SEPINTAS LALU

Asas JavaScript

(untuk frontend)

Bagaimana Kursus Ini Akan Dijalankan...

Dapatkan kod sumber di
https://github.com/kidino/js-sepintas-lalu

1. Ini merupakan Crash Course / Sepintas Lalu
2. Kita takkan belajar semua perkara tentang JavaScript
3. Kita akan pelajari perkara penting untuk bermula.

Objektif dan Hasil Yang Diharapkan

1. Pelatih mula mengenali JavaScript
2. Pelatih mengetahui keupayaan dan apa yang boleh dibina dengan JavaScript
3. Pelatih boleh membina aplikasi ringkas dengan JavaScript
4. Pelatih tahu di mana untuk mencari rujukan berkaitan JavaScript

Apa itu JavaScript (frontend)

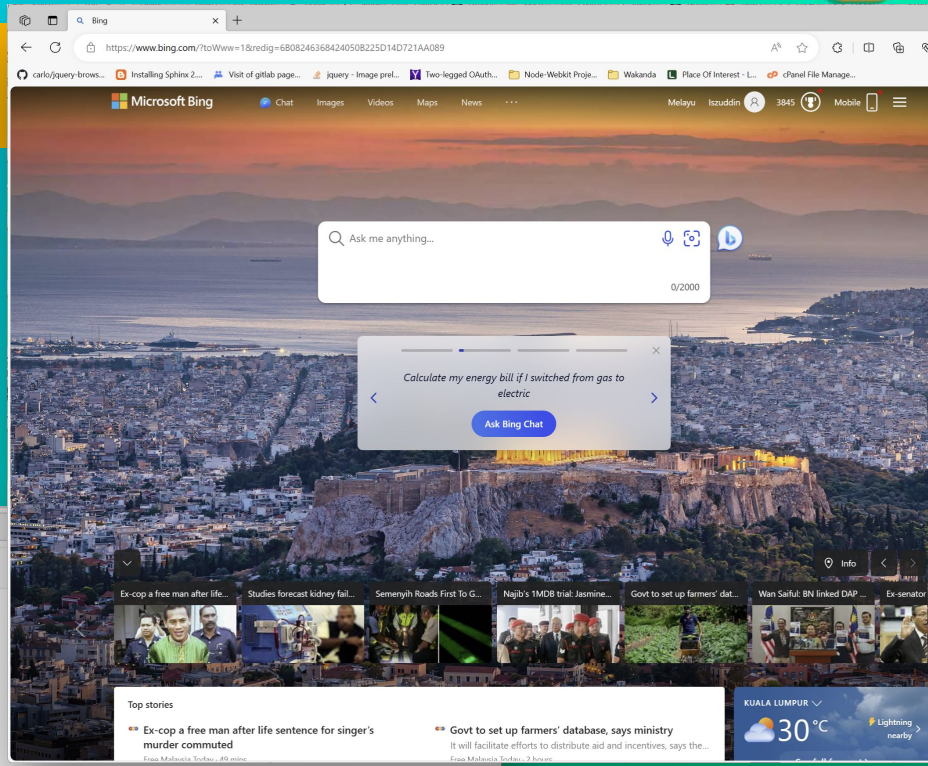
1. Bahasa pengaturcaraan yang disertakan bersama pelayar Internet seperti Google Chrome, Edge, Firefox dan lain-lain.
2. Versi server dengan NodeJS dibina oleh Ryan Dahl pada 2009.

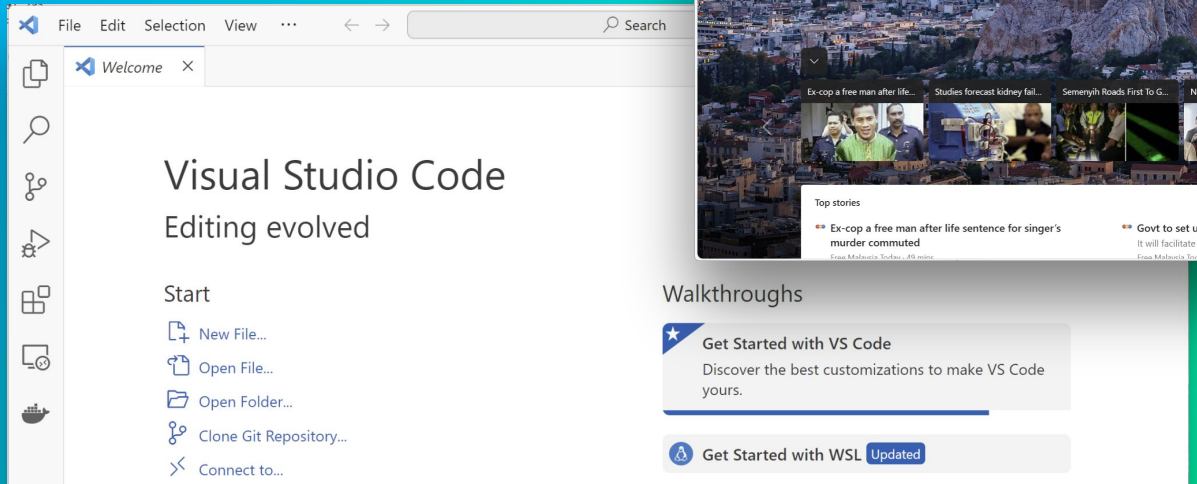
Apa itu JavaScript (frontend)

1. Tidak boleh membaca atau menulis *file* di komputer
2. Tidak boleh membuat sambungan ke database secara terus.
3. Boleh membuat sambungan ke server luar (API)
4. Hidup di dalam Pelayar Internet

Persiapan

- Pelayar Internet
  (Edge / Chrome)
- VS Code
  https://code.visualstudio.com/





Pengaturcaraan (Topik Asas)

1. 4 Cara Guna JavaScript
2. Sintaks
3. Variable
4. Operator & *Expression*
5. *Conditionals*
6. Gelung *(Loop)*
7. Fungsi *(Functions)*
8. Modal
   alert, prompt, confirm
9. Menggunakan Data
   Array, Objek & JSON
10. Manipulasi *DOM*
11. Peristiwa *(Event Handling)*
12. Module, Import, Export
13. Menggunakan *Strings*
    Concatenation, *Template Literal*
14. Menggunakan Nombor
    parseInt, parseFloat, dll

Pengaturcaraan (Topik Lanjutan)

Untuk makluman. *Tidak terkandung dalam kursus ini.*

- Object Oriented Programming
- External Communication
  Fetch, WebSocket, EventSource
- Menggunakan Tarikh & Masa
- Web API
  Camera, Geolocation, Notification, Screen Capture, History, Canvas, WebStorage, LocalStorage
- Web Worker
- WebRTC
  Multimedia Communication
- Fungsi-fungi Matematik
- Web Animation
- WebGL
- WebXR

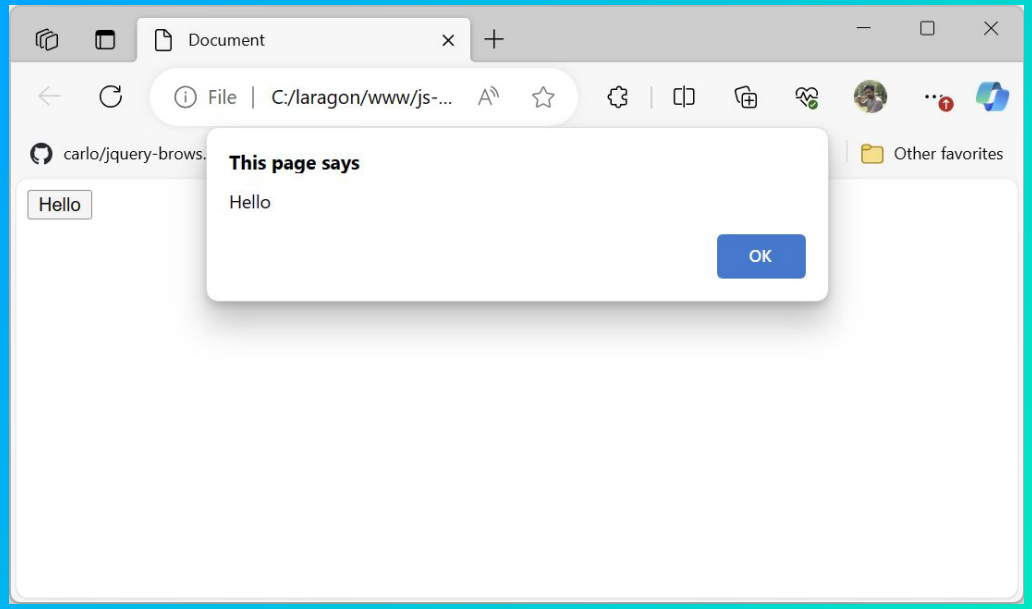
4 Cara Guna JavaScript

4 Cara Guna JavaScript

1. *Inline* - digunakan dalam HTML dengan peristiwa (event)
2. *Embedded* - digunakan dalam file HTML dengan tag <script> ... </script>
3. *External* - Kod JavaScript ditulis dalam file lain dengan ektensyen .js
4. *Console di Developer Tool* - Tekan butang F12 di kebanyakan Pelayar Internet

JavaScript Inline

```
<button onclick="alert('Hello')">Hello</button>
```



JavaScript Embedded

```
<button>Hello</button>
```

```
<script>
    let button = document.getElementsByTagName('button')[0];
    button.addEventListener('click', function(){ alert('Hello') });
</script>
```

JavaScript External

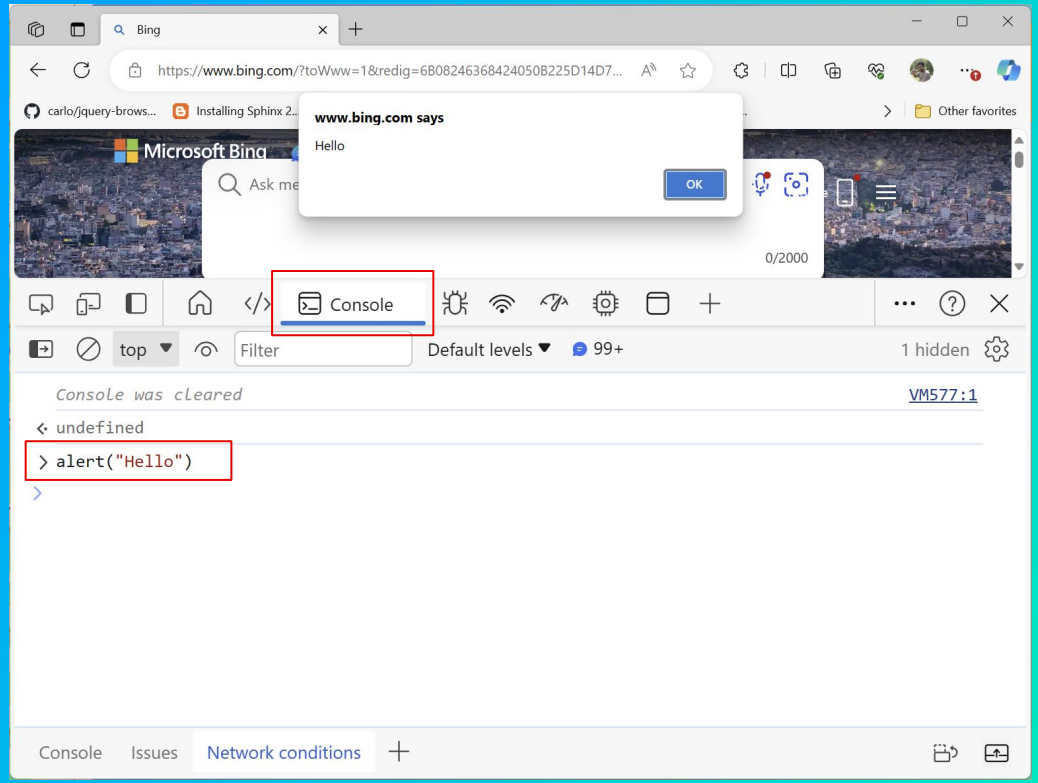
File : index.html

```
<button>Hello</button>
<script src="app.js"></script>
```

File : app.js

```
let button = document.getElementsByTagName('button')[0];
button.addEventListener('click', function(){ alert('Hello') });
```

Console di Developer Tool



- Tekan kekunci F12
- Cari butang Console
- Tulis kod di ruangan Console and tekan <enter>

Sintaks

Sintaks

1. JavaScript bersifat case-sensitive
   `Pelajar` dan `pelajar` adalah *variable* yang berbeza

```
let Pelajar = "Ali";
let pelajar = "Muthu";
```

Sintaks - Komen

```
// Ini komen satu baru -- tidak akan dilaksanakan

/* -- ini juga komen banyak baris
     alert("Hello");
*/
```

1. Komen tidak akan dilaksanakan
2. Kita boleh menulis komen dengan `//`
3. Atau dengan `/* ... */`

Sintaks - Bracket

```
function hello() {
    alert("Hello");
}
```

```
if(student == "Ali") {
    alert("Hello Ali");
}
```

```
let student = {
    name : "Ali",
    age : 18
}
```

1. Bracket { ... } digunakan untuk blok arahan
2. Biasa digunakan dengan fungsi, blok conditional (if, switch), blok gelung, data JSON dan objek.

Sintaks - Semicolon

1. JavaScript mempunyai satu ciri yang dipanggil ASI atau *Automatic Semicolon Insertion*.
2. Secara am, ASI akan meletakkan semicolon di hujung baris
3. ASI menjadikan meletakkan semicolon (;) di hujung pernyataan kod tidak wajib
4. Namun kita perlu berhati-hati dalam beberapa keadaan tertentu

Sintaks - Semicolon

Isu dengan Automatic Semicolon Insertion

```
function getPerson() {
    return
    {
      name: 'Johan'
    };
  }

console.log(getPerson());
```

ASI meletakkan semicolon di sini

Hasil : undefined

Sintaks - Semicolon

Kaedah yang lebih baik

```
function getPerson() {
    return {
      name: 'Johan'
    };
  }

console.log(getPerson());
```

ASI tidak meletakkan semicolon selepas braket {

Hasil : Johan

Pembolehubah (Variable)

Pembolehubah (Variable)

1. Menyimpan nilai untuk operasi
2. Nilai boleh diterima daripada pengguna, sumber luar, atau dinyatakan sendiri
3. Tiga cara untuk menyatakan (declare) pembolehubah
    a. `let`
    b. `var`
    c. `const`

Pembolehubah (Variable)

Penggunaan `let`

1. Nilai boleh diubah
2. Bersifat *block-scope*
3. Bersifat *function-scope*
4. Tidak boleh *declare* semula

```
// OK - nilai boleh diubah
let pelajar1 = "Abu";
pelajar1 = "Atan";

// RALAT - penggunaan pelajar2 di luar blok
{ let pelajar2 = "Aminah" }
console.log(pelajar2);

// RALAT - penggunaan pelajar4 di luar fungsi
function test() {
    let pelajar4 = "Muthu";
}
test();
console.log(pelajar4);
```

Pembolehubah (Variable)

Penggunaan var

1. Nilai boleh diubah
2. Bersifat *global-scope*
3. Bersifat *function-scope*
4. Boleh *declare* semula

```
// OK - nilai boleh diubah
var pelajar1 = "Abu";
pelajar1 = "Atan";

// OK - penggunaan pelajar2 di luar blok
{ var pelajar2 = "Aminah" }
console.log(pelajar2);

// RALAT - penggunaan pelajar4 di luar fungsi
function test() {
    var pelajar4 = "Muthu";
}
test();
console.log(pelajar4);
```

Pembolehubah (Variable)

Penggunaan const

1. Nilai boleh diubah
2. Bersifat *block-scope*
3. Bersifat *function-scope*
4. Tidak boleh *declare* semula

```
// RALAT - nilai tidak boleh diubah
const pelajar1 = "Abu";
pelajar1 = "Atan";

// RALAT - penggunaan pelajar2 di luar blok
{ const pelajar2 = "Aminah" }
console.log(pelajar2);

// RALAT - penggunaan pelajar4 di luar fungsi
function test() {
    const pelajar4 = "Muthu";
}
test();
console.log(pelajar4);
```

Pembolehubah - Jenis Data (Data Type)

1. Variable ada jenis-jenisnya. Ia tertakluk kepada bagaimana ia dibina.
2. Antara jenis data adalah :
    a. Nombor Integer
    b. Nombor Perpuluhan (float)
    c. String (rentetan aksara / teks)
    d. Boolean (true / false)
    e. Objek
    f. Array

Pembolehubah - Jenis Data (Data Type)

```
let a = 14; // ini nombor integer

let b = 3.14134; // ini nombor perpuluhan (float)

let student = 'Ali'; // ini string

let registered = false; // ini boolean

let car = { // ini objek
    "plate_no" : "AB 1234",
    "make" : "Proton",
    "model" : "Iriz"
}
```

```
// ini Array dengan kandungan String
let pokok = [
    "Cengal",
    "Jati",
    "Merbau"
];

// ini undefined
let nothing;
```

Menggunakan Nombor

1. Ada beberapa operasi yang biasa melibatkan nombor yang perlu diketahui.
2. Input dari web biasanya bersifat teks. Ia perlu diubah dengan type casting untuk membantu operasi.
3. Contoh operasi :
   a. parseInt
   b. parseFloat
   c. .toString()
   d. .toFixed()

Menggunakan Nombor

1. parseInt - menukar teks ke nombor Integer *(tanpa nilai perpuluhan)*
2. parseFloat - menukar teks ke nombor Float *(dengan nilai perpuluhan)*
3. .toString() - method pada nombor untuk diubah ke jenis string
4. .toFixed() - menetapkan ketepatan titik perpuluhan

Menggunakan Nombor - Dari Teks ke Nombor

parseInt()

```
var inputElement =
document.getElementById('intInput');
var inputValue = inputElement.value;

// Using parseInt to convert the string to an
integer
var intValue = parseInt(inputValue);
```

parseFloat()

```
var inputElement =
document.getElementById('floatInput');
var inputValue = inputElement.value;

// Using parseFloat to convert the string to a
floating-point number
var floatValue = parseFloat(inputValue);
```

Menggunakan Nombor - Dari Teks ke Nombor

.toString()

```
// boolean tidak boleh dipaparkan
// maka ia dijadikan string

var status = false;
console.log( status.toString() );
```

.toFixed()

```
var floatValue = parseFloat(inputValue);

// Menetapkan nombor pada 2 titik perpuluhan
var formattedNumber = floatValue.toFixed(2);
```

Pembolehubah - Mengenalpasti Jenis Data (typeof)

typeof

```
let berat_input = "tiga puluh kilo";
let berat = parseInt(berat_input); // akan memulangkan NaN

if (typeof(berat) != 'number') {
    alert("Input tidak sah")
}
```

Operator

Operator

1. Operator adalah simbol seperti `+ - / * = . > <`
2. Operator membantu dalam operasi matematik
3. Operator membantu mengubah nilai pembolehubah
4. Operator membantu memproses logik

Operator (Matematik)

```
x = 15 - 4;  // tolak, hasil 11
x = 15 + 4;  // tambah, hasil 19
x = 15 / 4;  // bahagi, hasil 3.75
x = 15 * 4;  // darab, hasil 60
x = 15 % 4;  // modulus (baki), hasil 3
x = 15 ** 4; // kuasa, hasil 50,625
```

Operator (Assignment, Meletakkan Nilai)

```
x = 15;   // memberi nilai 15
x += 15;  // ditambah 15
x -= 15;  // ditolak 15
x *= 15;  // didarab
x /= 15;  // dibahagi 15
x %= 15;  // operasi modulus
```

Operator (Assignment, Meletakkan Nilai)

```
x++;      // memberi nilai, kemudian ditambah 1
++x;      // ditambah 1, kemudian memberikan nilai
x--;      // memberi nilai, kemudian ditolak 1
--x;      // ditolak 1, kemudian memberikan nilai
```

Operator (Perbandingan)

```
if (x == y) { ... }  // sama dengan
if (x === y) { ... } // sama dengan, dan membandingkan data type sekali
if (x != y) { ... }  // tidak sama dengan
if (x !== y) { ... } // tidak sama dengan, dan membandingkan data type sekali
if (x > y) { ... }   // lebih besar
if (x >= y) { ... }  // lebih besar atau sama dengan
if (x < y) { ... }   // lebih kecil
if (x <= y) { ... }  // lebih kecil atau sama dengan
```

Conditional

Conditionals

1. Conditionals mengawal aplikasi mengikut situasi tertentu
2. 2 kaedah yang paling biasa adalah dengan pernyataan :
    a. if-else
    b. switch-case

Conditional - if-else

1. if () - menguji penyataan
2. else if () - menguji penyataan seterusnya
3. else - untuk lain-lain kes

```
umur = 33;

if(umur < 13) {
    console.log('kanak-kanak')
} else if ((umur >= 13) && (umur < 19)) {
    console.log('remaja')
} else if ((umur >= 19) && (umur < 56)) {
    console.log('dewasa')
} else {
    console.log('warga emas')
}
```

Conditional - switch-case

1. switch() - menyatakan variable yang untuk diuji
2. case() - menyatakan kes untuk operasi seterusnya
3. break - menghentikan operasi untuk kes-kes sebelumnya
4. default - untuk lain-lain kes

```
let saiz = 'M';
switch(saiz) {
    case 'S' :
    case 'M' :
        do_small();
        break;
    case 'L' :
    case 'XL' :
        do_normal();
        break;
    default :
        do_big();
        break;
}
```

Gelung (Loop)

Loop (Gelung)

1. Loop membantu mengulangi kod sehingga syarat tertentu
2. Jenis loop
   a. for
   b. for-in
   c. for-of
   d. while
   e. do ... while
   f. Array.forEach
   g. Array.map
   h. Array.filter

Loop (for)

```
for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

Loop (for-in)

```
const obj = { a: 1, b: 2, c: 3 };
for (let key in obj) {
  console.log(key, obj[key]);
}
```

- Untuk mendapatkan kekunci dari array atau objek

Loop (for-of)

```
const arr = [1, 2, 3];
for (let value of arr) {
  console.log(value);
}
```

- Untuk mendapatkan nilai dari array atau objek

Loop (while)

```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

Loop (do...while)

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
```

Loop (Array.forEach)

```
const arr = [1, 2, 3];
arr.forEach(function (value) {
  console.log(value);
});
```

- Digunakan bersama Array

Loop (Array.map)

```
const arr = [1, 2, 3];
const doubled = arr.map(function (value) {
  return value * 2;
});
console.log(doubled);
```

- Digunakan bersama Array
- Operasi akan memulang Array baru

Loop (Array.filter)

```
const arr = [1, 2, 3, 4, 5];
const evenNumbers = arr.filter(function (value) {
  return value % 2 === 0;
});
console.log(evenNumbers);
```

- Digunakan bersama Array
- Operasi akan memulang Array baru

Fungsi (function/method)

Fungsi (function/method)

1. Fungsi mengumpulkan kod dalam satu set untuk tujuan yang khusus
2. JavaScript mempunyai banyak fungsi-fungsi untuk pelbagai kegunaan seperti `console.log()`
3. Kita boleh menulis fungsi kita sendiri
4. Fungsi boleh menerima input (parameter)
5. Fungsi boleh memulangkan hasil

Fungsi (function)

```
function tambah ( a, b ) {
    return a + b;
}
console.log( tambah(4, 3) );
```

3 Cara Membina Fungsi

```
function salam(nama) {
    alert('Salam, '+nama);
}
```

```
const salam = function(nama) {
    alert('Salam, '+nama);
};
```

```
const salam = (nama) => { // Sintaks ECMAScript
    alert('Salam, '+nama);
}
```

Menerima Parameter (Rest Parameter)

Fungsi boleh dibina untuk menerima jumlah parameter yang tidak terhad.

```
function sum(...numbers) { // perhatikan operasi 3 titik
    let sum = 0;
    numbers.forEach(num => sum += num);
    return sum;
}

const result = sum(1, 2, 3, 4);
console.log(result); // Hasil: 10
```

Menerima Parameter (Nilai Lalai)

Fungsi boleh menerima parameter dengan nilai lalai (default) sekiranya tiada parameter diberikan.

```
function greet(nama = "Saudara") {
    console.log('Hello, '+nama);
}

greet(); // Hasil: Hello, Saudara
greet("Johan"); // Hasil: Hello, Johan
```

Modal

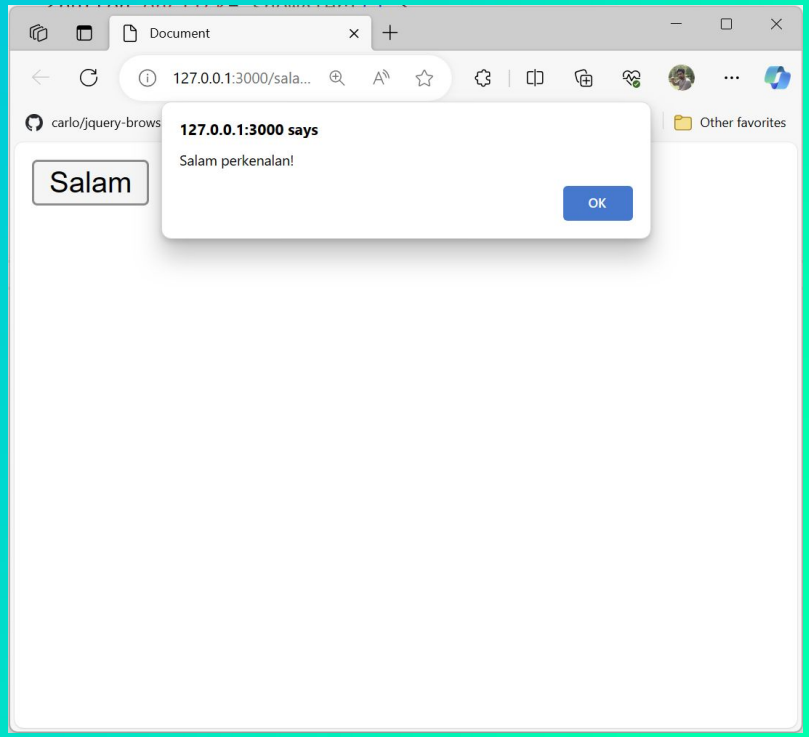
(Alert, Prompt, Confirm)

Modal

1. Modal memberikan cara mudah untuk berinteraksi dengan pengguna.
2. 3 modal yang tersedia dalam JavaScript
   a. alert() : memberi makluman kepada pengguna
   b. prompt() : meminta input daripada pengguna
   c. confirm() : meminta pengesahan

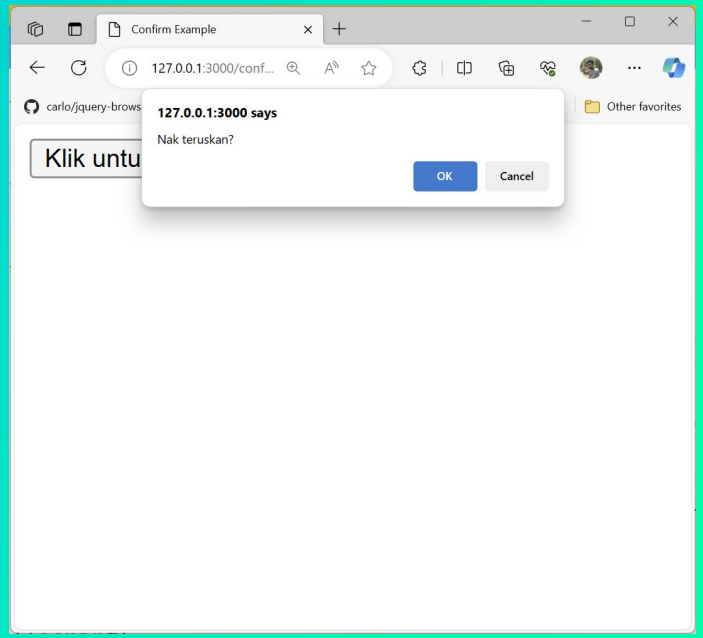
Modal - alert

```
<button onclick="showAlert()">
    Salam
</button>

<script>
    function showAlert() {
        alert("Salam perkenalan!");
    }
</script>
```



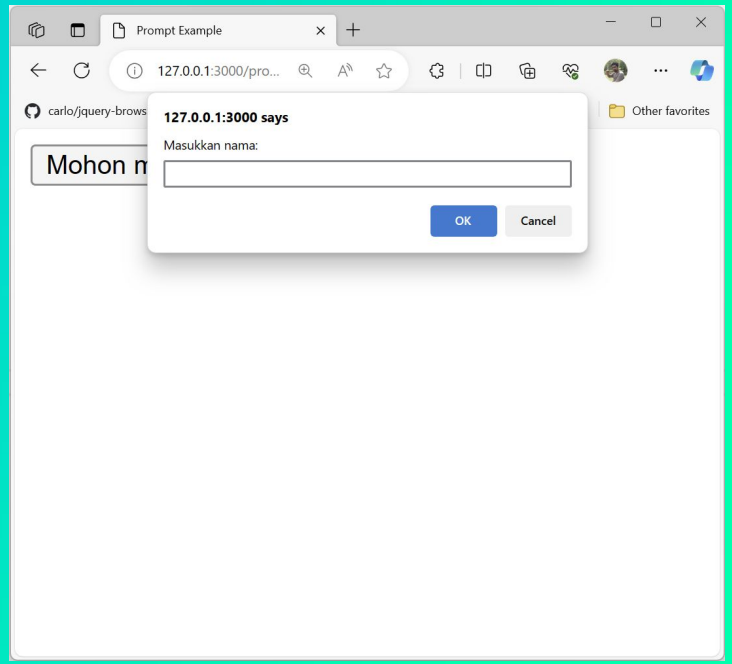
Modal - confirm

```
<button onclick="showConfirmation()">
    Klik untuk Pengesahan!
</button>

<script>
    function showConfirmation() {
        var isConfirmed = confirm("Nak teruskan?");
        if (isConfirmed) {
            alert("Anda klik OK! Mari teruskan...");
        } else {
            alert("Anda klik Cancel. Operasi dibatalkan.");
        }
    }
</script>
```



Modal - prompt

```
<button onclick="showPrompt()">Mohon maklumbalas</button>

<script>
    function showPrompt() {
        var userInput = prompt("Masukkan nama:");

        if (userInput !== null) {
            alert("Salam, " + userInput );
        } else {
            alert("Maklumbalas tidak diterima.");
        }
    }
</script>
```



Menggunakan Data
(Array)

Menggunakan Data - Array

1. Array adalah himpunan data yang disimpan dalam satu pembolehubah
2. Elemen di dalam Array biasanya adalah dari jenis data yang sama
3. Elemen di dalam Array boleh juga berjenis objek.
4. Nombor indeks untuk Array bermula dari 0

Menggunakan Data - Array

Membina & Mengakses Array

```
// Gaya literal
const fruits = ['Apple', 'Banana', 'Orange'];

// Gaya objek
const numbers = new Array(1, 2, 3, 4, 5);

console.log(fruits[0]);  // Hasil: 'Apple'
console.log(numbers[2]); // Hasil: 3
```

Menggunakan Data - Array

Mengubah Kandungan Array

```
fruits[1] = 'Mango'; // Mengubah kandungan
console.log(fruits); // Hasil: ['Apple', 'Mango', 'Orange']

fruits.push('Grapes'); // Menambah item di hujung
console.log(fruits); // Hasil: ['Apple', 'Mango', 'Orange', 'Grapes']

fruits.unshift('Pineapple'); // Menambah item di awal
console.log(fruits); // Outputs: ['Pineapple', 'Apple', 'Mango', 'Orange', 'Grapes']
```

Menggunakan Data - Array

Membuang Kandungan Array

```
fruits.shift(); // membuang item dari awal
console.log(fruits); // Outputs: ['Apple', 'Mango', 'Orange']
```

```
fruits.pop(); // membuang item dari hujung
console.log(fruits); // Outputs: ['Pineapple', 'Apple', 'Mango', 'Orange']
```

Menggunakan Data - Array

Mendapatkan Kandungan Array

```
// mendapatkan kandungan dan menghasilkan array baru
const slicedFruits = fruits.slice(1, 3);
console.log(slicedFruits); // Hasil: ['Mango', 'Orange']
```

```
// menambah di celah Array
fruits.splice(1, 1, 'Peach', 'Kiwi');
console.log(fruits); // Hasil: ['Apple', 'Peach', 'Kiwi', 'Orange']
```

Menggunakan Data - Array

Mencari Item di dalam Array

```
const index = fruits.indexOf('Kiwi');
console.log(index); // Hasil: 2
```

```
const hasKiwi = fruits.includes('Kiwi');
console.log(hasKiwi); // Hasil: true
```

Menggunakan Data - Array

1. Kita boleh membuat gelung dan mendapatkan item dari Array satu persatu.
2. Ia menggunakan fungsi
   - for-in
   - for-of
   - filter
   - forEach
   - map
3. Rujuk pelajaran Gelung (Loop)

Menggunakan Data
(Objek)

Menggunakan Data - Objek

1. Hampir setiap elemen di dalam JavaScript adalah objek. Ini termasuklah variable, string, dan lain-lain.
2. Objek boleh mempunyai salah satu ada keduanya :
   a. Property
   b. Function / Method

Membina Objek

Literal

```
const person = {
    firstName: 'John',
    lastName: 'Doe',
    age: 30,
    isStudent: false,
    sayHello: function() {
      console.log('Hello!');
    }
};
```

Object Constructor

```
const person = new Object();

person.firstName = 'John';
person.lastName = 'Doe';
person.age = 30;
person.isStudent = false;
person.sayHello = function() {
    console.log('Hello!');
};
```

Membina Objek

Object.create

```
const personPrototype = {
    sayHello: function() {
        console.log('Hello!');
    }
};

const person =
Object.create(personPrototype);
person.firstName = 'John';
person.lastName = 'Doe';
person.age = 30;
person.isStudent = false;
```

Function Constructor

```
function Person(firstName, lastName, age, isStudent) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
    this.isStudent = isStudent;
    this.sayHello = function() {
      console.log('Hello!');
    };
}

const person = new Person('John', 'Doe', 30, false);
```

Membina Objek

Singleton

```
const person = (() => {
    const privateVariable = 'I am private';
    return {
        firstName: 'John',
        lastName: 'Doe',
        age: 30,
        isStudent: false,
        sayHello: function() {
            console.log('Hello!');
        },
        getPrivateVariable: function() {
            return privateVariable;
        }
    };
})();
```

Class

```
class Person {
    constructor(firstName, lastName, age, isStudent) {
      this.firstName = firstName;
      this.lastName = lastName;
      this.age = age;
      this.isStudent = isStudent;
    }

    sayHello() {
      console.log('Hello!');
    }
}

const person = new Person('John', 'Doe', 30, false);
```

Mengakses Property

Property Objek boleh diakses dengan dua (2) cara :

```
// gaya objek property
console.log(person.firstName);

// gaya Array key
console.log(person['firstName']);
```

Menggunakan Data
(JSON)

Menggunakan Data - JSON

1. JSON - JavaScript Object Notation
2. JSON boleh berubah dari bentuk teks ke bentuk objek mengikut kesesuaian.
3. Bentuk teks memudahkan data dihantar dan diterima antara server.
4. Bentuk objek memudahkan untuk pengaturcaraan.
5. JSON juga mudah dibaca dan difahami manusia.

Menggunakan Data - JSON

Contoh-contoh JSON

Data Objek

```
{
  "name": "Johan",
  "age": 30,
  "city": "Seremban"
}
```

Mengandungi Array

```
{
  "fruits": ["apple", "orange", "banana"],
  "numbers": [1, 2, 3, 4, 5]
}
```

Objek dalam Array

```
[
   { "name": "Johan" },
   { "name": "Atan" },
   { "name": "Abu" }
]
```

Menggunakan Data - JSON

JSON.stringify() & JSON.parse()

JSON.parse()

```
let data =
'{"fruits":["apple","orange","banana"],"numbers":[1,
2,3,4,5]}';

obj_data = JSON.parse(data);

console.log(obj_data.fruit);
```

JSON.stringify()

```
let data = {
  "fruits": ["apple", "orange", "banana"],
  "numbers": [1, 2, 3, 4, 5]
}

json_str = JSON.stringify(data);

console.log(json_str);
```

Manipulasi DOM

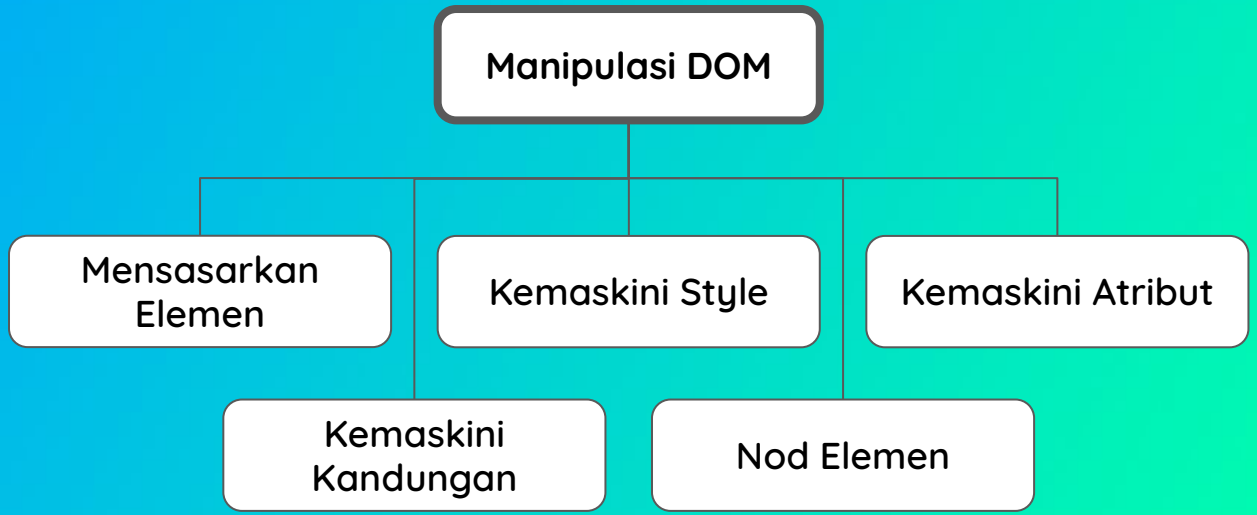
Manipulasi DOM

1. Dengan Manipulasi DOM, kita boleh mengubah kandungan di dalam halaman web.

2. Perkara yang boleh diubah :
   a. Mengubah *CSS / style* seperti `font-size`
   b. Mengubah *property* seperti `src` untuk gambar `<img>`
   c. Mengubah kandungan seperti teks dalam `<h1>`
   d. Menambah atau membuang elemen

Manipulasi DOM

Dengan Manipulasi DOM, kita boleh mengubah kandungan di dalam halaman web.



Manipulasi DOM - Mencari Elemen

Langkah Pertama : mensasarkan elemen

Memulangkan satu (1) elemen

- `document.getElementById()`
- `document.querySelector()`

Memulangkan satu atau lebih element secara Array

- `document.getElementsByClassName()`
- `document.getElementsByTagName()`
- `document.getElementsByName()`
- `document.querySelectorAll()`
- `document.forms`

Manipulasi DOM - querySelector() & querySelectorAll()

querySelector() & **querySelectorAll()** boleh digunakan dengan CSS Selector seperti mana kita menulis arahan CSS.

```
// Mensasarkan <div id="example1">...</div>
var element = document.querySelector("#example1");
```

```
// Mensasarkan kesemua <div class="example2">...</div>
var elements = document.querySelectorAll(".example2");
```

```
// Mensasarkan kesemua <p>...</p>
var paragraphs = document.querySelectorAll("p");
```

Manipulasi DOM

Memanipulasi DOM untuk banyak elemen

```
// Mencari <span class="blue-text">...</span>
var elements = document.getElementsByClassName("blue-text");

// Menukar warna untuk semua elemen, satu persatu
for (var i = 0; i < elements.length; i++) {
    elements[i].style.color = "blue";
}
```

Manipulasi DOM

Memanipulasi DOM satu element

```
// Mencari <div id="myDiv">...</div>
var myElement = document.getElementById("myDiv");

// Menukar satu elemen sahaja
myElement.style.backgroundColor = "yellow";
```

Manipulasi DOM
(Mengubah Style)

Manipulasi DOM - Mengubah Style

Manipulasi terus

```
// Selecting an element
var element = document.querySelector("#example");

// Changing style properties
element.style.color = "blue";
element.style.fontSize = "16px";
element.style.backgroundColor = "yellow";
```

Rujukan : https://www.w3schools.com/jsref/dom_obj_style.asp

Manipulasi DOM - Mengubah Style

Menambah / Membuang Class CSS

```
// Mensasarkan <div id="example">...</div>
var element = document.querySelector("#example");

// Menambah class
element.classList.add("newClass");

// Memadam class
element.classList.remove("oldClass");

// Memeriksa class
element.classList.contains("exists");
```

Manipulasi DOM - Mengubah Style

Mengemaskini className

```
// Mensasarkan <div id="example">...</div>
var element = document.querySelector("#example");

// Menetapkan class
element.className = "newClass";

// Menambah class
element.className += " additionalClass";
```

Manipulasi DOM - Mengubah Style

Mengemaskini style dengan setAttribute

```
// Mensasarkan <div id="example">...</div>
var element = document.querySelector("#example");

// Mengemaskini style dengan arahan CSS inline
element.setAttribute("style", "color: green; font-size: 18px;");
```

Manipulasi DOM - Mengubah Style

Mengemaskini style dengan cssText

```
// Mensasarkan <div id="example">...</div>
var element = document.querySelector("#example");

// Mengemaskini cssText dengan arahan CSS inline
element.style.cssText = "color: brown; font-size: 20px;";
```

Manipulasi DOM
(Mengemaskini Atribut)

Manipulasi DOM - Mengemaskini Attribute

Fungsi mengubah butang supaya boleh diklik

```
function enableInput() {
    // Mensasarkan <button id="myInput">Hantar</button>
    var inputElement = document.getElementById("myInput");
    inputElement.disabled = false;
    inputElement.value = "I am enabled!";
}
```

Manipulasi DOM - Mengemaskini Attribute

Fungsi mengubah nilai untuk <input>

```
function changeInputValue() {
    // Mensasarkan <input type="text" id="myTextInput" />
    var inputElement = document.getElementById("myTextInput");
    inputElement.value = "New Value";
}
```

Manipulasi DOM - Mengemaskini Attribute

Fungsi mengubah gambar

```
function changeImageSource() {
    // Mensasarkan <img src="..." id="myImage" />
    var imageElement = document.getElementById("myImage");
    imageElement.src = "new-image.jpg";
    imageElement.alt = "New Image";
}
```

Manipulasi DOM - Mengemaskini Attribute

Fungsi mengubah checkbox

```
function toggleCheckboxes() {
    // mensasarkan <input type="checkbox" class="myCheckbox" />
    var checkboxes = document.querySelectorAll(".myCheckbox");

    // Menukar status satu persatu
    checkboxes.forEach(function (checkbox) {
      checkbox.checked = !checkbox.checked;
    });
}
```

Manipulasi DOM - Borang

Ada beberapa *property* yang lazim digunakan dengan borang (form)

`inputText.value`	`String`
`textArea.value`	`String`
`selectOption.selected`	`True | False`
`inputRadio.checked`	`True | False`
`button.disabled`	`True | False`

Manipulasi DOM - Borang

HTML

```
<input type="checkbox" value="nasi ayam">
Nasi Ayam<br>
<input type="checkbox" value="mi goreng">
Mi Goreng<br>
<input type="checkbox" value="nasi lemak">
Nasi Lemak<br>
<input type="checkbox" value="roti canai">
Roti Canai<br>

<button onclick="getChecked()">Get
Checked</button>
<div id="info"></div>
```

JavaScript

```
function getChecked() {
    let checked_boxes =
document.querySelectorAll('input[type=checkbox]:checked');
    let info = document.getElementById('info');
    let checked_values = [];

    checked_boxes.forEach(function(checkbox){
        checked_values.push( checkbox.value )
    })

    info.innerText = checked_values.join(', ');
}
```

Manipulasi DOM
(Mengemaskini Kandungan)

Manipulasi DOM - Mengemaskini Kandungan

```
var paragraph = document.getElementById("myParagraph");
paragraph.innerText = "New content added!";
```

```
var divElement = document.getElementById("myDiv");
divElement.innerHTML = "<strong>New content added!</strong>";
```

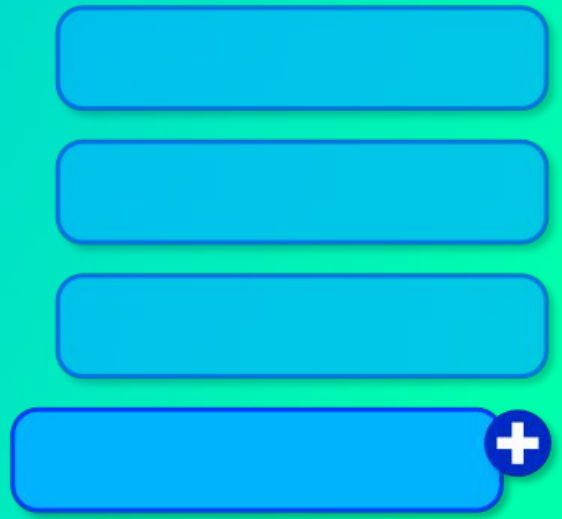
Manipulasi DOM

(Manipulasi Elemen)

Manipulasi DOM - Manipulasi Nod Elemen

Menambah elemen di hujung

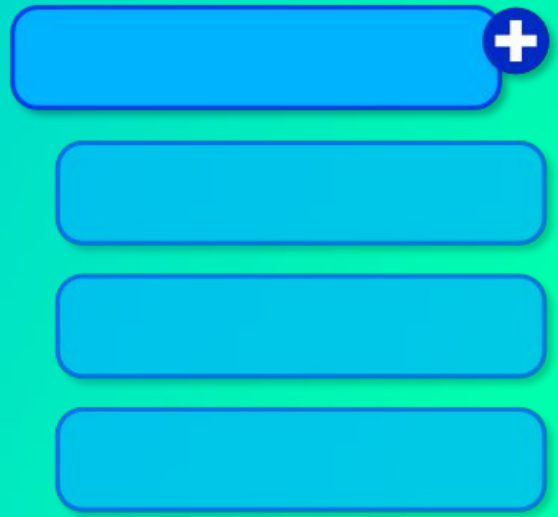
```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membina element <li> baru
var newItem = document.createElement("li");
newItem.textContent = "New Item at the End";

// Menambah <li> baru di hujung
myList.appendChild(newItem);
```



Manipulasi DOM - Manipulasi Elemen

Menambah elemen di awal

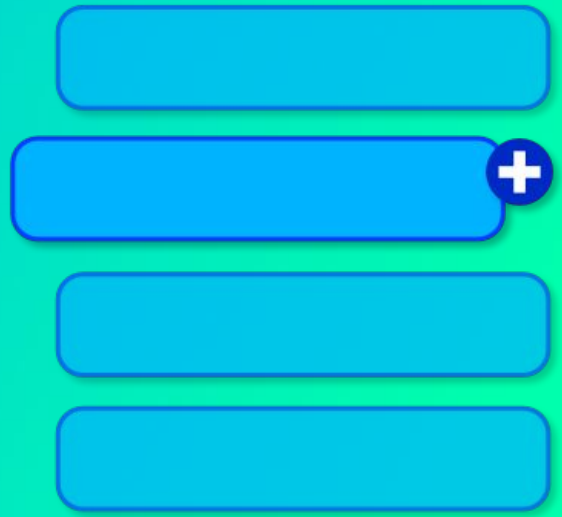
```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membina element <li> baru
var newItem = document.createElement("li");
newItem.textContent = "New Item in the Front";

// Menambah <li> baru di awal
myList.insertBefore(newItem, myList.firstChild);
```



Manipulasi DOM - Manipulasi Elemen

Menambah element di celah-celah

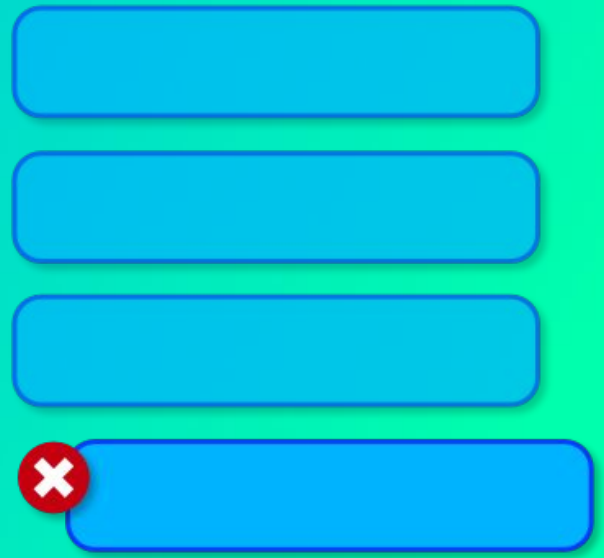
```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membina element <li> baru
var newItem = document.createElement("li");
newItem.textContent = "New Item in the Front";

// Menambah <li> baru di sebelum elemen kedua
var secondChild = myList.children[1];
myList.insertBefore(newItem, secondChild.nextSibling);
```



Manipulasi DOM - Manipulasi Elemen

Membuang elemen di hujung

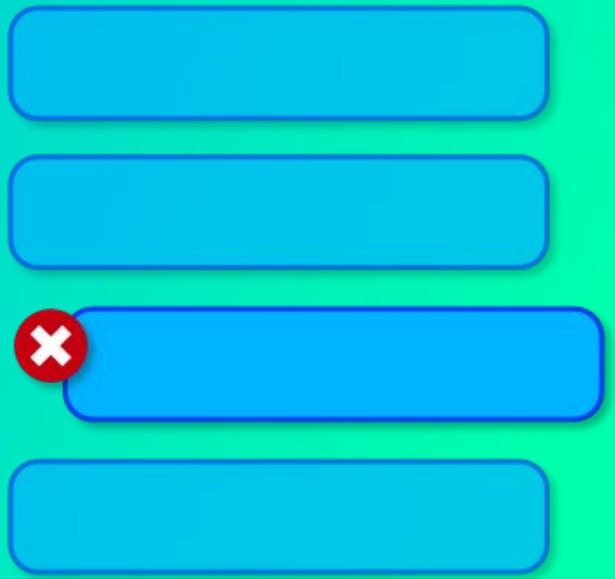
```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membuang elemen di hujung
var lastChild = myList.lastChild;
myList.removeChild(lastChild);
```



Manipulasi DOM - Manipulasi Elemen

Membuang elemen di tengah-tengah

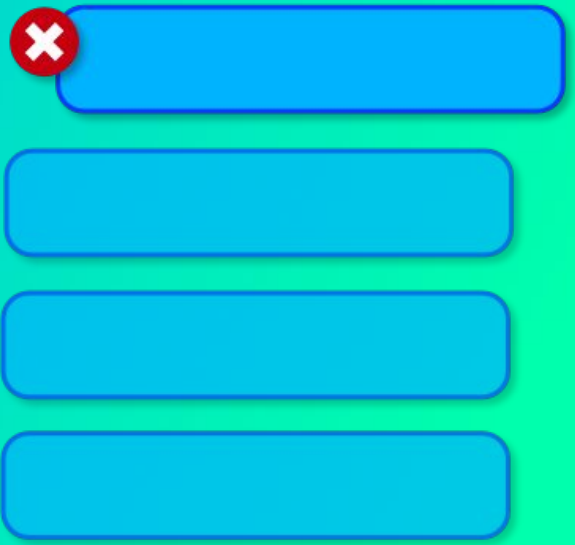
```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membuang elemen yang ketiga
var thirdChild = myList.children[2];
myList.removeChild(thirdChild);
```



Manipulasi DOM - Manipulasi Elemen

Membuang elemen di awal

```
// Mensasarkan <ul id="myList"> ... </ul>
var myList = document.getElementById("myList");

// Membuang elemen di awal
var firstChild = myList.firstChild;
myList.removeChild(firstChild);
```



Manipulasi DOM - Manipulasi Elemen

Membuang elemen itu sendiri

```
var elementToRemove = document.getElementById("myDiv");

// menyemak bahawa elemen wujud
if (elementToRemove) {
  elementToRemove.remove();
}
```

Peristiwa
(Event Handling)

Peristiwa - Event Handling

1. Kita boleh menetapkan arahan tertentu apabila sesuatu peristiwa itu berlaku
2. Contoh peristiwa (event)
   - onClick
   - onBlur
   - onFocus
   - onChange
   - onKeyUp
   - onKeyDown
   - onMouseOver
   - onMouseOut
   - onSubmit
   - onContextMenu
   - onScroll
   - DOMContentLoaded
3. Rujukan : https://www.w3schools.com/jsref/dom_obj_event.asp

Peristiwa - Event Handling

Menyatakan *event* secara inline dalam HTML

```
<button onclick="myFunction()">Click me</button>
```

Menyatakan *event* melalui elemen dengan JavaScript

```
var button = document.getElementById("myButton");
button.onclick = function() {
  // Your event handling code here
};
```

Peristiwa - Event Handling

Menyatakan *event* melalui elemen dengan JavaScript

```
var button = document.getElementById("myButton");
button.addEventListener("click", function() {
  // Your event handling code here
});
```

Peristiwa - Event Handling

Menyatakan *event* dalam fungsi berasingan

```
function myEventHandler() {
    // Your event handling code here
}

var button = document.getElementById("myButton");
button.addEventListener("click", myEventHandler);
```

Peristiwa - Event Handling

Berhenti melaksanakan *event*

```
function myEventHandler() {
    // Your event handling code here
}

var button = document.getElementById("myButton");
button.addEventListener("click", myEventHandler);

// Remove the event listener
button.removeEventListener("click", myEventHandler);
```

Peristiwa - Event Handling

Event Delegation - Event pada *parent* untuk sasaran *child*

HTML

```
<ul id="myList">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

JavaScript

```
// Mensasarkan elemen <ul id="myList">
document.getElementById("myList")
    .addEventListener("click", function(event) {
        if (event.target.tagName === "LI") {
            // Mensasarkan elemen <li>
            console.log("Clicked on",
            event.target.textContent);
        }
    });
```

Module, Import, Export

Module, Import, Export

1. Aplikasi JavaScript boleh disusun dalam beberapa file yang berasingan mengikut modul
2. Fungsi yang dibina untuk kegunaan di tempat lain memerlukan arahan `export`
3. Kod JavaScript yang ingin menggunakan fungsi dari file berbeza akan menggunakan arahan `import`
4. File JavaScript yang mengandungi import dan export perlu dipanggil dengan atribut `type="module"`

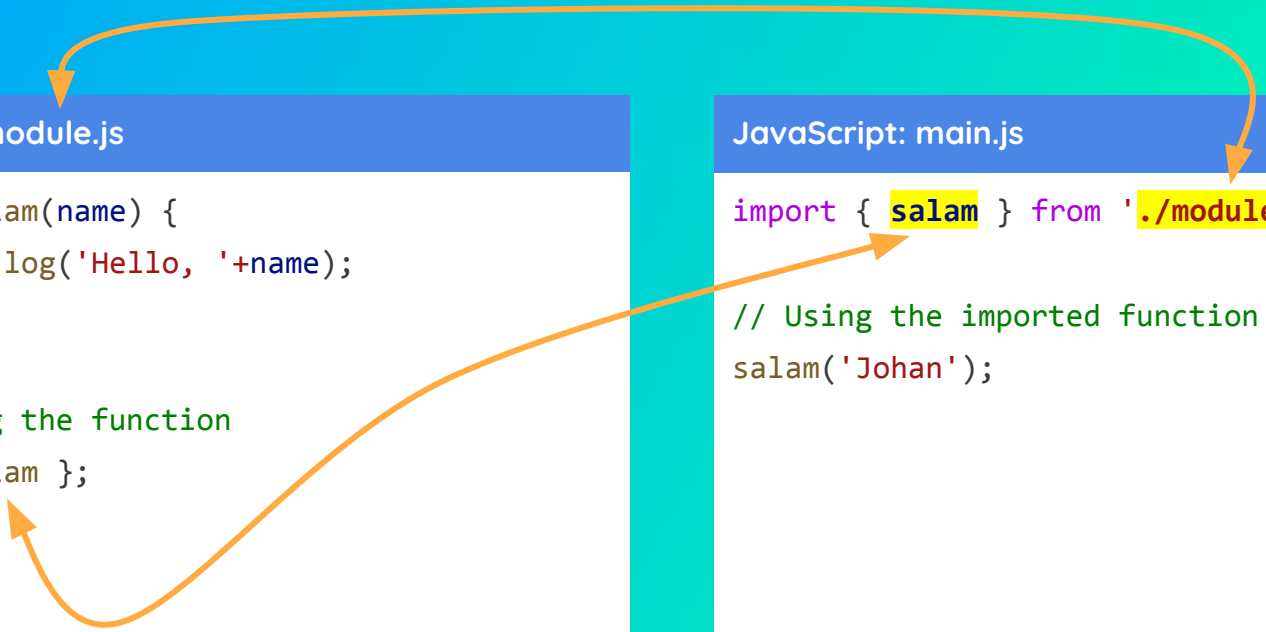
Module, Import, Export

JavaScript: module.js

```
function salam(name) {
    console.log('Hello, '+name);
}

// Exporting the function
export { salam };
```

JavaScript: main.js

```
import { salam } from './module';

// Using the imported function
salam('Johan');
```



Module, Import, Export

HTML: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Module Example</title>
</head>
<body>
  <script type="module" src="./module.js"></script>
  <script type="module" src="./main.js"></script>
</body>
</html>
```

Menggunakan String

Menggunakan String

1. Ada berapa amalan yang wajar diketahui apabila menggunakan string bersama dengan variable.
2. Dua pendekatan biasa adalah :
   a. Concatenation
   b. Template Literal

Menggunakan String - Concatenation

Menggunakan operator tambah (+), kita boleh mencantum beberapa rentetan string menjadi string yang lebih panjang.

```
// Using the '+' operator for string concatenation
var firstName = "Johan";
var lastName = "Razak";
var fullName = firstName + " " + lastName;
console.log(fullName); // Hasil: Johan Razak
```

Menggunakan String - Template Literal

1. Template Literal menggunakan simbol backtick (`) untuk membina string
2. Pembolehubah digunakan bersama ${ *nama_variable* }
3. Template Literal juga digunakan untuk string yang merangkumi beberapa baris.

Menggunakan String - Template Literal

Dengan pembolehubah dalam string

```
var age = 25;
var message = `Saya ${age} tahun.`;
console.log(message); // Output: Saya 25 tahun.
```

Template literal dengan berbilang baris

```
var multilineString = `
    This is a multiline string.
    It can span multiple lines.
    Very convenient!
`;
console.log(multilineString);
```

Menggunakan String - String Methods

1. Ada banyak lagi method / fungsi yang boleh digunakan untuk manipulasi String.
2. Antara method dan fungsi dalam String

- length
- slice()
- substring()
- substr()
- replace()
- replaceAll()
- toUpperCase()
- toLowerCase()
- concat()
- trim()
- trimStart()
- trimEnd()
- padStart()
- padEnd()
- charAt()
- charCodeAt()
- split()
- join()

3. Rujuk : https://www.w3schools.com/js/js_string_methods.asp

Terima Kasih