SEKOLAH FULLSTACK

COURSE 2

Belajar Dasar JavaScript (Sisi Frontend + Backend)

Material Details

Modul Sekolah Fullstack Cilsy

Penulis : Muhammad Lukman Hakim
Editor : Muhammad Fakhri Abdillah,Iqbal Ilman Firdaus

Penerbit : **PT. Cilsy Fiolution Indonesia**
Web Site : https://cilsyfiolution.com , https://sekolahfullstack.com

Daftar Isi

3.
Learn JavaScript: Hello JavaScript!

3.1. Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Mengatahui apa itu JavaScript.
2. Memahami penggunaan JavaScript.
3. Memiliki skill dalam menggunakan JavaScript.
4. Mengatahui cara penggunaan string methods
5. Mengetahui dan memahami penggunaan metode untuk manipulasi string
6. Mengatahui cara penggunaan Number Methods
7. Mengetahui cara penggunaan Array
8. Mengetahui jenis jenis metode pada array.
9. Mengetahui logika conditional statement pada JavaScript
10. Mengetahui logika perulangan (loop) pada JavaScript
11. Mengetahui DOM pada JavaScript

3.2. Outline Materi

1. Hello JavaScript
2. Penggunaan
3. Output
4. Statement

5. Variable
6. Operator
7. Arithmetic
8. Data Types
9. Function
10. Object
11. Array
12. Functions
13. Object
14. Event
15. String Methods
16. Number Methods
17. Array
18. Arrays Methods
19. Array Iteration
20. Conditional Statement
21. Loop
22. DOM

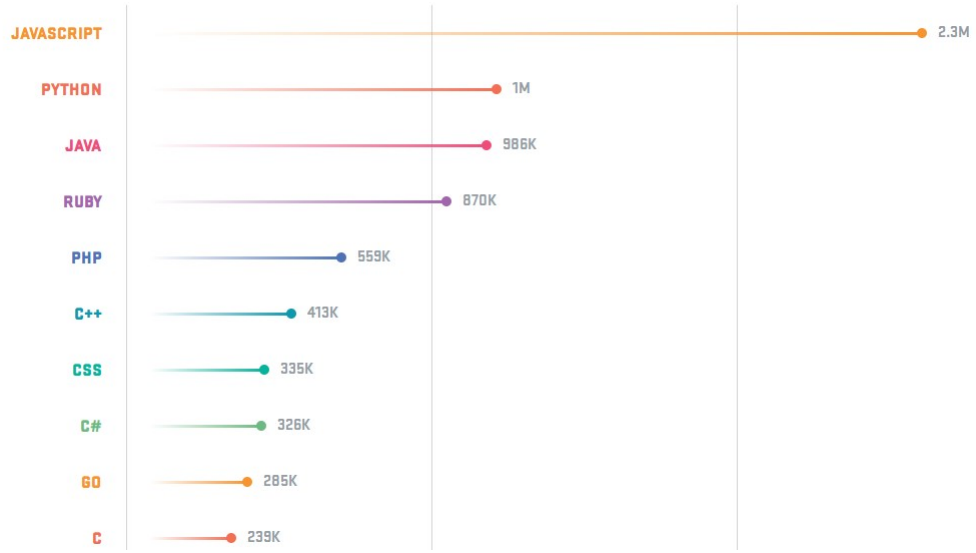
3.3. Hello JavaScript

JavaScript adalah bahasa pemrograman yang dapat digunakan pada HTML. JavaScript merupakan salah satu dari tiga bahasa yang perlu dikuasai oleh web developer karena dengan menggunakan JavaScript kita dapat memprogram tingkah laku dari sebuah halaman Web.

JavaScript dan Java adalah bahasa yang berbeda! Baik dari segi konsep maupun desain. JavaScript ditemukan oleh Brendan Eich pada tahun 1995.

Saat ini javascript tidak hanya digunakan di sisi client (browser) saja. Javascript juga digunakan pada server, console, program desktop, mobile, IoT, game, dan lain-lain.

Hal ini membuat javascript semakin populer dan menjadi bahasa yang paling banyak digunakan di Github.



Penggunaan JavaScript di Github

3.3.1. Apa itu JavaScript?

Javascript adalah bahasa pemrograman yang awalnya dirancang untuk berjalan di atas browser. Namun, seiring perkembangan zaman, javascript tidak hanya berjalan di atas browser saja. Javascript juga dapat digunakan pada sisi Server, Game, IoT, Desktop, dsb.

Javascript awalnya bernama **Mocha**, lalu berubah menjadi **LiveScript** saat browser Netscape Navigator 2.0 rilis versi beta (September 1995). Namun, setelah itu dinamai ulang menjadi Javascript.

Terinspirasi dari kesuksesan Javascript, Microsoft mengadopsi teknologi serupa. Microsoft membuat ‘Javascript' versi mereka sendiri bernama JScript. Lalu di tanam pada Internet Explorer 3.0.

Hal ini mengakibatkan ‘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. Saat ini ECMAScript sudah mencapai versi 8 (ES8).

Versi ECMAScript	Tahun Rilis

ES 1	Juni 1997
ES 2	Juni 1998
ES 3	Desember 1999
ES 4	Terbengkalai
ES 5	Desember 2009
ES 5.1	Juni 2011
ES 6	Juni 2015
ES 7	Juni 2016
ES 8	Juni 2017

3.3.2. Peralatan untuk Belajar Javascript

Apa saja perlatan yang harus disiapkan untuk belajar Javascript? "Alat Perang" yang perlu kita siapkan masih sama pada saat kita belajar HTML, yaitu:

1. Web Browser (Google Chrome, Firefox, Opera, dll)
2. Teks Editor

3.3.3. Membuat Hello JavaScript

Penggunaan JavaScript pada halaman HTML dapat ditulis secara inline yang artinya kode JavaScript dituliskan pada dokumen HTML dan disisipkan pada sebuah element. Berikut contoh kode JavaScript pada HTML:

```
<!DOCTYPE html>
<html>
```

```
    <body>
        <h2 id="demo">Apa yang dapat JavaScript lakukan?</h2>
        <button type="button"
onclick='document.getElementById("demo").innerHTML = "Hello
JavaScript!"'>
            Click Me!
        </button>
    </body>
</html>
```

Buat sebuah dokumen HTML dengan isi sama dengan contoh diatas dan simpan dalam format file HTML (.html).

Jika halaman HTML dibuka dengan web browser, maka akan keluar tulisan yang terdapat element <h2> dan sebuah button. Ketika button di click, button akan menjalankan metode yang sudah kita buat didalam fisrt tag nya yaitu "onclick". Konten dari onclick adalah kode JavaScript 'document.getElementById("demo").innerHTML = "Hello JavaScript!"' yang artinya JavaScript akan mencari element dengan id = "demo" menggunakan perintah (document.getElementById("demo").innerHTML) selanjutnya konten yang berada pada element yang terpilih akan diganti dengan value baru yang di definisikan (pada contoh ini konten yang berada pada element h2 dengan id = "demo" akan berubah menjadi "Hello JavaScript").

3.3.4. Exercise

Buatlah "hello world" menggunakan bahasa javascript yang disimpan pada file HTML.

3.4. Penggunaan

Terdapat tiga cara penggunaan javascript pada dokumen HTML yaitu, iniline, internal dan external. Namun biasanya penggunaan inline dihindari demi kemudahan dalam membaca kode untuk pengembangan maka dari itu inline diganti menjadi internal.

3.4.1. Internal

Penggunaan JavaScript secara internal dapat ditulis pada element <script>.

```
<script type="text/javascript">
    document.getElementById("demo").innerHTML = "Hello
JavaScript";
</script>
```

Element script dapat dimasukan pada element <head> atau <body>. Berikut contoh penempatan element <script> pada <head>:

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            function myFunction() {
                document.getElementById("demo").innerHTML = "Hello
JavaScript";
            }
        </script>
    <head>

    <body>
        <p id="demo">Text ini akan diganti</p>
```

```
        <button type="button" onclick="myFunction()">Klik
Bung</button>
    </body>
</html>
```

Berikut contoh penemapatan element <script>pada <body>:

```
<!DOCTYPE html>
<html>
    <head>
<head>
    <body>
        <p id="demo">Text ini akan diganti</p>

        <button type="button" onclick="myFunction()">Klik
Bung</button>

        <script type="text/javascript">
            function myFunction() {
                document.getElementById("demo").innerHTML = "Hello
JavaScript";
            }
        </script>
    </body>
</html>
```

3.4.2. External

JavaScript bisa juga ditempatkan secara external. Keunggulannya adalah satu kode dapat digunakan oleh beberapa halaman HTML. Dengan menggunakan element yang sama <script> namun dengan tambahan attribute 'src'.

```
<script src="myScript.js"></script>
```

```
<!DOCTYPE html>
<html>
  <head>
    <script src="myScript.js"></script>
  <head>

  <body>
    <p id="demo">Text ini akan diganti</p>
    <button type="button" onclick="myFunction()">Klik Bung</button>
  </body>
</html>
```

index.html

Penempatan element <script> sama dengan penggunaan JavaScript secara internal, dapat disimpan di element <head> atau element <body>. Selanjutnya kode JavaScript ditulis pada dokumen berbeda menggunakan nama yang sama dengan nama yang dituliskan pada attribute 'src'. Disimpan dengan extension file .js untuk menandakan bahwa dokumen tersebut adalah dokumen javascript.

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello JavaScript";
}
```

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello JavaScript";
}
```

myScript.js

Pada dokumen JavaScript tidak lagi perlu menuliskan element <script>, langsung saja tuliskan kode javascript yang akan digunakan.

3.4.3. Exercise

Buatlah latihan pertama (hello world) menggunakan teknik external!

3.5. Output

Output digunakan untuk mengembalikan nilai yang sudah selesai diproses yang akan ditampilkan pada halaman web. Terdapat empat cara ouput yang dapat digunakan, yaitu:

3.5.1. InnerHTML

Digunakan untuk mengakses element HTML, JavaScript dapat menggunakan document.getElementById(id) untuk memilih element mana yang akan dilakukan perubahan.

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Ini Judul</h1>
        <p>Ini paragraph</p>

        <p id="demo"></p>
        <script>
            document.getElementById("demo").innerHTML = 5 + 6;
        </script>
    </body>
</html>
```

3.5.2. Document.write()

Digunakan untuk menampilkan output pada documen html.

```
<!DOCTYPE html>
<html>
```

```
    <body>
        <h1>Ini Judul</h1>
        <p>Ini paragraph</p>

        <script>
            document.write(5 + 6);
        </script>
    </body>
</html>
```

Ingat! Dengan menampilkan output menggunakan document.write() setelah halaman HTML dimuat, makan document.write() akan menghapus semua HTML yang ada dan menulis output pada halaman HTML.

3.5.3. Window.alert()

Window.alert() digunakan untuk menampilkan output berupa alert box.

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Ini Judul</h1>
        <p>Ini paragraph</p>

        <script>
            window.alert(5 + 6);
        </script>
    </body>
</html>
```

3.5.4. Console.log()

Untuk mode debugging, output ditampilkan menggunakan console.log yang bisa dilihat pada developer tools web browser.

```
<!DOCTYPE html>
<html>
```

```
    <body>
        <h1>Ini Judul</h1>
        <p>Ini paragraph</p>

        <script>
            console.log(5 + 6);
        </script>
    </body>
</html>
```

3.6. Statement

Sebuah program komputer terbangun dari daftar instruksi yang di eksekusi oleh komputer. Pada bahasa pemrograman, instruksi yang ditulis dengan sebuah bahasa pemrograman disebut sebagai statement, sama halnya dengan program yang dibangun menggunakan bahasa javascript yang didalamnya berisi daftar statement menggunakan bahasa javascript. Pada HTML, program javascript di eksekusi oleh web browser.

Javascript statement terdiri dari value, operator, expression, keyword dan comment. Dalam proses eksekusinya, statement akan di eksekusi satu per satu sesuai dengan urutan penulisannya.

3.6.1. Semicolon

Semicolon digunakan untuk memisahkan statement pada javascript. Penempatan semicolon berada di akhir sebuah statement.

```
var a, b, c;

a = 5;
b = 10;
c = 15;
```

Karena terdapat karakter pemisah berupa semicolon maka penulisan beberapa statement dalam satu baris dapat dilakukan.

```
var a, b, c; a = 5; b = 10; c = 15;
```

Pada beberapa kasus mungkin terdapat contoh statement yang tidak menggunakan semicolon, namun direkomendasikan diberi semicolon sebagai pemisah agar mudah untuk di baca dikemudian hari.

3.6.2. White Space

Javascript akan mengabaikan beberapa space. Space ditambahkan untuk kemudahan dalam membaca kode. Berikut adalah contoh penggunaan white space yang tidak mempengaruhi performa javascript:

```
var name = "Lukman";

var name="lukman";
```

Dalam penulisan yang baik, berikan spasi diantara operator (= + - * /):

```
var x = y + z;
```

3.6.3. Line

Panjang satu line yang direkomendasikan adalah 80 karakter demi kemudahan dalam membaca kode yang ditulis. Jika statement sudah mendekati atau melebihi 80 karakter pada satu line maka cari operator terdekat dan berikan line break (line baru).

```
document.getElementById("demo").innerHTML =
"Hello Lukman";
```

3.6.4. Code Block

Statement javascript dapat dikelompokkan pada code block, didalam kurung kurawal { ... }. Penggunaan code block adalah untuk mendefinisikan bahwa statement yang ada didalam kurung kurawal akan dieksekusi secara bersamaan. Salah satu penggunaan javascript code block adalah saat pembuatan sebuah function().

```
function myFunction() {
    document.getElementById("hello1").innerHTML = "Hello Lukman";
    document.getElementById("hello2").innerHTML = "Hello there general!";
}
```

3.6.5. Keyword

Statement javascript dimulai dengan keyword untuk mengidentifikasi aksi apa yang akan dilakukan oleh javascript.

Kata yang digunakan untuk keyword tidak bisa digunakan sebagai nama sebuah variable.

3.6.6. Exercise

A. Diskusi

Carilah keyword javascript sebanyak-banyaknya dan diskusikan dengan mentor anda.

B. Latihan

Cobalah gunakan keyword sebagai nama variable dan amati apa yang terjadi!

3.7. Syntax

Javascript syntax adalah sebuah set aturan, bagaimana program javascript terbentuk:

```
var x, y, z;          // Bagaimana variable di deskripsikan
x = 5; y = 10;     // Bagaimana cara memberikan nilai pada variabel
z = x + y;            // Bagaimana cara menghitung nilai menggunakan variabel
```

Syntax pada javascript dibagi menjadi dua jenis, fixed value dan variable value. Fixed value biasanya disebut sebagai literal, dan variable value disebut sebagai variable.

3.7.1. Literal

Literal merupakan fixed value atau dengan kata lain tidak bisa diubah atau diganti. Hal penting yang perlu diperhatikan saat menulis literal adalah:

1. Nomor ditulis menggunakan atau tidak menggunakan desimal.
2. String (text) ditulis menggunakan single quotes atau double quotes.

```
<!DOCTYPE html>
<html>
    <body>
        <p id="testNumber"> </p>
        <p id="testString></p>

        <script>
           document.getElementById("testNumber").innerHTML =
10.50;
           document.getElementById("testString").innerHTML =
"Hello Lukman";
        </script>
    </body>
</html>
```

3.7.2. Variabel

Pada bahasa pemrograman, variable digunakan untuk menyimpan nilai. Pada Javascript digunakan var untuk keyword sebagai sebuah deklarasi untuk variable. Tanda sama dengan digunakan sebagau assignment. Variable adalah kontainer untuk menyimpan nilai data. Contohnya x, y dan z adalah sebuah variable:

```
var x = 5;
var y = 10;
var z = x + y;
```

Dari contoh diatas dapat dijelaskan:

- x menyimpan nilai 5
- y menyimoan nilai 10
- z menyimpan nilai 15

Seperti sebuah perhitungan aljabar:

```
// menghitung luas segi tiga siku-siku
var tinggi = 10;
var alas = 20;
var luas = 0.5 * alas * tinggi;
```

3.7.3. Expression

Sebuah expression adalah kombinasi dari value, variable, dan operator yang menghitung sebuah nilai. Perhitungannya disebut evaluation. Contohnya 5 * 10 di evaluation menjadi 50:

```
5 * 10
```

Expression dapat berisi nilai variable:

```
x * 50
```

Nilai dapat berupa berbagai bentuk seperti number atau string. Contohnya:

```
"Lukman" + " " + "Hakim"
```

akan di evaluate menjadi "Lukman Hakim".

3.7.4. Comment

Tidak semua javascript akan dieksekusi. Kode yang ditulis setelah tanda double slash // atau diantara /* dan */ di sebut sebagai comment atau komentar. Comment tidak akan dianggap dan tidak akan di eksekusi.

3.7.5. Identifier

Identifier adalah nama. Pada JavaScript, identifier adalah nama dari sebuah variable. Aturan dari penamaan pada javascript hamper sama dengan aturan penamaan pada bahasa pemrograman lainnya. Pada javascript, karakter pertama haruslah berupa huruf atau underscore _, atau dollar sign $.

3.7.6. Case Sensitive

Semua identifier javascript bersifat case sensitive. Artinya untuk mendapatkan nilai yang sama, kedua identifier haruslah sama persis. Contoh:

```
var nama, Nama;
nama = "Lukman";
Nama = "Hakim";
```

Variabel name dan Name adalah dua variable yang berbeda.

Pada kasus penulisan keyword (contohnya keyword yang digunakan untuk membuat variable), javascript tidak akan menginterpretasikan VAR dan Var sebagai var.

3.7.7. Menghubungkan Kata

Sebenarnya, beberapa programmer memiliki berbagai cara untuk menghubungkan beberapa kata menjadi nama variable.

1. Hyphens

Menggunakan strip (-) sebagai penghubung. first-name, last-name.

2. Underscore

Menggunakan underscore (_) sebagai penghubung. first_name, last_name.

3. Upper Camel Case

Menggunakan huruf capital diawal kata. FirstName, LastName.

4. Lower Camel Case

Menggunakan huruf kecil di kata pertama dilanjutkan huruf besar pada kata berikutnya. firstName, lastName.

Dari ke-empat cara untuk menghubungkan nama, programmer javascript memiliki kecenderungan untuk menulis dengan menggunakan **Lower Camel Case.**

3.8. Tipe Data

Variable javascript dapat menyimpan beberapa jenis tipe data: number, string, object dan berbagai macam lagi.

```
var length =16;                                    // Number
var lastName ="Hakim";                             // String
var x = {firstName: "John", lastName: "Doe"};      // Object
```

Pada bahasa pemrograman, tipe data ada konsep utama. Agar dapat digunakan pada variable, sangat penting untuk mengetahui tentang tipe data yang hendak digunakan. Tanpa tipe data, sebuah komputer tidak dapat menyelesaikan permasalahan seperti dibawah dengan baik:

```
var x = 16 + "Volvo";
```

3.8.1. Tipe JavaScript Bersifat Dinamis

Javscript memiliki tipe yang berfisat dinamis. Yang artinya satu varibel dapat digunakan untuk menyimpan tipe data yang berbeda.

```
var x;                 // Sekarang x adalah undifined
x = 5;                // Sekarang x adalah number
x = "John";           // Sekarang x adalah string
```

3.8.2. String

String atau text string adalah sebuah serial dari karakter seperti "Lukman Hakim". String ditulis menggunakan quotes. Quotes yang digunakan dapat berupa single atau double. String digunakan untuk menyimpan dan memanipulasi text.

```
var mobil1 = "Lada"; // Menggunakan double quotes
var mobil2 = ‘Lada'; // Menggunakan single quotes
```

Anda dapat menggunakan quote didalam string, selama quote yang digunakan tidak sama dengan quote yang digunakan untuk membungkus string.

```
var hari1 = "Sekarang hari jum'at";    // Penggunaan single quote
didalam double quote

var hari2 = "Sekarang hari ‘libur'";     // Penggunaan single
quote didalam double quote

var hari3 = 'Sekarang hari "santuy"'; // Penggunaan double quote
didalam single quote
```

3.8.3. Escape Character

Bagaimana jika ingin menggunakan double quotes didalam double quotes atau single quotes didalam single quotes?

```
var x = "Aku seorang "kapiten" yang mempunyai pedang panjang.";
```

String akan terpotong menjadi "Aku seorang" dan tidak dilanjutkan. Dalam kasus seperti ini digunakan escape character dengan bantuan backslash (\ ). Dengan backslash special karakter akan diubah menjad string biasa.

```
var x = "Aku seorang \"kapiten\" yang mempunyai pedang panjang.";
```

```
var x = ‘Aku seorang \'kapiten\' yang mempunyai pedang panjang.';
```

Dengan kondisi seperti diatas, maka variable x akan bernilai:

```
Aku seorang "kapiten" mempunyai pedang panjang.
```

Tidak terputus, sama halnya dengan penggunaan single quote.

Length

Cara untuk mendapatkan berapa panjang atau berapa karakter yang ada dalam sebuah variable string, dapat digunakan property length.

```
var name = "Lukman";
name.lenght;                          // return 6
```

3.8.4. Number

Javascript hanya memiliki satu tipe number. Number dapat ditulis menggunakan decimal atau tanpa decimal.

```
var x = 34.00;  // Ditulis menggunakan desimal (separator berupa titik)
var y = 34;       // Ditulis tanpa desimal
```

Penomoran yang besar atau yang kecil dapat ditulis menggunakan scientific (exponential) notation.

```
var x = 123e5;    // 12300000
var y = 123e-5; // 0.00123
```

3.8.5. Boolean

Boolean hanya memiliki dua nilai yaitu true atau false. Boolean sering digunakan untuk conditional testing.

```
var x = 5;
var y = 10;
```

```
var z = y - x;

(x == y)      // return false
(x == z)      //return true
```

3.8.6. Array

Array adalah kumpulan nilai yang digabungkan menjadi satu. Pada Javascript array ditulis berada didalam kurung siku [...] dengan separator berupa koma untuk membedakan nilai satu dengan lainnya.

```
var cars = ["Lada", "Volvo", "Scania"];
```

3.8.7. Object

Object pada javascript ditulis menggunakan kurung kurawal { ... }. Object memiliki properties yang ditulis dengan pasangan nama: nilai, dan dipisahkan oleh koma untuk setiap datanya.

```
var person = {firstName: "Lukman", lastName: "Hakim", age: 25};
```

3.8.8. Typeof

Untuk mengetahui tipe dari variable di javascript dapat menggunakan oprator typeof. Typeof akan mengembalikan nilai tipe dari sebuah variable atau expression.

```
typeof "";                  // return string
typeof "Lukman";    // return string
typeof 0;                    // return number
typeof (3 + 4);           // return number
```

3.8.9. Undefined

Pada bahasa pemrograman javascript, sebuah variable yang tidak memiliki nilai akan diberi nilai default "undefined". Tipe datanya juga akan "undefined".

```
var car;    // return undefined
```

Karena variable hanya di deklarasikan dan tidak di assign value, maka otomatis nilai yang akan di return adalah undefined.

3.8.10. Empty Value

Berbeda dengan undefined, empty value masih memilki nilai hanya kosong. Contohnya:

```
var car = "";    // nilainya adalah "", tipe datanya string.
```

3.8.11. Null

Null berarti "tidak ada" pada javascript menunjukan sesuatu yang tidak ada. Sayangnya pada javascript, tipe data dari null adalah object.

```
var person = {firstName: "Lukman", lastName: "Hakim", age: 25};
person = null;     // akan mengembalikan nilai null dengan tipe
data object
```

3.8.12. Perbedaan Null Dan Undefined

Secara tipe data undefined berbeda dengan null, namun secara nilai undefined sama dengan null.

```
typeof undefined;     // undefined
typeof null;                // object

null === undefined   // false
null == undefined     // true
```

3.9. Function

Pada bagian ini kita akan belajar function. Nah apa itu function? Function adalah sebuah blok kode yang didalamnya berisi beberapa statement, tujuannya adalah menjalankan statement yang terkandung didalamnya secara bersamaan untuk mencapai suatu hasil.

Masih bingung?

Mari kita lihat contoh berikut.

```
// Menghitung luas lingkaran

function luasLingkaran(diameter) {
  var r = diameter / 2;
  var cekJari = r % 7;

  if(cekJari == 0) {
    var phi = 22/7;
  } else {
    var phi = 3.14;
  }

  var luas = phi * r * r;

  return luas;
}
```

Ini merupakan contoh sebuah function yang menghitung luas dari sebuah lingkaran dimana masukannya berupa nilai dari diameter lingkarannya. Okay kita kembali ke penjelasan sebelumnya yang menyatakan bahwa function adalah sekumpulan statement yang akan dijalankan bersamaan pada satu waktu. Pada function luasLingkaran() terdapat 9 statement didalamnya. Nah semuanya akan dieksekusi ketika function berjalan untuk mencapai suatu hasil yaitu berapa nilai luas lingkaran yang di representasikan pada variable luas.

Kita coba jalankan function dengan statement berikut:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>
function luasLingkaran(diameter) {
  var r = diameter / 2;
  var cekJari = r % 7;

  if(cekJari == 0) {
    var phi = 22/7;
  } else {
    var phi = 3.14;
  }

  var luas = phi * r * r;

  return luas;
}
```

```
document.getElementById("demo").innerHTML = luasLingkaran(14);
</script>

</body>
</html>
```

Pada line :

```
document.getElementById("demo").innerHTML = luasLingkaran(14);
```

function yang kita buat dipanggil dengan diameter 14. Selanjutnya hasil dari function luasLingkaran akan ditampilkan pada element dengan id “demo”.

Nah function kita dipanggil dan selanjutnya akan menjalankan statement yang terkandung didalamnya dengan urutan sebagai berikut:

a. Akan ditentukan nilai dari variable r dengan rumus diameter dibagi menjadi 2.

b. Akan ditentukan nilai dari variable cekJari dengan rumus r % 7. % disini artinya adalah operator modulus. Dimana jika nilai r bekelipatan 7 (karena rumusnya r modulus 7) maka hasilnya akan 0 dan jika bukan kelipatan 7 maka hasilnya bukan 0.

c. Selanjutnya menentukan nilai phi namun kita cek dulu apakah variable r merupakan kelipatan 7 atau bukan karena berpengaruh pada penentuan nilai phi. Dengan menggunakan bantuan variable cekJari kita gunakan percabangan IF. Jika cekJari bernilai sama dengan 0 (artinya r berkelipatan 7) maka nilai phi yang kita gunakan 22/7 sedangkan kalua bukan (else) maka nilai ph adalah 3.14.

d. Selanjutnya kita hitung variable luas dengan memasukan rumus luas lingkaran, yaitu phi * r * r.

e. Langkah terakhir kita definisikan return untuk mengembalikan hasil dari apa yang sudah dikerjakan pada sebuah function. Disini kita melakukan return terhadap nilai variable luas yang merepresentasikan nilai luas lingkaran.

Cukup mudah kan penggunaan function ini, nah pertanyaan selanjutnya adalah kenapa sih kita membuat sebuah function? Kenapa tidak langsung saja menulis rumus?.

Oke, bayangkan jika kamu memiliki sebuah website yang dapat menghitung 5 luas lingkaran dalam satu waktu. Artinya kamu memiliki 5 buah input, 5 buah rumus dan 5 buah output. Kita hitung saja penggunaan rumusnya. Jika satu rumus memiliki 9 baris kode, kamu perlu menulis sekitar 45 baris kode (5 buah rumus * 9 baris setiap rumus). Bandingkan jika kamu menggunakan function! Kamu cukup menulis satu function lalu memanggilnya 5 kali dan dapat disatukan dengan baris kode output, artinya kamu hanya perlu menulis 11 baris saja! (9 baris rumus + 1 baris initial function + 1 baris untuk return nilai). Sangat efisien bukan!

Jadi dengan penggunaan function ini kamu dapat menulis hanya satu kali saja, selanjutnya tinggal panggil function yang sudah dibuat. Hasilnya akan berbeda jika nilai parameter yang kalian masukan berbeda. **Resuable**.

Yang perlu digaris bawahi, ketika kalian menulis sebuah function pada javascript namun kalian tidak memanggilnya maka semua baris statement yang ada didalam function tidak akan dieksekusi oleh komputer.

3.9.1. Syntax

Sebuah function dalam javascript diawali dengan keyword function, diikuti dengan nama function nya dan kurung buka-tutup atau parentheses (). **Penamaan function memiliki aturan yang sama dengan aturan penamaan variable.** Didalam parentheses dapat diisi parameter. Jumlah parameter yang disimpan pada parentheses dapat lebih dari satu tergantung dengan kebutuhan. Pemisah antara satu parameter dan parameter lainnya menggunakan koma (,):

```
(paremeter1, parameter2, ...)
```

blok kode yang hendak dijalankan disimpan didalam kurung kurawal {}. Jadi bentuk umum function adalah:

```
function name(parameter1, parameter2, parameter3) {
  // kode yang akan dijalankan.
}
```

3.9.2. Invokes & Return

Kode yang berada didalam sebuah function akan dieksekusi ketika ada yang melakukan invokes (memanggil) function tersebut.

Sebuah function akan berhenti memproses ketika mencapai statement return. Apasih return itu? Secara harfiah return adalah pengembalian, lalu apa yang dikembalikan dan kepada siapa kembalian itu diberikan?

Mari kita analogikan, terdapat dua orang yang bernama Bambang dan Budi.

“Ketika Bambang memanggil Budi, Budi merespon panggilan Bambang dengan melambaikan tangan.”

Dari analogi yang ada, diketahui bahwa terdapat tiga kondisi dasar:

a. Bambang memanggil Budi.

b. Budi menerima panggilan Bambang.

c. Budi memberikan respon dengan melambaikan tangan.

Ketiga kondisi dasar ini berkaitan dengan function pada javascript. Kondisi pertama saat Bambang memanggil Budi itu sama dengan invokes (memanggil) function yang ada. Kodisi kedua saat Budi menerima panggilan Bambang adalah proses berjalannya function dan kodisi ketika ketika Budi memberikan respon dengan melambaikan tangan sama dengan return atau pengembalian terhadap input yang diberikan.

Jadi return akan dikembalikan kepada pemanggilnya atau siapa yang melakukan invokes dan kembaliannya berupa hasil output dari function yang dipanggil.

3.9.3. Parameter

Pada bagian pertama kalian pasti sudah mendengan parameter pada function. Parameter pada function ditujukan untuk menerima masukan yang diberikan oleh yang memanggil function.

Oke begini, kita ambil lagi contoh pada bahasan kita mengenai luasLingkaran().

```
// Menghitung luas lingkaran
function luasLingkaran(diameter) {
  var r = diameter / 2;
  var cekJari = r % 7;

  if(cekJari == 0) {
    var phi = 22/7;
  } else {
    var phi = 3.14;
  }

  var luas = phi * r * r;

  return luas;
}
```

Kita lihat line pertama:

```
function luasLingkaran(diameter) {
```

Terdapat parameter yang bernama diameter, untuk penamaan parameter aturannya sama dengan aturan penamaan variable. Kalian bebas memberi nama apa saja untuk parameter namun untuk mempermudah dalam menulis kode berikanlah nama parameter yang relevan dengan apa yang akan kita kerjakan. Yang perlu diperhatikan selain nama parameter adalah jumlah parameter yang hendak digunakan.

Pada kondisi seperti ini, **ketika function tidak ada yang memanggil, nilai dari diameter adalah undefined.**

Oke, kita lihat bagaimana cara kita memanggil function luasLingkaran():

```
document.getElementById("demo").innerHTML = luasLingkaran(14);
```

Kita ambil inti pemanggilan functionnya saja:

```
luasLingkaran(14)
```

Pada saat memanggil function yang kita miliki, terdapat angka 14 pada parenthesis nya. Dengan demikian nilai dari parameter yang bernama diameter adalah 14.

Bagaimana jika terdapat dua atau lebih parameter?

Function luasLingkaran(diameter, jari) {

Kita asumsikan terdapat sebuah function dengan dua parameter diameter dan jari (untuk jari-jari). Bagaimana cara memanggilnya dan berapa nilai dari masing-masing parameter?

luasLingkaran(14, 7)

Pada saat pemanggilan function, berikan dua nilai atau berikan nilai sesuai dengan jumlah parameter yang ada pada definisi function dengan pemisah antara nilai berupa karakter koma (,). Dengan kondisi seperti ini maka nilai parameter diameter adalah 14 dan parameter jari adalah 7. Jadi pemberian nilai pada parameter akan sama persis seperti urutan parameternya.

Bagaimana jika salah satu dari dua parameter yang dibutuhkan tidak di definisikan?

Misal kita panggil function seperti ini: luasLingkaran(14)

Maka nilai parameter diameter akan menjadi 14 dan nilai parameter jari akan tetap undefined yang menyebabkan hasil perhitungan menjadi ‘NaN’ atau ‘Not a Number’ karena terjadi operasi antara number dan undefined.

Parameter tidak melakukan pengecekan tipe data terhadap apa yang dimasukan saat pemanggilannya. Jadi pastikan apa yang dimasukan saat pemanggilan function relevan dengan apa yang dibutuhkan function.

Kembali lagi pada function luasLingkaran(). Jika kita memanggil function tersebut dengan cara:

```
luasLingkaran(‘Budi’)
```

Maka hasil yang keluar adalah 'NaN' singkatan dari 'Not a Number'. Kenapa karena terjadi operasi bilangan dimana string di proses bersama number.

3.9.4. Latihan

Buatlah beberapa function yang dapat menghitung:

1. Luas Lingkaran.
2. Luas Persegi.
3. Luas Segi Tiga.
4. Volume Kubus.
5. Volume Tabung.

3.10. Object

Untuk memahami apa itu object mari kita analogikan pada kehidupan nyata: Mobil adalah object. Mobil memiliki properties seperti berat dan warna, dan mobil memiliki methods seperti nyala atau mati.

Properties	Methods
mobil.nama : Tesla	mobil.start()
mobil.model: 3	mobil.drive()
mobil.warna: merah	mobil.stop()

Semua mobil dapat memiliki properties yang sama namun dengan value yang berbeda. Semua mobil memiliki methods, tetapi method itu dijalankan berbeda setiap waktu.

Kita sudah pernah belajar variable javascript, variable adalah sebuah tempat atau container yang dapat memuat data values. Untuk membuat sebuah variable yang menyimpan nama mobil kalua kalian masih ingat bentuk kodenya akan seperti berikut:

```
var mobil = "Tesla";
```

Objek adalah variable, tetapi object dapat menyimpan lebih dari satu value. Contohnya kita akan memasukan detail dari mobil pada satu object mobil:

```
var mobil = {nama: “Tesla”, model: 3, warna: “Merah”}
```

Value pada object ditulis berpasang-pasangan (name:value) dan dipisahkan menggunakan titik dua (:).

Okay jadi sampai disini kita sudah bisa menganalogikan sebuah object, menentukan properties dari object yang kita miliki hingga object tersebut memiliki methods apa saja. Ya sederhananya object adalah variable yang dapat menampung banyak values. Baik kita lanjutkan ke bahasan teknis dari object.

3.10.1. Membuat Object

Pada bahasa pemrograman javascript, kita dapat membuat object kita sendiri. Ada dua cara untuk membuat object, yang pertama menggunakan object literal, dan yang kedua menggunakan keyword new.

3.10.2. Object Literal

Cara termudah untuk membuat sebuah object adalah menggunakan object literal. Dengan object literal kita bisa membuat dan mendefinisikan sebuah object hanya dengan satu statement. Object literal adalah daftart dari pasangan name:values didalam kurung kurawal {...}.

```
var mobil = {nama: “Tesla”, model: 3, warna: “Merah”}
```

Pada penulisannya kita tetap perlu memperhatikan tipe data apa yang kita gunakan pada valuenya. Contoh pada properties nama kita menuliskan Tesla didalam petik dua yang artinya properties akan memiliki **value berupa string**. Berbeda dengan properties model yang value nya berupa angka 3 saja tanpa petik, tujuannya membuat **properties memiliki value number**.

Penggunaan spasi dan line breaks tidak berpengaruh, artinya kita bisa mendefinisikan object dengan bentuk seperti ini:

```
var mobil = {
```

```
  nama: “Tesla”,
  model: 3,
  warna: “Merah”
}
```

Tujuannya agar mudah dalam membaca properties dan value nya ketika dalam satu object menampung banyak properties.

3.10.3. Keyword New

Membuat object javascript menggunakan keyword new akan menjadi seperti berikut:

```
var mobil = new Object();
mobil.nama = “Tesla”;
mobil.model = 3;
mobil.warna = “Merah”;
```

Pertama-tama kita buat variable mobil dengan value “new Object()” dengan begitu tipe data untuk variable mobil adalah object.

Untuk memasukan name dan value pada object mobil bisa dilihat pada line selanjutnya:

```
mobil.nama = “Tesla”;
```

Tulis nama object diikuti tanda titik (.) lalu property name. Selanjutnya assign value pada object. Secara penulisan object literal lebih banyak digunakan karena lebih praktis dibandingkan penggunaan keyword new dalam pembuatan object.

3.10.4. Mengakses Object

Kita sudah mengetahui bentuk object dan bagaimana cara membuatnya, kali ini kita akan mengakses object yang kita miliki. Tujuannya tentu saja untuk memanfaatkan data yang tersimpan didalamnya.

Dengan menggunakan contoh sebelumnya, kita memiliki object mobil. Bagaimana caranya kita mendapatkan nama mobil?

Terdapat du acara untuk melakukan akses terhadap object. Cara pertama:

```
objectName.propertyName
```

Cara kedua adalah:

```
objectName[propertyName]
```

Okay objectName yang kita miliki adalah “mobil” lalu karena kita ingin mendapatkan namanya maka property yang kita ambil adalah “nama”. Jadi untuk mendapatkan nama dari object mobil adalah:

mobil.nama; **atau** mobil[“nama”];

```
var mobil = {
  nama: "Tesla",
  model: 3,
  warna: "Merah"
};

document.getElementById("demo").innerHTML = "Nama mobil pertama
saya adalah " + mobil.nama;
```

3.10.5. Menambah Property

Untuk menambahkan property pada sebuah object kita perlu mengetahui object apa yang kita miliki. Kembali kita gunakan contoh object mobil:

1. Ketahui nama object yang akan ditambahkan property nya. (mobil)
2. Ketahui value yang akan ditambahkan dan berikan nama property yang relevan. (pada contoh ini kita akan menambahkan kapasitas penumpang)
3. Tuliskan nama obect diikuti nama property dengan pemisah berupa titik (.).
4. Masukan value dengan menggunakan assignment operator.

Maka bentuk kode untuk mamasukan property baru pada object adalah sebagai berikut:

```
mobil.kapasitas = 4;
```

3.10.6. Mengubah Value Property

Sama halnya dengan menambahkan property pada sebuah object namun bedanya dalam pengubahan kita perlu mengetahui apakan property yang akan kita ubah sudah ada dalam object yang kita miliki. Jadi bahasa teknisnya pengubahan value property pada object adalah bentuk re-assignment value terhadap existing property.

Sebagai contoh kita gunakan object mobil yang sebelumnya sudah ditambahkan property kapasitas. Kita akan mengubah kapasitas yang semula 4 menjadi 6 dengan kode:

```
mobil.kapasitas = 6;
```

Sama persis bukan kodenya dengan yang digunakan untuk menambahkan property? Hanya saja ketika penambahan maka property sebelumnya belum ada pada object dan ketika pengubahan property sudah ada pada object namun kita re-assign valuenya.

3.10.7. Menghapus Property

Untuk menghapus property yang sudah ada pada object yang kita miliki dapat menggunakan keyword ‘delete’. Sama halnya seperti mengubah, kita perlu tau property apa yang akan kita hapus. Penghapusan akan menghilankan property dan valuenya, sedangkan object tidak akan terhapus.

Contohnya kita akan menghapus property kapasitas pada object mobil dengan kode:

```
delete mobil.kapasitas;
```

Keyword delete ini hanya berfungsi untuk property object dan tidak memiliki efek pada variable dan function.

3.10.8. Latihan

Buatlah property minimal 5 dari object berikut:

1. Person
2. Car
3. Animal
4. Computer
5. Student

3.11. Event

Event adalah sesuatu yang terjadi pada element HTML. Ketika JavaScript diterapkan pada sebuah halaman HTML maka apabila terjadi sebuah event, JavaScript akan melakukan sebuah reaksi.

Pernah tidak kalian mengakses sebuah website dan ketika melakukan klik tombol lalu muncul sebuah pop up?

Itu adalah contoh dari event! Saat element button di klik maka JavaScript akan melakukan reaksi dalam bentuk menampilkan pop up dan metode yang digunakan disebut onclick.

Contoh event HTML yang dapat kita manfaatkan adalah ketika halaman selesai di muat, ketika form masukan berubah atau ketika tombol di klik seperti pada contoh sebelumnya.

Jadi tujuan utama dari event ini adalah mengeksekusi sesuatu ketika terjadi sebuah event yang membuat halaman web yang kita miliki lebih aktif dan responsive terhadap apa saja yang dilakukan oleh penggunaan.

Penggunaannya pun cukup mudah, kita bisa menambahkan property event pada element HTML.

```
<element event="kode JavaScript">
```

Okay kita buat sebuah button yang ketika di klik akan menampilkan tanggal menggunakan metode onclick.

```
<!DOCTYPE html>
<html>
   <body>

    <button
onclick="document.getElementById('demo').innerHTML=Date()">Lihat
Waktu</button>

    <p id="demo"></p>

  </body>
</html>
```

Kita tambahkan event onclick pada element button. Ketika terdapat event klik pada element button maka script yang berada pada event akan dieksekusi. Pada contoh ini kita menampilkan tanggal pada element paragraph dengan id demo.

Contoh yang sama namun menggunakan function:

```
<!DOCTYPE html>
  <html>
    <body>

    <button onclick="displayDate()">Lihat Waktu</button>

    <script>
      function displayDate() {
        document.getElementById("demo").innerHTML = Date();
      }
    </script>
```

```
    <p id="demo"></p>

  </body>
</html>
```

Selain mengeksekusi script yang menjadi value dari event, kita juga dapat memanggil function seperti contoh diatas.

Beberapa event yang sering digunakan pada halaman web adalah:

Event	Deskripsi
onchange	Ketika terjadi perubahan pada sebuah element
onclick	Ketika pengguna melaukan klik
onmouseover	Ketika pengguna melakukan mouse over
Onmouseout	Ketika pengguna melakukan mouse out
onkeydown	Ketika pengguna menekan tombol pada keyboard
onload	Ketika browser selesai melakukan load halaman HTML

3.11.1. Latihan

Dengan menggunakan latihan function sebelumnya (3.4.7), buatlah function tersebut dapat berjalan ketika terdapat event:

1. Onclick
2. Onload

3.12. String Methods

String method akan membantu kita dalam bekerja untuk mengolah tipe data string. Nilai primitive seperti string tidak memiliki properties atau method. Tetapi menggunakan string method javascript akan memperlakukan value primitive menjadi object, tujuannya agar kita dapat memanipulasi atau mendapatkan detail dari string yang ada.

3.12.1. String Length

Kita akan mencari panjang sebuah string dengan menggunakan property length. Penggunaanya dengan cara menuliskan variable string yang kita akan cari panjangnya lalu beri tanda titik (.) dan tambahkan property length:

```
var txt = “lukman hakim”;
var hasil = txt.length; // akan menghasilkan 12
```

Property length akan menghitung setiap karakter yang ada pada variable yang dihitung. Selain karakter, tanda baca pun akan ikut terhitung, contohnya “lukman hakim” memiliki panjang 12.

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo"></p>
    <script>
      var txt = "lukman hakim";
      document.getElementById("demo").innerHTML = txt.length;
    </script>
  </body>
</html>
```

3.12.2. Find String in a String

Kita dapat mencari posisi string pada sebuah string! Maksudnya mencari posisi sebuah kata dari sebuah kalimat. Terdapat dua method untuk mencari indexOf() dan lastIndexOf().

indexOf() method mengembalikan nilai index dari posisi karakter pertama dari string pertama yang sesuai dengan apa yang dicari. Contohnya:

```
<!DOCTYPE html>
  <html>
    <body>

    <p id="demo"></p>

    <script>
      var str = "lukman hakim";
      var pos = str.indexOf("hakim");
      document.getElementById("demo").innerHTML = pos;
    </script>
  </body>
</html>
```

Hasilnya akan menampilkan angka 7. Bagaimana cara kerjanya?

Aturan yang perlu diingat adalah karakter pertama dari string dihitung dari mulai angka 0. Contohnya “lukman” karakter “l” adalah karakter ke-0 dan seterusnya hingga karakter “n” adalah karakter ke-5.

Kembali ke contoh awal, yang kita cari adalah posisi kata “hakim” maka yang keluar adalah nilai index 7 dimana 7 adalah index dari karakter “h” yang merupakan karakter awal dari kata “hakim”.

Bagaimana jika terdapat dua kata “hakim”?

```
var str = “lukman hakim bukan lah hakim”;
```

Jika menggunakan indexOf() maka index yang keluar adalah 7, mengapa? Karena yang dicari adalah kata “hakim” pertama dari sebuah kalimat.

lastIndexOf() method mengembalikan nilai index dari posisi karakter pertama dari string terakhir dari apa yang dicari.

Jika kita menggunakan contoh yang pertama untuk mencari string “hakim” pada kalimat “lukman hakim” maka nilai index yang di return akan sama yaitu 7. Namun ketika kita menggunakan contoh yang kedua hasilnya akan berbeda.

```
<!DOCTYPE html>
  <html>
    <body>

    <p id="demo"></p>

    <script>
      var str = "lukman hakim bukan lah hakim";
      var pos = str.lasIndexOf("hakim");
      document.getElementById("demo").innerHTML = pos;
    </script>
  </body>
</html>
```

Jika kode diatas menggunakan string method indexOf() maka return index akan bernilai 7 karena karakter “h” yang diambil indexnya berasal dari kata yang pertama. Sedangkan jika kita menggunakan string method lasIndexOf() maka return index akan bernilai 23 karena karakter “h” dari kata “hakim” yang cocok diambil dari posisi yang paling terakhir.

Sebanyak apapun kata “hakim” yang ada pada sebuah kalimat jika kita menggunakan indexOf() maka yang diambil adalah kata pertama, dan lastIndexOf() akan mengambil kata terakhir.

Bagaimana jika kata yang dicari tidak ada pada string yang digunakan?

```
<!DOCTYPE html>
  <html>
    <body>

    <p id="demo"></p>
```

```
    <script>
      var str = "lukman hakim bukan lah hakim";
      var pos = str.indexOf("john");
      document.getElementById("demo").innerHTML = pos;
    </script>
  </body>
</html>
```

Baik method indexOf() ataupun lastIndexOf() akan melakukan return index dengan nilai -1 jika kata yang dicari tidak ditemukan.

3.12.3. Slice

Slice adalah method untuk string yang digunakan untuk slicing atau memotong. Slice akan mengekstraksi string dan melakukan return berupa string baru hasil ekstraksi. Untuk menggunakan slice diperlukan 2 parameter, yang pertama adalah index posisi karakter yang akan dipotong dan yang kedua adalah index terakhir dari karakter yang akan dipotong ditambahkan satu. Contohnya:

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

Variable res akan melakukan return dengan value “Banana”, kenapa? Bagaimana cara kerjanya?

Oke Kita lihat pada variable res. Untuk menggunakan slice, pertama-tama Kita perlu menuliskan nama variable yang valuenya berupa string, pada kasus yang kita milki adalah variable str. Penulisannya menjadi:

```
str.slice()
```

Berikutnya kita masukan dua parameter yang digunakan oleh slice yaitu index karakter pertama dan index karakter terakhir plus satu yang akan Kita ekstrak. Kali ini kita akan mengekstrak kata “Banana”. Index karakter pertama pada kata “Banana” adalah 7 dan index karakter terakhir ya adalah 12. Selanjutnya index karakter terakhir kita +1 menjadi 13,

mengapa? Agar karakter yg di potong adalah karakter setelah huruf “a” terakhir milik “banana”. Jadi penulisan slice akan menjadi seperti berikut:

```
str.slice(7, 13)
```

Bagaimana jika parameter diisi angka negatif? Slice akan melakukan pemotongan dengan menghitung nomor index dari belakang. Dengan contoh yang sama Kita akan mengambil string “Banana”.

Langkah pertama kita hitung karakter pertama “Banana” dari belakang atau dari sebelah kiri. Index karakter pertama adalah 11 dari sebelah yang artinya kita perlu menulisnya dalam bentuk negatif yaitu -11. Karena kita ingin “B” pada “Banana” terambil valuenya maka kita geser index nya dengan melakukan – 1 dari index asalnya. Jadi kita dapat -13 untuk posisi pertama yang akan di potong. Selanjutnya posisi kedua adalah karakter terakhir dari “Banana” memiliki index 6 dari kiri, Kita ubah menjadi bentuk negatif -6.

Jadi untuk mengambil “Banana” dari belakang atau sebelah Kiri yang menggunakan nilai negatif menggunakan kode:

```
str.slice(-12, 6)
```

3.12.4. Substring

Substring method akan memotong string dan mengembalikan string hasil pemotongan. Cara kerjanya sama dengan slice namun substring tidak menerima parameter negatif. Mirip seperti slice, substring memerlukan 2 parameter dimana parameter pertama adalah index awal dari string yang akan di potong dan parameter kedua adalah index dari karakter terakhir plus satu dari string yang akan dipotong.

```
var str = “Apple, Banana, Kiwi”;
var res = str.substring(7, 13);
```

Maka variable res akan menyimpan value berupa string “Banana”. Karena cara kerjanya sama seperti slice, untuk penjelasannya silahkan bisa dibaca kembali method slice.

Bagaimana jika kita tidak memasukan parameter kedua? Apa yang akan terjadi?

Yang akan terjadi adalah string akan tetap diproses parameter pertama akan tetap digunakan untuk memotong string awal namun akan mengembalikan sisanya. Contohnya:

```
var str = “Apple, Banana, Kiwi”;
var res = str.substring(7);
```

Parameter pertama akan digunakan untuk memotong string pada index 7, dimulai dari “B”. Karena tidak didefinisikan parameter kedua, maka tidak terjadi pemotongan dan di return sampai akhir string.

```
“Banana, Kiwi”
```

3.12.5. Substr

Fungsi substr() sama dengan slice() hanya saja parameter kedua pad function substr() mendefinisikan panjang dari string yang akan dipotong. Contoh:

```
var str = “Apple, Banana, Kiwi”;
var res = str.substr(7, 6);
```

Akan menghasilkan respon “Banana”, bagaimana cara kerjanya?

Parameter pertama pada substr() mendefinisikan indeks keberapa pemotongan akan dimulai. Parameter kedua mendefinisikan berapa panjang parameter yang akan dipotong. Dari contoh kita lihat parameter pertama adalah 7 dimana 7 adalah indeks untuk “B”, selanjutnya parameter kedua adalah 6 yang artinya 6 karakter setelah parameter pertama akan diambil dan akhirnya variable res memiliki value “Banana”.

Apabila kalian lupa untuk mendefinisikan parameter kedua, maka substr() akan memotong string dari indeks yang didefinisikan sampai karakter terakhir pada string. Contohnya:

```
var str = “Apple, Banana, Kiwi”;
var res = str.substr(7);
```

Maka akan mengembalikan nilai “Banana, Kiwi”.

Apabila parameter pertama diisi dengan angka negatif maka pemotongan string akan dimulai dari belakang. Contohnya:

```
var str = “Apple, Banana, Kiwi”;
var res = str.substr(-4);
```

Akan mengembalikan nilai “Kiwi” untuk variable res.

3.12.6. Replace

Cara penggunaan fungsi replace() pada jvascript untuk mengganti string adalah sebagai berikut:

```
replace('yang ingin di ganti','ganti nya')
```

jadi pada parameter pertama, kita masukkan misalnya kata yang ingin kita ganti, dan pada parameter yang kedua kita masukkan kata lain yang menjadi penggantinya.

Contohnya, kita akan mengganti kata “mantan” menjadi “istri”.

```
<script>
    var kalimat = "Aku sangat cinta kepada mantan ku";
      var ganti = kalimat.replace('mantan', 'istri');
    document.getElementById("contoh").innerHTML = ganti;
</script>
```

Maka yang sebelumnya isi variabel kalimat adalah “Aku sangat cinta kepada mantan ku” maka akan berubah menjadi “Aku sangat cinta kepada istri ku”.

3.12.7. Converting to Upper and Lower

Jika kita ingin menguba isi string dengan huruf besar semua atau huruf kecil semua, kita bisa mengandalkan kedua fungsi ini, yaitu toUpperCase() untuk mengubah string menjadi huruf kapital dan toLowerCase() berguna untuk mengubah string menjadi huruf kecil semua.

Contoh:

```
<p id="besar"></p>
<p id="kecil"></p>

<script>
    var kalimat = "Belajar JAVASCRIPT ";

   // mengubah ke huruf besar
   var a = kalimat.toUpperCase();

   // mengubah ke huruf kecil
   var b = kalimat.toLowerCase();

    document.getElementById("besar").innerHTML = a;
   document.getElementById("kecil").innerHTML = b;
</script>
```

Maka akan menghasilkan output seperti berikut:

```
BELAJAR JAVASCRIPT
belajar javascript
```

3.12.8. Concat

Method concat() digunakan untuk menggabungkan dua atau lebih string. Penggunaannya membutuhkan dua parameter dimana parameter pertama adalah karakter yang digunakan untuk menghubungkan sedangkan parameter kedua adalah string berikutnya yang hendak disambungkan.

```
var str1 = “Hello”;
var str2 = “World!”;
var res = str1.concat(" ", str2);
```

Okay Kita bedah contoh kode diatas. Kita memiliki 2 string pada var str1 dan str2. Lalu kita akan menggabungkan str1 dan str2 yang disimpan pada var res dengan method concat dan penghubung antara str1 dan str2 adalah karakter spasi.

Untuk menggunakan method concat, pertama-tama kita tuliskan variable pertama yang hendak dihubungkan dengan variable lainnya, lalu berikan karakter titik untuk penghubung dengan method concat. Parameter yang digunakan pada method concat adalah karakter penghubung dan string berikutnya, pada contoh diatas kita gunakan parameter pemgganti spasi " ", dan string selanjutnya str2. Maka penulisan nya menjadi seperti berikut:

```
var res = str1.concat(" ", str2);
```

Dan variable res akan memiliki value “Hello World!”. Jika menggunakan cara sederhana, hasil dari method concat() akan sama dengan cara seperti berikut:

```
var res = str1 + " " + str2;
```

Namun lebih disarankan untuk menggunakan method concat().

3.12.9. Trim

Untuk menghilangkan white space diantara string dapat menggunakan method trim(). Contohnya sebagai berikut:

```
var str = "        Hello World!        ";
var res = str.trim();
```

Terdapat white space berupa spasi di samping kiri dan kanan dari kata Hello World! Pada variable str. Trim() berfungsi untuk menghilangkan ya dan sekarang value dari var res adalah “Hello World!”.

3.12.10. String to Array

Fungsi String to Array atau fungsi split() adalah untuk mengubah string menjadi bentuk array. Kalau kita ingat kembali bahwa array adalah tipe data untuk menyimpan banyak informasi. Pada bagian ini, kita akan membuat sebuah variable bernama huruf dan berisi ‘a, b, c’. Kemudian, kita akan memecahkan string ini menjadi array dengan fungsi split(). Contoh script yang kita gunakan adalah sebagai berikut.

```
<script>
    var huruf = "a,b,c";

   // string jadi array
    var convert = huruf.split(",");
    document.getElementById("contoh").innerHTML = convert[0];
</script>
```

Bisa kita lihat bahwa fungsi string yang digunakan adalah

```
var convert = huruf.split(",");
```

Maka, jadilah array yang tersimpan dalam variable convert dan menampilkan urutan pertama karena kita menggunakan script

```
document.getElementById("contoh").innerHTML = convert[0];
```

yang akan menampilkan isi array urutan ke-0.

3.12.11. Exercise

1. Buatlah sebuah variable dengan nama str1 yang memiliki value “Aku seorang Bajak Laut mempunyai pedang panjang!”
2. Hitung berapa panjang karakter str1.
3. Carilah indeksOf dari “Bajak Laut”
4. Slice ambil kata “Bajak Laut”

3.13. Number Methods

Dengan number methods kita dapat memanipulasi bentuk nomor untuk dimanfaatkan.

3.13.1. To String

Untuk mengembalikan value number menjadi string digunakan method toString(). Cara penggunaan method toString() adalah dengan menuliskan dulu angka yang bertipe number diikuti dengan titik (.) lalu berikan nama methodnya toString().

Perbedaan angka yang memiliki tipe number dan string adalah adalah seperti berikut:

```
var x = 123;           // type number
var y = '123';         // type string
var z = "123";        // type string
```

Jadi pada penulisan angka saat melakukan assignment pada variable, kita perlu memperhatikan penggunaan tanda petik baik yang single maupun double. Karena jika digunakan tanda petik maka angka akan dianggap sebagai data degan tipe string. Sedangkan data dengan tipe number adalah angka yang tidak memiliki tanda petik saat assignmentnya.

Oke balik lagi pada topik, mengubah number menjadi string menggunakan toString().

```
var x = 123;

x.toString();
```

Dengan begini variable x yang mulanya memiliki tipe number berubah menjadi string.

3.13.2. Exponential

Kita dapat mengubah angka decimal menjadi angka dengan bentuk exponensial. Exponential ini akan membulatkan angka sesuai dengan panjang yang kita tentukan. Value yang dikembalikan setelah proses konversi akan memiliki tipe string,

Kita memerlukan satu parameter berupa angka yang digunakan untuk menentukan pajang dari bentuk exponential yang akan dikeluarkan oleh method. Contoh:

```
var x = 9.656;

x.Exponential(2);     // return 9.66e+0
```

```
x.Exponential(4);     // return 9.6560e+0
x.Exponential(6);     // return 9.656000e+0
```

3.13.3. Fixed

Panjangnya sebuah decimal pada angka yang kita miliki dapat diatur menggunakan method toFixed(). Hasil return dari toFixed() akan berupa string. Memerlukan satu buah parameter yang digunakan untuk menspesifikasikan berapa panjang angka dibelakang koma.

Dalam prosesnya, toFixed() memiliki sifat pembulatan.

```
var x = 9.656;

x.toFixed(0);      // return 10
x.toFixed(2);      // return 9.66
x.toFixed(4);      // return 9.6560
x.toFixed(6);      // return 9.656000
```

3.13.4. Precission

Jika toFixed() dapat mendefinisikan panjangnya decimal atau angka dibelakang koma, maka toPrecission() mengembalikan panjangnya number (didepan dan dibelakang koma). Return berupa string dan dibulatkan.

```
var x = 9.656;

x.toPrecission();       // return 10
x.toPrecission(2);     // return 9.7
x.toPrecission(4);     // return 9.656
x.toPrecission(6);     // return 9.65600
```

3.13.5. Convert

Pada javascript terdapat 3 method yang dapat digunakan untuk melakukan konversi variable ke number.

a. Number()

b. ParseInt()

c. parseFloat()

Method ini tidak termasuk pada number method, tetapi global javascript method.

3.13.6. Number

Kita dapat mengubah tipe data selain number kedalam number.

```
Number(true);                  // return 1
Number(false);                 // return 0
Number(“10”);                 // return 10
Number("   10”);              // return 10
Number(“10   ”);              // return 10
Number("   10   ”);           // return 10
Number(“10.33”);            // return 10.33
Number(“10,33”);            // return NaN
Number(“10 33”);            // return NaN
Number(“Lukman”);       // return NaN
```

Untuk bentuk Boolean (true or false) maka return nilai number yang tersedia adalah 1 atau 0.

Untuk tipe data string namun kontennya berupa angka maka number akan melakukan return berupa angka tersebut walaupun didepan atau dibelakang angka tersebut terdapat spasi.

Jika terdapat titik diantara dua angka, maka return akan berupa angka tersebut dengan decimal.

Jika terdapat spasi diantara dua angka maka return berupa NaN atau not a number.

Jika diantara dua angka terdapat koma, maka return berupa NaN.

Jika bukan berupa angka maka return berupa NaN.

Jika string tidak dapat diubah kedalam bentuk numbe maka method akan melakukan return NaN.

3.13.7. ParseInt

Jika pada number() kita mendapatkan return berupa NaN jika string yang dikonversi memiliki spasi diantara kedua angka, maka pada parseInt() kita dapat mengkonversi angka tersebut namun hanya angka pertama saja yang akan di return.

```
parseInt("10");              // return 10
parseInt("10.30");          // return 10
parseInt("10 20");          // return 10
parseInt("10 years");    // return 10
parseInt("years 10");    // return NaN
```

3.13.8. parseFloat

Pada dasarnya parseFloat() sama seperti parseInt() hanya perbedaan paserFloat() dan parseInt() adalah ketika terdapat titik diantara dua angka. Jika pada parseInt() hanya angka sebelum titik saja yang di return, namun pada parseFloat() dapat direturn seluruhnya.

```
parseInt(10.30);          // return 10

parseFloat(10.30);    // return 10.30
```

3.14. Arrays

Array adalas variable special yang digunakan untuk menyimpan beberapa value pada satu variable. Bagaimana jika kita memiliki banyak item dalam sebuah daftar contohnya jenis mobil, untuk menyimpan jenis mobil dengan sebuah variable akan terlihat seperti berikut:

```
var car1 = "Toyota";
var car2 = "Volvo";
var car3 = "Nissan";
```

Bayangkan jika kita bukan hanya memiliki 3 mobil tetapi ada 300 mobil? Solusinya adalah array! Karena array dapat menampung lebih dari satu value dalam sebuah nama variable dan dapat diakses melalui nomor index nya.

3.14.1. Membuat Array

Cara palang mudah membuat sebuah array adalah dengan menggunakan metode literal. Caranya cukup menggunakan kurung kotak [...] dan value variable dituliskan didalamnya. Pemisah antara variable satu dan lainnya digunakan karakter koma (,).

```
var array_name = [item1, item2, ... ];
```

Contoh penulisan variable car yang sudah kita bahas sebelumnya kedalam bentuk array adalah sebagai berikut:

```
var car = [“Toyota”, “Volvo”, “Nissan”];
```

Spasi dan line break tidak berpengaruh pada bentuk penulisan array, bisanya penggunaan line break ditujukan agar lebih mudah dalam membaca array yang dibuat:

```
var car = [
  “Toyota”,
  “Volvo”,
  “Nissan”
];
```

Cara kedua untuk membuat array dapat dilakukan dengan menggunakan keyword new. Dengan contoh yang sama mari kita buat array dengan menggunakan keyword new:

```
var car = new Array(“Toyota”, “Volvo”, “Nissan”);
```

Dalam prakteknya, pembuatan array lebih banyak menggunakan metode literal [...] dibandingkan dengan menggunakan metode new Array() karena dianggap lebih cepat dalam proses pengerjaannya.

3.14.2. Mengakses Array

Kita dapat mengakses sebuah array dengan menggunakan index number. Setiap value yang ada pada sebuah array memiliki nomor index yang secara otomatis diberikan oleh array. Nomornya diberikan berdasarkan urutannya dan nomor pertama pada array dimulai dari 0.

Jika sebelumnya kita memiliki variable car dengan tiga value, maka urutan nya akan mendaji seperti berikut:

```
0 => “Toyota”
1 => “Volvo”
2 => “Nissan”
```

Untuk mengakses “Toyota” maka kita perlu memanggil array dan memberikan spesifikasi index berapa yang hendak kita ambil valuenya didalam kurung kotak [...] :

```
var new_car = car[0];       // akan mengembalikan nilai “Toyota”.
```

Sekarang coba untuk mengakses value “Nissan”!

Kita dapat mengakses semua value array dengan hanya menuliskan nama variable nya saja tanpa perlu menyertakan index. Contohnya kita ingin melihat semua value yang ada pada array car maka yang kita tuliskan adalah:

```
var car = [“Toyota”, “Volvo”, “Nissan”];


document.getElementById(“demo”).innerHTML = car;
```

3.14.3. Mengubah Array

Dengan menggunakan assignment operator " = " kita dapat mengganti value sebuah array. Langkah pertama yang kita lakukan adalah mengetahui pada index keberapa value akan diubah.

Contoh dengan menggunakan variable car, kita akan mengubah nilai “Volvo” menjadi “Lada”. Index dimana value “Volvo” berada adalah 1, berikut kode yang digunakan untuk mengubahnya:

```
var car = [“Toyota”, “Volvo”, “Nissan”];


car[1] = “Lada”;
```

Maka saat kita memanggil array car pada index ke satu:

```
car[1];      // return “Lada”
```

Value yang mulanya adalah “Volvo” berganti menjadi “Lada”.

Sekarang coba ubah value “Nissa” menjadi “Hyundai”!

3.14.4. Array adalah Object

Array adalah bentuk special dari sebuah object. Sama-sama sebuah variable yang dapat menampung banyak value namun perbedaan mendasar antara array dan object adalah array menggunakan nomor index untuk mengakses elementnya atau value yang ada didalamnya sedangkan object menggunakan nama.

3.15. Arrays Methods

Dengan menggunakan array method kita bisa memodifikasi bentuk sebuah array.

3.15.1. Array to String

Kita dapat mengubah bentuk array menjadi sebuah string dengan menggunakan method toString().

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];

document.getElementById("demo").innerHTML = fruits.toString();
```

Dengan method toString() semua value pada element array akan dijadikan satu baris string yang memiliki pemisah berupa koma (,).

```
Pisang,Jeruk,Apel,Mangga
```

Jika kita ingin mengubah pemisah setiap value dari element array, kita dapat menggunakan method join(). Method join() ini sama seperti method toString() yang akan mengubah value pada element array menjadi string hanya saja pada join kita dapat memasukan parameter yang nantinya digunakan untuk pemisah. Contoh, kita ingin pemisah setiap value adalah dash (-):

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];

document.getElementById("demo").innerHTML = fruits.join(" - ");
```

Maka hasil konversinya akan seperti berikut:

```
Pisang - Jeruk - Apel - Mangga
```

3.15.2. Pop

Pop atau poping digunakan untuk mengeluarkan element terakhir pada array.

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];


fruits.pop();       // Element terakhir (Mangga) hilang dari array.
```

Jadi setelah kita menjalankan pop() maka ketika kita panggil kembali array fruits hasilnya akan menjadi berikut:

```
Pisang,Jeruk,Apel
```

3.15.3. Push

Push digunakan untuk mendorong sebuah array untuk dimasukan element baru pada posisi akhir.

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];


fruits.push("Kiwi");       // Element terakhir akan bertambah.
```

Jadi setelah kita menjalankan push dan memberi value untuk element array baru didalamnya, maka saat kita panggil kembali array fruits akan menjadi seperti berikut:

```
Pisang,Jeruk,Apel,Mangga,Kiwi
```

3.15.4. Shift

Jika pada pop() kita mengeluarkan element terakhir pada array, dengan shift() kita dapat mengeluarkan element pertama pada array.

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];


fruits.shoft();       // Element pertama (Pisang) hilang dari
array.
```

Jadi setelah kita menjalankan shift() maka ketika kita panggil kembali array fruits hasilnya akan menjadi berikut:

```
Jeruk,Apel,Mangga
```

3.15.5. Unshift

Sama halnya dengan shift yang beroperasi di awal, unshift akan menambahkan element baru pada awal array.

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];


fruits.unshift("Kiwi");      // Element terakhir (Mangga) hilang
dari array.
```

Jadi setelah kita menjalankan unshift() maka ketika kita panggil kembali array fruits hasilnya akan menjadi berikut:

```
Kiwi,Pisang,Jeruk,Apel,Mangga
```

3.15.6. Splice

Kita dapat menambahkan element baru pada array dengan jumlah dan posisi yang dapat ditentukan menggunakan splice(). Splice() memerlukan dua parameter berupa angka, dimana parameter pertama menunjukkan pada posisi index keberapa value akan dimasukan sedangkan parameter kedua digunakan untuk mendefinisikan jumlah element yang hedak dihapus dari posisi index yang didefinisikan pada parameter pertama.

Kita memiliki variable fruits yang berupa array. Bagaimana jika pada index kedua kita masukan Anggur dan Semangka?

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];


fruits.splice(2, 0, "Anggur", "Semangka");
```

Pada contoh diatas parameter pertama bernilai 2 yang berarti pada index ke 2 kita akan melakukan perubahan pada array, parameter kedua bernilai 0 yang artinya tidak ada element yang akan dihapus, sedangkan parameter sisanya adalah yang akan dimasukan kedalam array. Jadi jika kita akses fruits maka arraynya menjadi:

```
Pisang,Jeruk,Anggur,Semangka,Apel,Mangga
```

Jika parameter kedua kita ganti dari 0 menjadi dua:

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];
```

```
fruits.splice(2, 2, "Anggur", "Semangka");
```

Maka dari index ke 2 (lihat parameter pertama) akan dilakukan penghapusan sebanyak 2 element (parameter kedua). Jadi jika kita akses element fruits:

```
Pisang,Jeruk,Anggur,Semangka
```

Splice() dapat juga digunakan untuk membuat "lubang" pada array atau menghilangkan sebuah element pada array dengan cara hanya memberi dua parameter tanpa penambahan.

Contoh kita kan menghapus element dengan value "Pisang" yang ada pada index 0:

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];

fruits.splice(0, 1);
```

Maka jika kita tampilkan variable fruits, kita tidak lagi menemukan value "Pisang". Parameter pertama menunjukan posisi dan parameter kedua menentukan berapa banyak penghapusan element yang akan dilakukan.

3.15.7. Merge

Jika terdapat dua buah array atau lebih, kita dapat menggabungkannya pada satu array dengan menggunakan method concat():

```
var f1 = ["Pisang", "Jeruk", "Apel", "Mangga"];
var f2 = ["Anggur", "Kiwi"];

var fruits = f1.concat(f2);
```

Maka variable fruits akan memiliki value gabungan array dari f1 dan f2:

```
Pisang,Jeruk,Anggur,Kiwi,Apel,Mangga
```

3.15.8. Sort

Untuk mengurutkan posisi element pada array berdasarkan value dapat digunakan method sort(). Sort() akan mengurutkan value secara alphabetically dan ascending.

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];
```

```
fruits.sort();
```

Maka hasilnya adalah:

```
Apel, Jeruk, Mangga, Pisang
```

3.15.9. Reverse

Dengan reverse() kita dapat mengurutkan element pada array berdasarkan value secara descending().

```
var fruits = ["Pisang", "Jeruk", "Apel", "Mangga"];

fruits.reverse();
```

Maka hasilnya adalah:

```
Pisang, Mangga, Jeruk, Apel
```

3.15.10. Numeric Sort

Pada tipe data number, kita tidak bisa mengurutkan menggunaka sort() atau reverse(). Tipe data number akan diurutkan sebagai string. Maksudnya begini jika kita memiliki nilai “25” dan “100” maka kalau diurutkan “25” lah yang paling besar, kenapa karena saat sorting yang dilihat adalah digit pertama. “25” memiliki digit pertama “2” dan “100” memiliki digit pertama “1” jika di dibandingkan maka “2” lebih besar dibanding “1” dan membuat sorting dengan method seperti ini untuk numbering akan tidak sesuai.

Karena penggunaan sort() dan reverse() tidak cocok untuk number maka digunakanlah compare function.

Untuk melakukan sorting ascending digunakan kode berikut:

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

Dan untuk descending digunakan kode berikut:

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return b - a});
```

3.16. Array Iteration

Array iteration method beroperasi pada setiap item array.

3.16.1. Array Foreach

Dengan menggunakan method foreach() function yang kita miliki akan dipanggil sebanyak item array yang kita miliki. Contohnya:

```
var numbers = [45, 5, 10, 25];

numbers.foreach(myFunction);

function myFunction(value) {
  return value * 2;
}
```

Pada setiap item array, myFunction akan dijalankan dan mengembalikan apa yang sudah diproses pada function tersebut. Pada contoh ini kita memiliki function yang mengalikan value dengan angka 2. Maka hasilnya akan seperti berikut:

```
[90, 10, 20, 50]
```

3.16.2. Array Map

Method map() mirip dengan foreach() hanya saja map() membuat array baru saat mengeksekusi function pada setiap element array, map() tidak menjalankan function apabila element pada array tidak memiliki value, karena map() membuat array baru maka array originalnya tidak diubah. Contohnya:

```
var numbers = [45, 5, 10, 25];
```

```
var res = numbers.map(myFunction);

function myFunction(value) {
  return value * 2;
}
```

Pada setiap item array, myFunction akan dijalankan dan mengembalikan apa yang sudah diproses pada function tersebut. Pada contoh ini kita memiliki function yang mengalikan value dengan angka 2. Maka hasilnya variable res akan memiliki value seperti berikut:

```
[90, 10, 20, 50]
```

3.17. Conditional Statement

Kita dapat menentukan sebuah “aksi” berdasarkan kondisi yang ada. Dengan menggunakan conditional statement kita dapat mendefinisikan aksi apa yang akan dilakukan jika kondisi terpenuhi atau tidak terpenuhi.

Conditional atau biasa disebut percabangan memerlukan perbandingan yang digunakan untuk menyatakan apakah kondisi terpenuhi atau tidak. Pada javascript kita dapat menggunakan comparison operator untuk membandingkan dua nilai atau lebih, selanjutnya logika yang digunakan dalam percabangan adalah if, if ... else, else if dan switch.

3.17.1. Comparison Operator

Comparison operator adalah operator yang dapat digunakan untuk membandingkan nilai yang keluarannya berupa kondisi “true” atau “false”.

Sebagai contoh kita memiliki variable x dengan nilai 10 (var x = 10;).

Operator	Deskripsi	Perbandingan	Kondisi
==	Sama dengan	x == 8 x == 10 x == “10”	False True True
===	Sama dengan value dan	x === 10	True

	tipe	x === “10”	False
!=	Tidak sama dengan	x != 8	True
!==	Tidak sama dengan value dan tipe	x !== 10 x !== “10”	False True
>	Lebih besar	x > 5	True
<	Lebih kecil	x < 5	False
>=	Lebih besar sama dengan	x >= 10	True
<=	Lebih kecil sama dengan	x <= 5	false

Contoh penggunaan dari comparison operator adalah sebagai berikut:

Kita akan membuat aplikasi yang mana aplikasi tersebut ada verifikasi umur, sehingga aplikasi tersebut hanya dapat diakses oleh orang dengan umur lebih dari 18 tahun. Maka script yang dapat kita tuliskan adalah sebagai berikut:

```
if (age < 18) text = "Maaf dik, kamu masih dibawah umur";
```

3.17.2. Logical Operator

Logical operator digunakan untuk menentukan logika antara variable dan value. Oke kita langsung saja ke contohnya, anggap saja kita memiliki variable x dengan nilai 7 (var x = 7;) dan variable y dengan nilai 8 (var y = 8;).

Operator	Deskripsi	Perbandingan	Kondisi
&&	Logika AND Dimana kedua nilai di sisi kiri dan kanan harus sama-sama bernilai true.	x > 7 && y < 10 x < 10 && y > 7	False True
||	Logika OR	x > 7 || y < 10	True

	Dimana salah satu dari kedua nilai kiri dan kana nada yang bernilai true	x < 10 || y > 7 x == 8 || y == 7	True False
!	Logika NOT Dimana akan membalikan nilai kondisi, Jika nilai kondisi !true akan sama dengan false dan sebaliknya.	!(x == 7) !(x == 8)	False True

3.17.3. If ... else

Pada conditional if terdapat tiga jenis yang berbeda, yang pertama if berdiri sendiri, yang kedua terdapat statement yang di eksekusi ketika kondisi tidak terpenuhi dengan menambahkannya pada blok else (if ... else) dan yang ketiga adalah ketika terdapat kondisi kedua menggunakan else if diikuti kondisi yang perlu dipenuhi.

Oke kita bahasa satu persatu.

3.17.3.1. If Condition

Pada if statement kita akan mendefinisikan blok kode yang akan dijalankan ketika kondisi terpenuhi atau true.

```
if (condition) {
  // Blok kode yang akan dieksekusi ketika kondisi true.
}
```

Contoh penggunaanya akan seperti berikut:

```
<!DOCTYPE html>
<html>
<body>
```

```
<p id="demo">Anda belum cukup umur.</p>

<script>
var age = 17;
if (age >= 17) {
  document.getElementById("demo").innerHTML = "SIlahkan lanjutkan proses pembuatan SIM";
}
</script>

</body>
</html>
```

3.17.3.2. If ... Else Condition

Pada if conditional jika kondisi tidak terpenuhi maka tidak akan ada statement yang dieksekusi, nah penggunaan else ini adalah sebagai fallback atau sebagai percabangan jika kondisi tidak terpenuhi. Kita langsung ke contoh saja biar lebih mudah memahaminya:

```
<!DOCTYPE html>
<html>
<body>

<button onclick="cekUmur()">Cek Status</button>

<p id="demo">Status</p>

<script>
function cekUmur() {
  var age = 17;
  var status;
```

```
  if (age >= 17) {
    status = “Umur anda memenuhi syarat”;
  } else {
    status = “Umut anda tidak memenuhi syarat”;
  }

  document.getElementById("demo").innerHTML = status;
}
</script>

</body>
</html>
```

3.17.3.3. If ... Else If ... Else Condition

Else If digunakan jika terdapat kondisi kedua yang perlu dipenuhi, contohnya:

```
<!DOCTYPE html>
<html>
<body>

<button onclick="cekUmur()">Cek Status</button>

<p id="demo">Status</p>

<script>
function cekUmur() {
  var age = 61;
  var status;

  if (age >= 17) {
    status = “Umur anda memenuhi syarat”;
```

```
  } else if (age >= 60) {
    status = “Anda tidak bisa lagi mengikuti ujian ini”;
  } else {
    status = “Umur anda tidak memenuhi syarat”;
  }

  document.getElementById("demo").innerHTML = status;
}
</script>

</body>
</html>
```

3.17.4. Switch

Switch statement pada javascript digunakan untuk melakukan akse berbeda berdasarkan kondisi yang berbeda.

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

Cara kerja switch adalah :

1. Switch mengevaluasi expression.
2. Value dari expression akan dibandingkan dengan setiap value dari case.
3. Ketika ada yang sesuai, maka blok kode yang ada pada kodisi tersebut di eksekusi.

Ketika kode javascript sampai pada keyword break, maka proses dari switch akan berhenti. Jika kita lupa menuliskan break, maka proses akan terus berjalan walaupun expression nya tidak sesuai dengan value pada case.

Jika tidak ada satu pun expression yang sesuai dengan value pada case, maka proses akan diarahkan pada keyword default.

Contoh penggunaan switch:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript switch</h2>

<p id="demo"></p>

<script>
var text;
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
```

```
</html>
```

3.17.5. Latihan

Nilai	Index
100 – 85	A
70 – 84	B
60 – 69	C
40 – 59	D
30 – 39	E
< 29	F

1. Dengan menggunakan if ... else ubahlah nilai kedalam index.
2. Dengan menggunakan switch ubahlah nilai kedalam index.

3.18. Loop

Loop atau biasa disebut perulangan dapat mengeksekusi sebuah blok kode berkali-kali atau secara berulang.

Loop akan sangat membantu jika kita ingin menjalankan kode yang sam berulang kali, setiap waktu dengan value yang berbeda. Contoh nya seperti berikut:

```
var mobil = [“Honda”, “Hyumdai”, “Hino”, “Toyota”, “Daihatsu”, “Nissa”, “Ford”, “BMW”, “Mercedes”];
var container = "";

container += mobil[0] + "<br>”;
container += mobil[1] + "<br>”;
```

```
container += mobil[2] + "<br>";
container += mobil[3] + "<br>";
container += mobil[4] + "<br>";
container += mobil[5] + "<br>";
container += mobil[6] + "<br>";
container += mobil[7] + "<br>";
container += mobil[8] + "<br>";

document.getElementById("demo").innerHTML = container;
```

Bayangkan jika mobil yang ada lebih dari 10, katakan 50! Berarti kita perlu menuliskan sebanyak 50 kali, cukup merepotkan bukan? Lihat kode berikut yang menghasilkan output yang sama:

```
var mobil = ["Honda", "Hyumdai", "Hino"];
var container = "";

for (I = 0; I < mobil.length; i++) {
  container += monil[i] + "<br>";
}

document.getElementById("demo").innerHTML = container;
```

Lebih simpel bukan?

3.18.1. For Loop

For loop atau perulangan dengan menggunakan "for". Berikut syntax yang digunakan:

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

Statement 1 di eksekusi satu kali sebelum meng-eksekusi blok kode.

Statement 2 mendefinisikan kondisi, perulangan akan terus berjalan selama kondisi pada statement kedua ini terpenuhi.

Statement 3 di eksekusi setiap kali perulangan terjadi.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
  text += "Nomor " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Oke kita bahas contoh kode diatas:

Statement 1: melakukan inisialisasi bahwa i = 0, statement ini hanya di eksekusi ketika memulai perulangan, selanjutnya tidak akan di eksekusi.

Statement 2: melakukan validasi, selama nilai i pada perulangan bernilai dibawah 5 (i < 5) maka perulangan akan terus berjalan. Proses validasi ini akan terus berjalan selama perulangan berjalan.

Statement 3: melakukan increment (atau dapat juga decrement) tujuannya untuk menambahkan nilai i, dieksekusi setiap perulangan terjadi. Mengapa? Karena jika nilai i tidak

bertambah disetiap perulangannya maka akan terjadi forever looping atau perulangan tak terbatas.

Mari kita lihat tabel berikut untuk memahami nilai i yang digunakan sebagai dasar perulangan.

Inisialisasi nilai i = 0;

Perulangan ke-	Nilai i	Validasi i < 5
1	0	True
2	1	True
3	2	True
4	3	True
5	4	True
6	5	False

Ketika nilai validasi false, maka perulangan akan berhenti.

3.18.2. While Loop

While loop atau perulangan menggunakan “While” akan dilakukan selama kondisinya memenuhi.

```
while (condition) {
  // code block to be executed
}
```

Condition akan selalu di check ketika looping berjalan. Berikut adalah contoh perulangan menggunakan while yang akan berjalan ketika nilai yang digunakan untuk validasinya (dalam kasus ini variable i) dibawah 10.

```
<!DOCTYPE html>
<html>
<body>
```

```
<p id="demo"></p>

<script>
var text = "";
var i = 0;
while (i < 10) {
  text += "<br>Nomor " + i;
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Berbeda dengan for yang mendefinisikan penambahan nilai i nya sebelum kode yang di eksekusi, pada while kita perlu mendefinisikan penambahan nilai i nya pada blok kode yang dieksekusi. Jika kita lupa untuk menambahkan nilai i maka perulangannya akan berlangsung selamanya.

3.18.3. Do While Loop

Prinsipnya sama denga while loop, tetapi do ... while ini menempatkan validasi di belakang.

```
do {
  // code block to be executed
}
while (condition);
```

Saat blok kode ini dijalankan maka kode yang ada didalam “do” akan di eksekusi, setelah selesai baru dilakukan validasi condition pada “while”, Jika hasilnya true maka perulangan akan berjalan kembali, jika false maka perulangan akan berhenti.

```
<!DOCTYPE html>
<html>
```

```
<body>

<p id="demo"></p>

<script>
var text = "";
var i = 0;

do {
  text += "<br>The number is " + i;
  i++;
}
while (i < 10);

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Persis seperti while, kita perlu menambahkan nilai perulangan i pada blok kode yang dieksekusi. Jika tidak bisa terjadi perulangan yang tak terbatas.

3.6.1. Latihan

Kita memiliki array dengan value = [“Apple”, “Microsoft”, “Google”, “Facebook”, “Spotify”, “Amazon”, “Netflix”, “Slack”].

1. Buatlah perulangan menggunakan for, untuk menampilkan semua element pada array yang kita miliki.
2. Buatlah perulangan menggunakan while, untuk menampilkan semua element pada array yang kita miliki.

3. Buatlah perulangan menggunakan do ... while, untuk menampilkan semua element pada array yang kita miliki.
4. Buatlah sebuah perulangan yang dapat menampilkan hasil sama seperti tulisan soal 1 sampai dengan 3.
5. Dengan menggunakan while dan do ... while, berikan kondisi i < 10 dan inisialisi nilai i = 10. Lalu perhatikan apa yang terjadi.

3.19. DOM

Dengan HTML DOM, Javascript dapat mengakses dan mengubah element dari sebuah dokumen HTML. Ketika web page di load, browser akan membuat Document Object Model atau DOM.

3.19.1. HTML DOM Method

HTML DOM method adalah aksi yang dapat dilakukan (pada HTML Element). HTML DOM property adalah value (HTML Element) yang dapat di set dan ubah. HTML DOM dapat diakses menggunakan javascript, pada DOM semua element HTML didefinisikan sebagai object. Contohnya:

Pada contoh berikut kita akan mengubah konten menggunakan innerHTML pada element paragraph dengan id = demo.

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Dari contoh diatas, getElementById adalah method dan innerHTML adalah property.

Cara yang paling umum untuk mengakses element HTML adalah menggunakan id, dan cara paling umum untuk memasukan konten pada element HTML adalah dengan menggunakan innerHTML.

3.19.2. Element

Sebelum melakukan pengubahan pada konten HTML, hal yang pertama kita lakukan adalah mencari bagaimana cara mengakses element HTML. Terdapat beberapa cara untuk mengakses HTML:

1. **Mencari berdasarkan id**

Cara yang paling mudah untuk mencari element HTML adalah dengan menggunakan element id.

```
document.getElementById("id name")
```

2. **Mencari berdasarkan tag name**

Untuk mencari semua tag dalam satu html dapat menggunakan:

```
document.getElementsByTagName("tag name")
```

Contohnya jika kita ingin mencari element paragraph adalah:

```
document.getElementsByTagName("p")
```

3. **Mencari berdasarkan class name**

Jika kita ingin mencari element HTML berdasarkan class name, kita dapat menggunakan:

```
document.getElementsByClass("class name")
```

3.19.3. HTML

Dengan menggunakan HTML DOM, Javascript dapat mengubah konten dari element HTML.

Di javascript kita dapat menulis langsung pada dokumen HTML dengan menggunakan document.write().

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write(Date());
</script>

</body>
</html>
```

Mengubah konten HTML dengan menggunakan property innerHTML. Berikut contoh penggunaan innerHTML:

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Pada contoh diatas, terdapat element paragraph dengan id=”p1”. Kita gunakan HTML DOM untuk mendapatkan element dengan id=”p1”. Javascript mengubah content dengan menggunakan innerHTML menjadi “New text!”.

Kita dapat mengubah attribute dari element HTML dengan menggunakan syntax:

```
document.getElementById(id).attribute = new value
```

berikut adalah contoh bagaiman javascript dapat mengubah attribute dari sebuah element html:

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

Pada contoh diatas, javascript mencari element HTML berdasarkan id = “myImage”. Javascript mengubah src nya dari “smiley.gif” menjadi “landscape.jpg”.