Department of Information Technology - State Polytechnic of Malang

Jobsheet-3: Javascript

Web Design and Programming Courses

Web Design and Programming Teaching Team

September 2024

Topic

- Introduction to Basic Concepts of JavaScript
- Data Types, Operators, Functions in Javascript
- Javascript in HTML

Purpose

Students are expected to:

1. Understanding the concept of Javascript
2. Understanding data types, operators, and functions in Javascript
3. Students are able to run Javascript on HTML

Introduction

JavaScript is a web programming language that is a *Client-Side Programming Language*. ***Client-Side Programming Language*** is a type of programming language whose processing is performed by *the client*. The *client* application in question refers to *web browsers* such as **Google Chrome and Mozilla Firefox**. *Client Side* programming languages are different from *Server Side* programming languages such as PHP, where for *the server side* all program code is executed on the server side.

To run **JavaScript**, we only need a *text editor* application and *a web browser*. **JavaScript** has the following features: *high-level programming language, client-side, loosely typed* and object-oriented. **JavaScript** at the beginning of its development functions to make the interaction between users and websites faster without having to wait for processing on the *web server*. Before *JavaScript*, every interaction from the user had to be processed by *a web server*.

Imagine when we fill out a *registration form* for a website, then click the *submit* button, wait for about 20 seconds for the website to process the form, and find a page stating that there is a form column that is still not filled in. It is for this purpose that **JavaScript** was developed. Processing to check whether all forms have been filled out or not, can be moved from the *web server* to the *web browser*.

In subsequent developments, *JavaScript* is not only useful for *form validation*, but also for a variety of more modern purposes. Various animations to beautify web pages, chat features, modern effects, games, can all be created using *JavaScript*. There are 3 types of javascript tag writing **methods**:

1. Write a tag with `<script type="text/javascript">` at the beginning and end with `</script>`. The attribute tells the browser that the script in the tag is JavaScript in text format.
2. Write a tag with `<script language="javascript">` at the beginning and end with `</script>`. This attribute is used to determine which version of JavaScript to use. For example, `<script language="javascript1.2">` indicates that the version of JavaScript used is 1.2.
3. Write tags with `<script language="javascript" type="text/javascript">` at the beginning and end with `</script>`. This mixed method combines old and new ways of writing, allowing compatibility for web browsers that support JavaScript but may not yet fully support HTML.

Practical Section 1: Creating a Javascript Program

Step	Description
1	Create a new folder with the name of the week3. In that folder, create a new file named **`helloworld.html`**

2	Type the program code below:
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>Hello World Javascript</title>`
	`</head>`
	`<body>`
	`<script>`
	`console.log("Saya belajar Javascript");`
	`document.write("Hello World!");`
	`</script>`
	`</body>`
	`</html>`

3	Save it as **`helloworld.html`**, then open the file with a web browser.
4	Observe what happens to the browser, then record your observations (Question No. 1)
5	Now try to open the javascript console, then look at the **`Inspect Elements > Console`**
6	Observe what happens in the Console tab, then record your results! (Question No. 2)

7	Earlier, we wrote the command: `console.log("Saya belajar Javascript");` Why do you think the command is not displayed? (Question No. 3)

Practical Section 2: How to Write Javascript Code in HTML

In the Part 1 Practicum, we have written JavaScript code in HTML, this method is an *embedded* writing method. Some other ways we need to know include:

1. *Embed* (Javascript code pasted directly into HTML)
2. *Inline* (Javascript code written on HTML attributes)
3. *External* (Javascript code is written separately from the HTML file)

1. Writing Javascript Code with Embed	
Step	**Description**
1	In this way, we use **the <script>** tag to embed *the* Javascript code in the HTML. These tags can be written in **the <head>** and **<body tags>**

2	Create **a embed.html** file. Type the program code below:
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`  <title>Belajar Javascript dari Nol</title>`
	`  <script>`
	`  // ini adalah penulisan kode javascript`
	`  // di dalam tag <head>`
	`  console.log("Hello JS dari Head");`
	`  </script>`
	`  </head>`
	`  <body>`
	`  <p>Tutorial Javascript untuk Pemula</p>`
	`  <script>`
	`  // ini adalah penulisan kode javascript`
	`  // di dalam tag <body>`
	`  console.log("Hello JS dari body");`
	`  </script>`
	`  </body>`
	`</html>`

3	Observe what happens to the browser? Record your observations (Question No. 4)

4	Which do you think is better, written in the **<head>** or **<body> tag**? (Question No. 5)

2. Inline *Javascript Code Writing*	
Step	**Description**
1	In this way, we'll write the javascript code inside the HTML attribute. This method is usually used to call a function on a specific event. One example is when clicked.

2	Create **a inline.html** file.
	Type the program code below:
	`<a href="#" onclick="alert('Yey!')">Klik aku!</a>`
	Or it can also be like this:
	`<a href="javascript:alert('Yey!')">Klik aku!</a>`

3	Observe what happens to the browser! Record your observations (Question No. 6)
4	What is the difference between the two program codes (Question No. 7)

3. External Javascript Code Writing	
Step	**Description**
1	In this way, we'll write the javascript code separately from the HTML file. This method is usually used for large projects, because it is believed that this way it can make it easier to manage project code.

2	Let's try, create two files, namely HTML and Javascript files.
	`├─ kode-program.js`
	`└─ index.html`

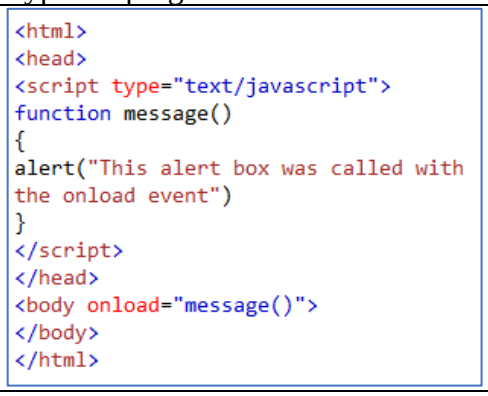
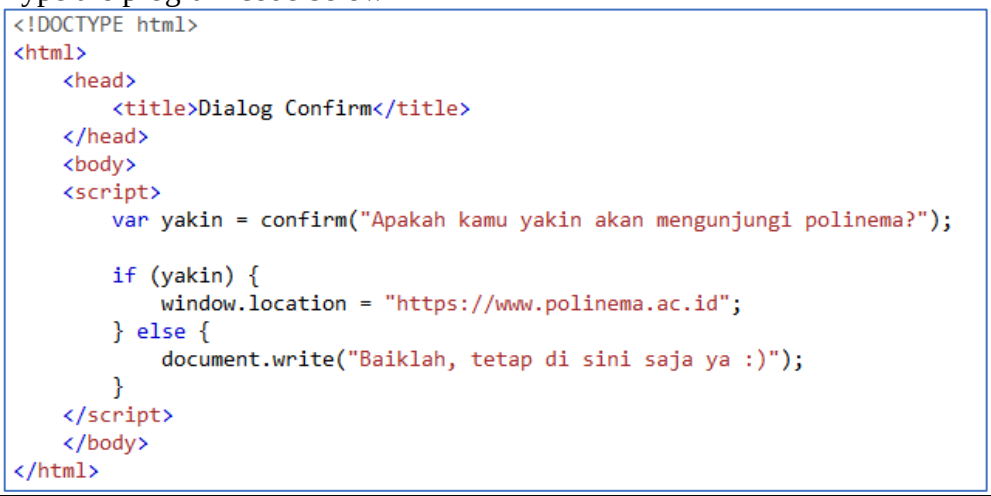
3	Contents of **the kode-program.js** file: `alert("Hello, ini adalah program JS eksternal!");`
4	Contents of **the index.html** file:

	<!DOCTYPE html> <html> <head> <title>Belajar Javascript dari Nol</title> </head> <body> Tutorial Javascript untuk Pemula <!-- Menyisipkan kode js eksternal --> <script src="kode-program.js"></script> </body> </html>
5	Observe what happens to the browser! Record your observations (Question No. 8)
6	In the experiment, we wrote separate javascript code with HTML code. Then in the HTML code we insert the **src** attribute in the **<script tag>** <!-- Menyisipkan kode js eksternal --> <script src="kode-program.js"></script> Then anything in **kode-program.js** file will be readable from **index.html**
7	Move **kode-program.js** file to another folder, what will happen if the javascript file is in a different folder? Observe and record your observations (Question No. 9)
8	Suppose we have a folder structure like this: ├── js/ │ └── kode-program.js └── index.html So to insert **the kode-program.js** file into the HTML, we can write the following code: <script src="js/kode-program.js"></script>

Practical Section 3: Dialogue Window

A dialog window is a window used to interact with users. There are three types of dialog windows in Javascript:

1. The alert() **dialog window**;
2. The confirm() **dialog window**;
3. The prompt() **dialog window**;

Step	Description
1	Buat File baru berana **alertjavascript.html**
2	Type the program code below
	
3	Observe what appears in the browser
4	Record your observations (Question No. 10)
5	Create a new file named **confirmjavascript.html**
6	Type the program code below 
7	Observe what appears in the browser
8	Record your observations (Question No. 11)
9	Create a new file named **promptjavascript.html** Type the program code below

	<!DOCTYPE html>
	<html>
	  <head>
	  <title>Dialog Promp</title>
	  </head>
	  <body>
	  <script>
	  var nama = prompt("Siapa nama kamu?", "");
	  document.write(" Hello "+ nama +" ");
	  </script>
	  </body>
	</html>

10	Observe what appears in the browser
11	Record your observations (Question No. 12)

Practical Section 4: Variables

The most commonly used way to create variables in JavaScript is to use the var keyword followed by the variable name and its value.

Example: var title = "Learn Javascript Programming";

Displaying the Contents of Variables

To display the contents of the variables, we can utilize functions to display outputs such as:

- The console.log() `function` returns the output to the JavaScript console;
- The `document.write() function` returns the output to the HTML document;
- and `alert()` returns the output to the dialog window.

Step	Description
1	Create a new file named `variable.html`
2	Type the program code below

	<!DOCTYPE html>
	<html lang="en">
	<head>
	  <title>Belajar Variabel dalam Javascript</title>
	  <script>
	  // membuat variabel
	  var name = "Javascript";
	  var visitorCount = 50322;
	  var isActive = true;
	
	  // menampilkan variabel ke jendela dialog (alert)
	  alert("Selamat datang di " + name);
	
	  // menampilkan variabel ke dalam HTML
	  document.write("Nama Situs: " + name + "
	");
	  document.write("Jumlah Pengunjung: " + visitorCount + "
	");
	  document.write("Status Aktif: " + isActive + "
	");
	
	  </script>
	</head>
	<body>
	</body>
	</html>

3	Observe what appears in the browser
4	Record your observations (Question No. 13)

Deleting Variables

Variable removal in Javscript is rare. However, for programs that require precision in memory allocation, variable deletion needs to be done so that memory usage is more optimal. Variable deletion can be done with the **keyword delete**.

Example:

```
var bookTitle = "Learn Javascript Programming";
delete bookTitle;
```

Then the bookTitle variable will disappear from memory.

Practical Section 5: FUNCTIONS

Functions are sub-programs that can be reused both within the program itself, and in other programs. A function in Javascript is an object. Because it has properties and also *methods*.

Step	Description
1	Create a new file named **`function1.html`**
2	How to call a function in Javascript code is usually written with: **`nameFunction();`**
3	Type the following program code

	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<script>`
	`// membuat fungsi`
	`var sayHello = () => alert("Hello World!");`
	`</script>`
	`</head>`
	`<body>`
	`<!-- Memanggil fungsi saat link diklik -->`
	`<a href="#" onclick="sayHello()">Klik Aku!</a>`
	`</body>`
	`</html>`

4	Observe what appears in the browser
5	Record your observations (Question No.14)

6	A parameter is a variable that stores a value for a process inside a function
	How to call a parameter in javascript is:
	`function kali(a, b){`
	`hasilKali = a * b;`
	`console.log("Hasil kali a*b = " + hasilKali);`
	`}`

7	Create a new file named **`function2.html`** Type the following program code
	`<html>`
	`<head>`
	`<script type="text/javascript">`
	`function total(numberA,numberB)`
	`{`
	`return numberA + numberB`
	`}`
	`</script>`
	`</head>`
	`<body>`
	`<script type="text/javascript">`
	`document.write(total(2,3))`
	`</script>`
	`</body>`
	`</html>`

8	Observe what appears in the browser
9	Record your observations (Question No.15)

Practical Section 6: Data Types

Data types are the types of data that we can store in variables. There are several types of data in Javascript programming:

- String (text)
- Integer or Number
- Float (number of Fractions)
- Boolean
- Object

JavaScript is a *dynamic typing* language, which means that we don't have to write data types when creating variables like in C, C++, Java, etc. which are *static typing*.
There are several rules for writing variables in Javascript:

- Variable naming **should not** use numbers in front of it.
  Example:

```
wrong
var 123name = "polinema";

true
var name123 = "polinema";
```

- Variable naming **can** use the initial underscore.
  Example:

```
var _nama = "Polinema";
```

- Variable naming **is recommended** using `camelCase` if it consists of two syllables.
  Example:

```
var _fullName = "Polinema";
```

- Variable naming **is recommended** using English
  Example:

```
var _postTitle = "Javascript Tutorials";
```

Step	Description
1	Create a new file named `tipedata.html`
2	Type the following program code

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

<h2>JavaScript Data Types</h2>

<p>Contoh Javascript Data Types:</p>

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

<script>
var x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";       // Now x is a String

document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

3	Observe what appears in the browser
4	Record your observations (Question No. 16)
5	Type the program below and save it with **string.html**

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

<h2>JavaScript Strings</h2>

<p>Membuat Javascript String</p>

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

<script>
var answer1 = "It's alright";
var answer2 = "He is called 'Dilan'";
var answer3 = 'He is called "Dilan"';

document.getElementById("demo").innerHTML =
answer1 + "<br>" +
answer2 + "<br>" +
answer3;
</script>

</body>
</html>
```

6	Observe what appears in the browser
7	Record your observations (Question No. 17)

8	Type the program below and save it with **boolean.html** `<!DOCTYPE html>` `<html>` `<body>` `<h2>JavaScript Booleans</h2>` `<p>Booleans hanya memiliki nilai true dan false</p>` `<p id="demo"></p>` `<script>` `var x = 5;` `var y = 5;` `var z = 6;` `document.getElementById("demo").innerHTML =` `(x == y) + " " + (x == z);` `</script>` `</body>` `</html>`
9	Observe what appears in the browser
10	Record your observations (Question No. 18)

11	Type the program below and save it with **array.html name**
	`<!DOCTYPE html>`
	`<html>`
	`<body>`
	
	`<h2>JavaScript Arrays</h2>`
	
	`<p>Array</p>`
	
	`<p id="demo"></p>`
	
	`<script>`
	`var cars = ["Satu","Dua","Tiga"];`
	
	`document.getElementById("demo").innerHTML = cars[0];`
	`</script>`
	
	`</body>`
	`</html>`

12	Observe what appears in the browser
13	Record your observations (Question No. 19)

Practical Section 7: Operators

An operator is a symbol used to perform operations on a value and variable. Operators in programming are divided into 6 types:

1. Arithmetic operator;
2. Assignment Operator;
3. relationship or comparison operators;
4. Logic Operators;
5. Operator Bitwise;
6. Operator Ternary;

An arithmetic operator is an operator to perform arithmetic operations such as addition, subtraction, division, multiplication, etc. Arithmetic operators consist of:

Operator Name	Symbol
Addition	+
Reduction	-
Multiplication	*
Exponentiation	**
Division	/
Modulus	%

Step	Description
1	Create a new File named **operator.html**
2	Type a program below
3	Observe what appears in the browser
4	Record your observations (Question No.20)

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

<h2>JavaScript Operators</h2>

<p>x = 5, y = 2, menghitung z = x + y, dan tampil z:</p>

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

<script>
var x = 5;
var y = 2;
var z = x + y;
document.getElementById("demo").innerHTML = z;
</script>

</body>
</html>
```

Practical Section 8: Branching

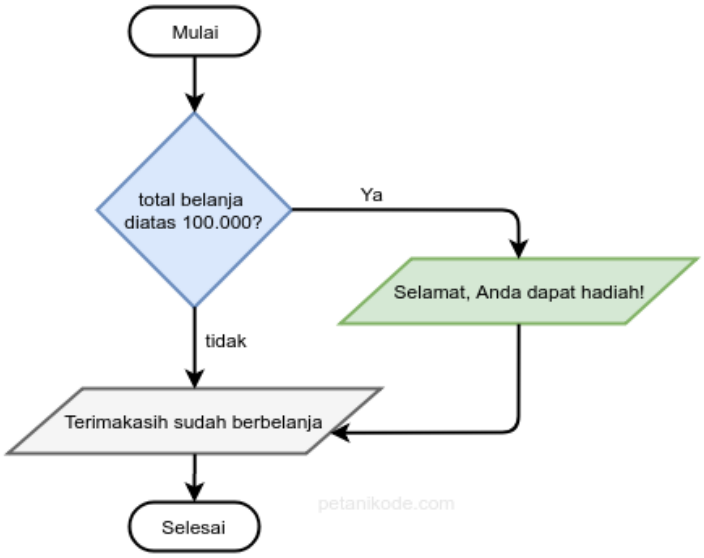
It can be said that branching and *looping* are one of the core methods in all programming languages in the world, because with branching and *looping* will produce a dynamic program, and not a linear and static program. Because JavaScript is one of the ways to do web programming on the client side, JavaScript also has this ability.

Some branching functions:

- Use **if** to specify the block of code to be executed, if the specified condition is true
- Use **else** to specify the block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to be tested, if the first condition is false
- Use **switches** to specify many alternative blocks of code to execute

Branching IF

An *if* branch is a branch that only has **one block of options** when the condition is true. Take a look at *the following* flowchart:



- If the total spend is greater than IDR 100,000, then display the message "Congratulations, you get a prize!".
- If it is below Rp 100,000, then the message "Congratulations, you got a prize!" will not be displayed.

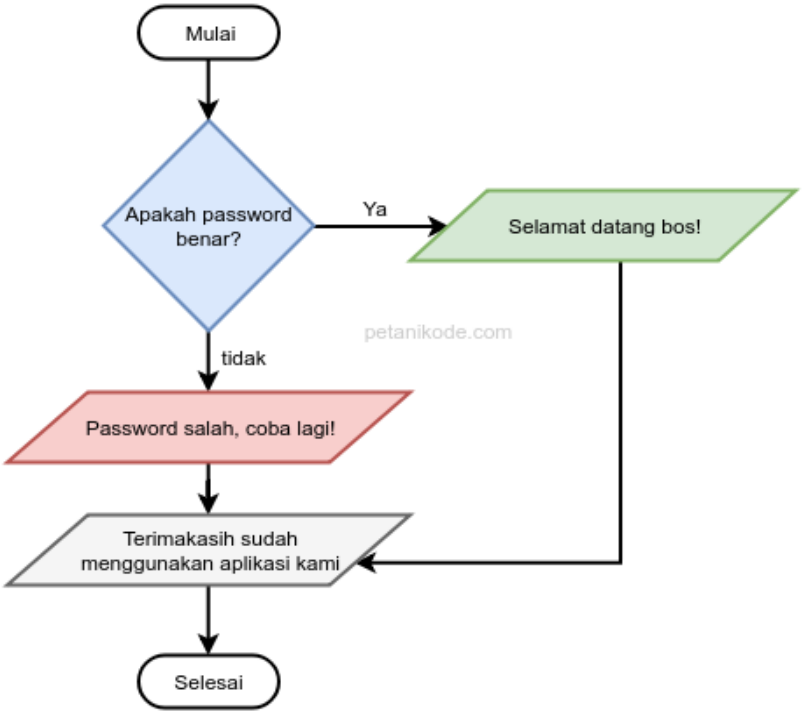
Step	Description
1	Create a new File named `if-javascript.html`
2	Type a program below

	<!DOCTYPE html>
	<html lang="en">
	<head>
	  <title>Percabangan if</title>
	</head>
	<body>
	  <script>
	  var totalBelanja = prompt("Total belanja?", 0);
	
	  if(totalBelanja > 30000){
	  document.write(" Selamat Anda dapat hadiah ");
	  }
	
	  document.write(" Terimakasih sudah berbelanja di toko kami ");
	  </script>
	</body>
	</html>

3	Observe what appears in the browser
4	Record your observations (Question No. 21)

Branching IF ELSE

An if/else *branch* is a branch that has **two blocks of choice**. The first option is for **the correct** condition, and the second option is for **the wrong** condition (*else*). Take a look at this flowchart:



This is a flowchart for checking passwords.

- If the password is correct, the message on the green block will be displayed: **"Welcome boss!"**
- If it is incorrect, then the message in the red block will be displayed: **"Incorrect password, try again!"**

Step	Description
1	Create a new File named **ifelse-javascript.html**

2	Type a program below
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`  <title>Percabangan if/else</title>`
	`</head>`
	`<body>`
	`  <script>`
	`  var password = prompt("Password:");`
	`  if(password == "teh"){`
	`  document.write("<h2>Selamat datang !</h2>");`
	`  } else {`
	`  document.write("<p>Password salah, coba lagi!</p>");`
	`  }`
	`  document.write("<p>Terima kasih sudah menggunakan aplikasi ini!</p>");`
	`  </script>`
	`</body>`
	`</html>`

3	Observe what appears in the browser
4	Record your observations (Question No. 22)

Branching SWITCH CASE

Switch/Case is an alternative form of **the if/else/if** branching structure. In **a switch/case** statement, instead of evaluating some if condition, the program evaluates the value of a variable or expression and compares it to some possible case. Each case represents a potential value, and when a match is found, the corresponding block of code is executed. If there are no matching cases, the **default** case is executed (if provided), similar to the **else** block in an **if/else statement**.

A **switch/case** structure can make code easier to read and organize, especially when dealing with multiple conditions based on a single variable. The structure looks like this:

```
switch(variabel){
    case <value>:
        // blok kode
        break;
    case <value>:
        // blok kode
        break;
    default:
        // blok kode
}
```

Step	Description
1	Create a new File named **switchcase.html**
2	Type a program below
3	Observe what appears in the browser
4	Record your observations (Question No. 23)

```
<!DOCTYPE html>
<html>
<head>
    <title>Percabangan switch/case</title>
</head>
<body>
    <script>
        var jawab = prompt("Kamu beruntung! Silahakn pilih hadiahmu dengan memasukan angka 1 sampai 5");
        var hadiah = "";
            switch(jawab){
            case "1":
                hadiah = "Tisu";
                break;
            case "2":
                hadiah = "1 Kotak Kopi";
                break;
            case "3":
                hadiah = "Sticker";
                break;
            case "4":
                hadiah = "Minyak Goreng";
                break;
            case "5":
                hadiah = "Uang Rp 50.000";
                break;
            default:
                document.write("<p>Opps! anda salah pilih</p>");}
        if(hadiah === ""){
            document.write("<p>Kamu gagal mendapat hadiah</p>");
        } else {
            document.write("<h2>Selamat kamu mendapatkan " + hadiah + "</h2>");
        }
    </script>
</body>
</html>
```

Nested Branching

Nested Branching refers to a condition in which one branching statement (such as **if**, **else**, switch, etc.) is placed inside another branching statement. This allows for more complex decision-making processes where multiple conditions need to be evaluated at different levels. In nested branching, the outcome of one condition can depend on the outcome of another, providing more granular control over the flow of the program.

For example, you can stack **an if** statement inside another if **statement** to first check one condition and then, based on that, check the second condition in the first block.

Step	Description
1	Create a new file named **nestedif.html**
2	Type a program below
3	Observe what appears in the browser
4	Record your observations (Question No.24)

```
<!DOCTYPE html>
<html>
<head>
    <title>Percabangan Ternary</title>
</head>
<body>
    <script>
        var username = prompt("Username:");
        var password = prompt("Password:");

        if(username == "mahasiswa"){
            if(password == "kopi"){
                document.write("<h2>Selamat datang </h2>");
            } else {
                document.write("<p>Password salah, coba lagi!</p>");
            }
        } else {
            document.write("<p>Anda tidak terdaftar!</p>");
        }
    </script>
</body>
</html>
```

Practical Section 9: Looping

Loops will help us execute code over and over again, however we want. There are five types of loops in Javascript. In general, this loop is divided into two: *counted loop* and *uncounted loop*. The differences are:

- **The Counted Loop** is an obvious loop and of course there are many loops.
- **An Uncounted Loop**, is a loop that is unclear how many times it has to repeat.

Loops included in *the Counted Loop*:

1. For Loops
2. Foreach Loop
3. Repeat Loop

Loops included in *the Uncounted Loop*:

1. Perulangan While
2. Perulangan Do/While

FOR Loops

A for loop is a loop that is included in a *counted loop*, because it is clear how many times it will repeat. It looks like this:

```
for(let i = 0; i < 10; i++){
    document.write("<p>Perulangan ke-" + i + "</p>")
}
```

Step	Description
1	Create a new file named **for-javascript.html**
2	Type a program below `<!DOCTYPE html>` `<html>` `<body>` `<h2>JavaScript Loops</h2>` `<p id="demo"></p>` `<script>` `var text = "";` `var i;` `for (i = 0; i < 5; i++) {` `  text += "The number is " + i + " ";` `}` `document.getElementById("demo").innerHTML = text;` `</script>` `</body>` `</html>`
3	Observe what appears in the browser
4	Record your observations (Question No. 25)

WHILE Loop

A while loop is a loop that is included in an uncounted loop. A while loop can also be a counted loop by providing a counter in it.

Step	Description
1	Create a new file named **`while.html`**
2	Type a program below
3	Observe what appears in the browser
4	Record your observations (Question No. 26)

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

<h2>JavaScript while</h2>

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

<script>
var text = "";
var i = 0;
while (i < 10) {
  text += "<br>The number is " + i;
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Perulangan DO WHILE

The do/while loop is a variation of the while loop in JavaScript. The main difference between the two is that a do/while loop will always execute the code inside the loop **at least once**, regardless of whether the condition is true or false. This is because the condition is evaluated **after** the code block is executed, not before, as in a standard while loop:

```
do {
    // blok kode yang akan diulang
} while (<kondisi>);
```

Main characteristics:

- The code inside the do block runs first, and then the condition is checked.
- If the conditions are correct, the loop repeats; If false, the loop stops.

This type of loop ensures that the code inside the loop is executed at least once, even if the condition is wrong in the first place.

Step	Description
1	Create a new file named **dowhile.html**
2	Type a program below `<!DOCTYPE html>` `<html>` `<body>` `<h2>JavaScript do ... while</h2>` `<p id="demo"></p>` `<script>` `var text = ""` `var i = 0;` `do {` `  text += " The number is " + i;` `  i++;` `}` `while (i < 10);` `document.getElementById("demo").innerHTML = text;` `</script>` `</body>` `</html>`
3	Observe what appears in the browser
4	Record your observations (Question No.27)

Reference:

1) Jason Beaird, The principles of Beautiful Web Design
2) Rian Ariona, Learn HTML and CSS (Fundamental Tutorial on learning HTML and CSS)
3) Adi Hadisaputra, HTML and CSS Fundamentals from Root to Leaf
4) John Duckett, HTML and CSS design and build websites