**DOKUMENTASI**

**ELODE.JS**

Panduan Penggunaan Library

**v1**

MUHAMMAD ALFAJRI ARRAIHAN

**1. PROSEDUR ( )**

**Elode.js** adalah prosedur pembuatan **Elemen HTML** dengan cara unik! Tanpa menggunakan Virtual DOM dan ini benar-benar ringan. Hanya 11-25kb untuk source dan minify.

**Dasar Prosedur**

Elode(query); Elode(query, property);

* **query** –> berisikan perintah prosedur
* **property** –> isi konten dari prosedur

Dalam penggunaan library **Elode.js**, kalian akan banyak menggunakan fungsi **Elode** sebagai prosedur untuk membuat suatu Elemen HTML.

**Contoh:**

Elode("h1 Halo Dunia ");

Elode("h1 Hai Dunia", {id: "judul"});

**Prosedur Lanjutan**

Elode(query).render();

* **render** –> fungsi untuk menampilkan Elemen HTML ke dalam komponen <body></body>

Prosedur lanjutan berada di dokumentasi **DEKLARATIF**

**2. SINTAKS UNIK <>**

Kode sintaks unik yang dihadirkan **Elode.js** untuk templating Elemen HTML dalam kode Javascript berbasis DOM (Document Object Model).

**Dasar Kode Sintaks**

Elode("h1 Halo Dunia").render();

* **h1** –> tag Elemen HTML <h1></h1>
* **Halo Dunia** –> isi konten dari Elemen HTML

Terdapat h1 dengan spasi sebagai separator/pembatas antar isi konten elemen “Halo Dunia”. (error jika tidak ada spasi)

**Mendalami Sintaks**

var nama = "Alfajri Arraihan";

Elode("h1 Halo Dunia "+nama).render();

* **nama** –> variabel string tidak null

Kode sintaks unik dapat divariasikan dengan berbagai variabel dalam kode Javascript. (int, boolean, dan lainnya)

var tag = "p", isi = "Apa Kabar? ", iya = true;

Elode(tag+" "+isi+" "+iya).render();

**Hasil :** Apa Kabar? true

**Sintaks Elemen**

Elode("h1 Halo Elemen").render();

Elode("p Belajar Elode.js").render();

**Hasil :**

**Halo Elemen**

Belajar Elode.js

Dalam pembuatan suatu Elemen HTML, prosedur sintaks hanya berlaku untuk satu tag saja.

**Sintaks Komponen**

Elode("div <h1 Halo Komponen> <p Belajar Membuat

komponen! dalam <b Elode.js> >").render();

**Hasil :**

**Halo Komponen**

Belajar membuat komponen! Dalam **Elode.js**

Dalam pembuatan suatu Komponen untuk Elemen HTML, prosedur sintaks dapat berlaku untuk multi tag (child) dalam satu query. Sebagaimana pembatasnya adalah simbol < dan >

**<tag \_spasi\_ isi konten>** dengan sintaks ini Elode dapat membuat beberapa elemen dalam satu wadah.

**Atribut [ ] Dalam Sintaks**

Sintaks unik menyediakan fitur atribut layaknya query dalam CSS Selector. Ada beberapa fitur sintaks atribut yaitu:

**ID – Identitas Elemen**

Elode("h1#judul Ini Judul Bro").render();

<h1 id=“judul”>Ini Judul Bro</h1>

* **tag#id** -> tata letak id dalam sintaks dekat dengan tag elemen. (bug jika ada spasi antara tag dan #id)

**Class – Properti Kelas**

Elode("button.btn.biru Download").render();

<button class=“btn biru”>Download</button>

* **.btn** -> ini adalah salah satu dari nama kelas dalam properti CSS yang merujuk kepada style button/tombol.
* **.btn.biru** -> kode sintaks harus menyambung dengan titik di setiap awalnya, tanpa ada spasi/jarak di setiap nama kelas. (bug jika ada spasi antara . titik dan nama kelas)

**Lanjutan – Sintaks Atribut**

Sintaks lanjutan atribut dapat kalian gunakan selain id dan kelas. Dengan prosedur sintaks yaitu simbol kurung kotak [ ]

**[atribut \_spasi\_ isi atribut]** -> aturan sintaks atribut

**Untuk Menambahkan Style :**

Elode("h1[style color:blue] Hai Biru").render();

<h1 style=“color:blue”>Hai Biru</h1>

**Memberikan EventListener :**

Elode("button[onclick alert(‘hai’)] Hai").render();

<button onclick=“alert(‘hai’)”>Hai</button>

**Dan Menggunakan Placeholder :**

Elode("input[placeholder Username]").render();

<input placeholder=“Username”>

Semua aturan dalam kode sintaks atribut [ ] dapat kalian gandakan. Dengan simbol pembatas/jarak antar atribut.

**[atribut1 \_spasi\_ isi atribut1 | atribut2 \_spasi\_ isi atribut2]**

sintaks atribut di atas adalah penggunaan multi atribut dalam satu wadah dengan pembatas | .

**Contoh :**

Elode("a[href https://www.google.com|style color:blue] Link Buka Google").render();

<a href=https://www.google.com” style=“color:blue”>

Link Buka Google</a>

**Javascript { } Dalam Sintaks**

Sintaks unik mampu menjalankan implementasi eksekusi kode Javascript eval(code); dalam kode query. Dengan ini, terciptalah sebuah fungsi Reaktivitas!

* Apa itu **Reaktivitas? –** itu adalah fungsi untuk template suatu kode Javascript dalam sintaks HTML.
* Bagaimana template kode Javascript bekerja? Jawabannya adalah **simbol kurung kurawal** { } yang akan menjadi dasar sintaks kode Javascript.

**Sintaks Operasional:**

Elode("h1 Hitung 1 + 1 = {1 + 1}").render();

<h1>Hitung 1 + 1 = 2</h1>

**Sintaks Logikal:**

Elode("h1 Apakah benar {1 == 2}").render();

<h1>Apakah benar false</h1>

**Sintaks Variabel:**

var nama = "Alf Equilfe", kode = "123";

Elode("h1 Hai {nama}! Kode kamu {kode}").render();

<h1>Hai Alf Equilfe! Kode kamu 123</h1>

Dan masih banyak lagi yang dapat kamu lakukan dengan sintaks template kode Javascript pada query Elode.

**Peraturan Sintaks <!>**

Ada sebuah peraturan simbol sintaks unik yang jika kamu tidak mengikutinya. mungkin akan berpotensi sebuah **BUG** mulai dari level ringan sampai fatal.

* **Ini penting!** Dan akan mengakibatkan bug yang fatal jika kamu tidak mengikuti aturan sintaks dasar ini. SETIAP SIMBOL KURUNG, BAIK ITU **< > KOMPONEN**, **[ ] ATRIBUT** DAN **{ } JAVASCRIPT**. HARUS ADA PEMBUKA DAN PENUTUP DALAM SINTAKS. AGAR TIDAK ADA BUG NO CLOSING SYMBOL ATAU SYMBOL NOT FOUND.
* Perlu diingat, bahwa pembatas/jarak antar tag dan isi konten dalam sintaks Elemen HTML **harus ada spasi**.

h1Hai Dunia <- salah h1 Hai Dunia <- benar

* Komponen < > berada pada isi konten dalam sintaks unik. Jika letaknya berada di tag itu salah. div<p Apa kabar?><marquee Saya baik> ini salah **harus ada jarak spasi** antara simbol < > komponen dengan tag agar dapat terbaca sebagai bagian node/child dari root/parent. div <p Benarkah?><marquee Iya benar> ini benar
* Sintaks **#id** tidak boleh ada spasi seperti h1# judul <- ini salah, karena nanti “judul” akan menjadi isi konten.

Dan tidak disarankan menggandakan #id seperti

h1#id1#id2 Hai Dunia<- karena yang akan terbaca hanya id2 saja dan id1 akan diabaikan.

* Nama kelas **.class** harus menyatu (tidak ada spasi) antara nama kelas lainnya. .biru . border <- ini salah. Seharusnya nama kelas dan titik **saling berdempetan** .biru.border <- ini benar.
* **Javascript { }** berada pada isi konten dalam sintaks. Tidak akan terbaca bila berada pada tag. Kecuali tag yang ada pada node/child dalam suatu wadah komponen. **Contoh:**

Tag h1 dengan kode Javascript { } menyatu/tidak ada spasi.

h1{nama} kamu menang! <- tidak terbaca

Kode javascript { } berada pada tag node/child.

h1 <span{atribut} kustom atribut> oleh {nama}

* **SIMBOL DALAM ISI KONTEN SINTAKS** adalah suatu kasus di mana simbol sensitif yang ada pada sintaks (terdeteksi sebagai bagian sintaks) akan menjadi bug bila penggunaannya untuk isi konten. **Contoh:**

h1 Aku tidak tau (>w<) ini akan terbaca sebagai simbol < > komponen. Maka dari itu, pencegahannya adalah dengan memasukan string kode Javascript { }.

h1 Aku tidak tau {‘(>w<)’} ini akan terbaca sebagai isi konten dari kode simbol Javascript { } dengan tanda kutip sebagai pembatas untuk string.

**3. DEKLARATIF (</>)**

**Prosedur Elode** itu deklaratif! Kamu dapat membuat variasi elemen dan komponen sebagai template atau objek berbasis HTML-DOM yang dapat di kembangkan secara berulang.

**Prosedur Deklaratif**

var nama = Elode("h1 Alfajri Arraihan"),

bio = Elode("p Indie Game Developer");

nama.render(); bio.render();

Variabel **nama** dan **bio** di deklarasikan sebagai prosedur **Elode** untuk pembuatan Elemen HTML **h1** dan **p** setelah itu di render secara berurutan.

**Hasil :**

**Alfajri Arraihan**

Indie Game Developer

Kamu dapat **memvariasikan** urutan render dengan mengubah susunan awal kode:

bio.render(); nama.render();

**Hasil :**

Indie Game Developer

**Alfajri Arraihan**

**Struktur Komponen ([ ])**

var website = Elode(["div",

"h1 Judul Website",

"p Deskripsi Website"

]);

website.render();

**Hasil :**

**Judul Website**

Deskripsi Website

Berdasarkan kode di atas. **Struktur Komponen** adalah prosedur untuk membuat komponen tanpa simbol < > komponen dalam sintaks. Digantikan dengan parameter **Array** pada **Elode(query):**

* Elode(query) -> Elode("h1 Halo Dunia");

Untuk prosedur query string,

* Elode(array) -> Elode(["div", "h1 Halo Dunia"]); Untuk prosedur query array.

**Aturan Struktur Komponen**

Elode([root, node1, node2, … node?]);

Elode([parent, child1, child2, … child?]);

Entitas array [0] adalah root/parent dan array[1] seterusnya adalah node/child tersusun berurutan [1,2,3,4,5].

* Entitas Array [0, 1, 2, 3, … n] dapat berupa string maupun Elemen HTML dan bahkan prosedur Elode.
* root/parent tidak boleh null. Yang berarti keberadaan entitas [0] harus ada dan terisi.
* Entitas Array tidak boleh undefined/null karena akan berpotensi error element creation null

**Variasi Struktur Komponen**

var nama = Elode("h1 Alf Equilfe"),

bio = Elode("p Digital Illustrator");

Elode(["div#UserData", nama, bio]).render();

Kode di atas adalah **contoh variasi deklaratif** dari suatu struktur komponen untuk menampilkan data user.

Di mana, variabel prosedur Elode yaitu: **nama** dan **bio** di masukan ke dalam entitas **Array** sebagai node/child dari prosedur struktur komponen.

**Hasil :**

**Alf Equilfe**

Digital Illustrator

**Komponen Deklaratif (a,b,..){}**

function BlogPost(judul, deskirpsi){

return Elode(["div.post",

"h1 "+judul,

"p "+deskripsi

]);

}

BlogPost("Tutorial Javascript","Belajar Pemrograman Dasar Javascript").render();

BlogPost("Upgrade Windows 11","Cara Upgrade Windows 10 Ke Windows 11 Tutorial Lengkap").render():

**Hasil :**

**Tutorial Javascript**

Belajar Pemrograman Dasar Javascript

**Upgrade Windows 11**

Cara Upgrade Windows 10 Ke Windows 11 Tutorial Lengkap

Berdasarkan kode di atas. **Komponen Deklaratif** adalah struktur komponen yang di deklarasikan sebagai fungsi. Yang di mana, fungsi tersebut **dapat digunakan kembali** (deklaratif) dan divariasikan sesuai data yang ada.

**BlogPost** adalah fungsi dengan 2 data (parameter) yaitu **judul** dan **deskripsi**. Dengan judul adalah data untuk **elemen h1** dan deskripsi untuk **elemen p**.

BlogPost(judul, deskripsi) <- **prosedur**

Elode([root, “h1 “+judul, “p “+deskripsi]) <- **hasilnya**

Jika kamu mengikuti alur prosedur Komponen Deklaratif. Maka parameter fungsi akan menjadi **data masukan** untuk membuat suatu struktur komponen.

**Penggunaan Komponen Deklaratif**

function Button(teks, warna){

return Elode("button[style color:"

+warna+"] "+teks);

}

Elode(["div",

Button("Kirim","blue"),

Button("Batal","red")

]).render();

**Hasil :**

**Batal**

**Kirim**

Jadi penggunaan komponen deklaratif **akan sangat membantu dalam membuat suatu UI/UX** dengan data yang dibutuhkan, secara berulang dan dapat divariasikan ke dalam wadah komponen lain. **Tanpa harus menulis ulang kode prosedur yang sama.**

**Properti ({ })**

Setiap elemen/komponen dalam Elode memiliki parameter **properti** setelah query. Dengan adanya properti, maka data yang ada dalam suatu wadah (**DOM**) dapat kita manipulasi.

**Dasar Properti**

Elode(query, property); -> Elode(string, object);

* Property adalah entitas object yang dapat menampung banyak jenis entitas lain.
* Parameter property **boleh null**, karena menyesuaikan keadaan prosedur.

**Memasukan ID:**

Elode("h1 Halo Teks", { id:"teks" });

**Menambahkan Style:**

Elode("h1 Halo Biru", {

style:{ color:"blue" }

});

**Memberikan EventListener:**

Elode("h1 Halo Dunia", {

onclick (){

alert("Hai Dunia");

}

});

Jadi, pada dasarnya, properti adalah tempat untuk data suatu komponen dimana **sifatnya adalah rahasia (private)** tidak dapat diakses secara global.

**Properti vs Variabel**

Jika kamu perhatikan kembali di bab **Sintaks Unik.** Penggunaan **variabel** untuk memasukan data secara langsung lewat sintaks adalah penggunaan yang tidak disarankan karena, akan menyebabkan sintaks error.

Maka dari itu, fungsi dan manfaat **properti** dapat mencegah terjadinya error pada sintaks tersebut. Dengan fungsi **Reaktivitas!**

**Data Variabel:**

var nama = "Alfajri Arraihan";

Elode(“h1 Benarkah Nama Kamu "+nama+"?").render();

**Data Properti:**

Elode("h1 Salam Kenal {{nama}}!",{

nama:"Alfajri Arraihan"

}).render();

**Hasil :**

**Benarkah Nama Kamu Alfajri Arraihan?**

**Salam Kenal Alfajri Arraihan!**

**Mendalami Properti**

Kamu sudah memasuki tahapan lanjut dari fungsi properti pada komponen. Sekarang, bagaimana fungsi **properti** secara kompleks dapat **dijadikan data** untuk komponen? Atau properti dapat dimanfaatkan sebagai fungsi lain?

* Properti dalam komponen adalah tipe data object JavaScript. Di mana, fungsinya adalah menampung data dengan syarat 2 sintaks yang harus ada yaitu. **KEY** dan **VALUE**.
* Dengan adanya properti, suatu komponen akan lebih mudah mengakses data dengan aman tanpa adanya bentrokan data lain. Dalam hal ini, properti **bersifat private atau rahasia**.
* Semua properti dalam komponen dapat di ubah datanya dan bersifat reaktif. Jadi tidak perlu lagi mengubah komponen berturut turut.

**Membuat Perhitungan:**

Elode(["#Perhitungan",

"p Jika Andi memiliki {{x}} buah dan Budi memiliki {{y}} buah <br> Maka berapa banyak buah keduanya?",

"p <b Jawaban>: {{x} + {y}}"],{x:8, y:7}).render();

**Hasil :**

Jika Andi memiliki 8 buah dan Budi memiliki 7 buah

Maka berapa banyak buah keduanya?

**Jawaban**: 15

**Membuat Teks Diskusi:**

Elode(["#TeksDialog",

"p {{b}}: {{a}} besok gimana nih!?",

"p {{c}}: Iya gimana {{a}}? udah siap nih.",

"p {{a}}: Besok ready dong. Kita ke rumah {{b}}...",

"p {{b}}: Tapi {{c}} jangan terlambat ya.",

"p {{c}}: Hehe... kalo bangun pagi."],{

a:"Andi", b:"Budi", c:"Cahyo"

}).render();

**Hasil :**

Budi: Andi besok gimana nih!?

Cahyo: Iya gimana Andi? udah siap nih.

Andi: Besok ready dong. Kita ke rumah Budi...

Budi: Tapi Cahyo jangan terlambat ya.

Cahyo: Hehe... kalo bangun pagi.

Dan masih banyak lagi yang dapat dilakukan dengan properti.

**Properti Terhadap JavaScript { }**

**JavaScript** sebagai **Reaktivitas** pada komponen dengan sintaks unik memiliki peran penting dalam membangun suatu bentuk **Reaktif** yang lebih kompleks. Sebagaimana sifat properti terhadap fungsi JavaScript:

**Bersifat reaktif dalam bentuk Sintaks Unik:**

"h1 {{sintaksUnik}} {{hai}}" -> { sintaksUnik:"<i Halo Dunia>", hai:"Hai Dunia" }

**Termasuk Jenis Variabel:**

"h1 Hasil {{a}+{b}+{c}} " -> { a:1, b:2, c:3 }

"h1 Nama {{a}+{b}} " -> { a:"Andi", b:"Budiman" }

"h1 Apakah benar? {{a} == {b}} " -> { a:1, b:2 }

**Properti Adalah Prototype DOM**

var h1 = Elode("h1 Im Red and Bordered",{

style:{ color:"red", border:"1px solid red"}

});

h1.style.fontSize = "18pt";

Properti adalah bagian dari prototype suatu elemen (DOM) dalam **HTML-DOM JavaScript**. Yang berarti sifatnya benar-benar mengikuti alur kerja DOM itu tersendiri. Maka dari itu setiap Elemen Elode merupakan **DOM-Based.**

Jika kamu mencoba mendeklarasikan Elemen Elode maka saat memasukannya ke suatu komponen DOM akan terlihat hasilnya bahwa **Elemen Elode** memang berasal dari **DOM** yang di manipulasi.

var div = document.getElementById("contoh");

div.appendChild(h1); // 🡨 Memasukan Elemen Elode

//Alternative

h1.render(div) // 🡨 Render ke DIV

h1.render("#contoh"); // 🡨 Render ke <id='contoh'/>

**Render (</>)**

**Render** adalah bentuk perintah yang digunakan saat ingin menampilkan suatu elemen/komponen pada **Elode.js** Penggunaannya sangat berpengaruh terhadap proses berjalannya elemen/komponen.

**Dasar Prosedur Render**

Elode("h1 Halo Elode.js").render();

var h1 = Elode("h1 Halo Render");

var p = Elode("p Belajar Render Elode.js");

h1.render(); p.render();

Jika kamu perhatikan lebih jelas, **render()** merupakan perintah murni dari **Elemen Elode**. Dan termasuk ke dalam jenis prototype prosedur. elemen.render();

**Render Target**

Prosedur render secara default akan memasukan Elemen Elode ke dalam komponen <body></body>. Jika kamu ingin memasukannya secara kustom (target) maka parameter prosedur render adalah jawabannya:

.render(target); -> .render(querySelector);

Elode("h1 Halo Dunia").render();

Elode("h1 Targetkan Ini").render("#app");

var root = document.getElementById("app");

Elode("h1 Elemen Teratas").render(root);

**Hasil:**

<!DOCTYPE html>

<html>

<head>…</head>

<body>

<div id="root">

<h1>Elemen Teratas</h1>

</div>

<div id="app">

<h1>Targetkan Ini</h1>

</div>

<h1>Halo Dunia</h1>

</body>

</html>

**Render Global**

Elode.js merender secara global merupakan cara untuk menyatukan seluruh komponen ke dalam satu wadah global yaitu <body></body>

Elode.render(

"h1 Halo Global",

"p Belajar Global Rendering",

SuatuKomponen(),

CounterButton(),

"footer Copyright © 2022 - Elode.js"

);

// Elode.render(elemen…);

// @param elemen adalah parameter argumen