

This is My Contribution for Open *Library* Concept.
First Published 06 August 2016,
Under License Attribution-NonCommercial-ShareAlike 4.0 International.

Tinju Cepat *Javascript* dengan *jQuery* version 1.0
Code Example : jQuery

Download full source code : <https://github.com/PUSRISTEK/learning-jQuery>

Kritik dan saran silahkan kirim ke : pusatrisetteknologi@gmail.com
Kritik dan saran terbaik dari pembaca akan saya tampilkan pada edisi revisi.
Kritik dan saran diperlukan agar ebook ini menjadi lebih baik lagi & terus berkembang.

Mohon Maaf jika ada pertanyaan yang tak sempat terbalas atau pertanyaan yang terbalas namun jawabanya tidak memuaskan, dikarenakan sebagai pelajar waktu saya terbatas dan sama sama masih belajar. Semoga buku ini memberi manfaat besar untuk pembacanya.

Special Thanks for..

Segala puji bagi Tuhan yang telah memudahkan saya untuk membuat buku ini agar bisa dibaca oleh orang-orang yang membutuhkan dan mempunyai tekad kuat untuk menuntut ilmu.

Thanks so much untuk Maudy Ayunda yang telah memotivasi dan menginspirasi saya untuk terus belajar dan berbagi agar kehidupan saya dan orang lain lebih baik lagi

“Kini saya sadar dan punya alasan besar untuk terus menulis dan banyak membaca sampai akhir hayat, ada manfaat besar yang tak terhitung, sulit didefinisikan dan sulit diekspresikan. Hanya bisa dirasakan dan dibayangkan. it's like loving someone.” - #GunGunFebrianza

Best Moment - 17/12/2015

Prepare

Ada beberapa poin – poin penting yang harus diketahui yaitu buku ini ditulis dengan asumsi anda telah memahami dasar tentang *HTML* dan *CSS*, selanjutnya ada beberapa hal yang harus dipersiapkan diantaranya :

Pertama sebelum memulai buat *file* .txt dengan nama kruna.txt buka *file* tersebut dengan notepad.

Kedua siapkan *code editor* favorit anda, bisa menggunakan ***notepad++*** atau ***sublime***.

Ketiga siapkan *browser* favorit anda, bisa *firefox* atau *chrome* namun penulis menyarankan anda menggunakan firefox.

Keempat instal *addons firebug* pada *firefox* :

<https://addons.mozilla.org/en-US/firefox/addon/firebug/>

Content Table

Chapter 1 : Mengenal Web Development

Subchapter 1 – Apa itu jQuery?

Subchapter 2 – Seperti Apa Cara Kerja Web application?

Subchapter 3 – Apa itu Client-Side Processing?

Subchapter 4 – Quick Fighting dengan HTML 5 & CSS 3

Subchapter 5 – Bagaimana menguji Javascript Application?

Subchapter 6 – Apa itu Javascript Shiv?

Chapter 2 : Mengenal Lebih Dalam Javascript

Subchapter 1 – Apa itu Javascript?

Subchapter 2 – Apa itu Web Console?

Subchapter 3 – Data Types pada Javascript

Subchapter 4 – Variable Declaration pada Javascript

Chapter 1 – Mengenal Web Development

Subchapter 1 : Apa itu jQuery?

jQuery adalah sebuah *javascript library* yang menyediakan banyak sekali *method* yang bisa langsung kita gunakan untuk mempermudah pemrograman *javascript*. *jQuery* adalah *project open source* yang dibuat oleh [John Resig](#) dan *jQuery Team*.^[1] Sebelum *jQuery* dirilis ke publik seluruh *method* ditulis dan diuji untuk memastikan kemampuan *cross-browser* berjalan sepenuhnya diseluruh *browser* tanpa mengalami *bug*.

jQuery adalah *javascript library* yang paling banyak digunakan, menurut beberapa sumber tercatat dari 10 juta *web* dengan tingkat pengunjung yang sangat tinggi 65% diantaranya menggunakan *jQuery*.^{[2][3]}

Cepatnya perkembangan didalam teknologi *web* membuat kebutuhan pengetahuan kita tidak hanya sebatas *HTML*, *CSS* dan *Javascript* saja. Melainkan juga *jQuery*, iya memang *jQuery* juga *javascript* karena ia telah dipermudah maka pengguna *front end* dapat dengan mudah menggunakannya tanpa harus mengerti cara kerja *javascript* dibelakangnya secara *back end*. Inilah alasan mengapa *jQuery* lebih sulit untuk dipelajari karena kita harus faham terlebih dahulu tentang *javascript*.

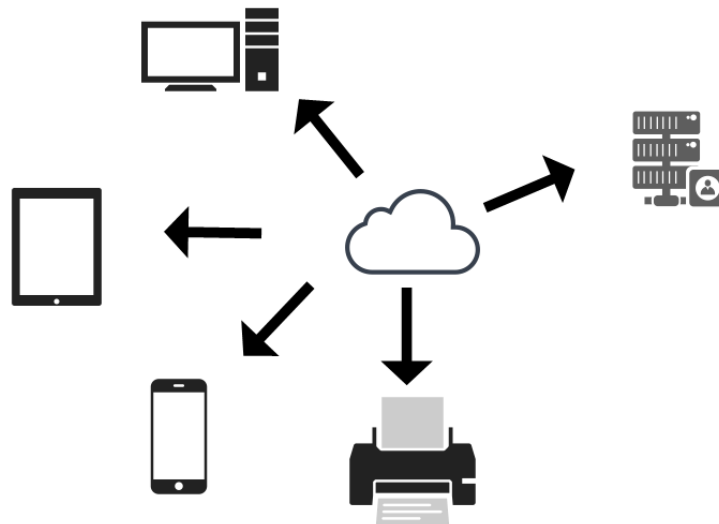
Tapi hal ini jangan dijadikan alasan untuk berhenti membaca buku ini :D sebab dalam buku ini kita juga akan mempelajari *javascript* agar bisa mengembangkan aplikasi-aplikasi menggunakan *jQuery* dan kajian menarik lainnya seperti *jQuery UI*, *AJAX*, *JSON* dan *jQuery Mobile*.

Pada *chapter 1* kita akan mengenal konsep dan terminologi yang sering digunakan dan harus diketahui didalam pembuatan kode untuk *javascript* dan aplikasi yang dibuat menggunakan *jQuery*.

Pada *chapter 2* kita akan menggunakan *javascript* untuk membuat aplikasi *jQuery* dan *jQuery UI*. Pada *chapter 3* kita akan belajar melakukan *DOM Scripting* menggunakan *jQuery*. Pada *chapter 4* kita akan mempelajari testing and debugging *javascript application* yang kita buat, bermain dengan *AJAX*, *JSON*, *Google Maps API*, *HTML 5 Geolocation*, *Web Storage* dan *Web Workers API*. *Chapter 5* kita akan belajar tentang *jQuery Mobile* yang bisa kita gunakan untuk membuat aplikasi di dalam *Mobile development*.

Subchapter 2 : Seperti Apa Cara Kerja Web application?

Sebuah **web application** terdiri dari beberapa elemen yang saling bekerja sama agar bisa memberikan suatu layanan pada sebuah komputer atau *mobile device*. Pada gambar dibawah ini sebuah **web application** terdiri dari **web server** dan **clients**. Sebuah **client** bisa berupa komputer, *tablet* atau *mobile device*. **Client** melakukan akses data pada **web server** menggunakan sebuah **web browser**.

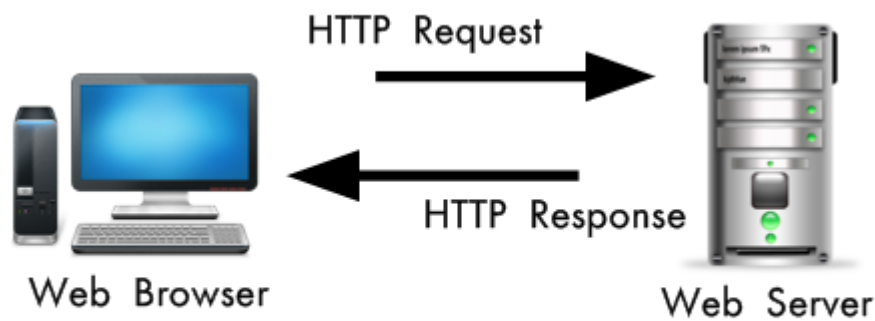


Learning jQuery - Created by Gun Gun Febrianza

Gambar 1.2.1 *Web application*

Network, sebuah sistem yang membuat *client* dan *server* dapat berkomunikasi. *Internet* adalah sebuah *network* skala besar yang terdiri dari sekumpulan *networks* kecil yang saling terhubung. Pada gambar 1.2.1 gambar **cloud** merepresentasikan *network* atau *internet* yang menghubungkan *client* dan *server*. Sebuah *network* bisa dikategorikan berdasarkan ukuran :

- 1.1. **LAN (Local Area Network)** sebuah *network* dengan skala yang sangat kecil yang membuat sekumpulan komputer dapat berkomunikasi dengan jarak yang dekat biasanya dalam satu gedung atau ruangan. *Network* seperti ini seringkali disebut dengan **Intranet**, dapat digunakan untuk menjalankan sebuah **web application** yang hanya bisa diakses oleh pegawai saja.
- 1.2. **WAN (Wide Area Network)** terdiri dari sekumpulan **LAN (Local Area Network)** yang saling terhubung. Untuk mengirimkan informasi dari satu *client* ke komputer lainnya sebuah *router* akan memastikan *network* mana yang paling dekat ke target komputer dan mengirimnya melalui *network* tersebut. Sebuah WAN bisa dimiliki oleh sebuah perusahaan privat atau lebih dari satu perusahaan privat.
- 1.3. **ISP (Internet Service Provider)** adalah sebuah perusahaan yang memiliki izin untuk memiliki dan mengatur WAN yang terhubung ke *internet* di seluruh dunia melalui **Internet Exchange Point**. Sebuah ISP akan menawarkan jasa kepada masyarakat agar bisa mengakses *network* yang dimilikinya.



Learning jQuery - Created by Gun Gun Febrianza

Gambar 1.2.2 *Static Web Page*

Static Web Page, adalah sebuah halaman yang kontennya tidak akan berubah setiap kali kita melakukan *HTTP Request*. Halaman seperti ini secara langsung dikirimkan *web server* ke *web browser* saat *browser* melakukan *request*. Pada *web browser* kita bisa mengetahui sebuah halaman bersifat *static* dengan melihat ekstensinya pada *address bar web browser*. Jika ekstensinya adalah *htm* atau *HTML* maka halaman tersebut adalah halaman *static*.

Pada gambar 1.2.2 menunjukkan bagaimana cara kerja *web application*, bagaimana sebuah *client* melakukan permintaan yang disebut dengan *HTTP Request* sebuah *web page static* ke *web server*. HTTP adalah kependekan dari *hyper text transfer protocol* dan dengan *HTTP Request* sebuah *web server* bisa mengetahui *file* mana yang diminta oleh *client*.

Saat *web server* menerima *HTTP Request*, *server* akan mencari *file* dalam disk drive yang dimilikinya. Setelah *file* ditemukan *web server* akan kembali mengirimkan *HTTP Response* ke *web browser* milik *client*. *Web browser* akan menerjemahkan kode *HTML (Hyper Text Markup Language)* menjadi suatu tampilan visual (*Render*) yang mudah difahami.



Learning jQuery - Created by Gun Gun Febrianza

Gambar 1.2.3 *Dynamic Web Page*

Dynamic Web Page, adalah sebuah halaman yang dihasilkan oleh suatu *program* atau *script* dalam *web server* setiap kali permintaan dilakukan. *Program* atau *script* tersebut akan dieksekusi oleh **application server** yang dimiliki oleh *server*. Sebagai contoh *Client* meminta suatu gambar yang ada didalam **database server** maka *HTTP Request* akan dibaca dan *script*

untuk mencari gambar yang diminta oleh *client* akan dieksekusi hasilnya akan diberikan kembali kepada *client*. (ada atau tidak ada gambar tersebut)

Permintaan yang dikirimkan ke *web server* termasuk data yang dibutuhkan ***application server*** untuk memproses permintaan. Seperti data yang ada didalam sebuah *form*, data tersebut sudah termasuk dalam *HTTP Request*. Saat *web server* menerima HTTP Request jika terdapat permintaan khusus pada *application server* maka *script* yang sesuai dengan permintaan pada *server* akan dieksekusi. Jika diperlukan *script* mampu melakukan permintaan pada *database server* sebagai data tambahan untuk menghasilkan sebuah halaman dinamis. Sebuah proses yang diselesaikan oleh *application server* kita bisa menyebutnya dengan ***server-side processing***.

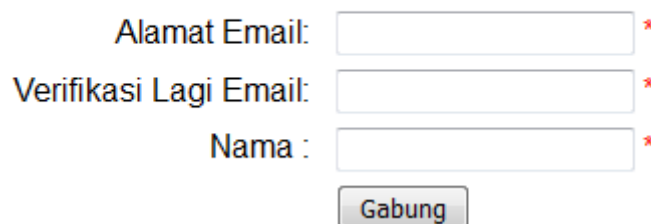
Subchapter 3 : Apa itu *client-side processing*?

Javascript adalah **scripting language** yang bisa dan awalnya digunakan sebagai *client-side processing*. Dengan *javascript* kita bisa mengubah suatu halaman secara langsung pada *browser client* menjadi dinamis tanpa memerlukan *server-side processing*. Namun hari ini *javascript* sudah berevolusi, ia tidak hanya digunakan untuk *client-side processing* namun juga bisa digunakan sebagai *server-side processing* semenjak ada **node.js**. Pada buku ini kita tidak akan membahas **node.js**, mungkin nanti saya akan menulis ebook tentang **node.js** 😊

Saat kita membuat sebuah *javascript application* kita pasti menggunakan *HTML (Hyper Text Markup Language)* untuk mengatur konten dan struktur, selanjutnya kita pasti menggunakan *CSS (Cascading Style Sheet)* untuk memberi *format* pada konten dan terakhir dengan *javascript* kita bisa melakukan *client-side processing*.

Hal pertama yang akan kita lakukan dengan *client-side processing* adalah melakukan **data validation**. Apa itu *data validation*? Kita akan mempelajarinya sekarang dengan membuat sebuah *interface* sederhana untuk mempraktekan *data validation*. Di bawah ini adalah tampilan *interface* yang akan kita gunakan untuk melakukan *data validation*:

Daftar Mailing List



Alamat Email: *

Verifikasi Lagi Email: *

Nama : *

Gambar 1.3.1 *Interface* untuk *Data Validation*

Pada *interface* diatas terdapat 3 kolom *entry* yang harus kita isi kemudian menekan tombol gabung. Bintang *asterisk* berwarna merah menunjukan bahwa kolom *entry* tersebut wajib diisi sebelum menekan tombol gabung. Ketika tombol gabung di tekan *javascript* akan memeriksa apakah data sudah valid atau belum. Jika belum valid informasi pemberitahuan untuk validasi akan muncul.

Dibawah ini adalah kode *index.HTML* untuk membuat *interface* :

Index.HTML

```
<!DOCTYPE HTML>
<HTML lang="en">
<head>
  <meta charset="UTF-8">
  <title>Data Validation</title>
  <link rel="stylesheet" href="email_validation.CSS">
  <script src="email_validation.js"></script>
</head>
<body>
  <main>
    <h1>Daftar Mailing List</h1>
    <form id="email_form" name="email_form" action="success.HTML" method="get">
      <label for="alamat_email1">Alamat Email:</label>
      <input type="text" id="alamat_email1" name="alamat_email1">
      <span id="alamat_email1_error">*</span><br>

      <label for="alamat_email2">Verifikasi Lagi Email:</label>
      <input type="text" id="alamat_email2" name="alamat_email2">
      <span id="alamat_email2_error">*</span><br>

      <label for="nama">Nama :</label>
      <input type="text" id="nama" name="nama">
      <span id="nama_error">*</span><br>

      <label>&nbsp;</label>
      <input type="button" id="bergabung" value="Gabung">
    </form>
  </main>
</body>
</HTML>
```

Pada kode *HTML* diatas diawali dengan deklarasi **<!DOCTYPE HTML>**, artinya kita akan menggunakan *HTML* 5. Selanjutnya kita membuat **meta tag** menggunakan **UTF-8** (sebuah *character encoding* untuk halaman). Di dalam *head section* kita menyimpan *format CSS* dan *script* untuk melakukan *client-side processing* secara eksternal. Di dalam *body section* kita menggunakan **main tag** sebuah *tag* dalam *HTML* 5 yang akan sering kita gunakan. Di dalamnya terdapat beberapa *tag* lainnya seperti **<h1>**, ****, **<label>** dan sebagainya.

Satu lagi ada perbedaan yang harus kita ingat kembali antara **tag** dan **element**.

<label> nama : **</label>**

Tag adalah *syntax* untuk membentuk struktur halaman (ditandai dengan warna merah) sementara **element** adalah *tag* dan nilai yang ada didalam sebuah *tag* (ditandai dengan warna biru).

Selanjutnya dibawah ini adalah kode *CSS* yang digunakan untuk membuat *interface* :

email_validation.CSS

```
body {
    font-family: Arial, Helvetica, sans-serif;
    background-color: white;
    margin: 0 auto;
    width: 770px;
    padding: 0 2em 1em;
    border: 3px solid blue;
}
h1 {
    color: green;
    margin-bottom: .5em;
}
label {
    float: left;
    width: 10em;
    text-align: right;
}
input {
    margin-left: 1em;
    margin-bottom: .5em;
}
span {
    color: red;
}
```

Pada kode CSS diatas kita bisa melihat *rule set* untuk *tag body*, *h1*, *label*, *input* dan *span* pada *rule set tag body* terdapat ***property*** untuk mengatur ***font-family***, ***background-color***, ***margin*** dan sebagainya.

Di bawah ini adalah kode *javascript* untuk melakukan *client-side processing* pada *interface* :

```
email_validation.js
var $ = function (id) {
    return document.getElementById(id);
}
var gabung = function () {
    var alamatEmail1 = $("alamat_email1").value;
    var alamatEmail2 = $("alamat_email2").value;
    var isValid = true;

    if (alamatEmail1 == "") {
        $("alamat_email1_error").firstChild.nodeValue = "Alamat Email Wajib
diisi..";
        isValid = false;
    } else {
        $("alamat_email1_error").firstChild.nodeValue = "";
    }

    if (alamatEmail1 != alamatEmail2) {
        $("alamat_email2_error").firstChild.nodeValue = "Email harus sama dengan
email sebelumnya..";
        isValid = false;
    } else {
        $("alamat_email2_error").firstChild.nodeValue = "";
    }

    if ($("nama").value == "") {
        $("nama_error").firstChild.nodeValue = "Nama wajib diisi..";
        isValid = false;
    } else {
        $("nama_error").firstChild.nodeValue = "";
    }

    if (isValid) {
        $("email_form").submit();
    }
}

window.onload = function () {
    $("bergabung").onclick = gabung;
    $("alamat_email1").focus();
}
```

Saat kita menggunakan sebuah *javascript* untuk memanipulasi *HTML* maka kita telah melakukan *DOM Scripting*. *DOM* adalah singkatan dari ***Document Object Model*** sebuah representasi yang di dalamnya terdapat representasi *element* dan *attribute* dari kode *HTML*. Ketika *javascript* mengubah sebuah aspek didalam *DOM* maka tampilan yang dihasilkan oleh *browser* juga akan ikut berubah.

Pada kode *javascript* diatas terdapat 3 fungsi, yaitu fungsi `$` untuk mendapatkan nilai dalam sebuah *element HTML* berdasarkan ID, fungsi **gabung** akan dieksekusi ketika tombol gabung dalam *HTML* diklik, fungsi terakhir adalah **window.onload** yang akan berjalan setelah *DOM* berhasil dimuat oleh *browser*.

Di dalam fungsi **gabung** kita bisa melihat terdapat empat *if statements* dengan *else clauses* yang menjadi logika untuk melakukan *data validation*. Pada *if statements* yang pertama kita bisa melihat kode dibawah ini :

```
if (alamatEmail1 == "") {  
    $("alamat_email1_error").firstChild.nodeValue = "Alamat Email Wajib diisi..";  
}
```

Jika kolom alamat *email* pertama kosong karena pengguna tidak memberi *input* maka *javascript* akan mengganti bintang asterisk ***** pada *interface* menjadi pemberitahuan informasi validasi.

Daftar Mailing List

Alamat Email: Alamat Email Wajib diisi..

Verifikasi Lagi Email:

Nama : Nama wajib diisi..

Gambar 1.3.2 Data Validation Alamat Email

Di bawah ini ketika pengguna memasukan *input* yang berbeda dengan alamat sebelumnya :

Daftar Mailing List

Alamat Email:

Verifikasi Lagi Email: Email harus sama dengan email sebelumnya..

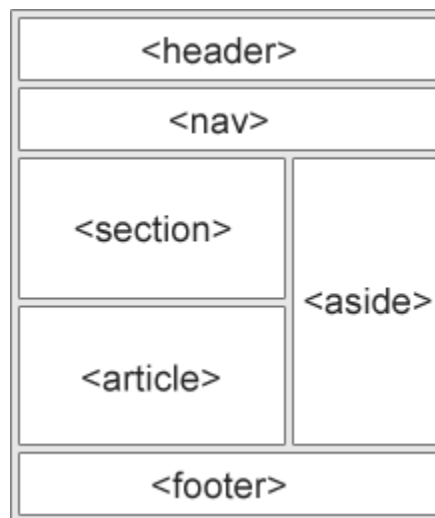
Nama : Nama wajib diisi..

Gambar 1.3.3 Data Validation Verifikasi Ulang Email

Subchapter 4 : Quick Fighting dengan HTML 5 & CSS 3

Seluruh implementasi yang ditulis didalam buku ini menggunakan *HTML 5 Semantic Element*, sebuah *element* yang maknanya jelas dan mudah dimengerti baik untuk *browser* dan *developer*. Saat dulu kita sering kali menandai suatu blok kode menggunakan *tag div* seperti `<div id="nav">` `<div class="header">` `<div id="footer">` sekarang pada *HTML 5* terdapat *semantic element* yang dapat mempermudah untuk menjelaskan hal tersebut yaitu :

`<article>`
`<aside>`
`<details>`
`<figcaption>`
`<figure>`
`<footer>`
`<header>`
`<main>`
`<mark>`
`<nav>`
`<section>`
`<summary>`
`<time>`



Gambar 1.4.1 Struktur *Semantic Elements*

Sebelumnya kita menggunakan *tag div* atau *division* untuk membagi sebuah konten didalam halaman kemudian memberinya *id attributes* agar *CSS* bisa mengaksesnya untuk memberikan sebuah *format*. Namun pada *HTML 5 tag div* (termasuk *tag span*) alangkah lebih baiknya tidak digunakan lagi untuk membuat struktur sebuah halaman kecuali digunakan untuk kepentingan *javascript application* seperti yang telah kita lakukan dalam *data validation* sebelumnya.

Selain itu secara historis *tag span* biasa digunakan untuk menjelaskan sebuah teks yang sering diberi *format* dengan *CSS* tetapi standar hari ini lebih baik menggunakan *tag* sesuai

dengan konten elemen yang diinginkan misal menggunakan *tag* `<q>` untuk membuat sebuah *double quotation mark*. Sehingga adanya *HTML 5 semantic element* memang digunakan untuk mengganti posisi *tag div*.

Selanjutnya kita akan mempelajari *HTML Attributes* yang sering digunakan dalam membuat *javascript application*. Kita akan menggunakan *id attribute* untuk mengidentifikasi sebuah *HTML element* dan menggunakan *class attribute* yang bisa diterapkan pada lebih dari satu *HTML element*. Kemudian pada *label* di *data validation* sebelumnya terdapat *for attribute* yang merelasikan sebuah *label* dengan sebuah *input* dan *title attribute* yang bisa kita gunakan untuk memberikan *tooltip* pada sebuah *HTML element*. Pada *server-side processing* sebuah *name attribute* digunakan untuk mengakses data yang dikirimkan.

Pada *CSS* atau (*Cascading Style Sheet*) sebelumnya kita menggunakan mode *external style sheet* yaitu kode *CSS* dibuat terpisah. Kita bisa menggunakan suatu *CSS* dalam suatu halaman ini dikenal dengan sebutan *embedded style sheet*. Seringkali dibuat *external style sheet* sehingga satu kode *CSS* bisa digunakan untuk banyak halaman kecuali kita memiliki suatu halaman dengan *CSS* yang harus *unique*.

Contoh penggunaan *external style sheet* :

```
<link rel="stylesheet" href="email_validation.CSS">
```

Contoh penggunaan *embedded style sheet* :

```
<style>
  body {
    font-size : 80%;
  }
</style>
```

Sekarang kita akan belajar menggunakan *CSS Selector* untuk menerapkan sebuah *style* pada *HTML Page*. Di bawah ini adalah kode *HTML* yang memiliki *semantic element main* dan *footer element*. Pada *main* terdapat dua buah *element* paragraf yang memiliki *class attribute* “blue” dan pada *footer* terdapat satu paragraf yang memiliki dua *class attribute* :

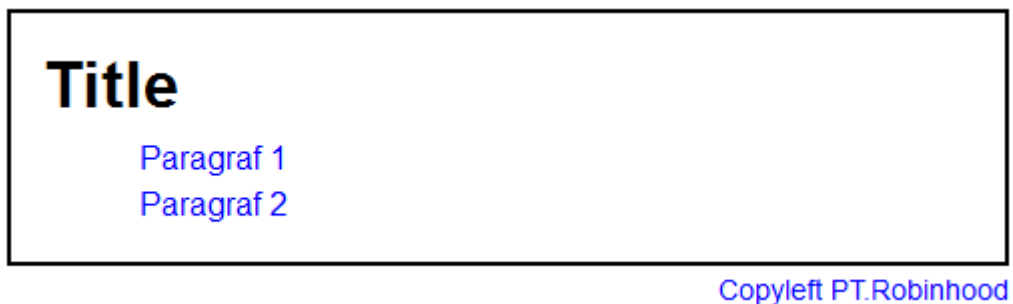
Index.html
<pre> <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Data Validation</title> <link rel="stylesheet" href="selector.css"> </head> <body> <main> <h1>Title</h1> <p class="blue"> Paragraf 1 </p> <p class="blue"> Paragraf 2 </p> </main> <footer> <p id="copyleft" class="blue right"> Copyleft PT.Robinhood</p> </footer> </body> </html> </pre>

Pada kode CSS dibawah ini terdapat 4 **rule set** yang ditandai dengan warna merah, disebut dengan **type selector** yang bisa kita gunakan untuk memberikan *style* pada sebuah *HTML element*. Selanjutnya terdapat 1 *rule set* yang ditandai dengan warna hijau, disebut dengan **id selector** yang bisa kita gunakan untuk memberikan *style* pada suatu *HTML element* berdasarkan *id attributes*. Terakhir terdapat 2 *rule set* yang ditandai dengan warna biru, disebut dengan *class selector* yang bisa kita gunakan untuk memberikan *style* pada lebih dari satu *HTML element*.

selector.css
<pre> body { font-family: Arial, Helvetica, sans-serif; margin: 1em auto; width: 500px; } main { display: block; padding: 1em; border: 2px solid black; } h1 { margin: 0 0 .25em; } p { margin: .25em 0 .25em 3em; } </pre>

```
}  
  
#copyleft {  
    font-size: 90%;  
}  
  
.blue {  
    color: blue;  
}  
  
.right {  
    text-align: right;  
}
```

Di bawah ini adalah halaman hasil dari kode HTML dan CSS sebelumnya :



Gambar 1.4.2 Implementasi *CSS Selector*

Table Uji Pemahaman A
Chapter 1

Pertanyaan	Hasil
Apa itu <i>Web application</i> ?	Faham/Hafal/Lupa
Apa itu <i>Client</i> dan <i>Server</i> ?	Faham/Hafal/Lupa
Apa itu Cloud?	Faham/Hafal/Lupa
Apa itu Intranet?	Faham/Hafal/Lupa
Sebutkan kategori <i>network</i> berdasarkan ukuran?	Faham/Hafal/Lupa
Apa itu <i>Internet</i> ?	Faham/Hafal/Lupa
Apa itu <i>Internet Service Provider</i> ?	Faham/Hafal/Lupa
Apa itu <i>Internet Exchange Point</i> ?	Faham/Hafal/Lupa
Apa itu <i>Web browser</i> ?	Faham/Hafal/Lupa
Apa itu <i>HTTP Request</i> ?	Faham/Hafal/Lupa
Apa kepanjangan dari HTTP?	Faham/Hafal/Lupa
Apa kepanjangan dari <i>HTML</i> ?	Faham/Hafal/Lupa
Apa itu <i>static web page</i> ?	Faham/Hafal/Lupa
Apa itu <i>dynamic web page</i> ?	Faham/Hafal/Lupa
Apa itu <i>application server</i> ?	Faham/Hafal/Lupa
Apa itu <i>database server</i> ?	Faham/Hafal/Lupa
Apa itu <i>server side processing</i> ?	Faham/Hafal/Lupa
Apa itu <i>scripting language</i> ?	Faham/Hafal/Lupa
Apa itu <i>data validation</i> ?	Faham/Hafal/Lupa
Apa itu <code><!DOCTYPE HTML></code> ?	Faham/Hafal/Lupa
Apa itu <i>character encoding</i> ?	Faham/Hafal/Lupa
Apa itu UTF-8 ?	Faham/Hafal/Lupa
Apa perbedaan <i>Tag</i> dan <i>Element</i> ?	Faham/Hafal/Lupa
Apa itu <i>DOM Scripting</i> ?	Faham/Hafal/Lupa
Apa itu <i>Document Object Model</i> ?	Faham/Hafal/Lupa
Apa itu <i>HTML 5 Semantic Element</i> ?	Faham/Hafal/Lupa
Apa itu <i>HTML Attributes</i> ?	Faham/Hafal/Lupa
Apa perbedaan <i>id attribute</i> dan <i>class attribute</i> pada <i>HTML Element</i> ?	Faham/Hafal/Lupa
Apa manfaat <i>for attribute</i> dan <i>title attribute</i> ?	Faham/Hafal/Lupa
Apa itu <i>CSS selector</i> ?	Faham/Hafal/Lupa
Apa perbedaan <i>id</i> , <i>type</i> dan <i>class selector</i> ?	Faham/Hafal/Lupa

Subchapter 5 : Bagaimana menguji Javascript Application?

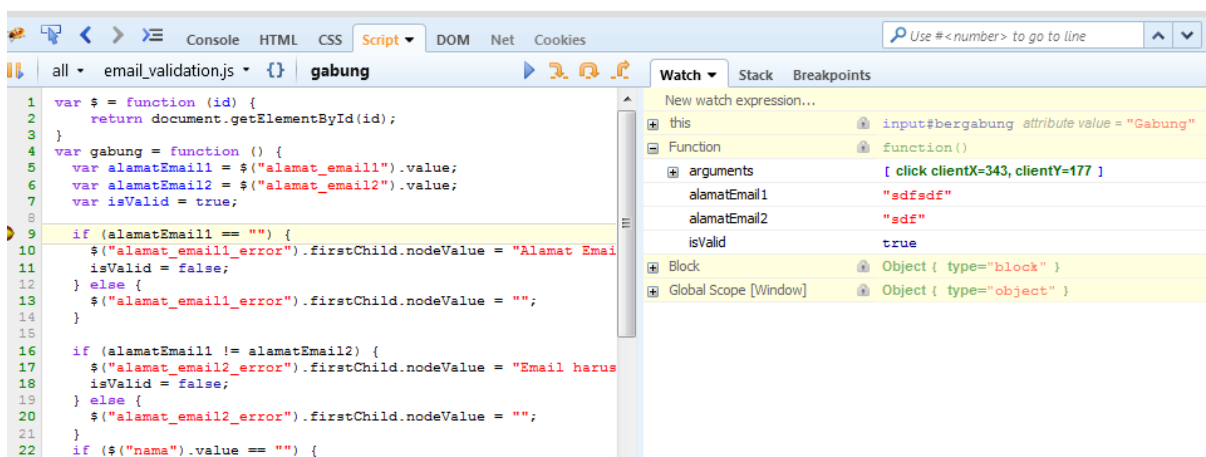
Jika sebelumnya kita berhasil melakukan instalasi *addons firebug*, kita akan menguji halaman sebelumnya saat kita melakukan *data validation*. Buka kembali halaman dibawah ini kemudian tekan F12 dan pilih *tab script* :

Daftar Mailing List

Alamat Email:

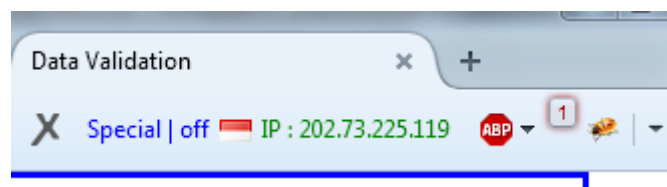
Verifikasi Lagi Email: Email harus sama dengan email sebelumnya..

Nama : Nama wajib diisi..



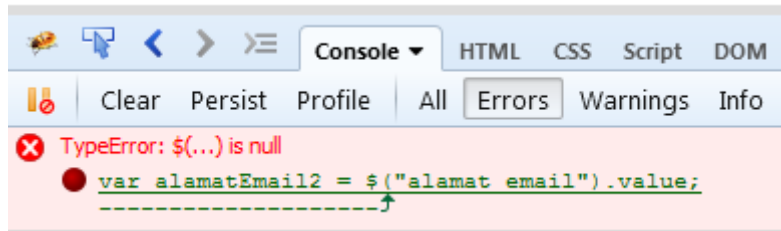
Gambar 1.5.1 Debugging Javascript Application

Lakukan *reload page* kemudian tekan tombol gabung jika tidak terdapat kesalahan kode pada *javascript* maka halaman akan berjalan seperti biasanya. Namun jika terdapat kesalahan kode pada *javascript* maka ketika menekan tombol Gabung akan muncul notifikasi angka pada *icon firebug* :



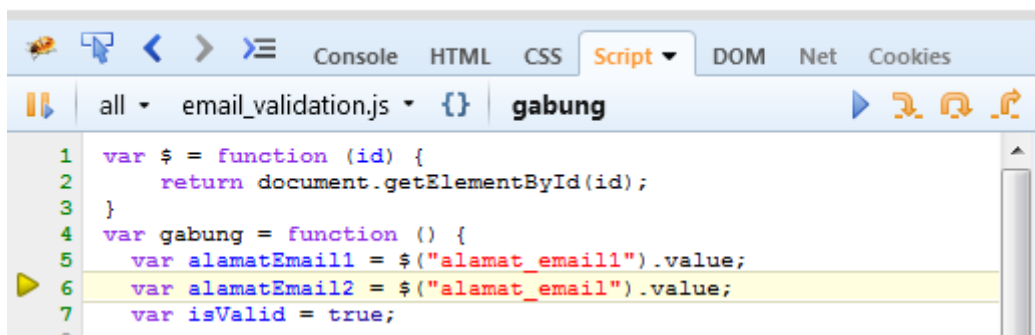
Gambar 1.5.2 Javascript Error Detected

Kemudian jika kita klik maka kita akan mengetahui *error* yang terjadi :



Gambar 1.5.3 *Error Location*

Pada gambar diatas disebutkan bahwa **fungsi \$** untuk mendapatkan nilai dari suatu *element* berdasarkan ID menghasilkan nilai **null**, artinya *element* dengan *id attribute* **alamat_email** tidak ada. Jika kita klik link yang berwarna hijau dan kembali menekan tombol **Gabung** maka kita akan langsung dibawa ke lokasi baris kode tempat terjadinya *error*.



Gambar 1.5.4 *Error Source*

Pada gambar 1.5.4 diatas lokasi kesalahan terjadi pada baris ke 6, fungsi \$ menggunakan *parameter id attribute* yang salah. Seharusnya **alamat_email2** bukan **alamat_email**.

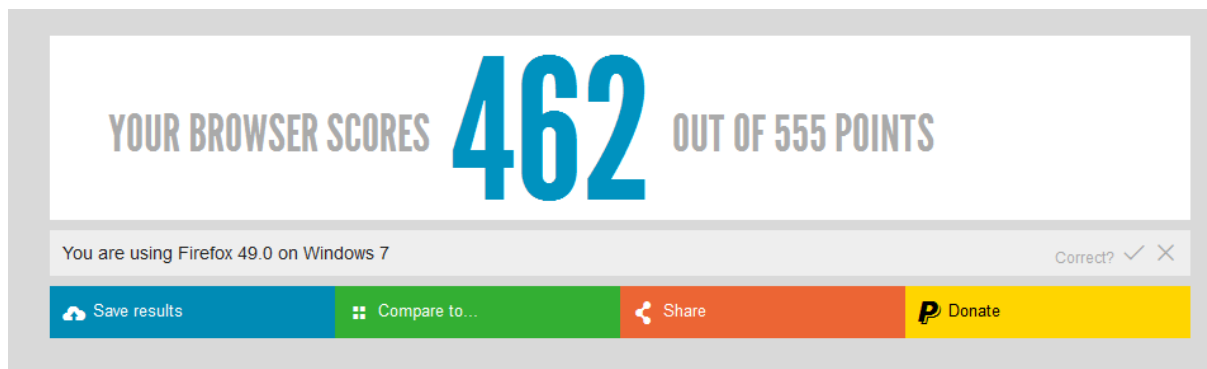
Subchapter 6 : Apa it Javascript Shiv?

Saat kita membuat sebuah *website* tentu kita ingin *website* yang kita buat bisa dikunjungi oleh siapa saja. Sehingga kita harus memastikan bahwa *website* yang kita buat *compatible* dengan seluruh *browser* yang ada, ini disebut dengan *cross-browser compability*. Ini artinya kita harus menguji *website* yang kita buat pada seluruh *browser* yang ada atau lima *browser* yang paling banyak digunakan.

Kita bisa mengetahui kemampuan HTML 5 *Compatibility browser* yang kita miliki saat ini (*actual vrowser version*) dengan mengakses situs :

<http://html5test.com/>

Hasil *score* atau *level of HTML 5 Compatibility* yang penulis dapatkan :



Gambar 1.6.1 *Browser Score*

Hari ini seluruh *browser modern* sudah mendukung HTML 5 *semantic element*. Namun pada browser lama seperti *Internet Explorer 7* dan *8* yang masih mempunyai peran besar di sebagian pasar aplikasi kita harus menggunakan *javascript shiv*. Penggunaan *javascript shiv* agar HTML 5 *element* tetap bisa digunakan pada *browsers* lama sehingga CSS tetap bisa memberikan style pada *elements* tersebut, untuk menggunakannya kita perlu menggunakan *html5.js* yang bisa kita masukan didalam element `<Head>`

```
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
```

Selain itu kita bisa menggunakan **normalize.css** yang dapat membantu kita memperbaiki tampilan dari *website* yang kita buat agar seragam pada semua *browser* sebab pada masing masing browser memiliki sedikit perbedaan arsitektur dalam melakukan *render* suatu halaman.

Table Uji Pemahaman B Chapter 1	
Pertanyaan	Hasil
Apa itu <i>Firebug</i> ?	Faham/Hafal/Lupa
Apa itu <i>cross-browser compability</i> ?	Faham/Hafal/Lupa
Apa itu <i>javascript shiv</i> ?	Faham/Hafal/Lupa
Apa manfaat <i>normalize.css</i> ?	Faham/Hafal/Lupa

Chapter 2 – Mengetahui Lebih Dalam Javascript

Subchapter 1 : Apa itu Javascript?

Javascript is an interpreted language.

Javascript adalah sebuah bahasa yang memerlukan *interpreter* agar bisa dimengerti oleh komputer, saat kita mengeksekusi *javascript code* dalam sebuah *browser* maka sebuah *interpreter* dalam *browser* tersebut akan mengkonversi *javascript code* menjadi sebuah *machine code*.

Machine code adalah sebuah bahasa yang dimengerti oleh komputer. *Machine code* adalah sebuah *string* yang terdiri dari bilangan biner 1(s) dan 0(s). Analoginya seperti mengkonversi bahasa inggris kedalam bahasa indonesia sehingga dapat kita mengerti. Hanya saja agar bisa difahami oleh komputer *javascript code* harus di interpret (terjemahkan) menggunakan *interpreter* pada *browser* setiap kali kita mengeksekusi sebuah *javascript code*.

Javascript memiliki sifat *Case-Sensitive*, *Cross-platform* dan mendukung *Design Pattern Object Oriented Programming*. *Javascript* adalah bahasa yang kecil dan ringan (*small & lightweight*), pada sebuah *browser javascript* sudah menyediakan beberapa *standar library* yang bisa kita gunakan seperti *array*, *date*, *math* dan sebagainya.

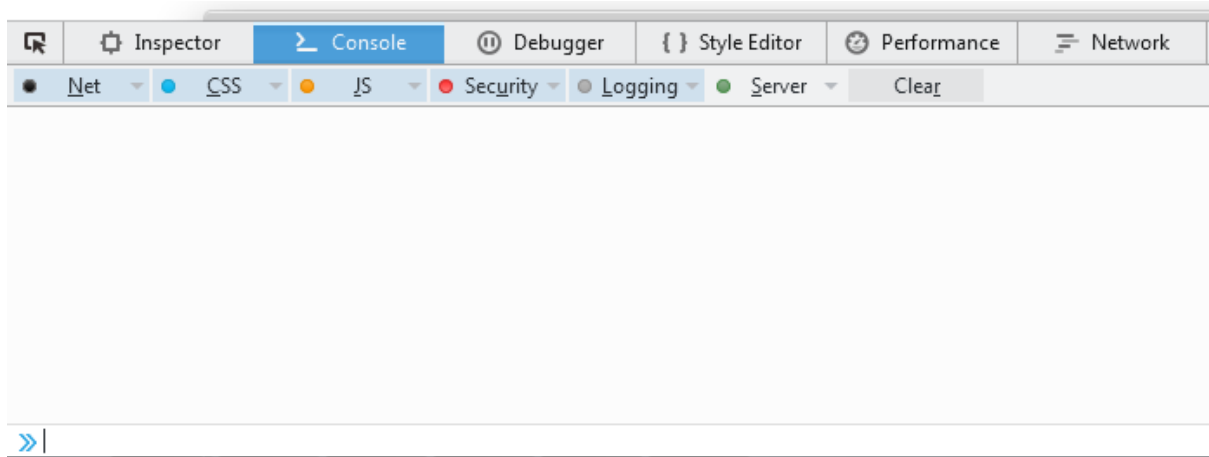
Ada lagi yang harus diketahui jika kita menggunakan *Javascript* pada sebuah *browser* yaitu versi *javascript* itu sendiri apakah sudah mengadopsi **ECMAScript** yang terbaru atau belum, pada kasus di buku ini kita akan menggunakan **Browser Firefox** yang sudah full support menggunakan latest ECMAScript 2015 atau biasa disebut dengan ECMAScript 6.[4]

ECMA adalah kependekan dari (***European Computer Manufacturers Association***). ECMA adalah sebuah organisasi *private* yang mengembangkan standar sistem informasi dan komunikasi. Standarisasi yang mereka lakukan adalah untuk *javascript* dan mereka menyebutnya dengan sebutan **ECMAScript**.

Kemudian organisasi yang mengatur standarisasi halaman *web* adalah **W3C** kependekan dari (***World Wide Web Consortium***) mereka bukan hanya mengatur standarisasi tentang HTML, XHTML dan XML tetapi juga untuk *javascript*.

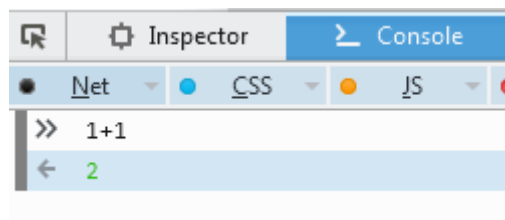
Subchapter 2 : Apa itu Web Console?

Pada *Browser Firefox* kita bisa menggunakan **Web Console** untuk mengeksekusi sebuah *Javascript Code*. Kita bisa memanggilnya dengan menekan CTRL+SHIFT+K maka akan muncul sebuah *interpreter* pada *browser* seperti gambar dibawah ini :



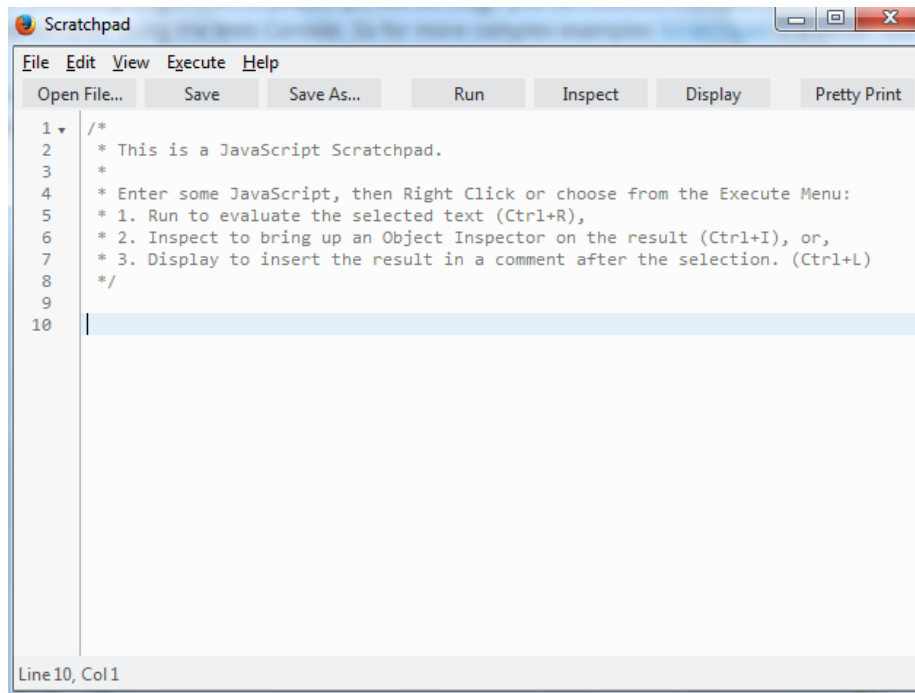
Gambar 2.2.1 Web Console

Untuk mengujinya coba ketik : 1 + 1 kemudian tekan enter. Hasilnya :



Gambar 2.2.2 Addition Test

Dengan **Web Console** kita hanya bisa mengeksekusi satu baris *javascript code* saja, untuk yang lebih *advance* kita bisa menggunakan **Scratchpad** yang disediakan oleh *firefox*. Untuk memanggilnya tekan tombol **SHIFT+F4** maka akan muncul seperti pada gambar dibawah ini :

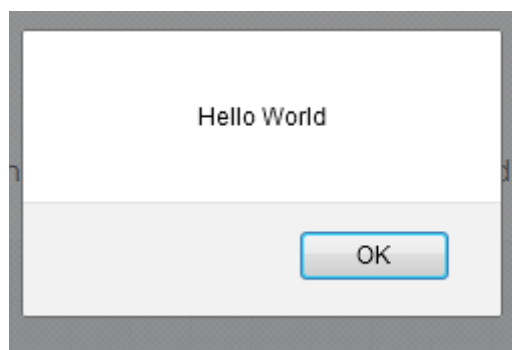


Gambar 2.2.3 Scratchpad pada Firefox

Sekarang kita akan melakukan *first coding hello world* ketik code dibawah ini kedalam code editor kemudian tekan CTRL + R untuk mengeksekusinya :

```
function sapa(pesan) {  
    alert("Hello " + pesan);  
}  
  
sapa("World");
```

Maka hasilnya adalah :



Gambar 2.2.4 Hello Word Pada Javascript

Subchapter 3 : Data Types pada Javascript

Variable adalah suatu tempat untuk menyimpan informasi, biasanya suatu *variable* memiliki *data types* yang spesifik. *Javascript* dikenal dengan ***Loosely Typed Language*** dikarenakan hanya mampu membuat *generic variable* tanpa harus menetapkan *Data Types* secara eksplisit. Berbeda dengan bahasa c, c++, c# yang tidak bisa membuat *generic variable* yang tidak memiliki *data types* yang jelas melainkan sebuah *variable* harus memiliki *data types* seperti *integer*, *float*, *double* dan sebagainya. Sehingga disebut dengan ***Strongly Typed Language***.

Meskipun begitu *Generic Variable* dalam *javascript* memiliki 7 *Data Types* yang mengacu pada *ECMAScript* terbaru diantaranya adalah :

1. **Number**, sebuah angka tanpa *quote*. Misal 42, 0.5 dan 2e-16
2. **String**, sekumpulan *character* dalam *quote*. Misal "hello"
3. **Boolean**, Nilai logika. Misal TRUE or FALSE
4. **Null**, sebuah *keyword* yang berarti tidak memiliki nilai
5. **Undefined**, sebuah *variable* yang nilainya belum didefinisikan.
6. **Symbol**, sebuah *data types* yang unik dan kekal.
7. **Object**, mengacu pada *object oriented programming*.

***Data Types* 6** merupakan tipe data terbaru pada **ECMAScript 6**.

Subchapter 4 : Variable Declaration pada Javascript

Seperti yang telah kita bahas sebelumnya *javascript* adalah **Loosely Typed Language** yang artinya kita tidak harus secara eksplisit menetapkan **Data Types** ketika membuat suatu *variable*. Berdasarkan **ECMAScript** terbaru Ada 3 tipe deklarasi dalam *javascript* yaitu :

1. **Var**, deklarasi *variable* secara umum.
2. **Let**, deklarasi *variable* berdasarkan ***block scope local variable***
3. **Constant**, deklarasi konstanta atau ***read-only variable***

Kita akan mempelajari ketiganya satu persatu, Dibawah ini adalah contoh ***variable declaration*** menggunakan **Var** pada *Javascript* :

```
var n = 1; // number
var b = true; // boolean
var s = "hello"; // string
var x; // undefined
```

Bisa juga menggunakan satu *statement* dengan syarat menggunakan *comma* untuk membedakanya dan *semicolon* diakhir *statement* seperti dibawah ini :

```
Var n = 1, var b = true, var s = "hello", var x;
```

Pemberian nama *variable* pada *javascript* bisa menggunakan abjad, angka, *underscore*, dan tanda *dollar*. Dibawah ini adalah contoh nama *variable* yang benar :

```
var _1a; // benar
var a1; // benar
var a_1; // benar
var 1a; // salah
```

Daftar Pustaka

- [1] <https://en.wikipedia.org/wiki/JQuery>
- [2] *"Usage of Javascript libraries for websites"*. Retrieved 2015-07-14.
- [3] *"jQuery Usage Statistics"*. Retrieved 2013-05-17.
- [4] Developer.mozilla.org