalah kumpulan nilai, tetapi tidak memiliki *index* melainkan memiliki nama. Jadi, didalam *object* tidak ada *index* yang dimulai dari nol melainkan dia memiliki nama untuk membedakannya.

Untuk lebih memahami pembahasan mengenai *object*, dapat dilihat pada contoh kasus yang akan dibahas pada Gambar 59. Untuk membuat kumpulan data seperti Nama,

NPM, Kelas, Jurusan, dan Nilai (IP Semester) harus menggunakan variabel biasa tanpa menggunakan array atau object maka kita akan tulis seperti Gambar 59.

```
var nama = 'Zhafran Malik';
var npm = 55412921;
var kelas = '3IA14';
var jurusan = 'Teknik Informatika';
var IPSemester = [2.90, 3.10, 3.25, 2.88, 3.59];
```

Gambar 59. Kasus javascript 1.

Dalam kodingan tersebut kita memiliki beberapa variabel yang menampung data mahasiswa dimulai dari nama sampai IPSemester. Karena IPSemester menampung nilai-nilai dari setiap semester maka kita masukkan kedalam array. Lalu kemudian kita ingin memiliki sebuah variabel untuk menghitung IPKumulatif yang kita buat menggunakan *function* seperti pada kodingan dibawah ini

```
function IPKumulatif(IPSemester) {
  var total = 0;
  for( var i = 0; i < IPSemester.length; i++) {
    total += IPSemester[i];
  }
  return total/IPSemester.length;
};
```

Gambar 60.

Kita memiliki *function* yang namanya IPKumulatif kemudian yang menerima parameternya adalah IPSemester, lalu didalam *function* kita hitung nilai dari masing-masing semesternya dan kemudian kita jumlahkan kedalam variabel total dan terakhir totalnya kita bagi dengan jumlah semesternya.

Dari contoh diatas, kita memiliki masalah yaitu bagaimana kalau kita ingin membuat data mahasiswa yang kedua atau ingin menambah data mahasiswa dengan jumlah yang banyak. Jika kita menggunakan variabel biasa seperti contoh diatas, pastinya akan merepotkan. Untuk permasalahan tersebut bisa kita sedikit perbaiki menggunakan array. Contoh jika menggunakan array akan seperti ini

```
var mahasiswa =
['Zhafran Malik', 55412921, '3IA14', 'Teknik Informatika', [2.90, 3.10, 3.25, 2.88, 3.59]];
```

Gambar 61.

Kalau kita ingin menambah data mahasiswa maka kita buat variabel baru **var mahasiswa2** dan kita isi datanya sesuai dengan contoh diatas. Kemudian untuk menghitung IPKumulatifnya kita juga tetap menggunakan *function* secara terpisah atau diluar dari variabel mahasiswa tersebut. Contohnya seperti ini

```
var mahasiswa =
['Zhafran Malik', 55412921, '3IA14', 'Teknik Informatika', [2.90, 3.10, 3.25, 2.88, 3.59]];

function IPKumulatif(IPSemester) {
  var total = 0;
  for( var i = 0; i < IPSemester.length; i++) {
    total += IPSemester[i];
  }
  return total/IPSemester.length;
};
```

Gambar 62.

Sehingga kalau kita ingin tahu IPKumulatif dari seorang mahasiswa, kita bisa panggil dengan cara seperti ini

```
IPKumulatif(mahasiswa[2]);
```

Gambar 63.

Kita panggil *function* dengan nama IPKumulatif yang parameternya kita ambil dari variabel mahasiswa dengan index ke-2.

Jika menggunakan array kita hanya dapat memperbaikinya sedikit. Masih lebih baik ketimbang kita menggunakan variabel biasa seperti contoh pertama tadi. Namun tetap saja akan membuat banyak baris kode / baris program jika memiliki data mahasiswa dalam jumlah yang banyak. Maka dari itu, permasalahan seperti diatas tadi dapat atasi dengan menggunakan *object*. Tentu saja kita akan dapat mengetik kode atau program lebih ringkas lagi jika memiliki data dalam jumlah yang banyak. Dan contoh penulisannya seperti ini

```
// jika menggunakan object
var mahasiswa = {
  nama : 'Zhafran Malik',
  npm : 55412921,
  kelas : '3IA14',
  jurusan : 'Teknik Informatika',
  IPSemester : [2.90, 3.10, 3.25, 2.88, 3.59],
  IPKumulatif : function() {
    var total = 0;
    var ip = this.IPSemester;
    for( var i = 0; i < ip.length; i++ ) {
      total += ip[i];
    }
    return total/ip.length;
  }
}
mahasiswa.IPKumulatif();
```

Gambar 64.

Seperti contoh kodingan diatas, bisa dilihat bahwa didalam *object* tidak memiliki index seperti array melainkan memiliki sebuah nama(key) didalam *object* tersebut. Serta *function* yang kita buat juga ada didalam *object*, jadi *function* yang kita buat tidak terpisah atau menyatu bersama dengan key dan data didalam variabel mahasiswa.

3.2 Memahami *Property* dan *Method*

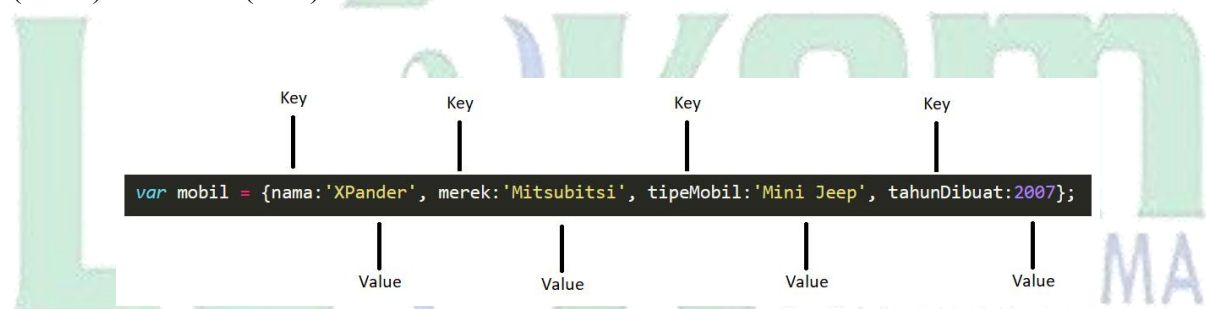
Agar lebih mudah dipahami apa itu *Property* dan *Method*. Kita coba buat analogi terlebih dahulu tentang *object*. Jadi kita analogikan kita memiliki sebuah objek berupa mobil. Mobil ini pastinya memiliki sebuah data contohnya adalah nama, merek, tipe mobil, tahun dibuat, warna, dan lain sebagainya. Dari data-data tersebut, itu merupakan *Property*. Kemudian objek kita yaitu mobil, dapat melakukan sesuatu. Contohnya adalah mobil bisa maju dengan kecepatan tinggi, bisa maju dengan kecepatan rendah, bisa mundur, dan bisa berhenti. Dari situ kita bisa pahami bahwa yang disebut *Method* adalah sebuah fungsi atau *function* nya.

Jadi, pengertian dari *Property* adalah ciri khas dari objek (variabel). Sedangkan *Method* adalah perilaku dari objek (fungsi). Dari analogi diatas tadi kita bisa modelkan kedalam kode program seperti ini :


```
// membuat object mobil
var mobil = {
  nama : 'XPander',
  merek : 'Mitsubitsi',
  tipeMobil : 'Mini Jeep',
  tahunDibuat : 2007,
  warna : 'putih',
  majuKencang : function () {
    console.log('Mobil ini melaju dengan kecepatan 300Km/h');
  },
  majuPelan : function () {
    console.log('Mobil ini melaju dengan kecepatan 10Km/h');
  },
  mundur : function () {
    console.log('Mobil ini melaju kebelakang');
  },
  stop : function () {
    console.log('Mobil ini berhenti');
  }
};
```

Gambar 65.

Objek pada javascript dapat dibuat dengan tanda kurung kurawal dengan isi berupa *key* (kunci) dan *value* (nilai). Contoh :



Gambar 66,

Key dengan *value* dipisahkan menggunakan titik dua. Jika nilai berupa karakter maka diberikan tanda kutip, bisa kutip satu (' ') atau kutip dua (" "). Jika berupa angka maka tidak perlu menggunakan tanda kutip. Dan objek diawali kurung kurawal buka ({) serta diakhiri kurung kurawal tutup (}) dan jangan lupa untuk memberikan titik koma (;) diakhir.

Contoh *property* adalah seperti ini :

```
var mobil = {
  // properti
  nama : 'XPander',
  merek : 'Mitsubitsi',
  tipeMobil : 'Mini Jeep',
  tahunDibuat : 2007,
  warna : 'putih',
};
```

Gambar 67.

Contoh *Method* adalah seperti ini :

```
// method
majuKencang : function () {
    console.log('Mobil ini melaju dengan kecepatan 300Km/h');
},
majuPelan : function () {
    console.log('Mobil ini melaju dengan kecepatan 10Km/h');
},
mundur : function () {
    console.log('Mobil ini melaju kebelakang');
},
stop : function () {
    console.log('Mobil ini berhenti');
}
};
```

Gambar 68.

3.3 Membuat dan Memanggil *Object*

Membuat *object* pada javascript ada beberapa cara diantaranya *Object Literal*, *Function Declaration*, *Constructor Function(keyword new)*. Mari kita langsung menuju kepembahasan :

1. *Object Literal*

Cara ini merupakan cara yang paling mudah ketika ingin membuat sebuah objek. Lalu misalkan kita ingin membuat sebuah objek mahasiswa menggunakan cara ini. Dan contohnya seperti ini:

```
var mhs = {
}
```

Gambar 69.

Dari contoh diatas, itu merupakan sebuah objek yang belum dimasukkan sebuah *property* didalamnya. Kita juga bisa menyebutnya dengan objek kosong. Jika kita masukkan *property* nya maka akan seperti ini

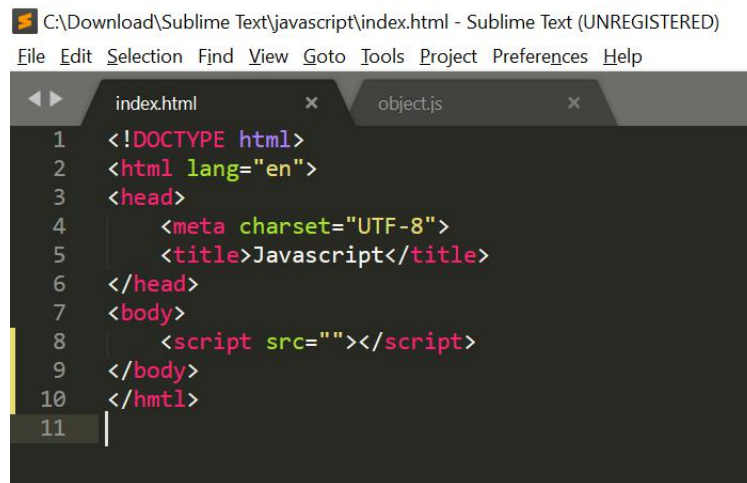
```
var mhs = {
    nama : 'Muhammad Alif',
    npm : '51627187',
    email : 'alif@student.gunadarma.ac.id',
    jurusan : 'Teknik Informatika'
}
```

Gambar 70.

Dengan menggunakan cara seperti itu, maka kita memiliki sebuah objek yang namanya 'mhs' yang didalamnya berisi 4 buah *property* yaitu nama, npm, email, dan jurusan.

Kemudian untuk memanggil data mahasiswa tersebut, caranya sebagai berikut :

Langkah 1 : Buka Teks Editor, buat file dengan nama index.html dan koding seperti dibawah ini.




```
C:\Download\Sublime Text\javascript\index.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.html x object.js x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Javascript</title>
6 </head>
7 <body>
8   <script src=""></script>
9 </body>
10 </html>
11
```

Gambar 71.

Langkah 2 : Buat file baru dengan nama object.js kemudian koding seperti ini.

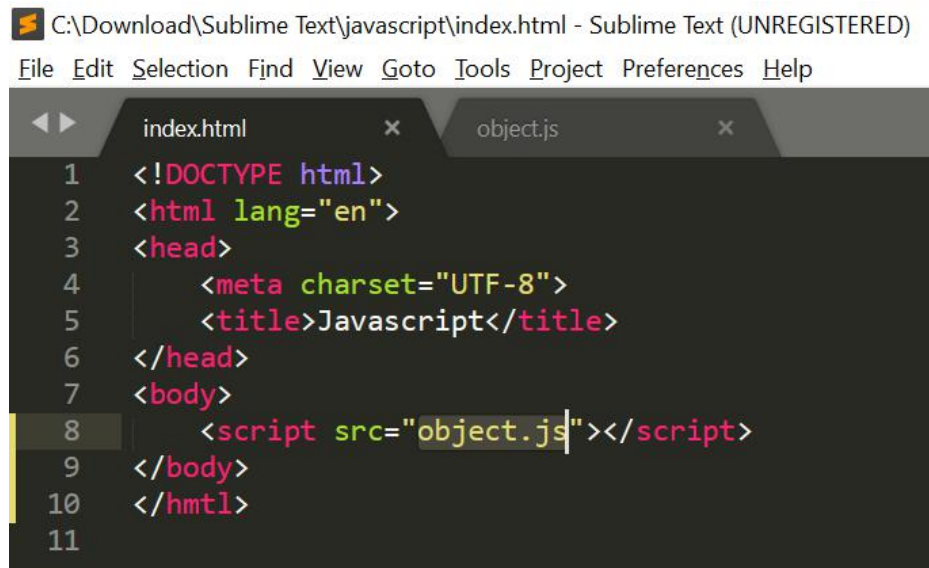


```
C:\Download\Sublime Text\javascript\object.js - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.html x object.js x
1 // Membuat Object
2 // Object Literal
3 var mhs = {
4   nama : 'Muhammad Alif',
5   npm : '51627187',
6   email : 'alif@student.gunadarma.ac.id',
7   jurusan : 'Teknik Informatika'
8 }
9
10
```

Gambar 72.

Langkah 3 : Didalam file index.html kita masukkan nama file yang berisi data mahasiswa dengan kodingan seperti ini.



C:\Download\Sublime Text\javascript\index.html - Sublime Text (UNREGISTERED)

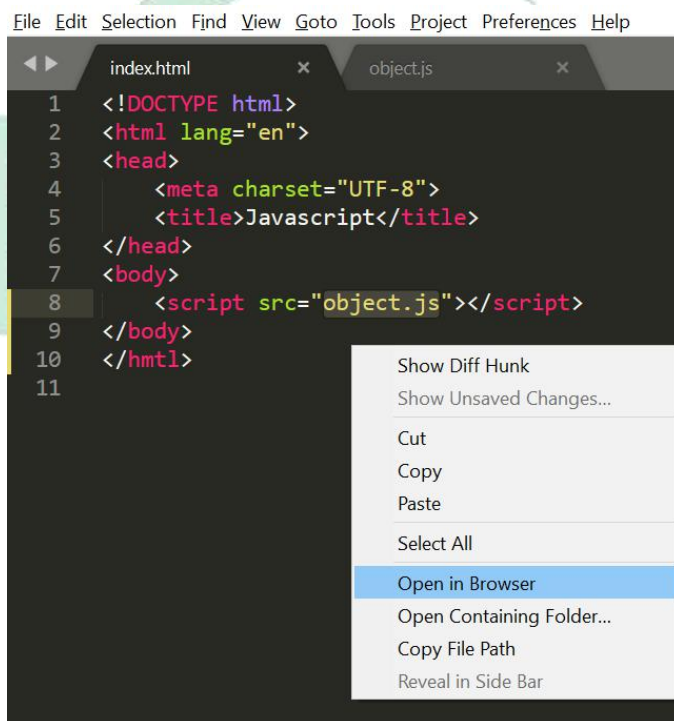
File Edit Selection Find View Goto Tools Project Preferences Help

index.html x object.js x

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Javascript</title>
6 </head>
7 <body>
8   <script src="object.js"></script>
9 </body>
10 </html>
11
```

Gambar 73.

Pastikan nama file dengan kodingan yang kita ketik di index.html itu sama. Dan kalau sudah klik kanan dan *Open in Browser*.



File Edit Selection Find View Goto Tools Project Preferences Help

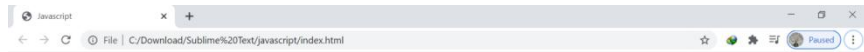
index.html x object.js x

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Javascript</title>
6 </head>
7 <body>
8   <script src="object.js"></script>
9 </body>
10 </html>
11
```

- Show Diff Hunk
- Show Unsaved Changes...
- Cut
- Copy
- Paste
- Select All
- Open in Browser
- Open Containing Folder...
- Copy File Path
- Reveal in Side Bar

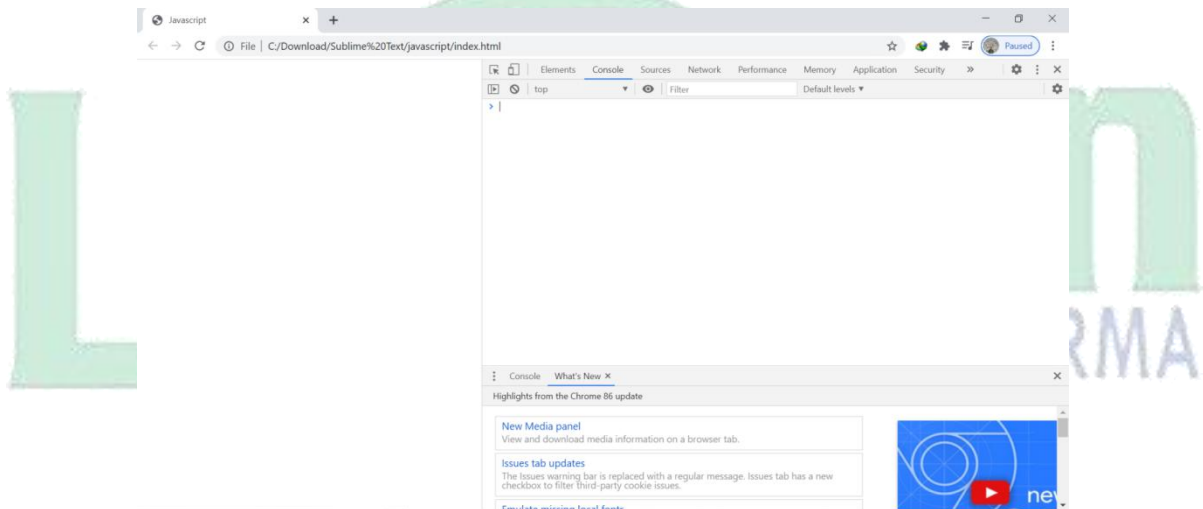
Gambar 74.

Langkah 4 : Jika sudah berhasil terbuka *browser* nya. Tampilannya seperti ini



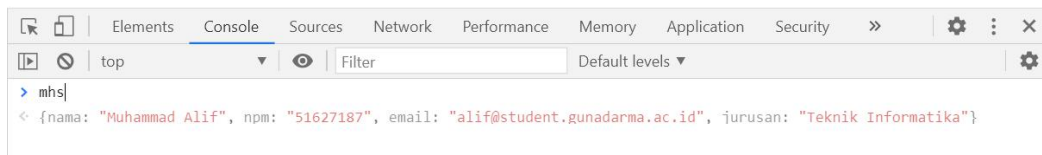
Gambar 75.

Browser yang kita buka tidak ada tampilan apapun, kalau sudah seperti itu kita klik kanan dan pilih *inspect* atau bisa juga `ctrl+shift+i` kemudian kita pilih *console* dan tampilannya seperti ini.



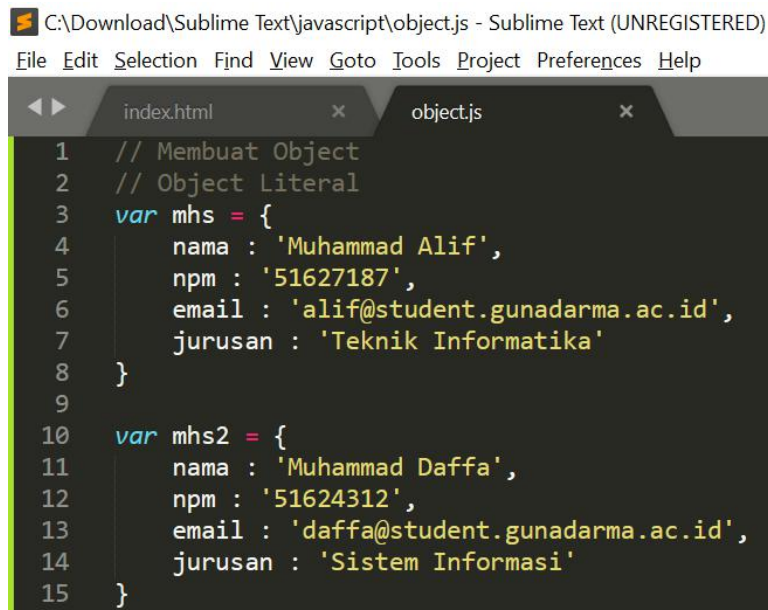
Gambar 76.

Langkah 5 : Didalam *console* kita ketik nama objek yang kita buat tadi yaitu 'mhs' dan tekan enter. Maka tampilannya akan seperti ini.



Gambar 77.

Jika berhasil melakukan langkah-langkah diatas. Maka proses pemanggilan berhasil dan kita bisa melihat data yang kita ketik didalam teks editor tadi. Apabila kita menambahkan data lain, maka buat objek baru, lakukan seperti langkah diatas, dan contoh hasilnya seperti ini.

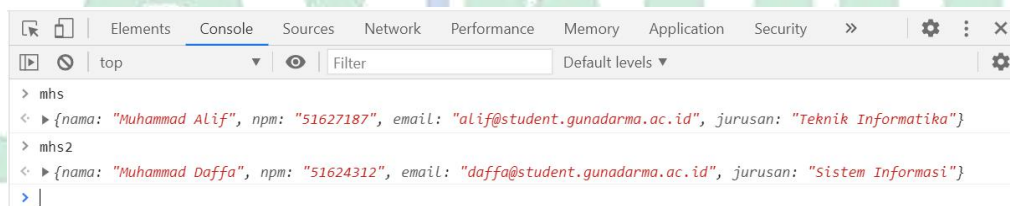


C:\Download\Sublime Text\javascript\object.js - Sublime Text (UNREGISTERED)

```
File Edit Selection Find View Goto Tools Project Preferences Help
```

```
1 // Membuat Object
2 // Object Literal
3 var mhs = {
4     nama : 'Muhammad Alif',
5     npm : '51627187',
6     email : 'alif@student.gunadarma.ac.id',
7     jurusan : 'Teknik Informatika'
8 }
9
10 var mhs2 = {
11     nama : 'Muhammad Daffa',
12     npm : '51624312',
13     email : 'daffa@student.gunadarma.ac.id',
14     jurusan : 'Sistem Informasi'
15 }
```

Gambar 78.



Elements Console Sources Network Performance Memory Application Security >> ⚙️ ⋮ ✕

top Filter Default levels ⚙️

```
> mhs
< ▶ {nama: "Muhammad Alif", npm: "51627187", email: "alif@student.gunadarma.ac.id", jurusan: "Teknik Informatika"}

> mhs2
< ▶ {nama: "Muhammad Daffa", npm: "51624312", email: "daffa@student.gunadarma.ac.id", jurusan: "Sistem Informasi"}

> |
```

Gambar 79.

2. Function Declaration

Pada cara ini, pembuatan objek pada javascript berbeda dengan cara yang pertama tadi. Kalau cara pertama membuat objek satu per satu dan isinya ada didalam objek. Jika datanya berbeda maka objeknya juga akan berbeda. Sedangkan dengan cara *function declaration* kita cukup membuat satu objek saja dan untuk membedakan datanya hanya perlu menambahkan data tanpa harus membuat objek yang baru. Contohnya seperti ini.

```
// Function Declaration
function buatObjectMahasiswa(nama, npm, email, jurusan) {
  var mhs = {};
  mhs.nama = nama;
  mhs.npm = npm;
  mhs.email = email;
  mhs.jurusan = jurusan;
  return mhs;
}
```

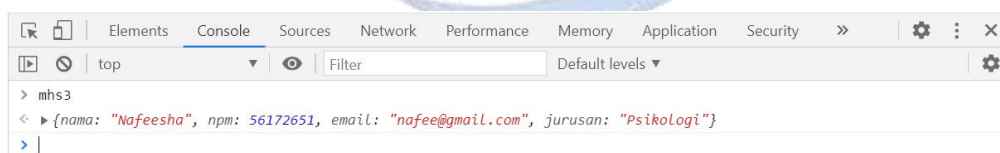
Gambar 80.

Gambar diatas adalah dimana kita hanya membuat objeknya saja. Kita buat sebuah *function* dengan nama 'buatObjectMahasiswa' lalu didalam kurung kita kirimkan propertinya yaitu nama, npm, email, dan jurusan. Kemudian didalamnya, deklarasikan variabel dengan nama 'mhs' dan deklarasikan sebagai objek. Lalu objek itu kita isi properti-propertinya dengan argument yang sudah kita tulis didalam *function* nya dan terakhir kita kembalikan nilai objek nya. Lalu untuk datanya kita isi dengan cara seperti ini.

```
var mhs3 = buatObjectMahasiswa('Nafeesha', 56172651, 'nafee@gmail.com', 'Psikologi')
```

Gambar 81.

Pastikan urutannya sesuai dengan properti yang sudah dimasukkan didalam *function* nya tadi. Jika sudah, save file nya lalu buka kembali browser nya kemudian refresh dan ketikkan nama datanya dan tekan enter. Maka hasilnya akan seperti ini.



Gambar 82.

3. Constructor Function

Constructor ini merupakan *function* yang khusus membuat *object*. Pada pendeklarasian nya sama saja seperti mendeklarasikan *function* yang kita buat sebelumnya. Akan ada kemiripan yang akan bisa kalian lihat. Contohnya seperti ini.


```
// Constructor
function Mahasiswa(nama, npm, email, jurusan) {
  this.nama = nama;
  this.npm = npm;
  this.email = email;
  this.jurusan = jurusan;
}
```

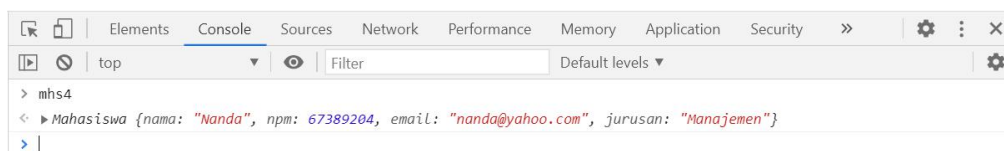
Gambar 83.

Kita buat *function* terlebih dahulu dan kita berikan namanya bebas, namun untuk cara penulisannya ketika menamakan *constructor*, kita ikuti aturan sama seperti kita menamakan sebuah kelas dalam bahasa pemrograman lain. Bisa dilihat dari contoh yaitu dengan huruf awalnya capital lalu kita tulis nama objeknya langsung. Lalu argumennya kita isi dengan properti nama, npm, email, dan jurusan. Kemudian kita ketik sesuai dengan kodingan diatas. Perbedaan dengan cara *function declaration* sebelumnya adalah tidak adanya deklarasi objek atau variabel dan juga tidak ada *return*. Lalu kita coba isi datanya dan panggil dengan cara seperti dibawah ini.

```
var mhs4 = new Mahasiswa('Nanda', 67389204, 'nanda@yahoo.com', 'Manajemen');
```

Gambar 84.

Untuk urutan datanya tetap diperhatikan agar sesuai. Kita buat variabel dengan nama 'mhs4' kemudian untuk *constructor* ada *keyword new* lalu ketik nama *function* nya yaitu 'Mahasiswa' lalu kita masukkan datanya. Hal penting pada saat pemanggilan adalah *keyword new* nya. Karena jika tidak ada 'new' maka javascript akan berasumsi bahwa kalian memanggil objeknya dengan *function declaration* sedangkan didalam *constructor* deklarasi objek dan juga *return* tidak ada. Dan mari kita lihat hasilnya.



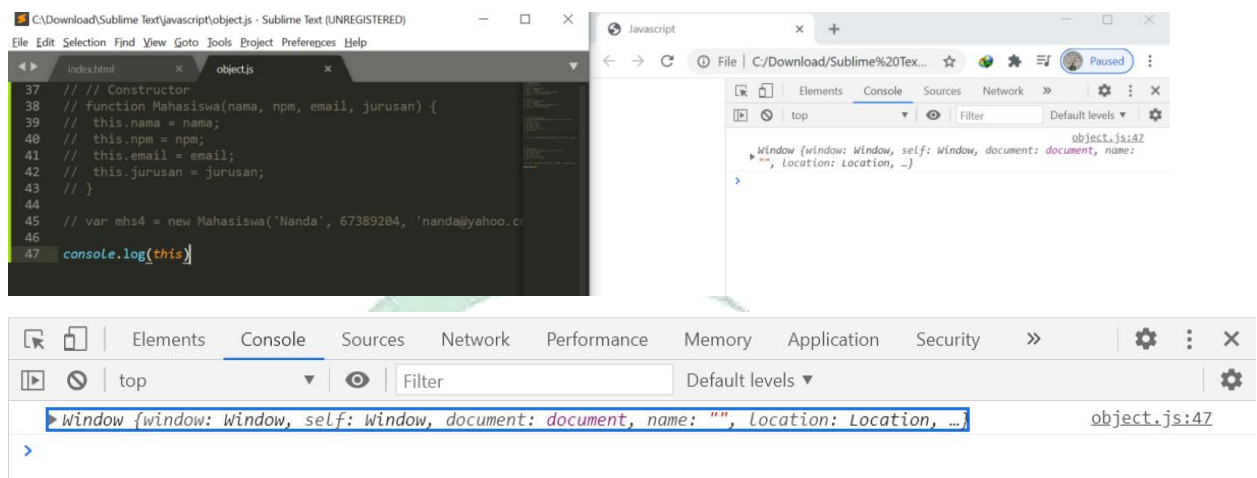
Gambar 85.

Dapat diperhatikan bahwa perbedaan antara *constructor* dengan *function declaration* yang sebelumnya adalah pada saat deklarasi variabel objek dan *return*. Kalau dengan *constructor* kita tidak perlu menambahkan variabel objek dan *return* karena sudah dibuat secara otomatis

oleh javascript sedangkan kalau *function declaration* kita perlu membuatnya sendiri dan kita berikan nama objeknya juga sendiri.

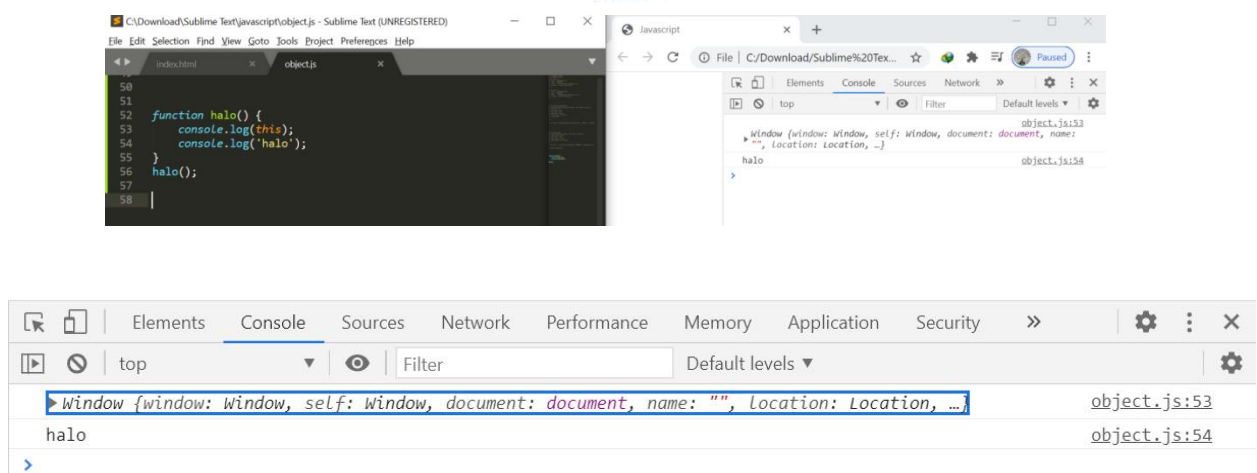
3.4 Memahami *This*

This merupakan *keyword* spesial yang secara otomatis didefinisikan pada sebuah *function*. Jadi maksudnya adalah ketika kalian membuat sebuah *function* atau *object* dengan cara apapun maka akan ada *keyword this* yang didefinisikan. Jadi sebagai contoh misalkan kita ketik 'console.log(this)' artinya kita ingin melihat apa si yang ada didalam *this* itu.



Gambar 86.

Dan setelah kita cari tau *this* itu apa, ternyata *keyword this* itu ada isinya dan isinya adalah window atau disebut dengan *object global* jadi sama saja ketika kita tulis console.log(window). Contoh lain coba kita tulis seperti ini dan akan tampil seperti ini.



Gambar 87.

Dan hasilnya juga sama seperti membuat *this* di *scope global*. Kalau kita membuat seperti ini:

```
var a = 10;
console.log(window.a);
```

Gambar 88.

Scope global itu adalah 'window'. Saat kita bikin var a, kita bisa bilang *object window* didalamnya ada properti 'a' yang isinya adalah 10. Maka akan tampil hasil '10' didalam *console*. Mari kita kembali kedalam kodingan yang tadi.

```
function halo() {
  console.log(this);
  console.log('halo');
}
halo();
```

Gambar 89.

Ketika kita ketik 'halo()' artinya kita menjalankan *function*. Dan itu juga sama saja artinya ketika kita mengetikkan 'window.halo()' karena dia memiliki *scope global*:

```
window.halo();
```

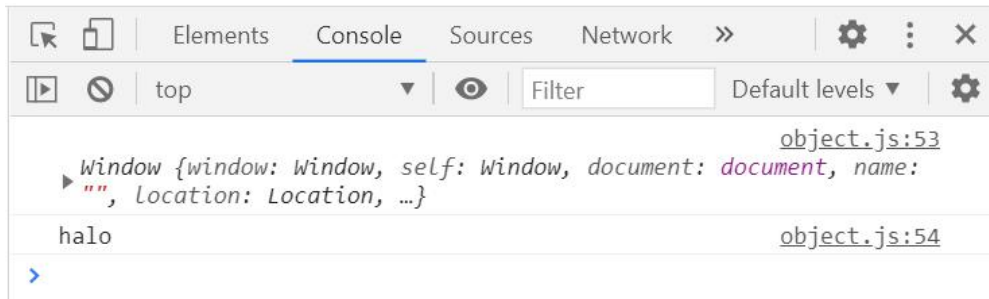
Gambar 90.

Dan juga, sama saja ketika kita ketikkan 'this.halo()'. Ketika kita membuat *function declaration*, *this* mengembalikan *object global*. Jadi *this* dalam *function declaration* konteksnya adalah mengembalikan *object global*.

```
this.halo();
```

Gambar 91.

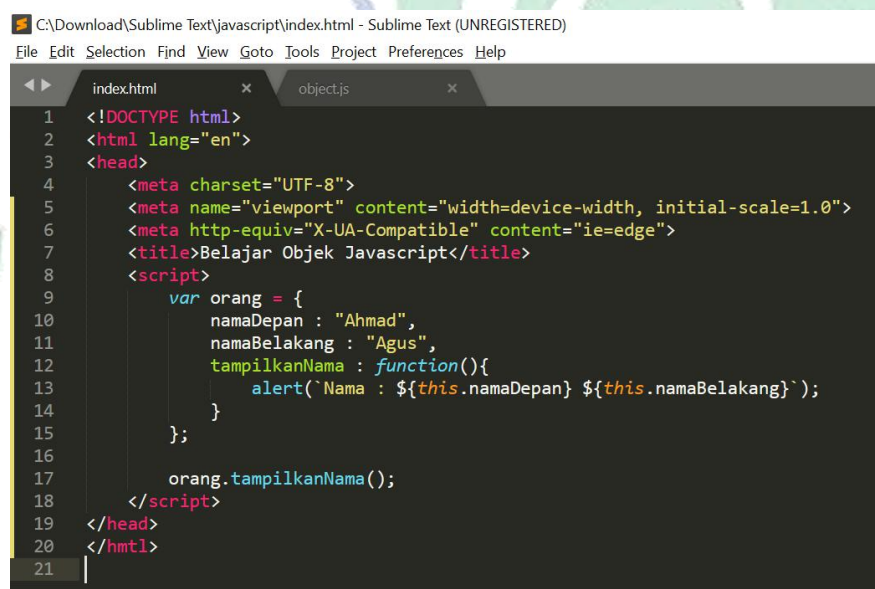
Jika kita lihat hasilnya akan sama, yaitu seperti ini



Gambar 92.

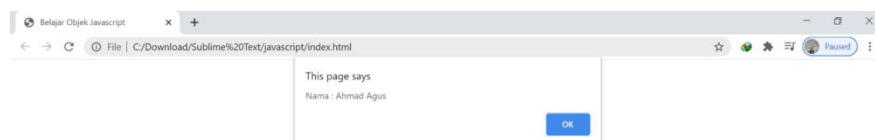
Selanjutnya adalah kita mencoba contoh lain. Selain arti *this* yang sudah dijelaskan diatas. Keyword *this* ini digunakan untuk mengakses *property* dan *method* dari dalam *method(object)*.

Contoh :



Gambar 93.

Maka hasilnya adalah seperti ini



Gambar 94.

Kata kunci *this* pada contoh diatas akan mengacu pada objek 'orang'

Referensi :

- <https://www.petanikode.com/javascript-objek/>
- WebProgrammingUnpas/OBJECT pada Javascript



LAMPIRAN

Program pertama

```
Goto Tools Project Preferences Help

program.js x tampil.html x
1 var n = prompt("Masukan nama anda:");
2 var c = confirm("Hai "+n+"! Apakah saya tampan?");
3 if (c == true) {
4     alert('Oh Thanks!!');
5 }else{
6     alert('Why?!!');
7 }

Goto Tools Project Preferences Help

program.js x tampil.html x
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <script type="text/javascript" src="program.js"></script>
5     </head>
6 </html>
```

Program kedua

```
alert.html x
1 <!DOCTYPE>
2 <html>
3     <head>
4         <title> Dialog ALert </title>
5     </head>
6     <body>
7         <script>
8             alert("Hello kawan! Selamat datang di MessageBox JavaScript!");
9         </script>
10    </body>
11 </html>
```



```
alert.html  confirm.html x  Untitled-1 ●

1  <!DOCTYPE>
2  <html>
3      <head>
4          <title> Dialog Confirm </title>
5      </head>
6      <body>
7          <script>
8              var yakin = confirm("Apakah kamu yakin akan meninggalkan halaman ini?");
9              if (yakin){
10                  document.write("ya!");
11              } else {
12                  document.write("Saya tetap di sini");
13              }
14          </script>
15      </body>
16  </html>

alert.html  confirm.html  prompt.html x

1  <!DOCTYPE>
2  <html>
3      <head>
4          <title> Dialog Prompt </title>
5      </head>
6      <body>
7          <script>
8              var nama = prompt("Siapa nama kamu?", "");
9              document.write("<p> Hello " + nama + "</p>");
10          </script>
11      </body>
12  </html>
```

Program ketiga

```
return!  argumenarray.js x  array.html x  returningarray.js  kelas

1  <html lang="en">
2  <head>
3      <title>Mengambil data dari Array</title>
4  </head>
5  <body>
6      <script>
7          //Membuat Array
8          var products = ["Pensil", "Pulpen", "Penghapus"];
9
10         //Mengambil Hasil Array
11         document.write(products[1]);
12     </script>
13 </body>
14 </html>
```

Program keempat

```
1 <html lang="en">
2 <head>
3   <title>Mengambil data dari Array</title>
4 </head>
5 <body>
6   <script>
7     //Membuat Array
8     var products = ["Pensil", "Pulpen", "Penghapus", "Penggaris"];
9     document.write("<ol>");
10    //Menggunakan Perulangan untuk Mencetak Semua Isi Array
11    for(let i=0; i<products.length; i++){
12      document.write('<li>' +products[i] + '</li>');
13    }
14    document.write("</ol>");
15  </script>
16 </body>
17 </html>
```

Program kelima

```
1 <html>
2 <body>
3   <script language="JavaScript" type="text/javascript">
4     var angka;
5     while(angka != 5)
6     {
7       angka = prompt("Masukkan sebuah angka : ");
8     }
9     document.write("Anda telah memasukkan angka 5");
10  </script>
11 </body>
12 </html>
13
```

Program keenam

```
Welcome JS Scripts.js X index.html
JS Scripts.js > myFunction
1 var txt = "";
2 var numbers = [45, 4, 9, 16, 25];
3 numbers.forEach(myFunction);
4
5 function myFunction(value, index, array)
6   document.write(txt + value + "<br>");
7 }
```

Program ketujuh

```
Scripts.js
1  switch (new Date().getDay()) {
2      case 0:
3          console.log("Sunday");
4          break;
5      case 1:
6          console.log("Monday");
7          break;
8      case 2:
9          console.log("Tuesday");
10         break;
11        case 3:
12            console.log("Wednesday");
13            break;
14        case 4:
15            console.log("Thursday");
16            break;
17        case 5:
18            console.log("Friday");
19            break;
20        case 6:
21            console.log("Saturday");
22            break;
23        default:
24            console.log("WHERE ARE YOU LIVING!?");
25    }
26
```

Program kedelapan

```
Welcome  Scripts.js  X  index.html
Scripts.js
1  alert("Selamat datang di Javascript pertamaku");
```

Program kesembilan

```
Scripts.js > ...
1  var confirmation = confirm("Do you want to go to another world?");
2  if (confirmation) {
3      alert("You pick to go to another world received blessings from that world");
4  } else {
5      alert("You pick to stay on earth and survive to achieve your life goal");
6  }
```

Program kesepuluh

```
js Scripts.js > ...  
1  var age = prompt("How old are you");  
2  alert("Your are " + age + " years old");|
```

Program kesebelas

```
// jika menggunakan object  
var mahasiswa = {  
    nama : 'Zhafran Malik',  
    npm : 55412921,  
    kelas : '3IA14',  
    jurusan : 'Teknik Informatika',  
    IPSemester : [2.90, 3.10, 3.25, 2.88, 3.59],  
    IPKumulatif : function() {  
        var total = 0;  
        var ip = this.IPSemester;  
        for( var i = 0; i < ip.length; i++ ) {  
            total += ip[i];  
        }  
        return total/ip.length;  
    }  
}  
mahasiswa.IPKumulatif();
```

Program keduabelas

```
// membuat object mobil  
var mobil = {  
    nama : 'XPander',  
    merek : 'Mitsubitsi',  
    tipeMobil : 'Mini Jeep',  
    tahunDibuat : 2007,  
    warna : 'putih',  
    majuKencang : function () {  
        console.log('Mobil ini melaju dengan kecepatan 300Km/h');  
    },  
    majuPelan : function () {  
        console.log('Mobil ini melaju dengan kecepatan 10Km/h');  
    },  
    mundur : function () {  
        console.log('Mobil ini melaju kebelakang');  
    },  
    stop : function () {  
        console.log('Mobil ini berhenti');  
    }  
};
```

Program ketigabelas

C:\Download\Sublime Text\javascript\index.html - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

```
index.html x object.js x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Javascript</title>
6 </head>
7 <body>
8   <script src="object.js"></script>
9 </body>
10 </html>
11
```

C:\Download\Sublime Text\javascript\object.js - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

```
index.html x object.js x
1 // Membuat Object
2 // Object Literal
3 var mhs = {
4   nama : 'Muhammad Alif',
5   npm : '51627187',
6   email : 'alif@student.gunadarma.ac.id',
7   jurusan : 'Teknik Informatika'
8 }
9
10 var mhs2 = {
11   nama : 'Muhammad Daffa',
12   npm : '51624312',
13   email : 'daffa@student.gunadarma.ac.id',
14   jurusan : 'Sistem Informasi'
15 }
```

Program keempatbelas

```
// Function Declaration
function buatObjectMahasiswa(nama, npm, email, jurusan) {
  var mhs = {};
  mhs.nama = nama;
  mhs.npm = npm;
  mhs.email = email;
  mhs.jurusan = jurusan;
  return mhs;
}
```



```
var mhs3 = buatObjectMahasiswa('Nafeesha', 56172651, 'nafee@gmail.com', 'Psikologi')
```

Program kelimabelas

```
// Constructor
function Mahasiswa(nama, npm, email, jurusan) {
    this.nama = nama;
    this.npm = npm;
    this.email = email;
    this.jurusan = jurusan;
}
```

```
var mhs4 = new Mahasiswa('Nanda', 67389204, 'nanda@yahoo.com', 'Manajemen');
```

Program keenambelas

📁 C:\Download\Sublime Text\javascript\index.html - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

```
index.html x object.js x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Belajar Objek Javascript</title>
8     <script>
9         var orang = {
10             namaDepan : "Ahmad",
11             namaBelakang : "Agus",
12             tampilkanNama : function(){
13                 alert(`Nama : ${this.namaDepan} ${this.namaBelakang}`);
14             }
15         };
16
17         orang.tampilkanNama();
18     </script>
19 </head>
20 </html>
21
```