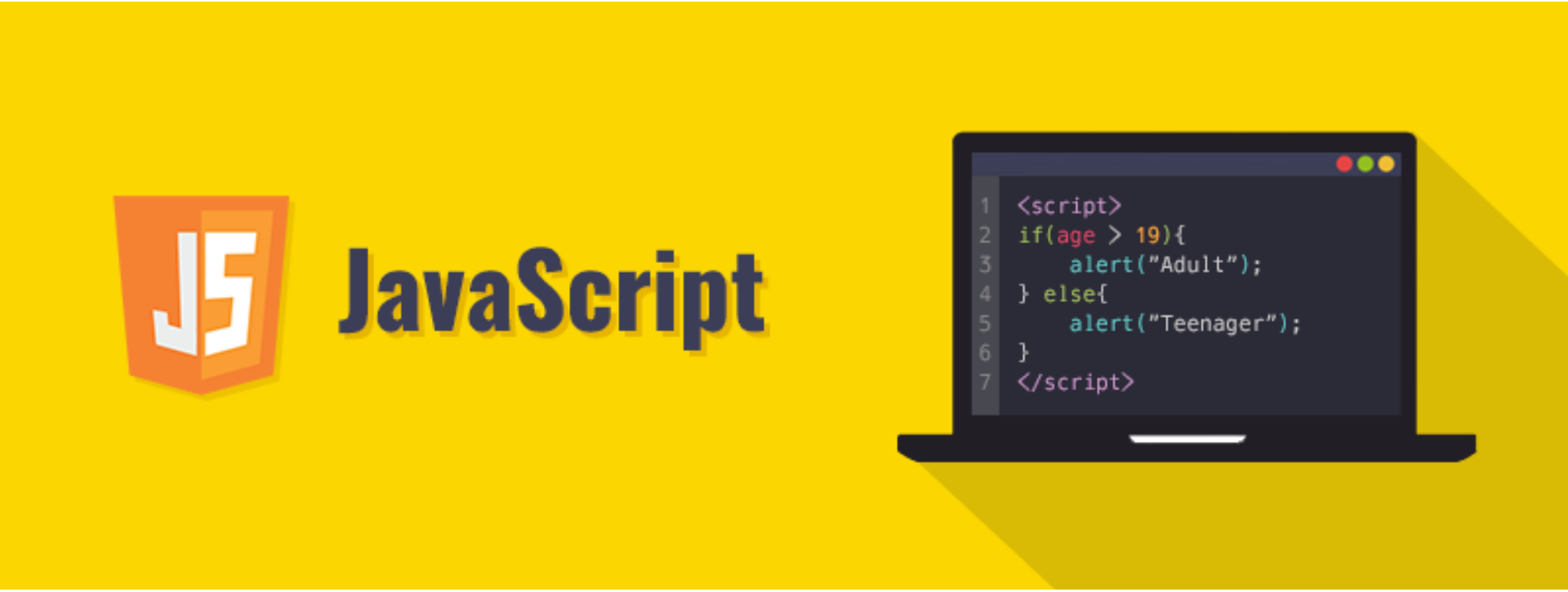
JS
JavaScript

Assalamualaikum



Konversi JavaScript

Jenis Data JavaScript :

- string
- number
- boolean
- object
- function

Dan 2 tipe data yang tidak dapat berisi nilai:

- null
- undefined

Mengubah Angka menjadi String

Fungsi **String()** dapat mengonversi angka menjadi string.

Ini dapat digunakan pada semua jenis angka, literal, variabel, atau ekspresi:

Metode **toString()** melakukan hal yang sama.

```
var angka = 123;
document.write(typeof angka);
document.write("<br>"+ typeof String(angka));
```

```
var angka = 123;
document.write(typeof angka);
document.write("<br>"+ typeof angka.toString());
```

Mengubah String menjadi Angka

Ada 3 metode JavaScript yang dapat digunakan untuk mengonversi variabel menjadi angka:

- Number(x)
- parseInt(x)
- parseFloat(x)

X adalah parameter string yang akan dirubah.

Mengubah String menjadi Angka

```
document.write("<br>" + Number("3.14") );    // returns 3.14
document.write("<br>" + Number(" ")  );      // returns 0
document.write("<br>" + Number("")   );      // returns 0
document.write("<br>" + Number("99 88") );   // returns NaN

document.write("<br>" + parseInt("3.14"));    // returns 3
document.write("<br>" + parseInt(" "));      // returns NaN
document.write("<br>" + parseInt(""));      // returns NaN
document.write("<br>" + parseInt("99 88"))  ;  // returns 99

document.write("<br>" + parseFloat("3.14"));    // returns 3.14
document.write("<br>" + parseFloat(" "));      // returns NaN
document.write("<br>" + parseFloat(""));      // returns NaN
document.write("<br>" + parseFloat("99 88"))    ;  // returns 99
```

Try Catch – throw – finally

- Pernyataan **try** memungkinkan Anda menguji blok kode kesalahan.
- Pernyataan **catch** memungkinkan Anda menangani kesalahan.
- Pernyataan **throw** memungkinkan Anda membuat kesalahan kustom.
- Pernyataan **finally** memungkinkan Anda mengeksekusi kode, setelah mencoba dan menangkap, terlepas dari hasilnya.

Try Catch - Error

Pernyataan **try** memungkinkan Anda untuk menentukan blok kode yang akan diuji untuk kesalahan ketika sedang dijalankan.

Pernyataan **catch** memungkinkan Anda untuk menentukan blok kode yang akan dieksekusi, jika terjadi kesalahan dalam blok **try**.

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

<script>
    try {
      adddlert("Welcome guest!");
    }
    catch(err) {
      document.getElementById("demo").innerHTML = err.message;
    }
</script>
```

Pernyataan throw

Pernyataan **Throw** memungkinkan Anda untuk membuat teks kesalahan kustom.

Secara teknis Anda bisa **melempar pengecualian (melempar kesalahan)** .

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

Pernyataan throw

Contoh ini memeriksa input. Jika nilainya salah, pengecualian (err) dilemparkan.

```
<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="p01"></p>

<script>
  function myFunction() {
    var message, x;
    message = document.getElementById("p01");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
      if(x == "") throw "empty";
      if(isNaN(x)) throw "not a number";
      x = Number(x);
      if(x < 5) throw "too low";
      if(x > 10) throw "too high";
    }
    catch(err) {
      message.innerHTML = "Input is " + err;
    }
  }
</script>
```

Pernyataan finally

Pernyataan **Finally** memungkinkan Anda mengeksekusi kode, setelah mencoba dan menangkap, terlepas dari hasilnya:

```
<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>

<p id="p01"></p>

<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "is empty";
    if(isNaN(x)) throw "is not a number";
    x = Number(x);
    if(x > 10)   throw "is too high";
    if(x < 5)  throw "is too low";
  }
  catch(err) {
    message.innerHTML = "Input " + err;
  }
  finally {
    document.getElementById("demo").value = "";
  }
}
</script>
```

Scoup JavaScript

Dalam JavaScript ada dua jenis ruang lingkup variabel:

- Ruang lingkup variabel **lokal.**
- Ruang lingkup variabel **global.**

Lingkup menentukan aksesibilitas (visibilitas) dari variabel-variabel ini.

Variabel yang didefinisikan di dalam suatu fungsi tidak dapat diakses (terlihat) dari luar fungsi.

Variabel lokal

Variabel dideklarasikan dalam fungsi JavaScript, menjadi **LOKAL** ke fungsi.
Variabel lokal memiliki **lingkup fungsi** : Mereka hanya dapat diakses dari dalam fungsi.

```
// kode disini tidak dapat menggunakan var car

function myFunction() {
    var car = "Volvo";

    // code disini dapat menggunakan var car

}
```

Karena variabel lokal hanya dikenali di dalam fungsinya, variabel dengan nama yang sama dapat digunakan dalam fungsi yang berbeda.
Variabel lokal dibuat ketika fungsi dimulai, dan dihapus ketika fungsi selesai.

Variabel global

Variabel yang dinyatakan di luar fungsi, menjadi **GLOBAL** .
Variabel global memiliki **cakupan global** : Semua skrip dan fungsi pada halaman web dapat mengaksesnya.

```
var car = "Volvo";

// code disini dapat menggunakan var car

function myFunction() {

        // code disini dapat menggunakan var car

}
```

Automatis global

Jika Anda menetapkan nilai ke variabel yang belum dideklarasikan, itu akan secara otomatis menjadi variabel **GLOBAL** .

Contoh kode ini akan mendeklarasikan variabel global car, bahkan jika nilainya ditetapkan di dalam suatu fungsi.

```
myFunction();

// code disini dapat menggunakan var car

function myFunction() {
        car = "Volvo";
}
```

Panduan gaya penulisan

Aturan Pernyataan

Aturan umum untuk pernyataan sederhana:

- Selalu akhiri pernyataan sederhana dengan titik koma.

Aturan umum untuk pernyataan kompleks (gabungan):

- Letakkan braket pembuka di akhir baris pertama.
- Gunakan satu ruang sebelum braket pembuka.
- Letakkan braket penutup di baris baru, tanpa spasi di depan.
- Jangan akhiri pernyataan kompleks dengan titik koma.

Panduan gaya penulisan

Aturan Objek

Aturan umum untuk definisi objek:

- Tempatkan braket pembuka pada baris yang sama dengan nama objek.
- Gunakan titik dua plus satu ruang antara masing-masing properti dan nilainya.
- Gunakan tanda kutip di sekitar nilai string, bukan di sekitar nilai numerik.
- Jangan tambahkan koma setelah pasangan nilai properti terakhir.
- Tempatkan braket penutup pada baris baru, tanpa spasi di depan.
- Selalu akhiri definisi objek dengan titik koma.

```
//objek
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Panduan gaya penulisan

Panjang Garis <80

Agar mudah dibaca, hindari garis yang lebih panjang dari 80 karakter.

Jika pernyataan JavaScript tidak cocok pada satu baris, tempat terbaik untuk memutusnya, adalah setelah operator atau koma.

```
//<80
document.getElementById("demo").innerHTML =
"Hello Dolly.";
```

Panduan gaya penulisan

Konvensi Penamaan

Selalu gunakan konvensi penamaan yang sama untuk semua kode Anda. Sebagai contoh:

- Nama variabel dan fungsi ditulis sebagai **camelCase**
- Variabel global yang ditulis dalam **UPPERCASE** (Kami tidak, tapi itu cukup umum)
- Konstanta (seperti PI) ditulis dalam **UPPERCASE**

Memuat JavaScript dalam HTML

Gunakan sintaksis sederhana untuk memuat skrip eksternal (atribut type tidak diperlukan):

<script src="myscript.js"></script>

Referensi :

https://www.w3schools.com/js/