Belajar JavaScript

#Tipe Data

1. Number
2. String
   1. Pada string diberikan kebebasan untuk penggunaan kutip satu (‘ ‘) ataupun kutip dua (“ “). Tetapi ada kondisi dimana jika ingin menggunakan keduanya dalam deklarasi tipe data string maka harus menggunakan “***escape character***” (\’ atau \”)

contohnya: ‘“Gerakan Pungut Sampah” dilakukan pada hari Jum\’at’

1. Boolean
   1. Pada tipe dat boolean terdapat 2 jenis data yaitu “truthy” dan “falsy” dimana dari truthy adalah: (true, non-zero number, "string", object, arrays, functions) dan falsy adalah: (false, 0, "", undefined, null, NaN)
2. Object
3. Function
4. Undefined

#Operator

1. Aritmatika
2. Penugasan
3. Perbandingan
4. Logika
5. String → Binary
6. Kondisional → Ternary
7. Typeof → Unary : (kondisi) ? benar : salah → contoh: (5 % 2 == 0) ? “ganjil” : “genap” hasilnya: ‘genap’

→ Binary: membutuhkan 2 operand (*operand1* **operator** *operand2*)

→ Unary: hanya membutuhkan 1 operand (**operator** *operand* (atau) *operand* **operator**)

→ Ternary: membutuhkan 3 operand

#Pengulangan / Loop / Iteration

1. For
2. While
3. Do - While

→ Tabel Penelusuran:

#Pengkondisian (video nomor 23)

1. If
2. If.. Else
3. If.. Else If.. Else
4. Switch

#Function ***Vid 27***

* Jika pada bahasa pemrograman lain ada yang namanya Method, class, cunstructor, Module, maka di JS bisa dilakukan menggunakan ‘function’.
* Function: sebuah sub-program / sub-routine yang dapat ‘dipanggil’ dibagian lain pada program
* Function merupakan “fundamental building blocks” atau bisa disebut struktur dasar pembentuk dari JavaScript.
* Function juga disebut sebagai prosedur (kumpulan statement untuk melakukan tugas atau menghitung sebuah nilai)
* Untuk dapat menggunakan sebuah ‘Function’, kita harus membuat terlebih dahulu function tersebut lalu memanggilnya. (meskipun nantinya ada function yang dibuat sendiri ataupun function yang dibuat oleh jvaScript)
* Function juga termasuk kedalam First-Class Object.

Kenapa harus membuat ‘function’?

* Reusability (DRY: Do not Repeat Yourself)
* Dekomposisi / Abstraksi – untuk menyembunyikan kompleksitasnya yang memudahkan dalam penelusuran ‘debuging’

Kategori ‘function’

* Built-in Function
  + Fungsi yang sudah disediakan oleh javaScript
  + Kita tinggal memanggilnya saja
  + Contohnya adalah: alert(); confirm(); prompt(); dan masih banyak yang lain (cek dokumentasi javaScript)
  + Built-in Function: String
    - charAt(), slice(), split(), toSring(), toLowerCase(), toUpperCase(), dll (<https://www.w3schools.com/jsref/jsref_obj_string.asp)>
    - Contohnya: let name = ‘Sudi’; console.log(name.charAt(3)); 🡪 dimana hasilnya akan ‘I’, jika ingin memanggil 2 nilai maka harus dipanggil lagi functionnya (console.log(name.charAt(0), name.charAt(2));
  + Built-in Function: Math
    - Sin(), cos(), tan(), random(), round(), floor(), ceil(), log(), dll (<https://www.w3schools.com/jsref/jsref_obj_math.asp>)
    - Contohnya: let numb = Math.random(); console.log(numb);
* User-Defined Function
  + Fungsi yang kita buat sendiri
  + Menggunakan keyword ‘function’
  + Contoh – function contoh():
  + Didalam function kita bisa berikan parameter / argument yang bisa digunakan dalam sebuah function
    - Parameter disimpan di dalam ()
    - Boleh ada atau tidak, jika ada boleh lebih dari 1
    - Dipisahkan dengan tanda koma ‘,’
  + Function body, ‘dibungkus’ dengan {}
  + Function juga dapat mengembalikan nilai (return value) atau tidak mengembalikan nilai
* Membuat User-Deined Function
  + Dengan deklarasi / function declaration
    - Contoh:
      * function jumlahDuaBilangan(a,b){  
         let total;  
         total = a + b;  
         return total;  
        }
      * A computer code with text

        AI-generated content may be incorrect.
  + Dengan ekspresi / function expression
    - Contoh:
      * let jumlahDuaBilangan = function(a,b){  
         let total;  
         total = a + b;  
         return total;  
        }
* Memanggil / menjalankan function
  + A black background with red text

    AI-generated content may be incorrect.

#Vid 28

Output

Menghasilkan sesuatu

Function

Melakukan sesuati terhadap bahan / material

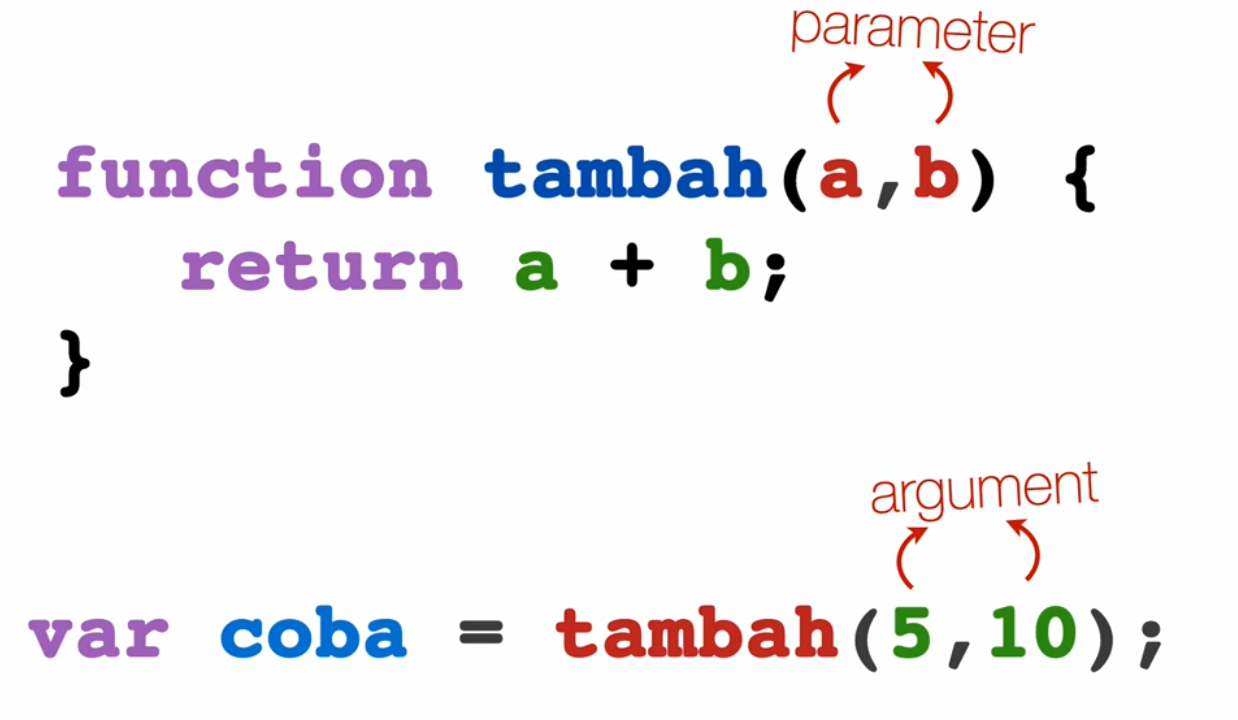
Input

Masukkan bahan / material ke dalam function

* Ketika kita bekerja dengan sebuah function, maka biasanya kita akan memiliki sebuah input / sebuah nilai yang nantinya kita akan masukkan kedalam function
* Function itu sendiri akan mengerjakan sesuatu terhadap bahan baku / material yang diinput
* Sehingga akan menghasilkan suatu output
* “Function” yang baik hanya mengerjakan 1 hal saja secara spesifik
* Function
  + Blok kode yang dibuat untuk melakukan tugas spesifik
  + Dapat dipanggul berungkali
  + Agar memudahkan dalam penelusuran
  + Reusability
* Contoh kasus:
  + Menjumlahkan volume 2 kubus
    - Ketahui sisi masing2 kubus (a: 8, b: 3)
    - Hitung volume masing2 kubus ((83 = 512) + (33 = 27))
    - Jumlahkan hasilnya (512+27)
    - Kembalikan nilai jawabannya (539)

#parameter dan argument

* Paramter: variable yang ditulis didalam kurung pada saat function dibuat, digunakan untuk menampung nilai yang dikirimkan saat function dipanggil.
* Argument: Adalah nilai yang dikirimkan ke parameter pada saat function nya dipanggil.
* Contohnya:



* **Jika** parameter **lebih sedikit dari** argument, **maka** argument **kelebihannya** **akan diabaikan -** *karekteristik dari javaScript*
* **Jika** parameter **lebih banyak dari** argument, **maka** parameter **kelebihannya akan diisi dengan nilai** undefined (nilai default dari JS). – *karekteristik dari javaScript*
* **Khusus Didalam javaScript** ada sebuah **variabel** khusus yang disebut **‘arguments’** yang definisinya adalah array yang berisi nilai yang dikirimkan saat fungsi dipanggil.

#Refactoring – vid 30

* Sebuah proses mengubah kode agar menjadi lebih ‘baik’ tanpa mengubah fungsionalitasnya
* Kenapa harus refactoring?
  + Readability 🡪 agar mudah dibaca oleh orang lain
  + DRY (Don’t Repeat Yourself)
  + Testability 🡪 penulisan kode agar lebih mudah saat nantinya dilakukan testing
  + Performance 🡪 meningkatkan kode agar lebih performance lebih baik
  + Maintainability
* Refactoring sederhana

#Scope - vid 31

* Bagaimana sebuah variabel dapat diakses dalam program, ada bahasa pemrograman yang memiliki konsep “block scope” dan ada juga yang memiliki konsep “function scope”
  + Block scope : C, java
  + Function scope: javaScript
* Pada javaScript suatu variable menggunakan konsep global scope / window scope, jadi jika **memanggil variable global** ke suatu **function itu diperbolehkan dan bisa**, tetapi jika **variable pada suatu function** (variable local) di panggil **diluar scope**, **maka hasilnya tidak bisa / error**
* Jika nama variable antara **global scope** dan **variable local (didalam function)**, maka tidak dinamakan name conflict, sehingga jika ingin mengambil varible global harus ditambahkan ‘window.*{nama variable}*’.
* Jika ingin membuat **variable local**, **harus selalu dideklarasikan** dalam sebuah function, karena javaScript akan menganggap variable local jika tidak ada deklarasi didalam function
  + Contohnya: A screen shot of a computer program

    AI-generated content may be incorrect. 🡪 ***gunakan let / const***

#Rekursif – vid 32

* Sebuah fungsi yang memanggil dirinya sendiri
  + Contoh tanpa adanya ‘**Base Case**’:  
    function test(n){  
    console.log(n); //digunakan untuk mencetak jika nilai sesuai argumen  
    return tes(n-1); //digunakan untuk mengurangi nilai argumen 1  
    }  
    tes(10);
  + Pada baris kode rekursif harus ada yang namanya **Base Case**, dimana itu adalah suatu kondisi batasan akhir jika ingin menggunakan rekursif
  + Contoh:   
    function test(n) {  
    if (n === 0) { //bisa menggunakan kurung kurawal atau langsung tulis ‘**return;**’  
    return; // tidak mengandung nilai karena untuk memberhentikan fungsi yang ada  
    }  
    console.log(n)  
    test(n-1);  
    }  
    test(10);
  + Rekursif biasanya digunakan untuk ‘**faktorial**’
    - Contoh:  
      function faktorial (n) {  
      if ( n === 0 ) return 1;  
      return n \* faktorial (n-1);  
      }  
      faktorial (5);
  + Semua looping bisa dibuat rekursif, tetapi tidak sebaliknya
* Implementasi Rekursif digunakan untuk:
  + Menggantikan looping
  + Fibonacci
  + Pencarian dan penelusuran pada struktur data list dan tree
  + Bahasa pemrograman yang tidak memiliki pengulangan (Haskell, Erlang, Prolog, dll)

Belajar React

JS & React

- DOM (Document Object Model) adalah memodelkan struktur code html menjadi hirarki sesuai dengan struktur HTML

Strukturnya seperti berikut

HTML → DOM → UI

***p.s*** Representasi object dari element HTML pada halaman web

- JavaScript (Vanilla JavaScript) digunakan untuk memanipulasi html dengan penulisan script (step-by step)

- React sebuah library JS untuk mempermudah dalam memanipulasi html dengan konsep declarative, dimana hal tersebut adalah konsep menuliskan apa yang akan dihasilkan

- contohnya saat kita memesan pizza sesuai menu tanpa menghiraukan langkah demi langkah

# Memulai React

- untuk menggunakan react dalam project kita, kita butuh 2 script react yang bisa kita panggil lewat unpkg.com (via CDN)

- untuk penggunaan React **via CDN** terbatas pada **pembelajaran** dan konsep project yang sangat kecil, dikarenakan untuk file CDN sangat **terbatas fiturnya**.

- untuk penggunaan CDN hanya perlu penggunaan **script React dan React-DOM**, dan juga **ditambah** dengan **script untuk babel**.

<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<script src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

Untuk contoh penggunaan babel sebagai berikut:

    <script type="text/babel">

        const container = document.getElementById('root');

        const tittle = ReactDOM.createRoot(container);

        tittle.render(<h1>Belajar React 🚀</h1>);

    </script>

- untuk keperluan pembelajaran lanjutan perlu untuk penggunaan vite (npm) dan **harus instalasi dan konfigurasi terkait container reactnya**.

#JSX

- JSX adalah ekstensi sintaks untuk javascript yang dapat mendeskripsikan UI dengan sintaks seperti HTML

- Meskipun masih ada cara lain untuk membuat komponen, kebanyakan developer React menggunakan JSX

- Sintaks JSX terlihat seperti HTML, tetapi lebih strict dalam penulisannya dan bisa menampilkan informasi yang dinamis

- Untuk menulisnya, kita harus ingat 3 aturan JSX

- https://react.dev/learn/writing-markup-with-jsx

ketentuannya sebagai berikut:

1. Selalu mengembalikan satu elemen

2. Selalu tutup semua tag (contoh <img src="img.png" alt="myImage" />

3. Penggunaan camelCase untuk atribut

#Prerequisite React

1. HTML & CSS

2. Javascript Fundamentals

3. DOM

4. Modern JavaScript (ES6)

5. Git & CLI (Command Line Interface)

6. Package Manager (NodeJS + NPM)