JAVASCRIPT

part 2

01

Document Object Model (DOM)

02

Element, NodeList, and HTMLCollection

03

DOM Selection

04

DOM Manipulation

05

DOM Events

06

DOM Traversal

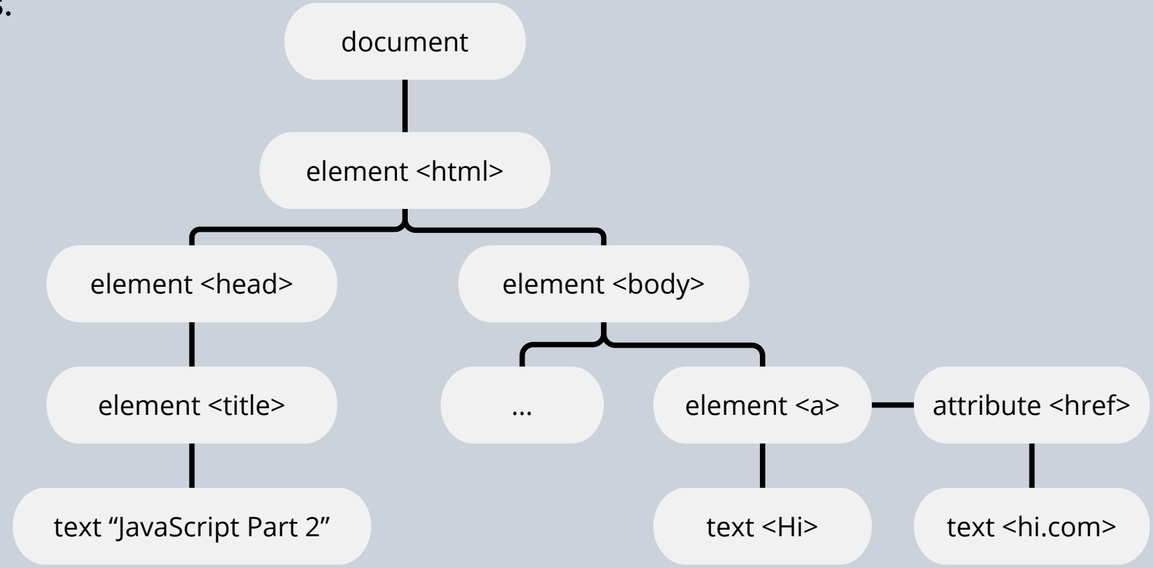
07

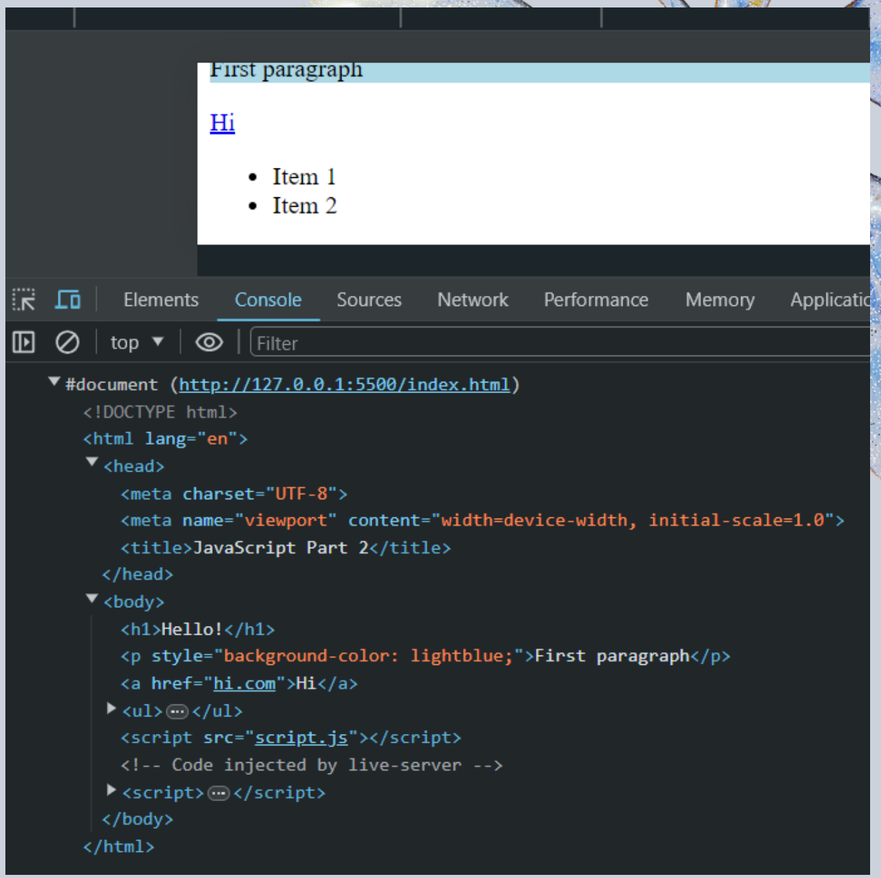
Event Bubbling & Capturing

DOCUMENT OBJECT MODEL (DOM)

DOM merupakan representasi dari elemen-elemen yang menyusun sebuah website dalam bentuk object yang dapat diakses melalui JavaScript.

DOM Tree merupakan susunan hirarki dari DOM yang terdiri dari nodes.





Prompt: DOM in JS

ELEMENT, NODELIST, AND HTMLCOLLECTION

Elemen merupakan sebuah node dengan tipe tertentu

NodeList merupakan kumpulan dari berbagai tipe nodes, direpresentasikan dalam bentuk array

HTMLCollection merupakan kumpulan dari node dengan tipe yang sama, direpresentasikan dalam bentuk array.

```
HTMLCollection(4) [p.p1, p.p2, p.p3, p]
  0: p.p1
  1: p.p2
  2: p.p3
  3: p
  length: 4
```

```
1  <h1 id="judul">Hello World</h1>          Element
2    <div id="container">
3        <section id="a">
4            <p class="p1">paragraf 1</p>
5            <a href="http://instagram.com/sandhikagalih">Instagram Fei</a>
6            <p class="p2">paragraf 2</p>          NodeList
7            <p class="p3">paragraf 3</p>
8        </section>
9        <section id="b">
10           <p>paragraf 4</p>
11           <ul>
12               <li>item 1</li>
13               <li>item 2</li>          HTMLCollection
14               <li>item 3</li>
15           </ul>
16       </section>
17   </div>
```

Prompt: Elements, nodeList, and HTMLCollection

DOM SELECTION

Memilih elemen dari sebuah DOM. Operasi (5):

getElementById() -> return sebuah elemen

```
1  const judul = document.getElementById('judul');
2  // melakukan modify pada elemen judul, inline css
3  judul.style.color = 'darkblue';
4  judul.innerHTML = 'Fei';
```

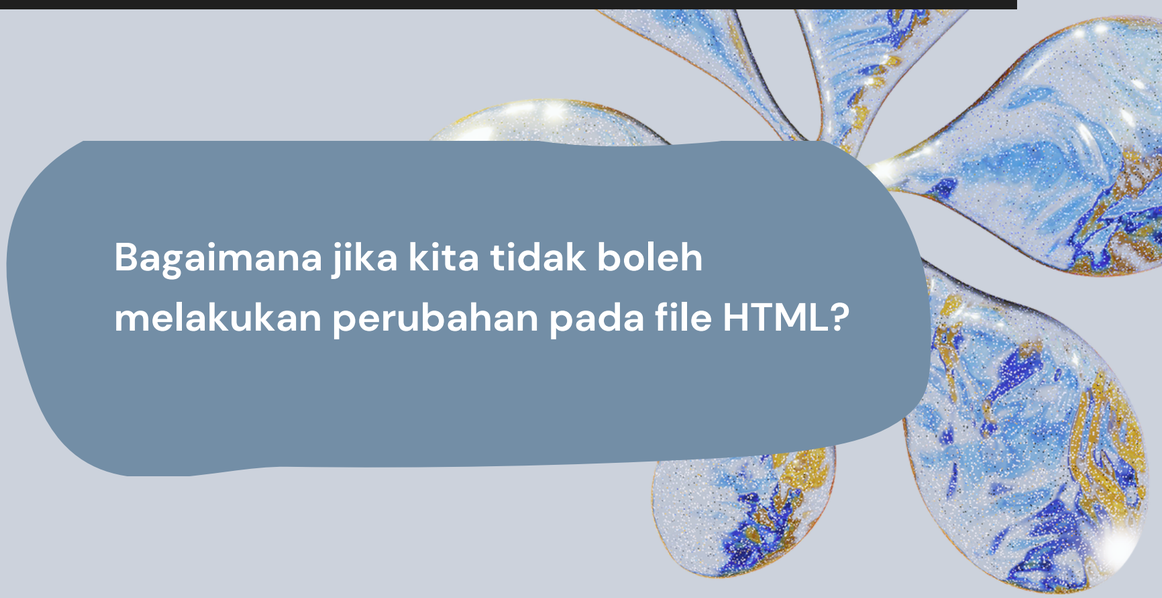
```
> judul
<h1 id="judul" style="color: darkblue; font-size: 60px;">Fei</h1>
```

getElementsByClassName() -> return HTMLCollection

```
1  const p1 = document.getElementsByClassName('p1');
2  p1[0].innerHTML = 'Diubah dari JS File';
```

getElementsByTagName() -> return HTMLCollection

```
1  const p = document.getElementsByTagName('p');
2  for(let i = 0; i < p.length; i++) {
3      p[i].style.backgroundColor = 'lightpink';
4  }
```

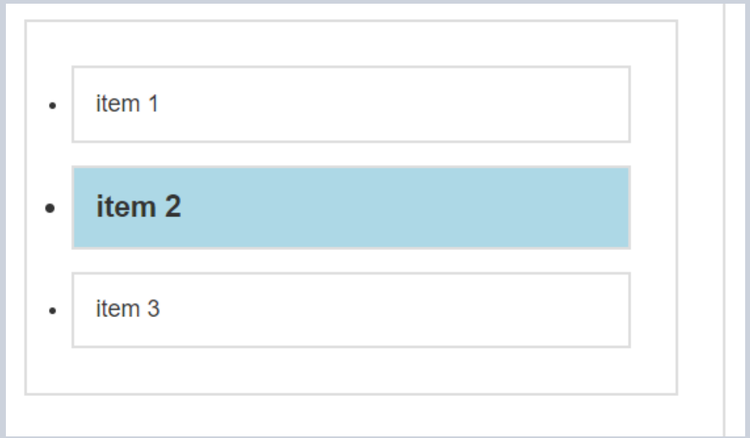


Prompt: DOM selection in JS

DOM SELECTION (2)

querySelector() –> return sebuah elemen

```
1  const li2 = document.querySelector('section#b ul li:nth-child(2)');
2  li2.style.backgroundColor = 'lightblue';
3  li2.style.fontSize = '20px'
4  li2.style.fontWeight = 'bold';
```



querySelectorAll() –> return NodeList

```
1  const sectionB = document.getElementById('b');
2  const ptag = sectionB.querySelectorAll('li');
3  for(let i = 0; i < ptag.length; i++) {
4      ptag[i].style.backgroundColor = 'lightpink';
5  }
```

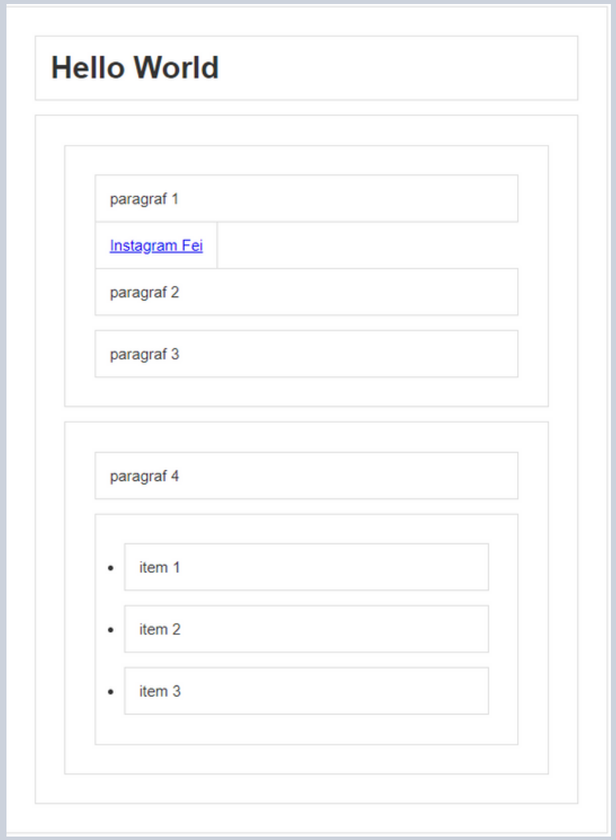
Prompt: DOM selection in JS

DOM MANIPULATION
(ELEMENT)

Element Manipulation
Digunakan untuk memanipulasi elemen-elemen yang diinginkan secara dinamis.

Methods:

- element.innerHTML
- element.style.<property>
- Attributes
- element.classList
- dll.




```
1  const judul = document.getElementsByTagName('h1')[0];
2  judul.setAttribute('name', 'hello'); //menambahkan attribute 'name'
3  judul.getAttribute('id'); //mendapatkan value dari attribute 'id'
4
5  const aLink = document.querySelector('section#a a');
6  aLink.removeAttribute('href'); //remove attribute 'href' dari a
7
8  const p2 = document.querySelector('.p2');
9  p2.classList.add('label'); //menambahkan class 'label''
10 p2.classList.remove('label'); //remove class 'label'
11 p2.classList.toggle('label'); //jika sudah ada, remove. jika belum, add
12 p2.classList.item(1); //mengetahui class pertama
13 p2.classList.contains('three'); //cek class
14 p2.classList.replace('three', 'four'); //mengganti nama class
```

Prompt: DOM manipulation in JS

DOM MANIPULATION
(NODE)

Node Manipulation

Digunakan untuk memanipulasi banyaknya elemen yang terdapat dalam bagian DOM yang diinginkan. Kita dapat menambahkan, mereplace, dan mengurangi elemen yang terdapat pada file HTML.

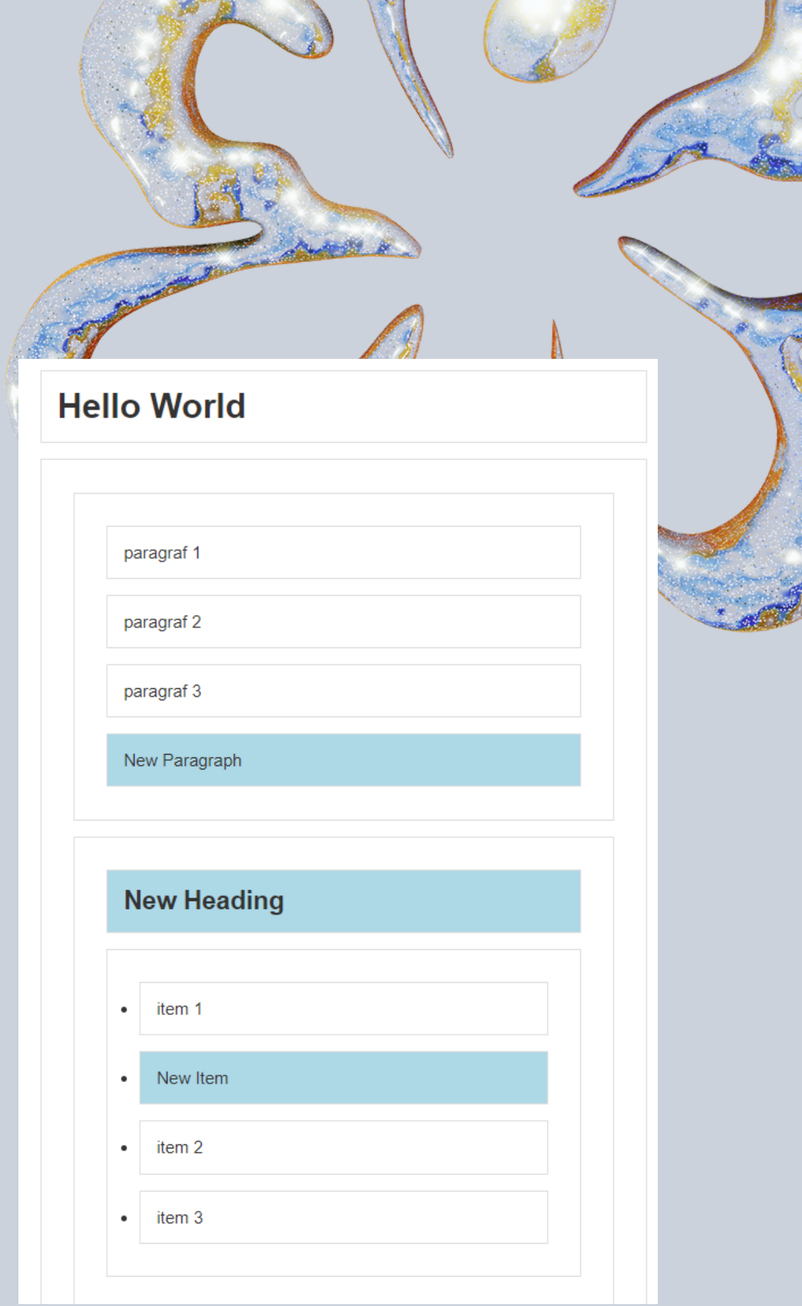
```
const newPhar = document.createElement('p');
const newPharValue = document.createTextNode('New Paragraph');
newPhar.appendChild(newPharValue);

const sectionA = document.getElementById('a');
sectionA.appendChild(newPhar);

const newLi = document.createElement('li');
const valueNewLi = document.createTextNode('New Item');
newLi.appendChild(valueNewLi);
const ul = document.querySelector('section#b ul');
const li2 = document.querySelector('section#b ul li:nth-child(2)');
ul.insertBefore(newLi, li2);

const link = sectionA.getElementsByTagName('a')[0];
sectionA.removeChild(link);

const sectionB = document.getElementById('b');
const p4 = sectionB.querySelector('p');
const newH2 = document.createElement('h2');
const valueNewH2 = document.createTextNode('New Heading');
newH2.appendChild(valueNewH2);
sectionB.replaceChild(newH2, p4);
```



Prompt: DOM manipulation in JS

DOM EVENTS

DOM Events merupakan perubahan yang dapat terjadi pada HTML elements yang disebabkan oleh adanya aksi dari user yang mentrigger suatu perubahan.

Cara meng-handle DOM Events (2):

- Event handler (inline HTML dan element method) -> onclick
- addEventListener()

Tidak disarankan untuk menggunakan inline HTML event handler karena membuat struktur HTML menjadi berantakan dan sulit untuk dipahami

Prompt: DOM events in JS

inline method (tidak disarankan)

```
<!-- inline method -->
<p class="p3" onclick="changeColor()">paragraf 3</p>
```

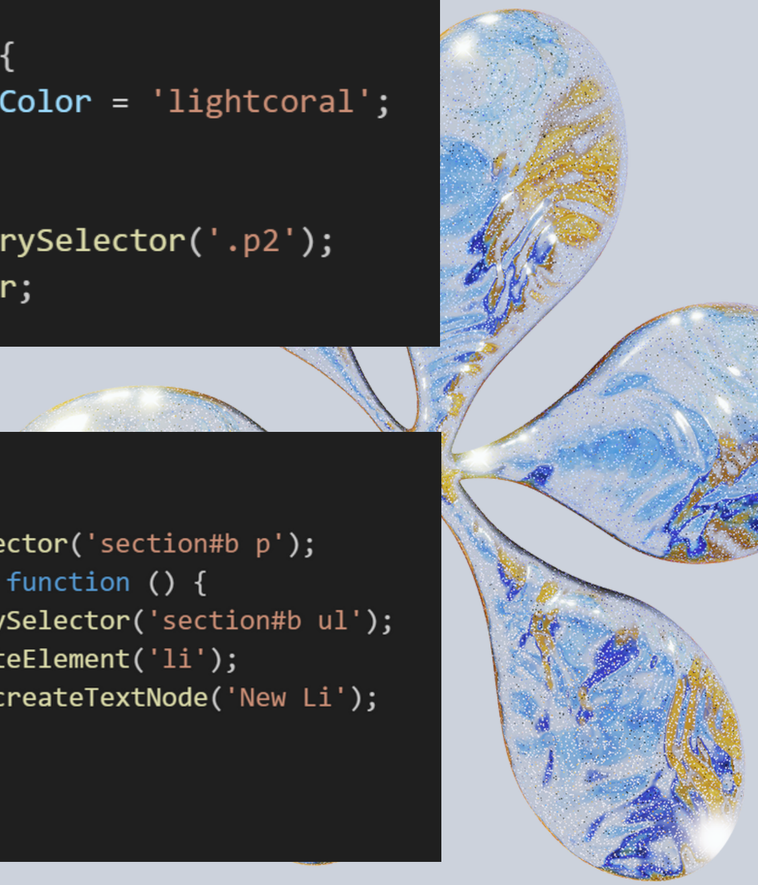
element method

```
function changeColor() {
    p2.style.backgroundColor = 'lightcoral';
}

const p2 = document.querySelector('.p2');
p2.onclick = changeColor;
```

addEventListener()

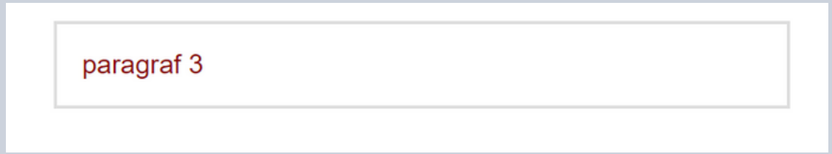
```
const p4 = document.querySelector('section#b p');
p4.addEventListener('click', function () {
    const ul = document.querySelector('section#b ul');
    const li = document.createElement('li');
    const textLi = document.createTextNode('New Li');
    li.appendChild(textLi);
    ul.appendChild(li);
});
```



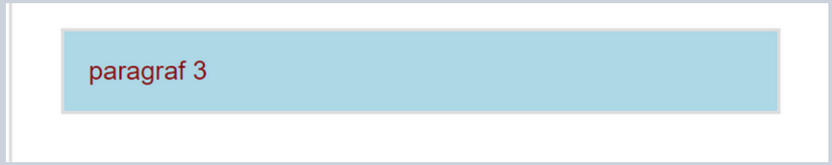
EVENT HANDLER VS ADDEVENTLISTENER()

Keduanya dapat mengeksekusi lebih dari satu event pada satu elemen yang sama namun event handler akan menimpa perubahan yang terjadi pada elemen tersebut sedangkan event listener akan menambahkan perubahan yang terjadi.

event handler



event listener




```
1  // Event handler
2  p3.onclick = function () {
3      p3.style.backgroundColor = 'lightblue';
4  }
5
6  p3.onclick = function () {
7      p3.style.color = 'darkred';
8  }
9
10 // addEventListener()
11 p3.addEventListener('click', function () {
12     p3.style.backgroundColor = 'lightblue';
13 });
14
15 p3.addEventListener('click', function () {
16    p3.style.color = 'darkred';
17 });
```

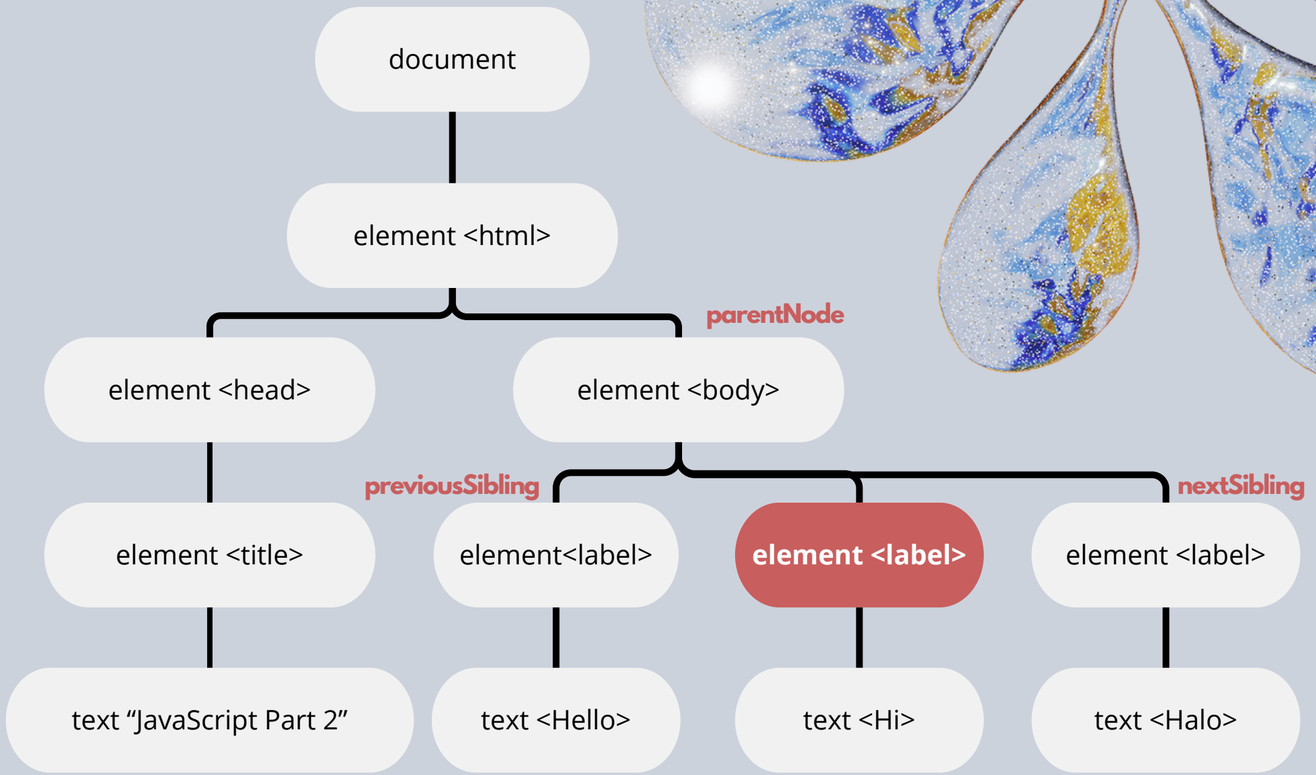
Prompt: DOM events in JS

DOM TRAVERSAL

Melakukan penelusuran pada suatu node atau elemen yang diinginkan dari DOM Tree dengan berasalkan dari elemen lain yang diketahui.

Method:

- parentNode()
- parentElement()
- nextSibling()
- nextElementSibling()
- previousSibling()
- previousElementSibling()



Prompt: DOM traversal in JS

DOM TRAVERSAL (2)

```html
<div class="container">
            <div class="card">
                <img src="img/cat3.jpeg" width="96" height="96" />
                <span class="nama">Fei</span>
                <span class="telp">08123456789</span>
                <span class="close">x</span>
            </div>
            ...
```

```js
const nama = document.querySelector('.nama');
console.log(nama);
/* node: whitespace dihitung sebagai node
element: whitespace tidak dihitung sebagai element, hanya yg ada tag */
console.log(nama.parentNode);
console.log(nama.parentElement.parentElement);
console.log(nama.nextElementSibling);
console.log(nama.nextSibling);
console.log(nama.previousSibling);
console.log(nama.previousElementSibling);
```

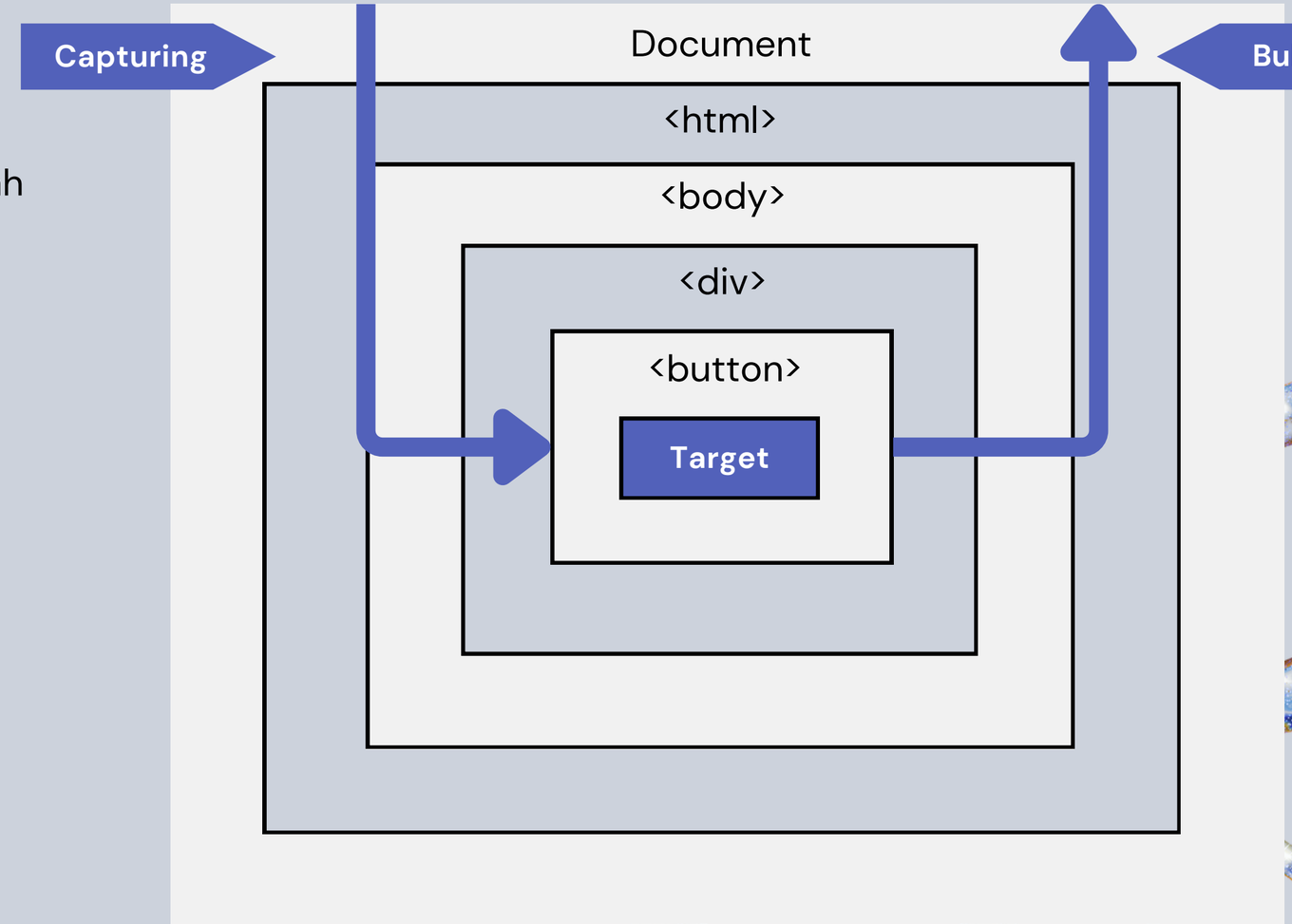
```
<span class="nama">Fei</span>
▶ <div class="card">…</div>
▶ <div class="container">…</div>
<span class="telp">08123456789</span>
▶ #text
▶ #text
<img src="img/cat3.jpeg" width="96" height="96">
```

```
▼ #text i
    assignedSlot: null
    baseURI: "http://127.0.0.1:5500/video10-12/index.html"
  ▶ childNodes: NodeList []
    data: "\n                "
    firstChild: null
    isConnected: true
    lastChild: null
    length: 17
  ▶ nextElementSibling: span.telp
  ▶ nextSibling: span.telp
    nodeName: "#text"
    nodeType: 3
    nodeValue: "\n                "
  ▶ ownerDocument: document
  ▶ parentElement: div.card
  ▶ parentNode: div.card
  ▶ previousElementSibling: span.nama
  ▶ previousSibling: span.nama
    textContent: "\n                "
    wholeText: "\n                "
  ▶ [[Prototype]]: Text
```

Prompt: DOM traversal in JS

EVENT BUBBLING AND CAPTURING

Sebuah konsep dalam DOM yang terjadi saat sebuah elemen mendapatkan trigger (event terjadi), maka event ini akan memperluas (propagate) pada parentsnya.
Event capturing merupakan kebalikan dari event bubbling yang berasal dari parents dan menyebar hingga ke leaf node.



Bagaimana cara membuat propagation eventsnya berhenti hanya pada target elemen saja?

event.stopPropagation()

Prompt: Event bubbling & capturing

THANK YOU

GitHub Repo: JavaScript 2