



Pemrograman Berbasis Web

Pertemuan 7

Nori Wilantika
Lya Hulliyyatus Suadaa
Yeni Rimawati

Politeknik Statistika STIS
Prodi DIV Komputasi Statistik

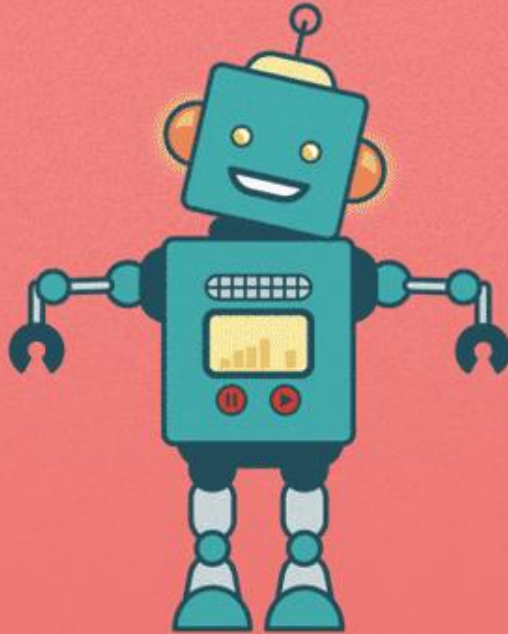


07

CLIENT-SIDE SCRIPTING (3)

Javascript;

DOM, Form Validation, BOM



MAKE IT MOVE
JAVASCRIPT



JAVASCRIPT

JavaScript memiliki semua **kekuatan** yang diperlukan untuk membuat halaman web dinamis dan interaktif:

- JavaScript dapat mengubah konten dan atribut elemen HTML
- JavaScript dapat mengubah style CSS dari elemen HTML
- JavaScript dapat menambah elemen dan atribut HTML baru
- JavaScript dapat menghapus elemen dan atribut HTML yang ada saat ini
- JavaScript dapat bereaksi terhadap event pada halaman
 - HTML form validation dapat dilakukan dengan JavaScript
- Javascript dapat membuat event HTML baru pada halaman

Bagaimana cara mengakses dan memanipulasi HTML dan CSS?

- Document Objects Model
- Window Object / Browser Object Model



The background is a light cream color with various abstract elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. On the left side, there is a large, textured grey circle with a black starburst line above it. In the bottom left, there is a pink brushstroke and a cluster of small black dots. On the right side, there is a large yellow circle, a thin black wavy line with two small yellow circles attached, and a grey brushstroke at the bottom right.

DOCUMENT OBJECT MODEL



HTML DOM

- Ketika sebuah halaman web dimuat, browser membangun model dari halaman web (dokumen) yang meliputi objek di dalam halaman tersebut -> **Document Object Model**
- ***"Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*** (W3C)



HTML DOM

- DOM adalah **standar object model** dan **programming interface** untuk HTML.
- DOM mendefinisikan HTML sebagai **objek**
 - HTML DOM methods: aksi-aksi yang dapat dilakukan (menambah atau menghapus elemen HTML).
 - HTML DOM properties: nilai (dari elemen HTML) yang dapat ditentukan dan diubah.
- Dengan kata lain: **HTML DOM adalah standar tentang cara mendapatkan, mengubah, menambah, atau menghapus elemen HTML.**



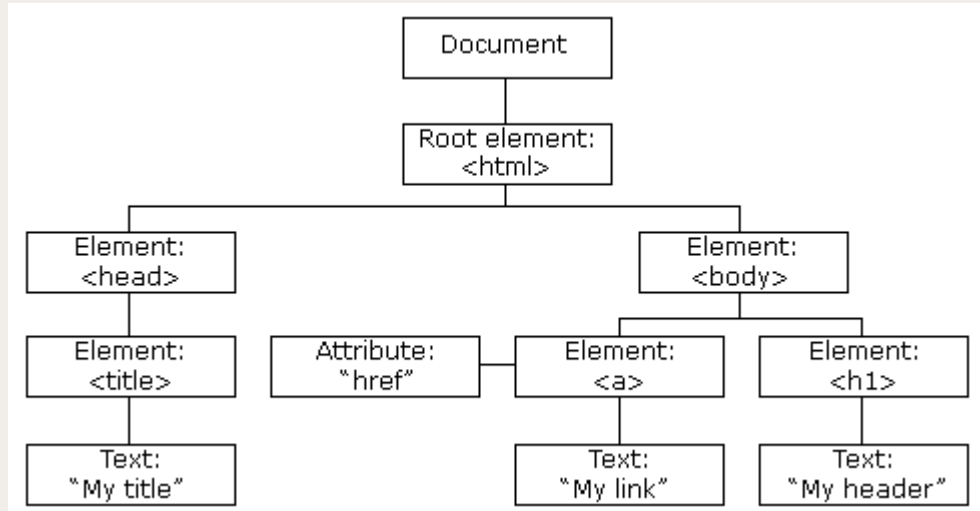


DOM NODES TREE

Berdasarkan standar W3C HTML DOM, segala sesuatu dalam dokumen HTML adalah suatu **node**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>

  <body>
    <a href="#">My link</a>
    <h1>My header</h1>
  </body>
</html>
```





DOM NODES TYPE

Node	Type	Example
ELEMENT_NODE	1	<code><h1 class="heading">W3Schools</h1></code>
ATTRIBUTE_NODE	2	<code>class = "heading" (deprecated)</code>
TEXT_NODE	3	<code>W3Schools</code>
COMMENT_NODE	8	<code><!-- This is a comment --></code>
DOCUMENT_NODE	9	The HTML document itself (the parent of <code><html></code>)
DOCUMENT_TYPE_NODE	10	<code><!Doctype html></code>

- Keseluruhan dokumen adalah **document node**
- Setiap elemen HTML adalah **element node**
- Setiap atribut HTML adalah suatu **attribute node** (deprecated)
- Teks di dalam elemen HTML adalah **text node**
- Semua komentar adalah **comment node**

https://www.w3schools.com/jsref/prop_node_nodetype.asp



FINDING HTML ELEMENTS

- By Nodes Traversal

parentNode

children

childNodes[*nodenumber*]

firstChild

lastChild

firstElementChild

lastElementChild

nextSibling

previousSibling

nextElementSibling

previousElementSibling

Jika struktur tree berubah, maka traversal path tidak lagi dapat dilakukan.



NODE TRAVERSAL EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <a href="#">My link</a>
</body>
</html>
```

```
> document.children
<< ▼ HTMLCollection [html] ⓘ
  ▶ 0: html
    length: 1
```

```
> document.childNodes
<< ▼ NodeList(2) [<!DOCTYPE html>, html] ⓘ
  ▶ 0: <!DOCTYPE html>
  ▶ 1: html
    length: 2
```

```
> document.children[0].children
<< ▶ HTMLCollection(2) [head, body]
```

```
> document.firstChild
<< <!DOCTYPE html>

> document.firstChildElementChild
<< <html>
  ▶ <head> ... </head>
  ▶ <body> ... </body>
  </html>

> document.lastChild
<< <html>
  ▶ <head> ... </head>
  ▶ <body> ... </body>
  </html>

> document.lastElementChild
<< <html>
  ▶ <head> ... </head>
  ▶ <body> ... </body>
  </html>
```



NODE TRAVERSAL EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <a href="#">My link</a>
</body>
</html>
```

```
> document.children[0].children
< ▶ HTMLCollection(2) [head, body]

> document.children[0].lastChild
< ▶ <body> ... </body>

> document.children[0].lastChild.children
< ▶ HTMLCollection [a]

> document.children[0].lastChild.childNodes
< ▼ NodeList(3) [text, a, text] ⓘ
  ▶ 0: text
  ▶ 1: a
  ▶ 2: text
  length: 3

> document.children[0].lastChild.parentNode
< <html>
  ▶ <head> ... </head>
  ▶ <body> ... </body>
  </html>
```



NODE TRAVERSAL EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <a href="#">My link</a>
</body>
</html>
```

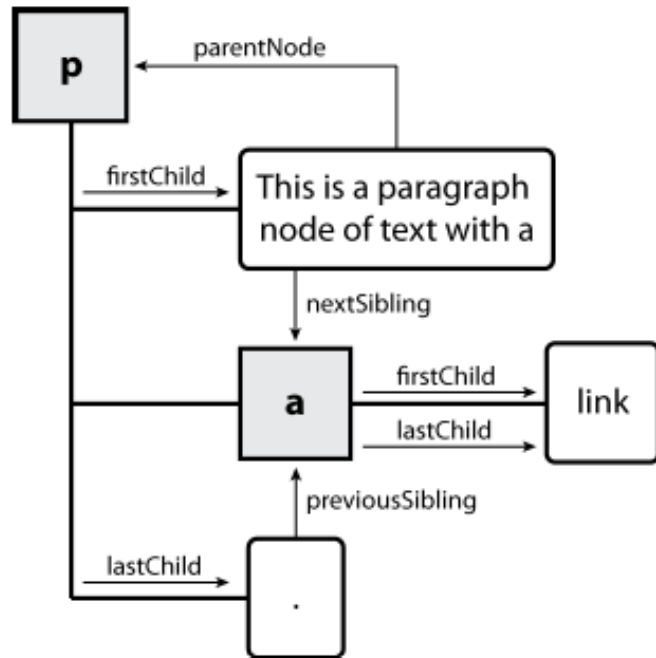
```
> document.children[0].firstChild
< ▶ <head> ... </head>
> document.children[0].firstChild.nextSibling
< ▶ #text
> document.children[0].firstChild.nextElementSibling
< ▶ <body> ... </body>
```



NODE TRAVERSAL EXAMPLE

```
<p id="foo">This is a paragraph of  
text with a  
<a  
href="/path/to/another/page.html">link  
</a>.</p>
```

HTML





FINDING HTML ELEMENTS

- By id, tag name, class name, CSS selectors

Method	Hasil
<code>getElementById()</code>	Element
<code>getElementsByTagName()</code>	HTMLCollection
<code>getElementsByClassName()</code>	HTMLCollection
<code>querySelector()</code>	Element
<code>querySelectorAll()</code>	NodeList

- HTMLCollection object adalah array-like list (collection) dari elemen HTML, yg dapat diakses dengan namanya, id, atau index number.
- NodeList object adalah list (collection) dari node yang diekstraksi dari suatu dokumen. NodeList object dapat terdiri dari attribute node dan text node.



FINDING HTML ELEMENTS EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <p id="par1">Paragraf 1</p>
  <p id="par2" class="parClass">Paragraf 2</p>
  <p id="par3" class="parClass">Paragraf 3</p>
</body>
</html>
```

```
> document.getElementById("par1")
< <p id="par1">Paragraf 1</p>

> document.getElementsByTagName("p")
< HTMLCollection(3) [p#par1, p#par2.parClass, p#par3.parClass, par1:
  p#par1, par2: p#par2.parClass, par3: p#par3.parClass]

> document.getElementsByTagName("p")[0]
< <p id="par1">Paragraf 1</p>

> document.getElementsByClassName("parClass")
< HTMLCollection(2) [p#par2.parClass, p#par3.parClass, par2:
  p#par2.parClass, par3: p#par3.parClass]

> document.getElementsByClassName("parClass")[1]
< <p id="par3" class="parClass">Paragraf 3</p>

> document.querySelector(".parClass")
< <p id="par2" class="parClass">Paragraf 2</p>

> document.querySelectorAll(".parClass")
< ▶ NodeList(2) [p#par2.parClass, p#par3.parClass]
```




CHANGING HTML ELEMENTS

- Mengubah konten HTML

```
document.getElementById(id).innerHTML = new HTML
```

- Mengubah nilai dari suatu Attribut

```
document.getElementById(id).attribute = new value
```

- Mengubah HTML Style

```
document.getElementById(id).style.property = new style
```

- Menambah dan Menghapus elemen HTML

```
document.getElementById(id).appendChild(new element)
```

```
document.getElementById(id).removeChild()
```



CHANGING HTML ELEMENTS

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <p id="par1">Paragraf 1</p>
  <p id="par2" class="parClass">Paragraf 2</p>
  <p id="par3" class="parClass">Paragraf 3</p>
</body>
</html>
```

Paragraf 1

Paragraf 2

Paragraf 3

```
document.getElementById("par1").innerHTML="Paragraf baru" //1
document.getElementById("par1").innerText="Paragraf baru" //2
document.getElementById("par1").innerHTML="<b>Paragraf baru</b>" //3
document.getElementById("par1").innerText="<b>Paragraf baru</b>" //4
```

1 Paragraf baru

Paragraf 2

Paragraf 3

2 Paragraf baru

Paragraf 2

Paragraf 3

3 Paragraf baru

Paragraf 2

Paragraf 3

4 Paragraf baru

Paragraf 2

Paragraf 3



CHANGING HTML ELEMENTS

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <p id="par1">Paragraf 1</p>
  <p id="par2" class="parClass">Paragraf 2</p>
  <p id="par3" class="parClass">Paragraf 3</p>
</body>
</html>
```

Paragraf 1

Paragraf 2

Paragraf 3

```
document.getElementById("par1").style="text-transform: uppercase" //5
document.getElementById("par1").style.color="red" //6
document.getElementsByClassName("parClass").style.color="red" //7
```

5 PARAGRAF 1

Paragraf 2

Paragraf 3

6 Paragraf 1

Paragraf 2

Paragraf 3

7 ✖ ▶ Uncaught TypeError: Cannot set properties of undefined (setting 'color')
at <anonymous>:1:56



CREATING HTML ELEMENTS

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>
<p>Add a new HTML Element.</p>

<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
const para = document.createElement("p");
const node = document.createTextNode("This is
new.");
para.appendChild(node);
const element = document.getElementById("div1");
element.appendChild(para);
</script>

</body>
</html>
```

JavaScript HTML DOM

Add a new HTML Element.

This is a paragraph.

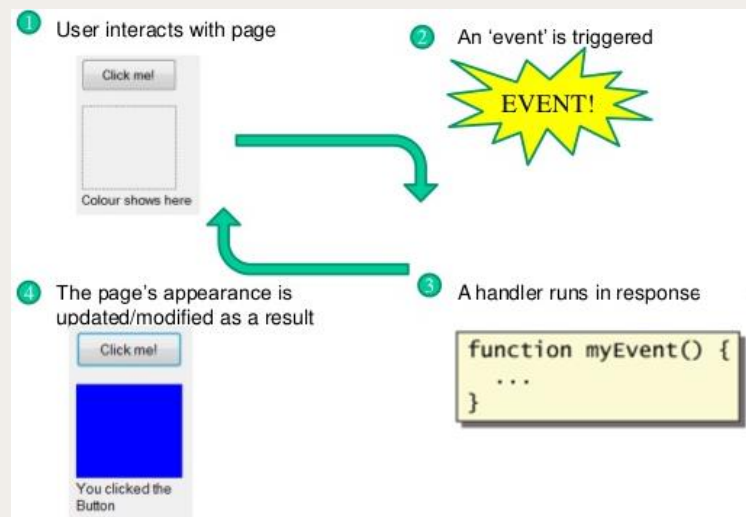
This is another paragraph.

This is new.



USING EVENTS

- Program JavaScript dieksekusi secara sekuensial
- program memiliki titik awal tertentu, dijalankan selangkah demi selangkah melibatkan **control structures** dan **function** mencapai titik di mana eksekusinya berhenti.
- HTML DOM memungkinkan kita mengatur kapan suatu kode dieksekusi
- JavaScript dapat dijalankan ketika sebuah peristiwa (event) terjadi, seperti:
 - Saat halaman web telah dimuat
 - Saat pengguna mengklik mouse
 - Saat gambar telah dimuat
 - Saat mouse bergerak di atas sebuah elemen
 - Ketika sebuah field input diubah
 - Saat formulir HTML dikirimkan
 - Saat pengguna menekan tombol





USING EVENTS

HTML events kebanyakan adalah aksi yang dilakukan pengguna. Fungsi JavaScript yang memproses event adalah **event handlers**.

- HTML Event Attributes

```
<button onclick="displayDate()">Try it</button>
```

- Menggunakan HTML DOM

```
<script>  
document.getElementById("myBtn").onclick = displayDate;  
</script>
```

- HTML DOM EventListener

```
document.getElementById("myBtn").addEventListener("click", displayDate);  
  
document.getElementById("myBtn").removeEventListener("click", displayDate);
```



EVENTS

Event	Keterangan	Event Attribute
load	triggered when the user enters the page	onload
click	occurs when an object is clicked	onclick
keydown	occurs when the user presses a key	onkeydown
mouseover	occurs once each time the mouse pointer moves over an HTML element	onmouseover
select	occurs when a user selects some of the text within a text or textarea field	onselect
focus	occurs when a form field receives input focus by tabbing with the keyboard or clicking with the mouse	onfocus
change	occurs when a select, text, or textarea field loses focus and its value has been modified	onchange
blur	occurs when an HTML element loses focus	onblur
submit	occurs when a user submits a form	onsubmit



EVENT EXAMPLE

```
<!DOCTYPE html>
<html>
<body>

<h1>HTML DOM Events</h1>
<h2>The onclick Event</h2>

<p>Click to display the time.</p>

<button
onclick="getElementById('demo').innerHTML
=Date()">What is the time?</button>

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

</body>
</html>
```

HTML DOM Events

The onclick Event

Click to display the time.

What is the time?

Mon Mar 20 2023 04:11:27 GMT+0700 (Western Indonesia Time)




The background is a light cream color with various abstract decorative elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. On the left side, there is a grey circular watercolor shape with a black starburst pattern above it, and a small yellow circular stamp below it. In the bottom left, there is a pinkish-red brushstroke and a cluster of small black dots. On the right side, there is a large yellow circular watercolor shape, a thin black wavy line with two small yellow circular stamps, and a grey brushstroke at the bottom right.

USER INPUT VALIDATION





USER INPUT VALIDATION

- Penggunaan umum JavaScript: validasi dari input pengguna pada HTML form sebelum diproses
 - Mengecek bahwa required field tidak dibiarkan kosong
 - Mengecek bahwa field hanya terdiri dari karakter yang diperbolehkan
 - Mengecek bahwa nilai yang dimasukkan berada dalam rentang yang diperbolehkan
- **Server side validation** dilakukan oleh web server, setelah input dikirim ke server.
- **Client side validation** dilakukan oleh web browser, sebelum input dikirim ke web server, berdasarkan:
 - HTML Input Attributes
 - CSS Pseudo Selectors




```
<form method="post" action="process.php"
      onSubmit="return validate(this)">
  <label>User name:      <input type="text" name="user"></label>
  <label>Email address: <input type="text" name="email"></label>
  <input type="submit" name="submit" />
</form>
<script>
function validate(form) {
  fail  = validateUser(form.user.value)
  fail += validateEmail(form.email.value)
  if (fail == "") return true
  else { alert(fail); return false } }
</script>
```





```
function validateUser(field) {  
  if (field == "") return "No username entered\n"  
  else if (field.length < 5)  
    return "Username too short\n"  
  else if (/[^a-zA-Z0-9_-]/.test(field))  
    return "Invalid character in username\n"  
  else return ""  
}  
  
function validateEmail(field) {  
  if (field == "") return "No email entered\n"  
  else if (!((field.indexOf(".") > 0) &&  
    (field.indexOf("@") > 0)) ||  
    /^[^a-zA-Z0-9\\.\\@\\_\\-]/.test(field))  
    return "Invalid character in email\n"  
  else return ""  
}
```



The background is a light cream color with various abstract decorative elements. In the top left, there is a horizontal red brushstroke and a cluster of small black dots. To the left of the center, there is a large, textured grey circle with a black starburst pattern above it. In the bottom left, there is a pink brushstroke and a cluster of small black dots. On the right side, there is a large yellow circle at the top, a thin black wavy line with two small yellow circles attached to it, and a grey brushstroke at the bottom right.

BROWSER OBJECT MODEL



BROWSER OBJECT MODEL

- Merepresentasikan jendela dari browser.
- Semua objek, fungsi dan variable global Javascript adalah anggota dari **window objects**.
- Mengizinkan **window objects** baru dibuat dan dimanipulasi:
window.screen **window.history** **Popup Boxes**
window.location **window.navigator**
- Mengizinkan **window properties** dibuat dan dimanipulasi
window.innerHeight
window.innerWidth
window.innerWidth
- **window methods** lainnya:
window.open() **window.focus()** **window.blur()**
window.close() **window.resizeTo()** **window.print()**
window.moveTo()



BROWSER OBJECT MODEL

Standard sequence untuk membuat objek window baru, bukanlah sebagai berikut:

```
// new instance of `Window` class  
var newWin = new Window(...)  
newWin.document.write('<html>...</html>')
```

Cara membuat objek window baru:

```
// new window created by using `open` with an existing one  
var newWin = window.open(...)  
newWin.document.write('<html>...</html>')
```



WINDOW SCREEN

Objek window.screen terdiri dari informasi mengenai screen pengguna.

- **screen.width** : mengembalikan width dari screen pengunjung (dalam pixel).
- **screen.height** : mengembalikan height dari screen pengunjung (dalam pixel).
- **screen.availWidth** : mengembalikan width dari screen pengunjung (dalam pixel), dikurangi interface feature seperti Windows Taskbar.
- **screen.availHeight** : mengembalikan height dari screen pengunjung (dalam pixel), dikurangi interface feature seperti Windows Taskbar.
- **screen.colorDepth** : mengembalikan jumlah bit yang digunakan untuk menampilkan satu warna.
- **screen.pixelDepth** : mengembalikan pixel depth dari screen.



WINDOW LOCATION

Objek `window.location` dapat digunakan untuk mendapatkan alamat halaman saat ini (URL) dan melakukan redirect browser ke halaman baru.

- **`location.href`** : mengembalikan URL dari halaman saat ini.
- **`location.hostname`** : mengembalikan nama dari internet host (halaman saat ini).
- **`location.pathname`** : mengembalikan pathname halaman saat ini.
- **`location.protocol`** : mengembalikan web protocol dari halaman.
- **`location.port`** : mengembalikan jumlah internet host port (halaman saat ini).

`https://stis.ac.id/`



WINDOW LOCATION EXAMPLE

https://stis.ac.id/

- `location.href` : 'https://stis.ac.id/'
- `location.hostname` : 'stis.ac.id'
- `location.pathname` : '/'
- `location.protocol` : 'https:'
- `location.port` : ''



WINDOW HISTORY

Objek `window.history` terdiri dari browser's history. Untuk melindungi privacy pengguna, terdapat batasan bagaimana JavaScript dapat mengakses objek ini.

- **`history.back()`**

Memuat URL sebelumnya pada history list. Sama seperti mengklik tombol Back pada browser.

- **`history.forward()`**

Memuat URL setelahnya pada history list. Sama seperti mengklik tombol Forward pada browser.



WINDOW NAVIGATOR

Objek window.navigator terdiri dari informasi mengenai browser pengunjung.

- navigator.appName : mengembalikan nama aplikasi dari browser
- navigator.appVersion : mengembalikan informasi versi dari browser
- navigator.userAgent : mengembalikan user agent dari browser

```
> navigator.appName
```

```
< 'Netscape'
```

```
> navigator.appVersion
```

```
< '5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36'
```

```
> navigator.userAgent
```

```
< 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36'
```



BROWSER DETECTION

```
function fnBrowserDetect(){  
  
    let userAgent = navigator.userAgent;  
    let browserName;  
  
    if(userAgent.match(/chrome|chromium|crios/i)){  
        browserName = "chrome";  
    }else if(userAgent.match(/firefox|fxios/i)){  
        browserName = "firefox";  
    } else if(userAgent.match(/safari/i)){  
        browserName = "safari";
```

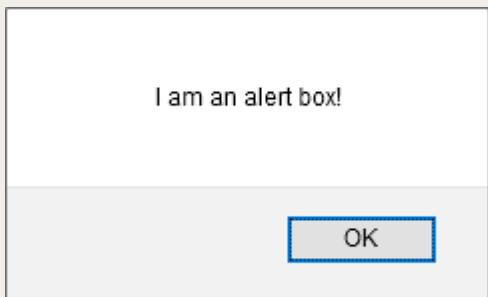
```
    }else if(userAgent.match(/oprV/i)){  
        browserName = "opera";  
    } else if(userAgent.match(/edg/i)){  
        browserName = "edge";  
    }else{  
        browserName="No browser detection";  
    }  
}
```

```
document.querySelector("h1").innerText="You  
are using "+ browserName + " browser";  
}
```



POPUP BOX

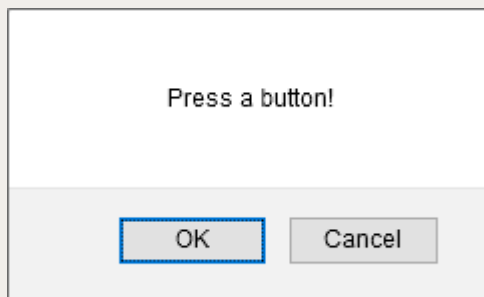
JavaScript memiliki tiga jenis popup boxes: Alert box, Confirm box, and Prompt box.



I am an alert box!

OK

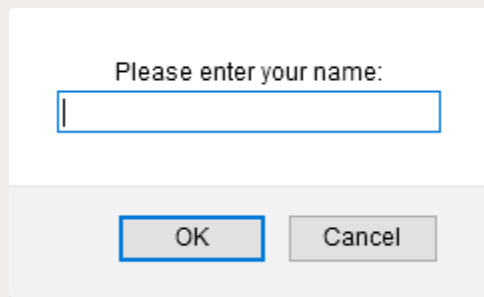
This is a screenshot of an alert box. It has a white background with a thin grey border. The text "I am an alert box!" is centered in the main area. At the bottom, there is a light grey bar containing a single button labeled "OK".



Press a button!

OK Cancel

This is a screenshot of a confirm box. It has a white background with a thin grey border. The text "Press a button!" is centered in the main area. At the bottom, there is a light grey bar containing two buttons: "OK" and "Cancel".



Please enter your name:


OK Cancel

This is a screenshot of a prompt box. It has a white background with a thin grey border. The text "Please enter your name:" is centered in the main area. Below the text is a text input field. At the bottom, there is a light grey bar containing two buttons: "OK" and "Cancel".


null `alert(message_string)`: menampilkan pesan

bool `confirm(message_string)`: menanyakan konfirmasi dari aksi

string `prompt(message string, default)`: meminta input



```
<!DOCTYPE html>
<html lang="en-GB">
  <head><title>Interaction example</title></head>
  <body>
    <script>
      do {
        string    = prompt("How many items do you want to buy?")
        quantity = parseInt(string)
      } while (isNaN(quantity) || quantity <= 0)
      do {
        string = prompt("How much does an item cost?")
        price  = parseFloat(string)
      } while (isNaN(price) || price <= 0)
      buy = confirm("You will have to pay "+
                    (price*quantity).toFixed(2)+
                    "\nDo you want to proceed?")
      if (buy) alert("Purchase made")
    </script>
  </body></html>
```





REFERENSI

1. JavaScript, The Definitive Guide by David Flanagan. Publisher O'Reilly.



2. W3C Document Object Model. <https://www.w3.org/TR/WD-DOM/>



3. COMP519 Web Programming (2020-21). University of Liverpool.
<https://cgi.csc.liv.ac.uk/~ullrich/COMP519/>



Thanks!

Do you have any questions?



wilantika@stis.ac.id

lya@stis.ac.id

yeni.rima@stis.ac.id

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik