UNIT 3 : OVERVIEW OF JAVASCRIPT

3.1 Overview of Client & Server-SideScripting

3.2 Structure of Java Script

3.3 Data types and Variables

3.4 Operators (Arithmetic, Assignment, Comparison, Logical and Conditional Operator)

3.5 Control Structure

- 3.5.1 If…Else, switch..case
- 3.5.2 While, Do…While, For Loop
- 3.5.3 break, continue

3.6 Java Script String and Events

3.6.1 Javascript Strings types

3.6.2 String functions:

concat(), split(), indexOf(), lastIndexOf(),substring(),

trim(), slice(), replace(), charAt()

3.6.3 Javascript Events :

3.6.3.1 Mouse Events : (click, mouseover, mouseremove, mouseout, mouseup)

3.6.3.2 keyboard Events : (keyup,keydown)

3.6.3.3 Form Event : (focus, submit, blur, change)

Webpage

- A web page is a document that acts as a web resource on the World Wide Web.
- In order to display a web page, a web browser is needed to retrieve web pages from the Internet.
- When accessed by a web browser it may be displayed as a web page on a monitor or mobile device.

Web Browser

- A web browser is a software application for accessing information on the World Wide Web.
- When a user requests a particular website, the web browser retrieves the necessary content from a web server and then displays the resulting web page on the user's device.
  - Google Chrome
  - Mozilla Firefox
  - Internet Explorer
  - Opera Mini

Categories of Webpage

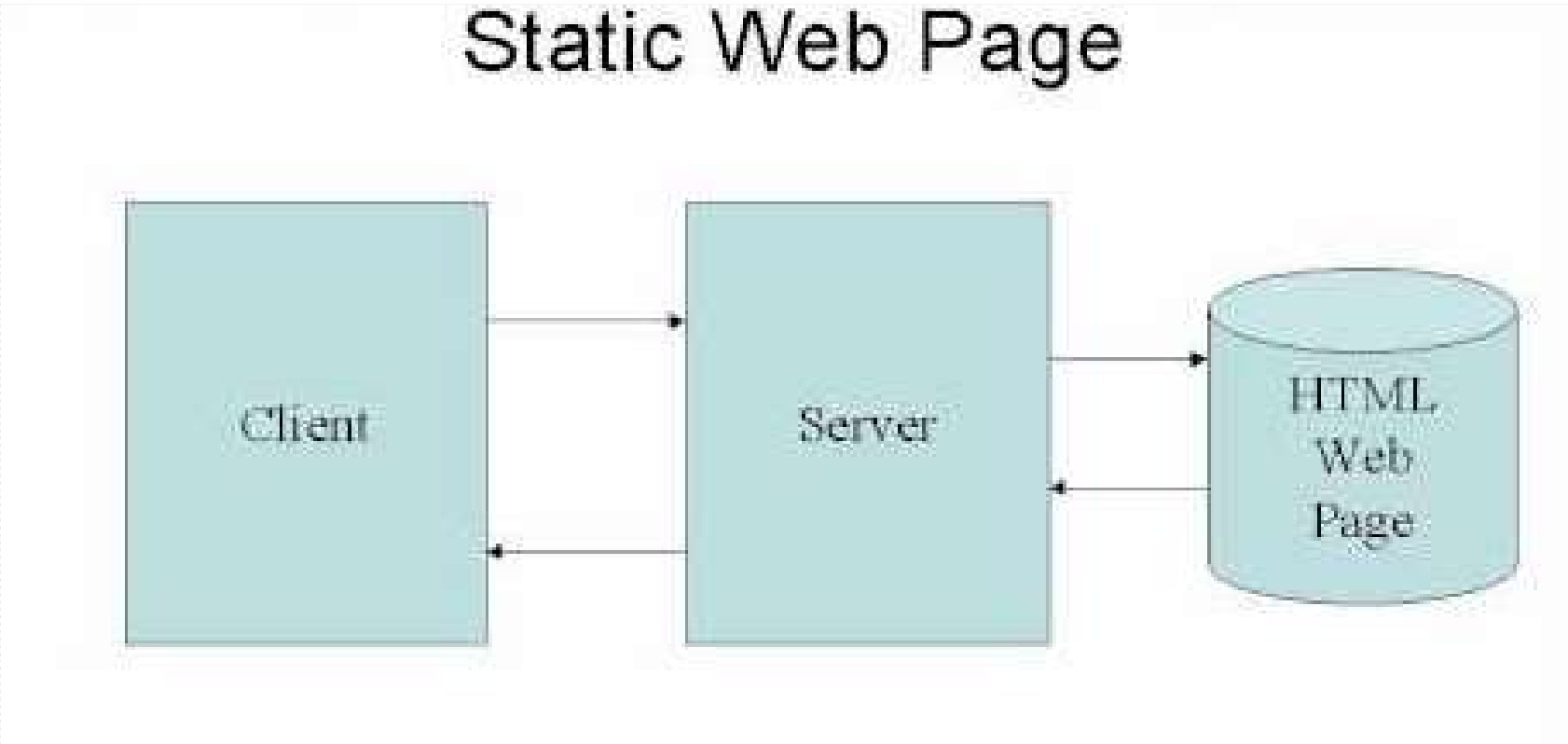
- Static Webpage
- Dynamic Webpage
- Active Webpage

Static, Dynamic and Active Page

Static Webpage

- A static webpage display the same information for all users.
- Static Webpage makes customization impossible.
- Static Webpages are suitable for the contents that never or rarely need to updated.
- Each request for a static document results in exactly the same response.

Architecture

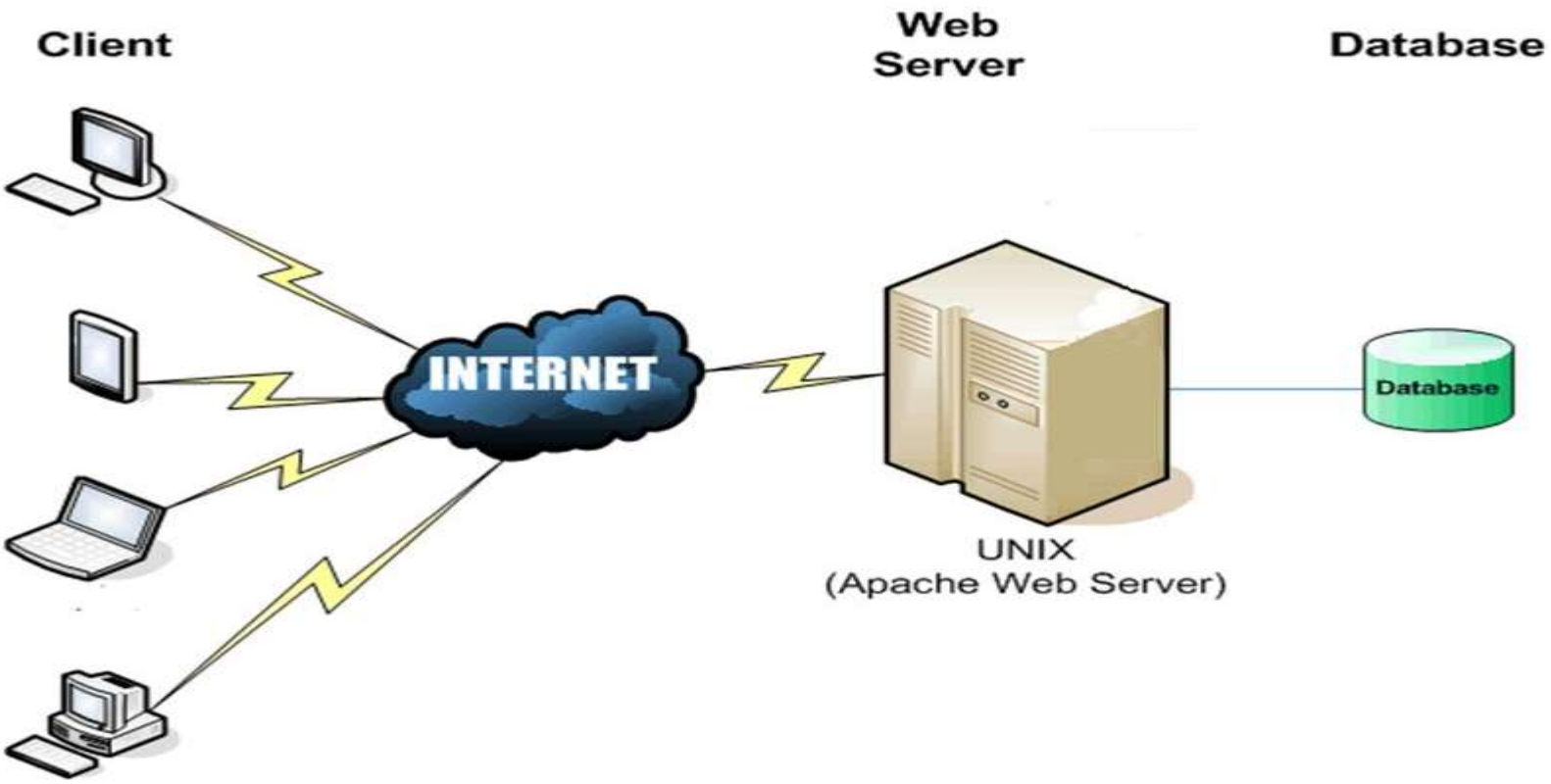


- Advantages
  - Quick to develop
  - Cheap to develop
  - Cheap to host
- Disadvantages
  - Require web development expertise to update website.

Dynamic Webpage

- Dynamic Webpages is a webpage whose constructions is controlled by an application server at server side scripting.
- They can be customized according to the need of users.
- Dynamic webpages can accept information from the users.

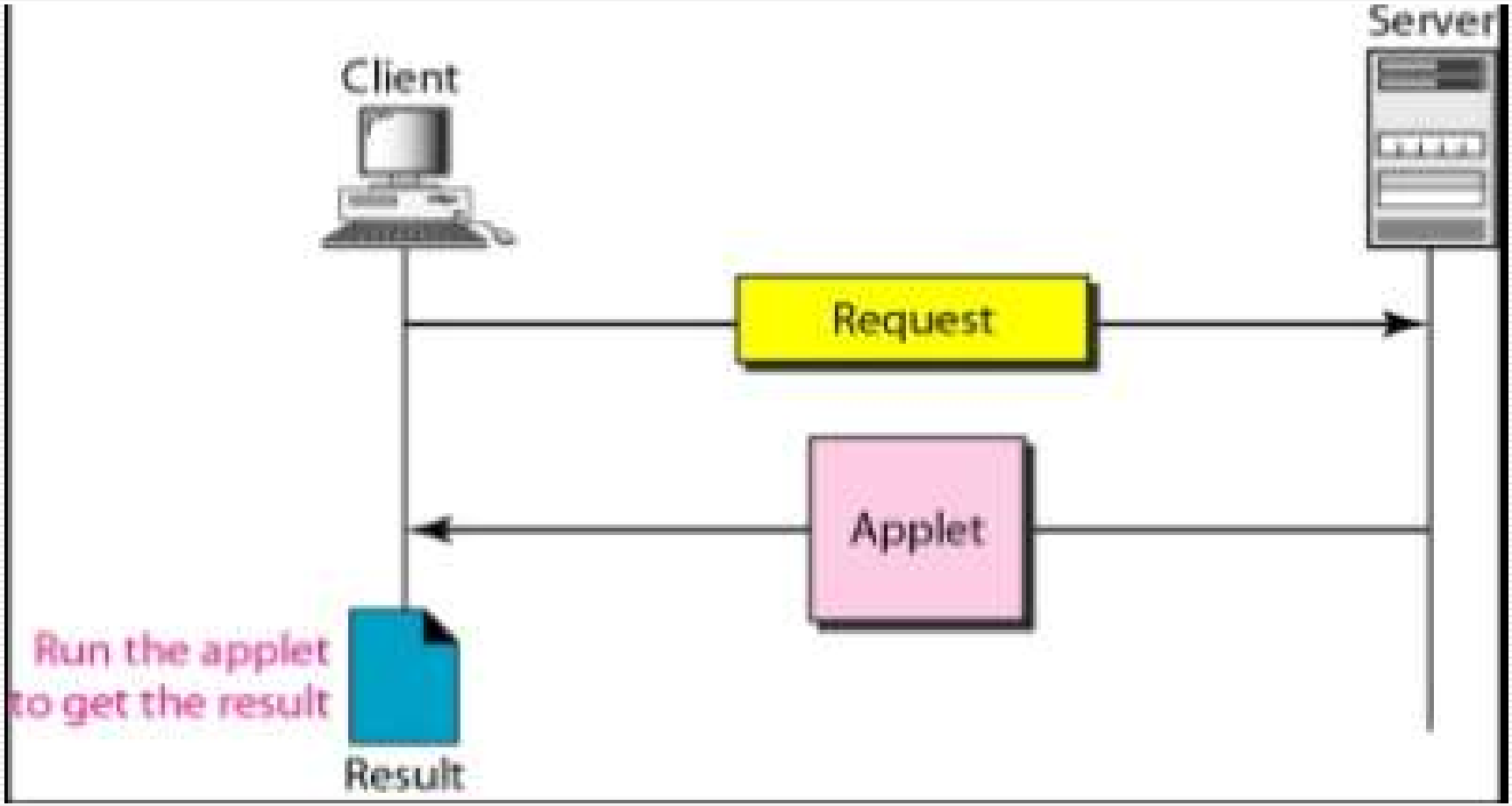
Architecture



- Advantages
  - Restructuring of Dynamic Website is easier
  - Can be developed or embedded as and when requires
  - Able to access content from any computer, anywhere in the world.
- Disadvantages
  - Longer Development time
  - For smaller company Dynamic Websites are expensive.

Active Webpage

- An **active web page** is a **page** where the browser performs the logic instead of the server.
- An Active web document consist of a computer program(Javascript, Applet) that the server sends to the browser and the browser must run locally.
- So for example when you've got a **page** where you're showing share prices, then you want it to update e.g. every 5 seconds. A solution would be to use AJAX with JavaScript.


Server
Client
Request
Applet
Run the applet
to get the result
Result

- Advantages
  - Ability to update information regularly.
- Disadvantages
  - An Active document is a potentially risk because the document can export or import information.
  - Some browser does not support Applet.

3. Javascript

3.3.1 Overview of Client and Server Side Scripting

3.3.2 Structure of Javascript

3.3.3 Data Types and Variables

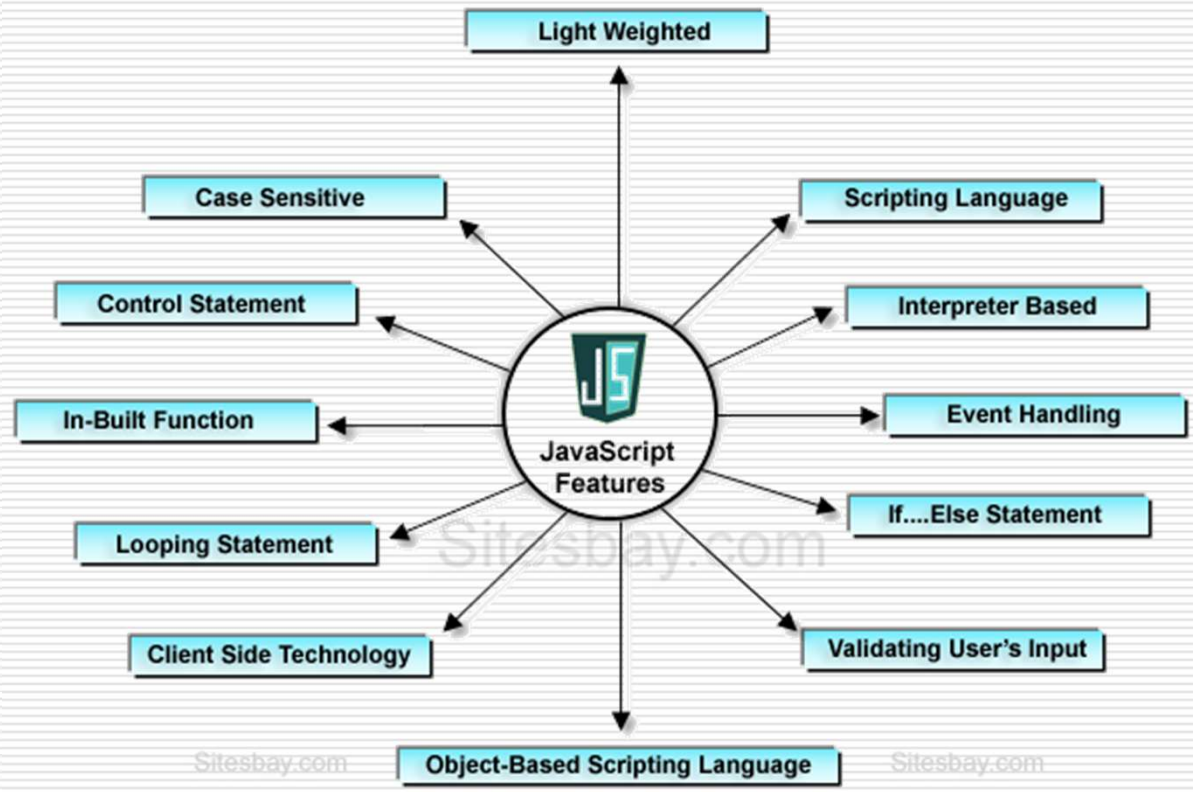
3.3.4 Operators

- Arithmetic Operator
- Assignment Operator
- Comparison Operator
- Logical Operator
- Conditional Operator

Javascript

- **JavaScript** was developed by **Brendan Eich** in **1995**, which appeared in Netscape, a popular browser of that time.
- The language was initially called LiveScript and was later renamed **JavaScript**.
- There are many programmers who think that **JavaScript** and Java are the same. In fact, **JavaScript** and Java are very much unrelated.

- JavaScript is a very powerful **client-side scripting language**. JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- In other words, you can make your webpage more lively and interactive, with the help of JavaScript.
- JavaScript is also being used widely in game development and Mobile application development.


Light Weighted
Case Sensitive
Scripting Language
Control Statement
Interpreter Based
In-Built Function
JavaScript Features
Event Handling
If....Else Statement
Looping Statement
Client Side Technology
Validating User's Input
Object-Based Scripting Language
Sitesbay.com
Sitesbay.com
Sitesbay.com

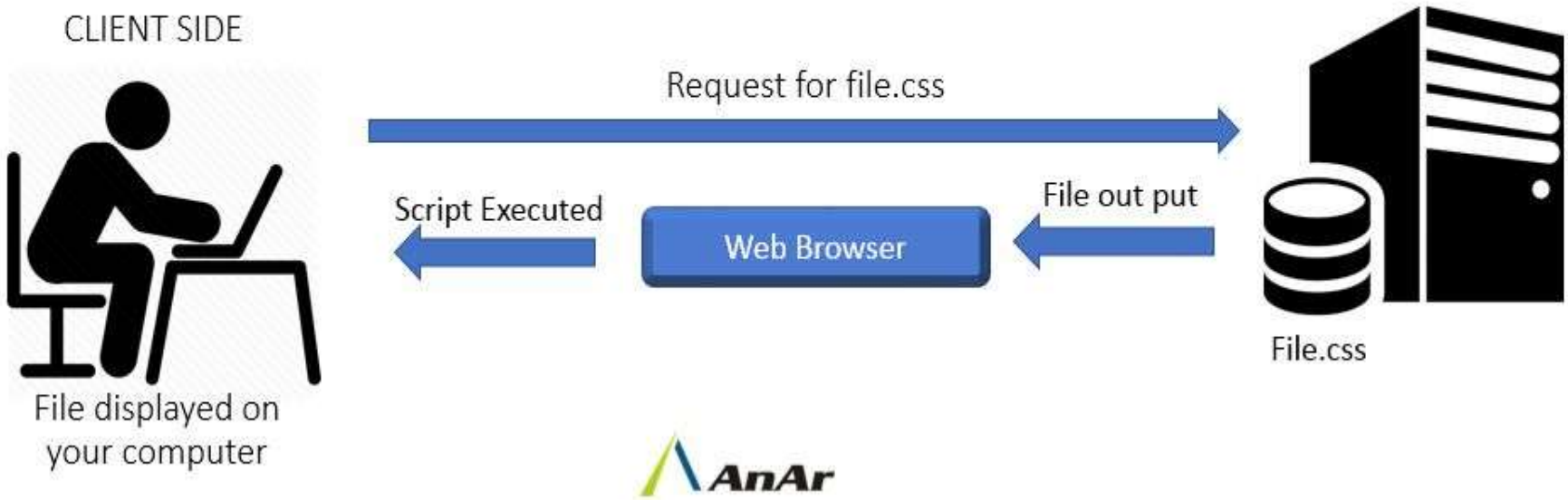
3.3.1 Overview of Client and Server Side Scripting

- A scripting or script language is a programming language for a special run-time environment that automates the execution of tasks; the tasks could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted.

Client Side Scripting

- **Client-side scripting**
- (embedded **scripts**) is code that exists inside the **client's** HTML page. This code will be processed on the **client** machine and the HTML page will NOT perform a PostBack to the web-**server**. Traditionally, **client-side scripting** is used for page navigation, data validation and formatting.

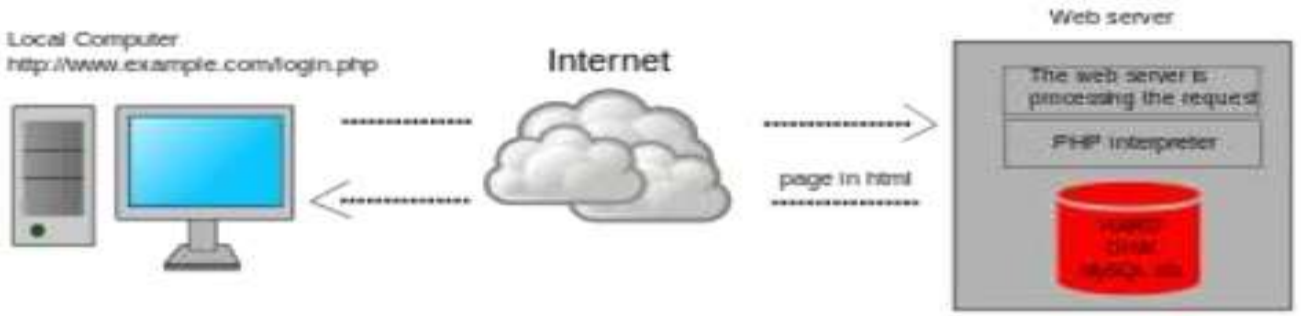
The process of client side scripting



Server Side Scripting

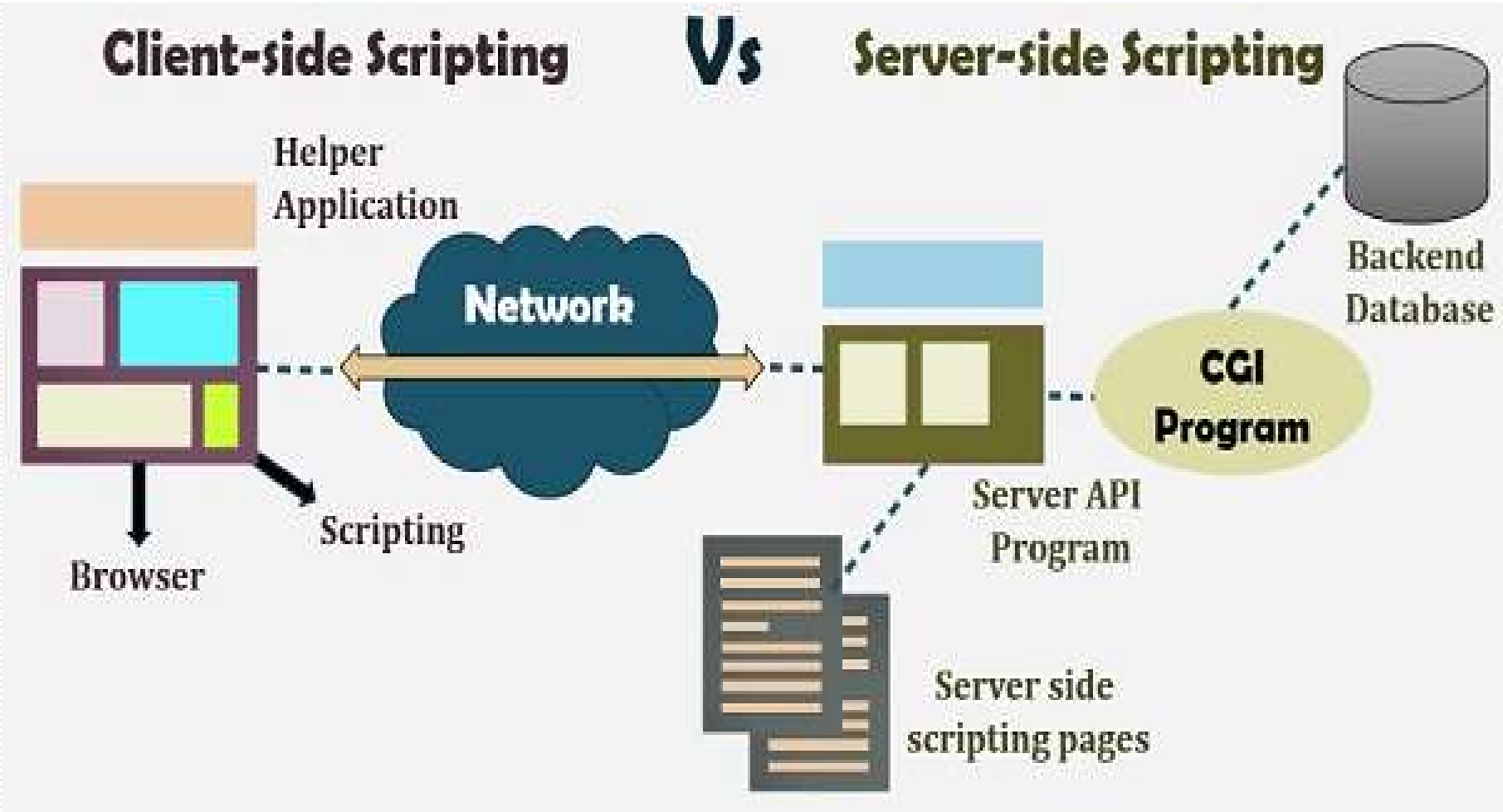
- Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's request to the website. The alternative is for the web server itself to deliver a static web page.

SERVER SIDE SCRIPTING



Server Side VS Client Side

BASIS FOR COMPARISON	SERVER-SIDE SCRIPTING	CLIENT-SIDE SCRIPTING
Basic	Works in the back end which could not be visible at the client end.	Works at the front end and script are visible among the users.
Processing	Requires server interaction.	Does not need interaction with the server.
Languages involved	PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.	HTML, CSS, JavaScript, etc.
Affect	Could effectively customize the web pages and provide dynamic websites.	Can reduce the load to the server.
Security	Relatively secure.	Insecure
Source Visibility	Not visible to any user	User can view in web browser


Client-side Scripting
Vs
Server-side Scripting
Helper
Application
Network
Backend
Database
CGI
Program
Server API
Program
Scripting
Browser
Server side
scripting pages

3.3.2 Structure of Javascript

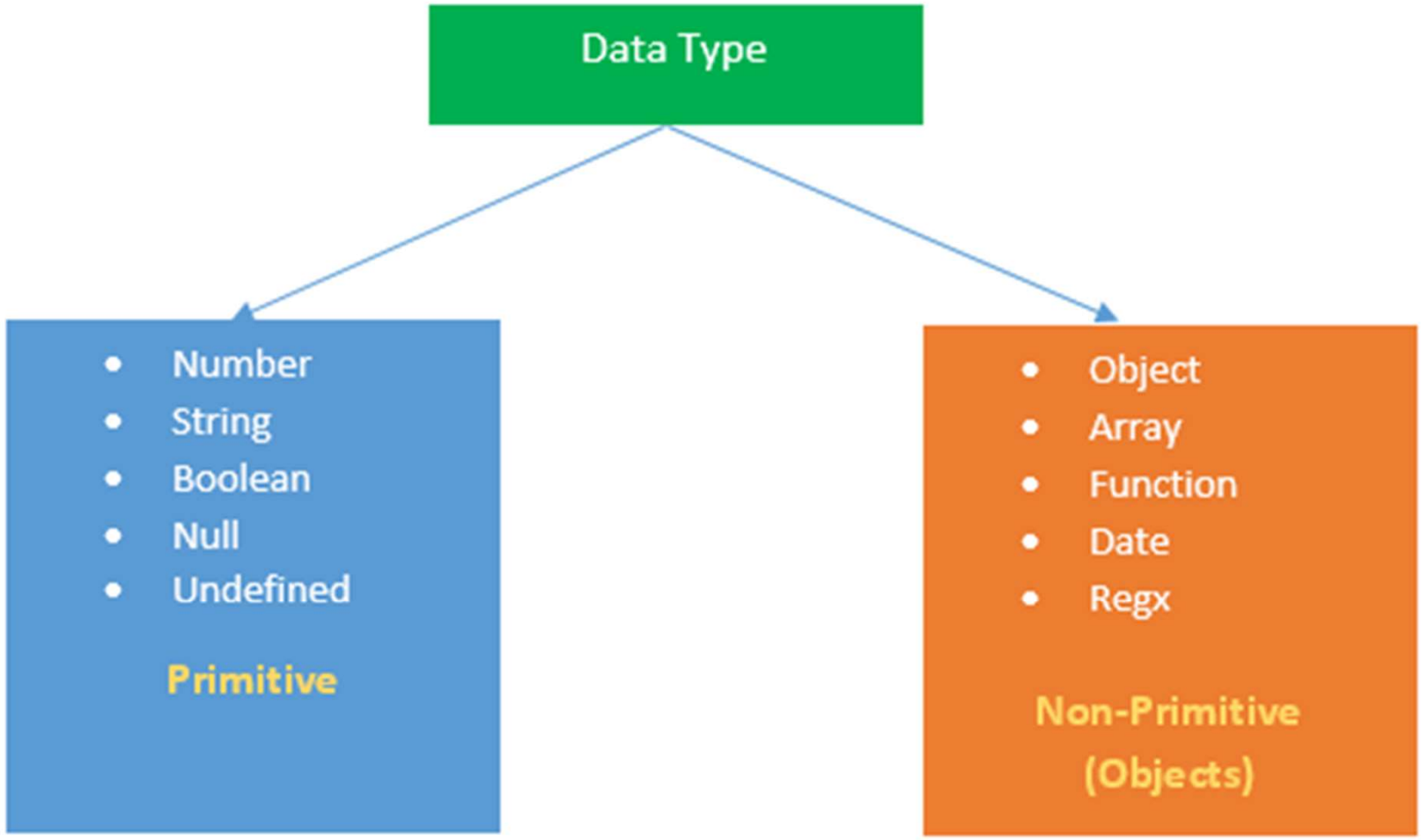
```
<HTML>
        <HEAD>
                <TITLE>JAVASCRIPT STRUCTURE</TITLE>
                <SCRIPT LANGUAGE="JAVASCRIPT">
                        //WRITE JAVASCRIPT CODE HERE
                </SCRIPT>
        </HEAD>
        <BODY>
                <FORM>

                </FORM>
        </BODY>
</HTML>
```

- Javascript is case sensitive language
- It is loosely typed language

Comments

- Single line Comment
  - // is used for Single Line Comment
- Multi line Comment
  - /* and */ is used for Multi Line Comment

3.3.3 Data Types and Variables



JavaScript Data Type

Number:

- JavaScript has only one type of numbers.
- Numbers can be written with, or without decimals:
  - var x1 = 34.00; // Written with decimals
  - var x2 = 34; // Written without decimals

String:

- A string (or a text string) is a series of characters like “SYBCA THE GREAT CLASS".
- Strings are written with quotes. You can use single or double quotes:
  - var carName1 = "Volvo XC60";// Using double quotes
  - var carName2 = 'Volvo XC60'; // Using single quotes

Boolean:

- Booleans can only have two values: true or false.
  - var a=true;
  - var flag=false;

Null:

- In JavaScript null is "nothing". It is supposed to be something that doesn't exist.
- Unfortunately, in JavaScript, the data type of null is an object.
  - var a = null;

undefined:

- In JavaScript, a variable without a value, has the value undefined. The type is also undefined.
  - var car; // Value is undefined, type is undefined

Variables

- A **JavaScript variable** is simply a name of storage location.
- There are two types of variables in JavaScript : local variable and global variable.
- There are some rules while declaring a JavaScript variable (also known as identifiers).
  1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
  2. After first letter we can use digits (0 to 9), for example value1.
  3. JavaScript variables are case sensitive, for example x and X are different variables.

- “var” is used to declare variable.
- Syntax :
  - var varname=value”;
- Example:
  - var n=10;
  - var s=“SYBCA”;
  - var flag=“true”;
  - var a=null;

3.3.4 Operators

1. Arithmetic Operator
2. Assignment Operator
3. Comparison Operator
4. Logical Operator
5. Conditional Operator
6. String Operator

1. Arithmetic Operator

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Example:

```
<SCRIPT>
                    var a=10;
                    var b=10;
                    document.write((a+b)+"<br>");
                    document.write(a-b+"<br>");
                    document.write(a*b+"<br>");
                    document.write(a/b+"<br>");
                    document.write(a%b+"<br>");
                    document.write(a**b+"<br>");
                    document.write((++a)+"<br>");
                    document.write((--b)+"<br>");
            </SCRIPT>
```

Output

20
0
100
1
0
10000000000
11
9

2. Assignment Operator

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Example:

```
<SCRIPT>
                    var a=parseInt(10);
                    var b=parseInt(10);
                    document.write((a+=b)+"<br>");
                    document.write((a-=b)+"<br>");
                    document.write((a*=b)+"<br>");
                    document.write((a/=b)+"<br>");
                    document.write((a%=b)+"<br>");
                    document.write((a**=b)+"<br>");

          </SCRIPT>
```

Output:

20
10
100
10
0
0

3. Comparison Operator

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Example:

```
<SCRIPT>
                    var a=parseInt(10);
                    var b=parseInt(10);
                    document.write((a==b)+"<br>");
                    document.write((a===b)+"<br>");
                    document.write((a!=b)+"<br>");
                    document.write((a<b)+"<br>");
                    document.write((a>b)+"<br>");
                    document.write((a<=b)+"<br>");
                    document.write((a>=b)+"<br>");

          </SCRIPT>
```

Output:

true
true
false
false
false
true
true

4. Logical Operator

Operator	Description
&&	logical and
||	logical or
!	logical not

Example:

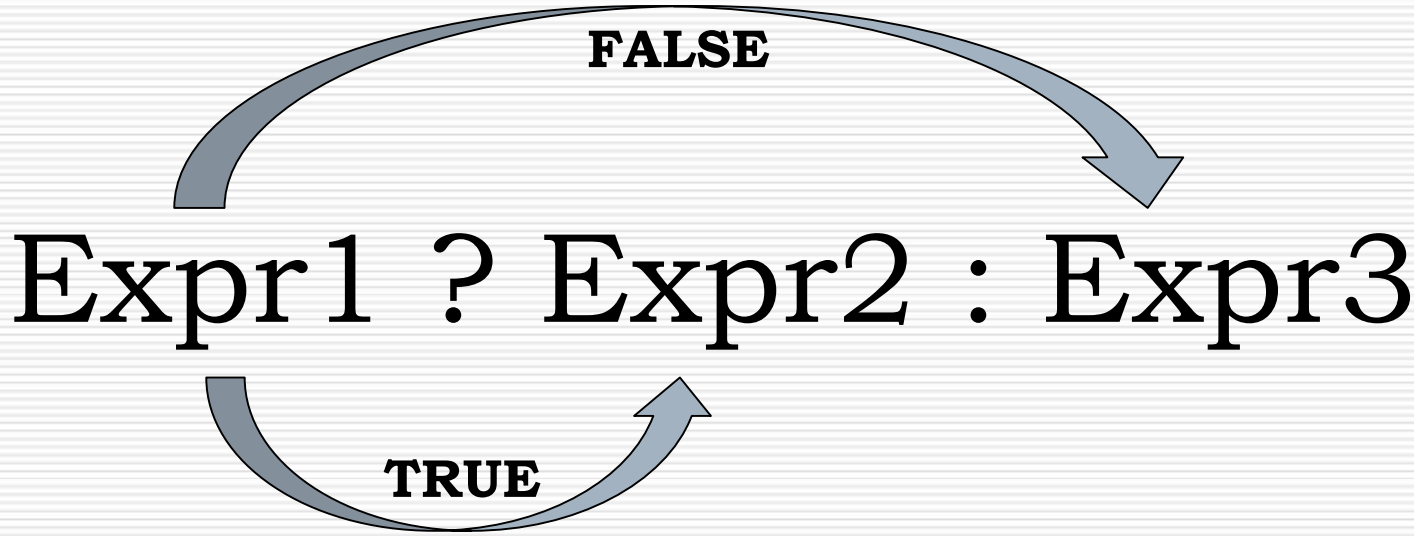
```
<SCRIPT>
                    var a=parseInt(10);
                    var b=parseInt(10);
                    document.write((a<b&&a>b)+"<br>");
                    document.write((a<b||a>b)+"<br>");
                    document.write(!(a<b)+"<br>");

          </SCRIPT>
```

Output

false
false
true

5. Conditional Operator

- ? : Operator
- Also known as Ternary Operator


FALSE
Expr1 ? Expr2 : Expr3
TRUE

Example:

```
<SCRIPT>
                    var a=parseInt(10);
                    var b=parseInt(10);
                    var c= a<b?a:b;
                    document.write(c);
              </SCRIPT>
```

Output: 10

6. String Operator

- The + operator can also be used to add (concatenate) strings.

Example:

```
<SCRIPT>
                    var a="VNSGU";
                    var b="SURAT";
                    var c= a+b;
                    document.write(c);
              </SCRIPT>
```

OUTPUT:

VNSGUSURAT

3.3.5 Control Structures

1. Branching
2. Looping
3. Jumping

Branching

- Simple if Statement
- if….else Statement
- Netsted if Statement
- Ladder if or elseif Statement
- Switch…case Statement

Looping

- Entry Control Loop
  - for loop
  - while loop
- Exit Control Loop
  - do…while Loop

Jumping

- break
- continue

Branching

- Simple if Statement
- if….else Statement
- Netsted if Statement
- Ladder if or elseif Statement
- Switch…case Statement

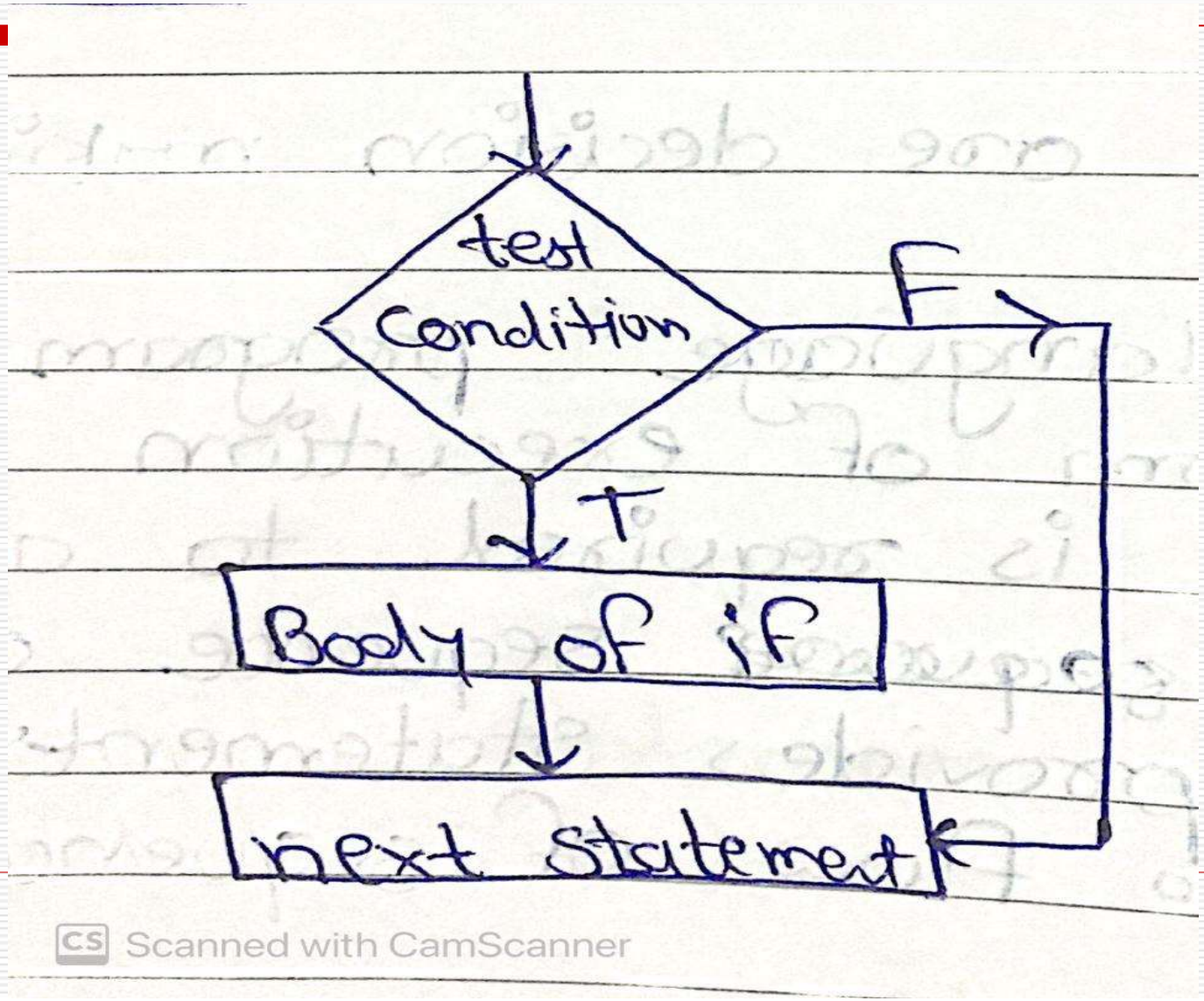
Simple if Statement

- When we need to execute a block of statements only when a given condition is true then we use if statement.

Syntax:

```
if (test - condition)
{
    //Statement
}
Next statements
```

Flowchart:



Example:

```
<SCRIPT>
                    var a=10;
                    var b=20;
                    if(a<b)
                                document.write("A is min");
          </SCRIPT>
```

OUTPUT:

A is min

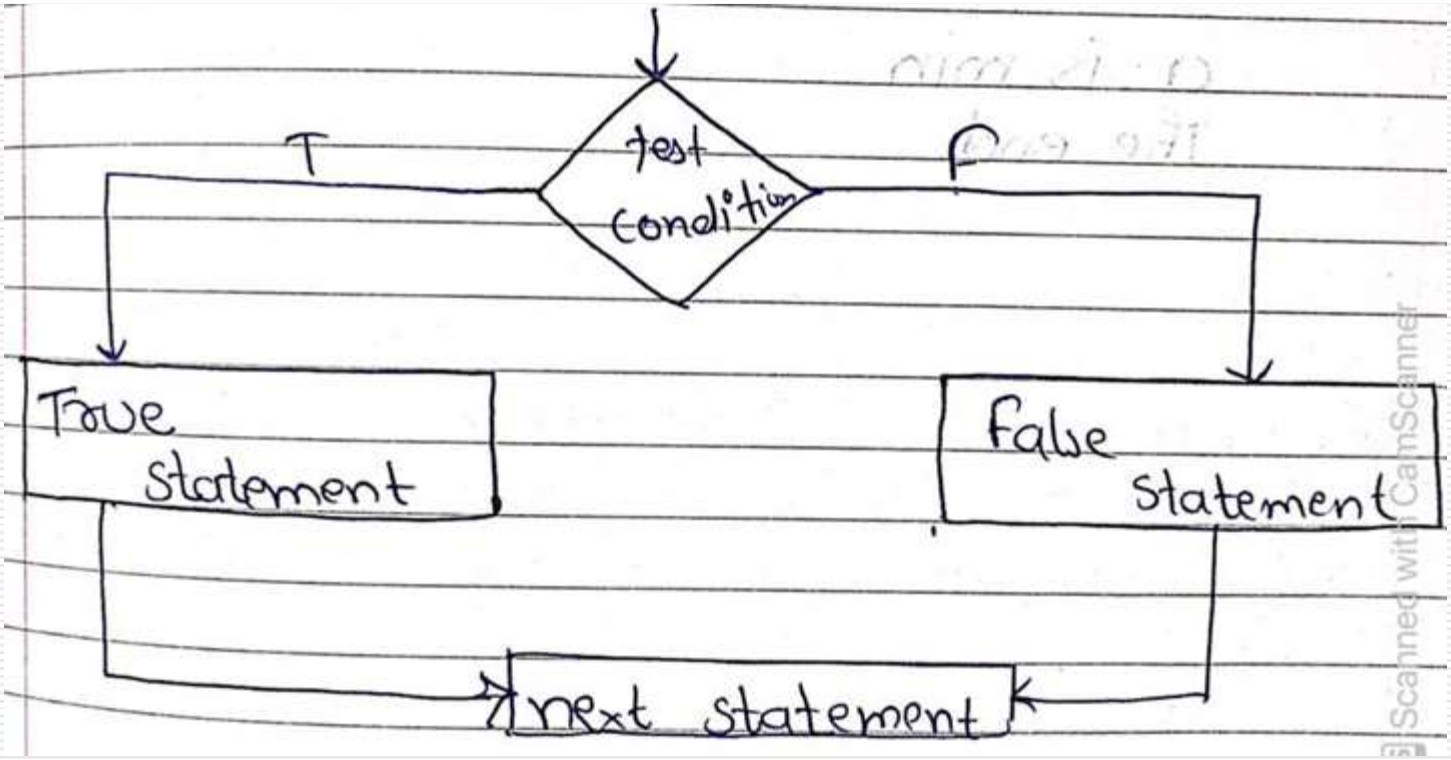
if...else Statement

- An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

Syntax:

```
if(test - condition)
{
    // True Statements
}
else
{
    // False Statements
}
```

Flowchart:


T
test condition
F
True statement
False statement
next statement

Example:

```
<SCRIPT>
                var a=100;
                var b=20;
                if(a<b)
                        document.write("A is min");
                else
                        document.write("B is min");
        </SCRIPT>
```

OUTPUT:

B is min

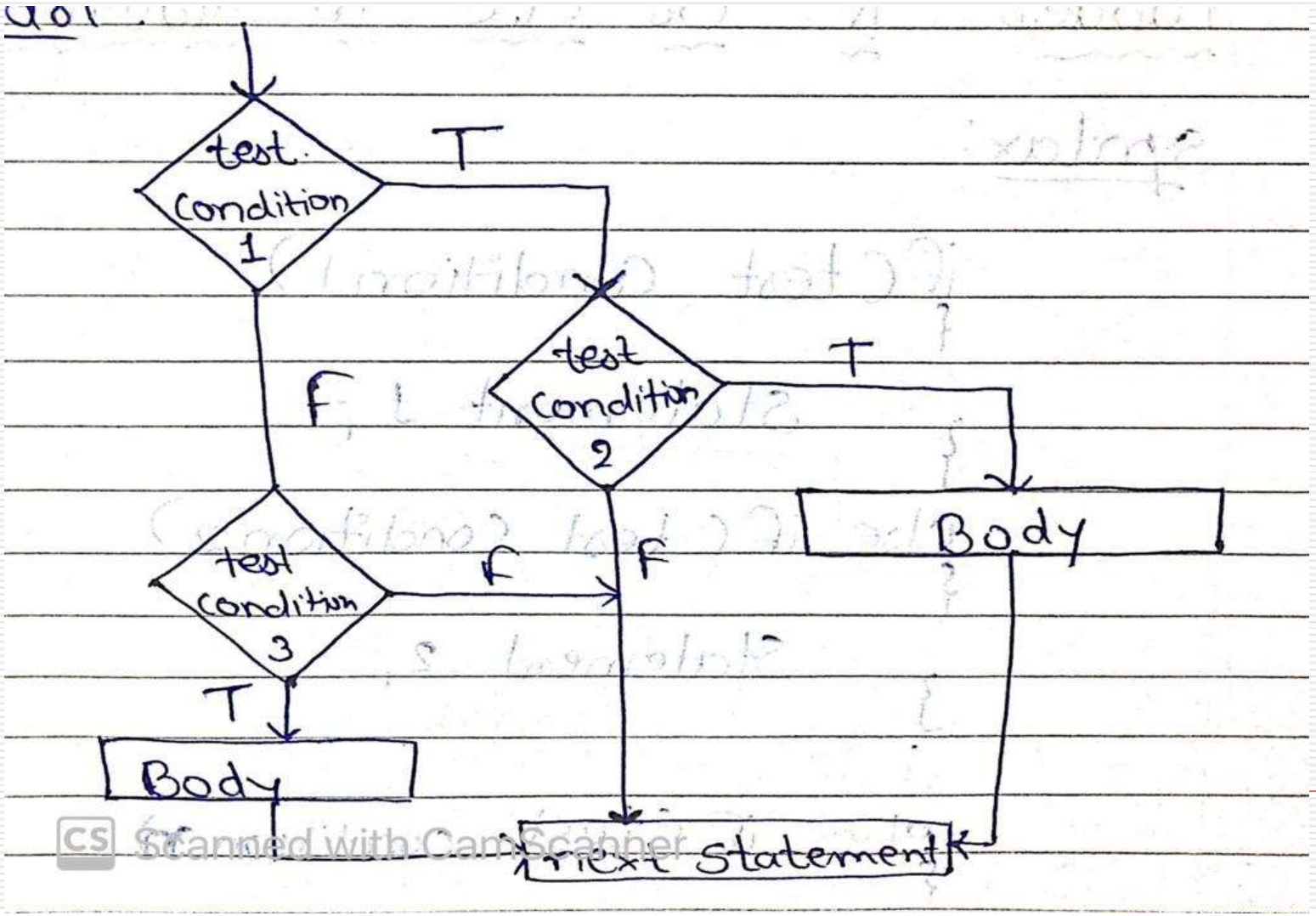
Nested if Statement

- If statement within if statement is known as nested if statement.

Syntax:

```
if(test - condition 1)
{
   if(test - condition 2)
   {
            // True Statement
   }
}
else
{
      if(test - condition 3)
      {
            // False Statement
      }
}
```

Flowchart:



Example:

```
<SCRIPT>
        var a=10,b=20,c=30;
        clrscr();
        if(a<b)
        {
                if(b<c)
                                document.write("Inside IF");
        }
        else
        {
                 document.write("Inside ELSE");
        }
}
</SCRIPT>
```

OUTPUT:

Inside IF

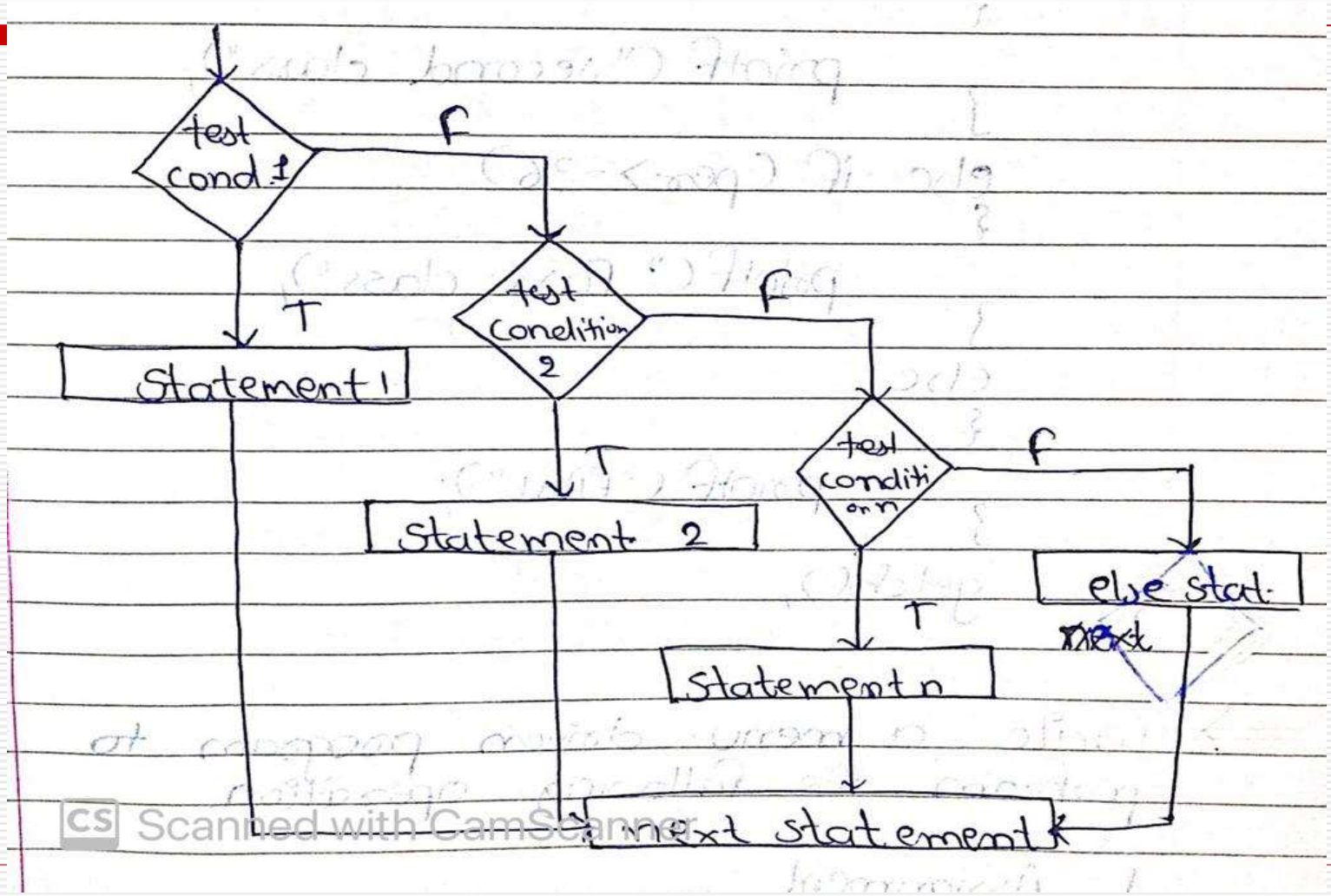
Ladder if or elseif Statement

- else-if statements in javascript is like another if condition, it's used in a program when if statement having multiple decisions.

Syntax:

```
if(test - condition 1)
{
   // Statement 1
}
else if(test - condition 2)
{
        // Statement 2
}
…
…
…
else if(test - condition n)
{
        // Statement n
}
else
{
        // Else Statement
}
next- statement
```

Flowchart:


test cond 1
F
T
Statement 1
test Condition 2
F
T
Statement 2
test conditi on n
F
T
Statement n
else stat.
next statement

Example:

```
<SCRIPT>
                    var per=80;
                    if(per>=70)
                              document.write("DIST");
                    else if(per>=60)
                              document.write("FIRST");
                    else if(per>=50)
                              document.write("SECOND");
                    else if(per>=35)
                              document.write("PASS");
          </SCRIPT>
```

OUTPUT:

DIST

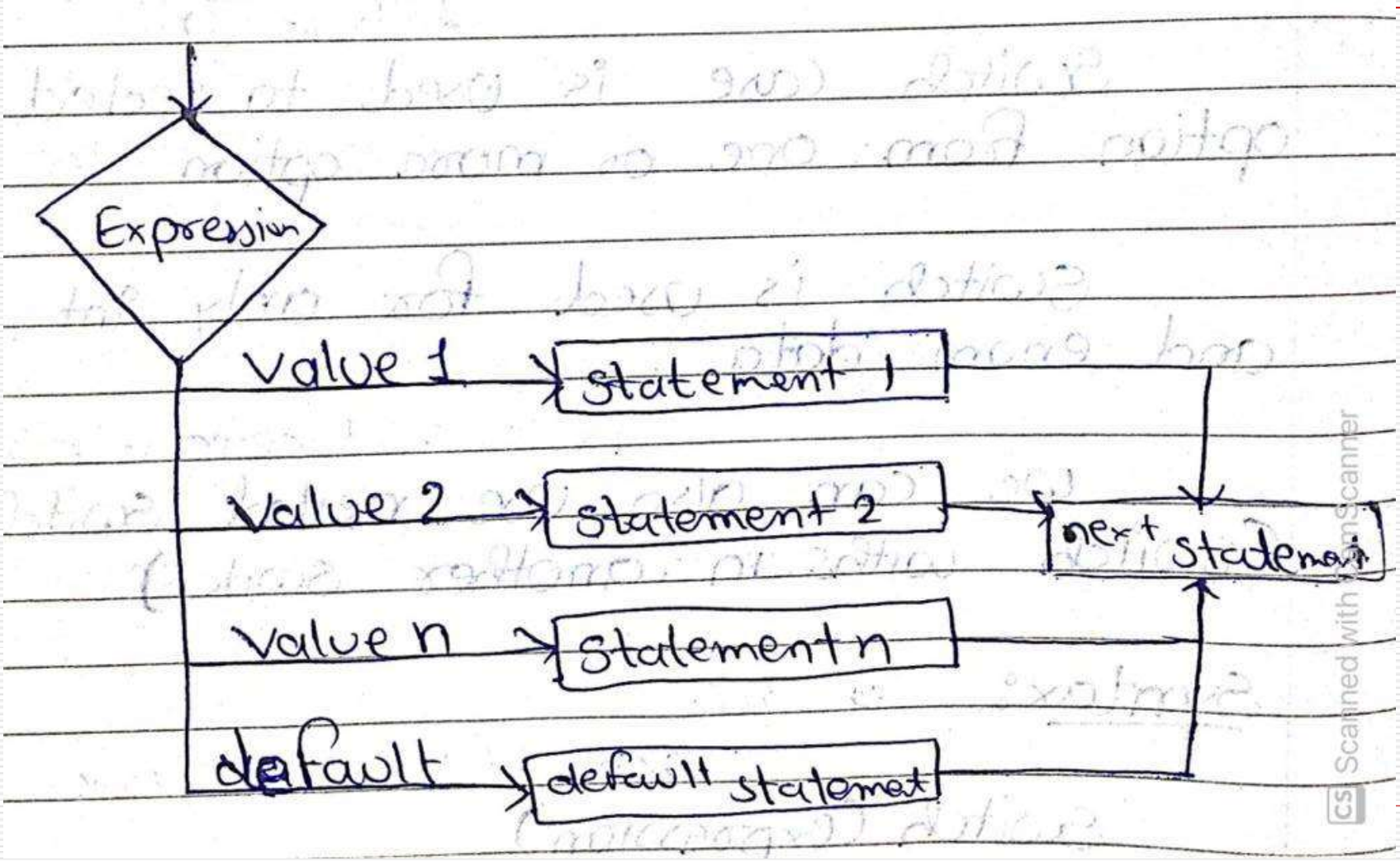
Switch Case or Selection Statement

- Switch case is used to select one option from one or more option.
- Switch is used for only int, char and enum data.
- We can also use nested switch (switch within switch).

Syntax:

```
switch(expression)
{
        case value1:
                Statement 1;
                break;
        case value2:
                Statement 2;
                break;
        …
        …
        …
        case valuen:
                Statement n;
                break;
        default:
                default Statement;
                break;
}
next- statement;
```

Flowchart:


Expression
value 1
statement 1
value 2
statement 2
value n
statement n
default
default statement
next statement

Example:

```
<SCRIPT>
                    var c=2;
                    switch(c)
                    {
                              case 1:
                                        document.write("QUIZ");
                                        break;
                              case 2:
                                        document.write("ASSIGNMENT");
                                        break;
                              case 3:
                                        document.write("VIVA");
                                        break;
                    }
          </SCRIPT>
```

OUTPUT:

ASSIGNMENT

Looping

- Entry Control Loop
  - for loop
  - while loop
- Exit Control Loop
  - do…while Loop

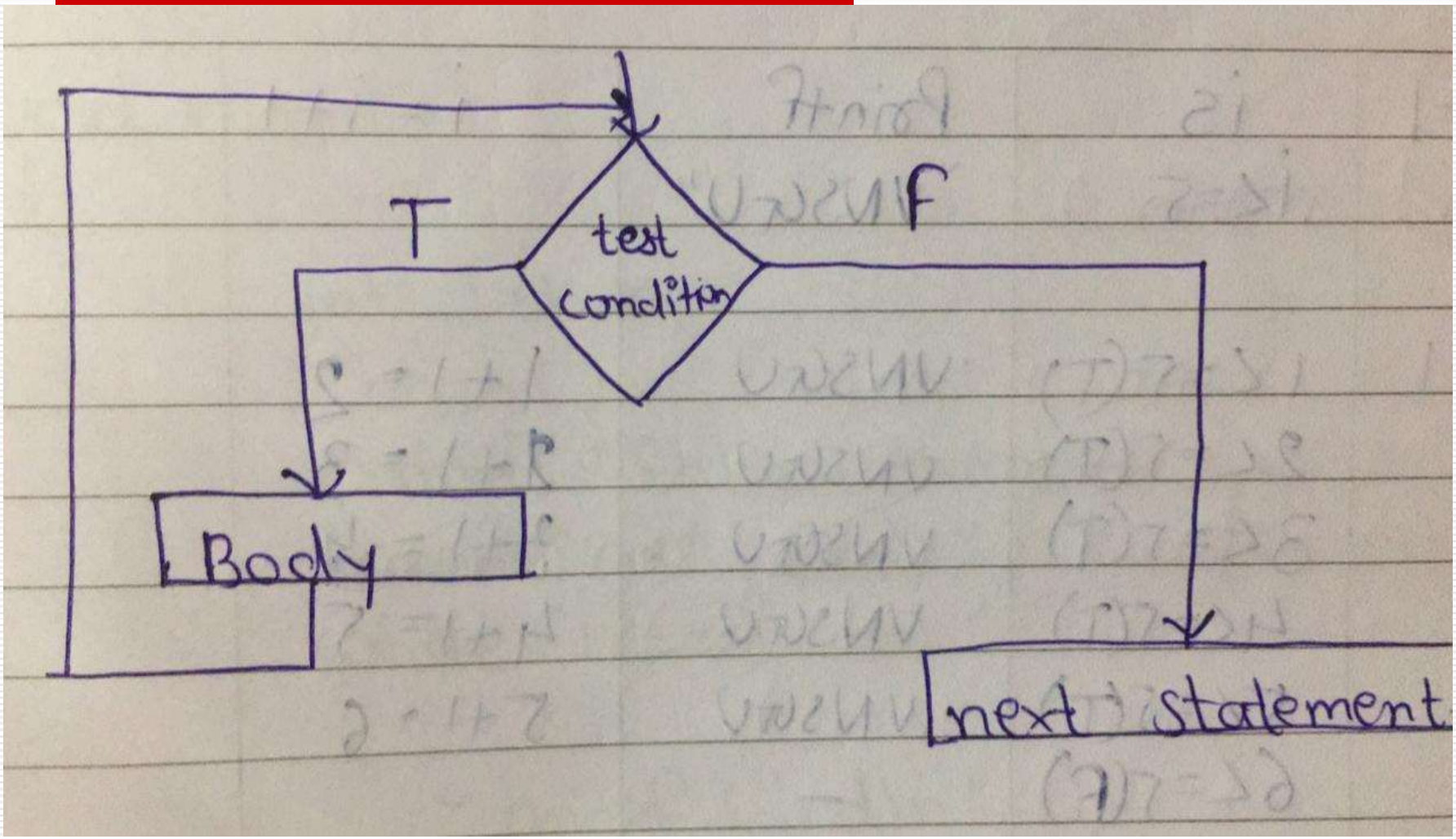
while loop

- A **while** loop in JAVASCRIPT repeatedly executes a target statement as long as a given condition is true.

Syntax:

```
while(Test Condition)
{
    //Body of Loop
}
Next-statements;
```

Flowchart:



Example:

```
<SCRIPT>
        var n=10,i=1;
        while(i<=n)
        {
                document.write(i);
                i++;
        }
    </SCRIPT>
```

OUTPUT:

12345678910

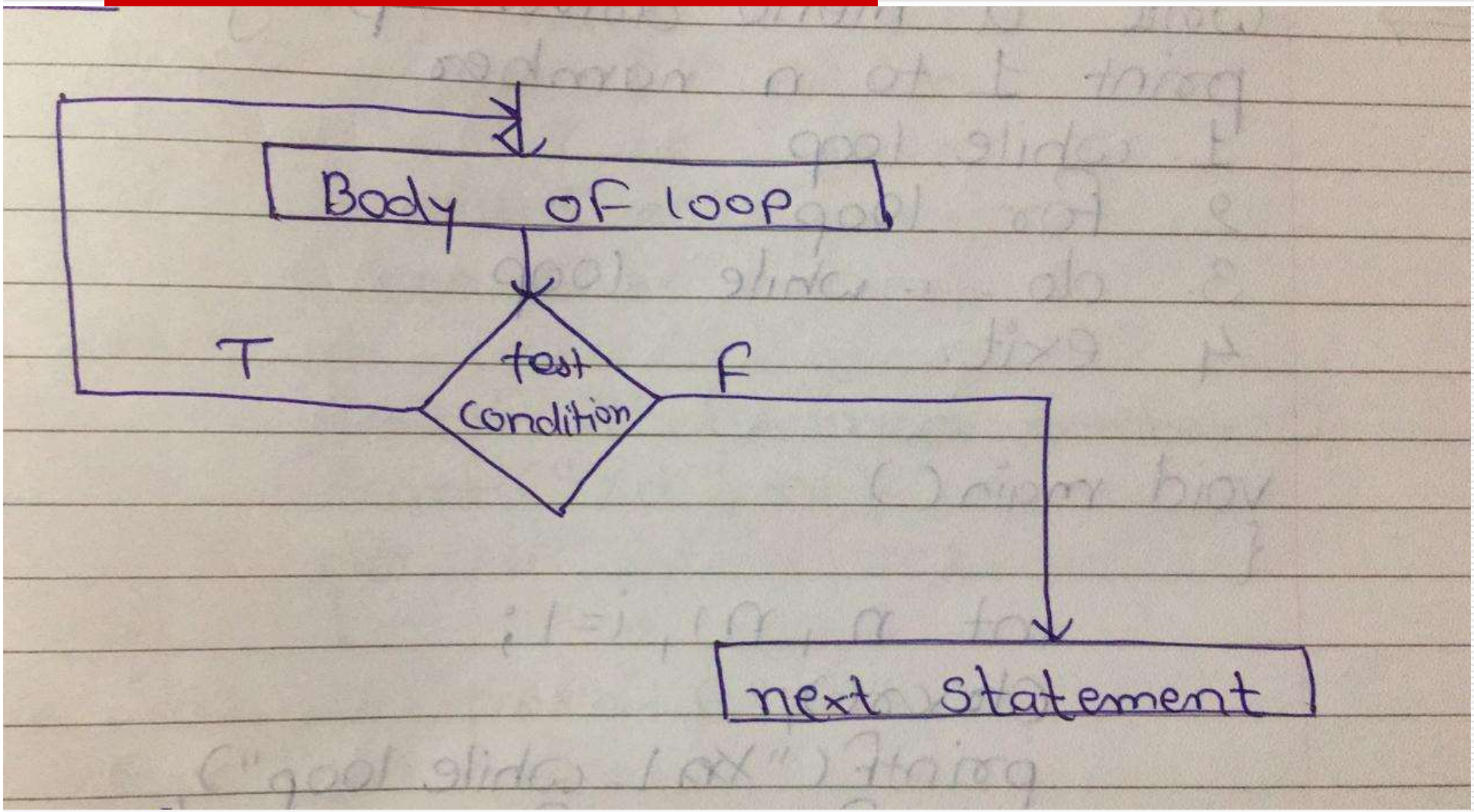
do..while loop

- The do..while loop is similar to the while loop with one important difference.
- The body of do...while loop is executed **at least once**. Only then, the test expression is evaluated.

Syntax:

```
do
{
    //body of loop
}while(Test-Condition);
```

Flowchart:



Example:

```
<SCRIPT>
        var n=10,i=1;
        do
        {
                document.write(i);
                i++;
        }while(i<=n);
    </SCRIPT>
```

OUTPUT:

12345678910

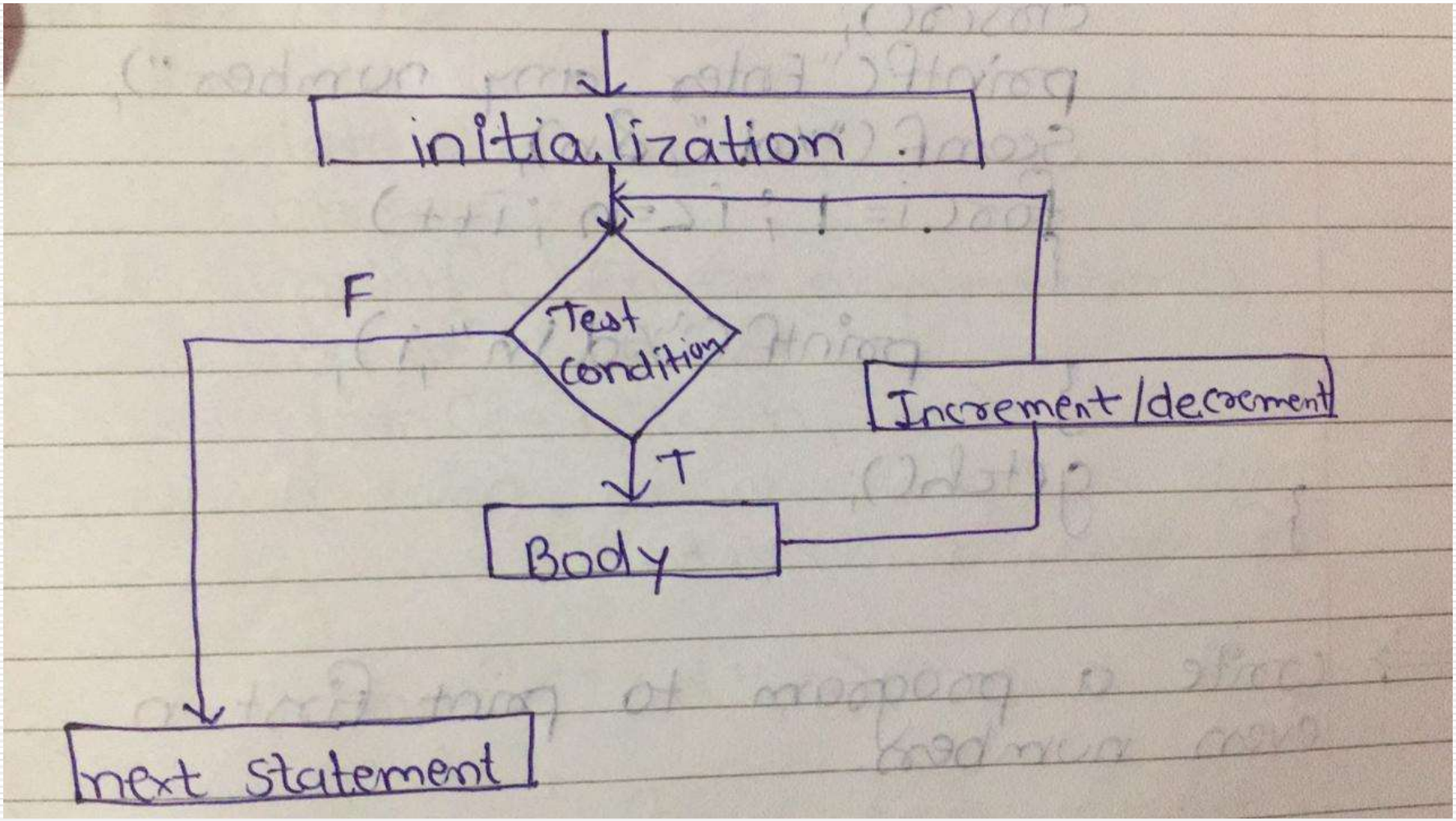
for loop

- A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for ( initialization; Test- Condition; Increment/Decrement )
{
   Body of Loop;
}
```

Flowchart:


initialization
Test condition
F
T
Body
Increment/decrement
next statement

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.

Example:

```
<SCRIPT>
                var n=10;
                for(var i=1;i<=n;i++)
                {
                        document.write(i);
                }
        </SCRIPT>
```

OUTPUT:

12345678910

Jumping

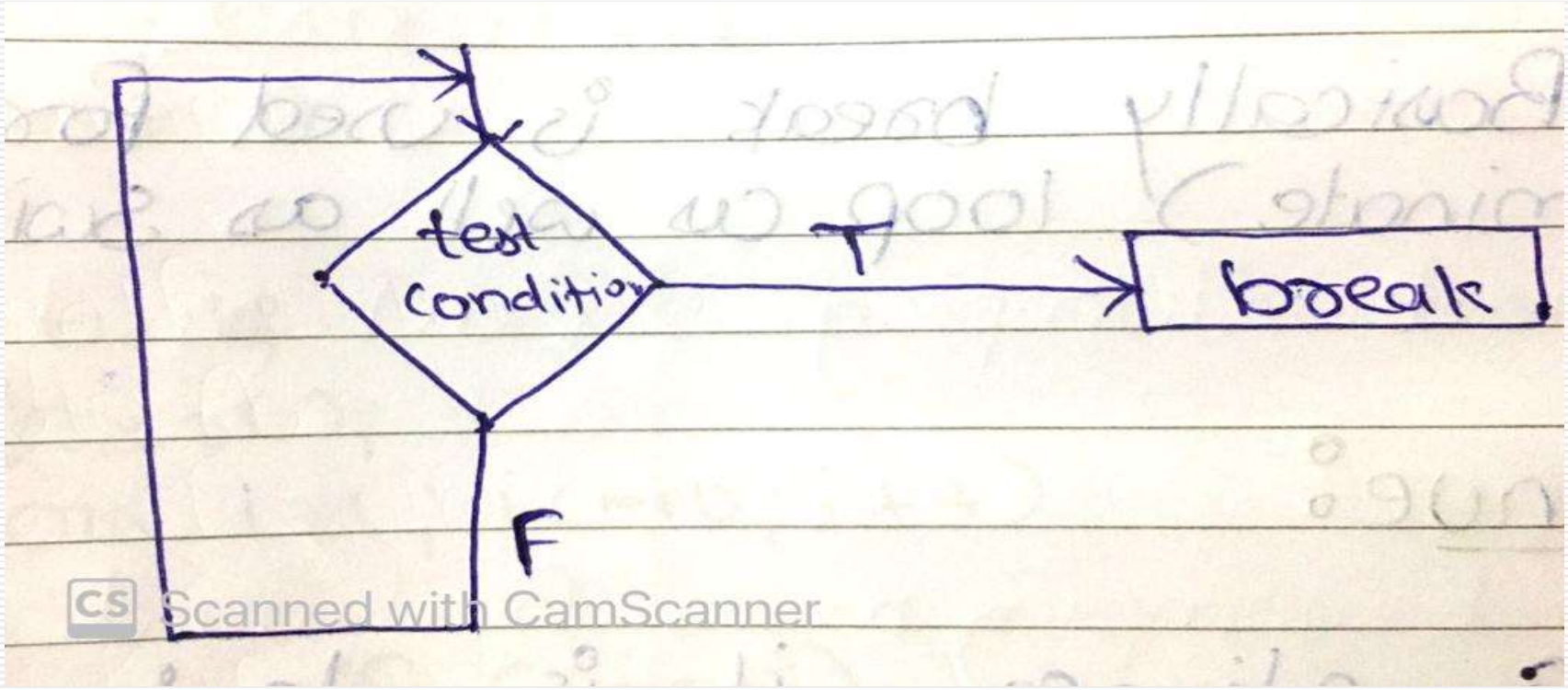
- break
- continue

Jumping statement: (break)

- When the break statement is executed inside a loop, the loop will be terminated and program control returns at the next statement of the loop.
- Syntax:
  - break;

- Note: Basically break is used for breaking(terminate) the loop as well as switch case.

Flowchart



Example

```
<SCRIPT>
                var n=10;
                for(var i=1;i<=n;i++)
                {
                        if(i==5)
                                break;
                        document.write(i);
                }
        </SCRIPT>
```

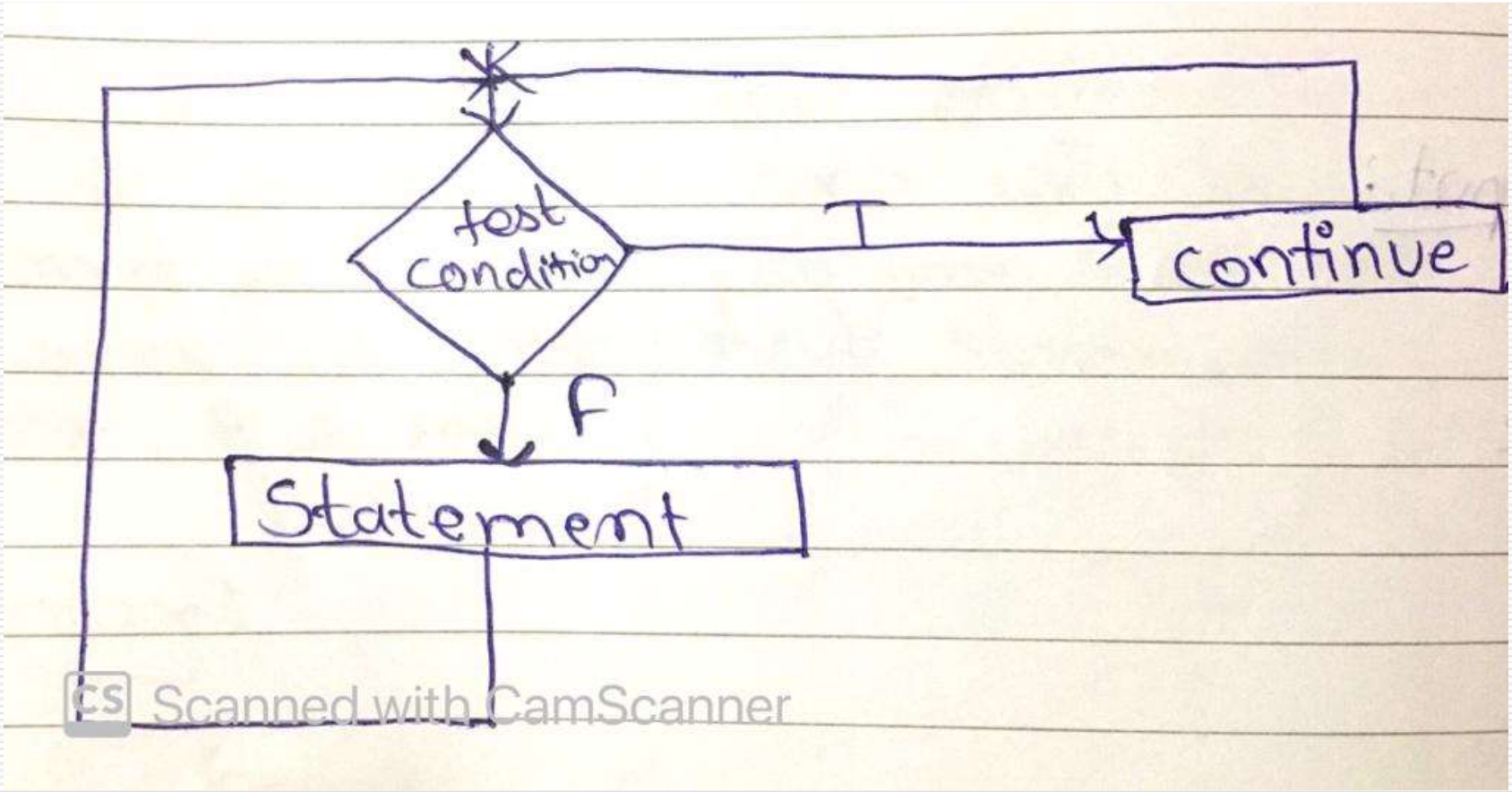
OUTPUT:

1234

Jumping statement: (continue)

- Sometimes it is desirable to skip some statement inside the loop, in such case **continue** statements are used.
- Continue forces the next iteration of the loop to take place and skipping any code in between.
- Syntax:
  - continue;

Flowchart


test condition
T
continue
F
Statement
Scanned with CamScanner

Example

```
<SCRIPT>
                    var n=10;
                    for(var i=1;i<=n;i++)
                    {
                              if(i==5)
                                        continue;
                              document.write(i);
                    }
          </SCRIPT>
```

OUTPUT:

1234678910

3.6 Javascript String and Events

3.6.1 Javascript String types

3.6.2 String Functions:

- concat()
- split()
- indexOf()
- lastIndexOf()
- substring()
- trim()
- slice()
- replace()
- charAt()

3.6.3 Javascript Events

- 3.6.3.1 Mouse Events
  - Click
  - mouseover
  - mouseremove
  - mouseout
  - mouseup
- 3.6.3.2 Keyboard Events
  - keyup
  - keydown
- 3.6.3.3 Form Events
  - Focus
  - Submit
  - Blur
  - change

3.6.1 Javascript String types

- The String object is used to represent and manipulate a sequence of characters.

3.6.2 String Functions:

- concat()
- split()
- indexOf()
- lastIndexOf()
- substring()
- trim()
- slice()
- replace()
- charAt()

concat()

- concat() joins two or more strings:
- The concat() method can be used instead of the plus operator.

Example:

```
<html>
        <head>
                <script>
                        var s="VNSGU";
                        var t="SURAT";
                        var st=s.concat(t);
                        document.write(st);
                </script>
        </head>
</html>
```

split()

- A string can be converted to an array with the split() method:

Example:

```
<html>
        <head>
                <script>
                        var s="very_good_morning";
                        var arr=s.split("_");
                        //document.write(arr);
                        document.write(arr[1]);
                </script>
        </head>
</html>
```

indexOf()

- The indexOf() method returns the position of the first occurrence of a specified value in a string.

- indexOf() returns -1 if the value is not found.

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning";

        document.write(s.indexOf("good"));
                </script>
        </head>
</html>
```

lastIndexOf()

- The indexOf() method returns the position of the first occurrence of a specified value in a string.

- indexOf() returns -1 if the value is not found.

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning good";

        document.write(s.lastIndexOf("good"));
                </script>
        </head>
</html>
```

substring()

- Extract characters from a string:
- the substring() method extracts characters, between to indices (positions), from a string, and returns the substring.

- The substring() method extracts characters between "start" and "end", not including "end".

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning good";
                        var sub=s.substring(5,9);
                        document.write(sub);
                </script>
        </head>
</html>
```

trim()

- The trim() method removes whitespace from both sides of a string.
- The trim() method does not change the original string.

Example:

```
<html>
        <head>
                <script>
                        var s="         very good morning          ";
                        document.write(s.trim());
                </script>
        </head>
</html>
```

slice()

- slice() extracts a part of a string and returns the extracted part in a new string.

- The method takes 2 parameters: the start position, and the end position (end not included).

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning";
                        var sub=s.slice(5,9);
                        document.write(sub);
                </script>
        </head>
</html>
```

replace()

- The replace() method searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced.

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning";
                        var
sub=s.replace("morning","afternoon");
                        document.write(sub);
                </script>
        </head>
</html>
```

charAt()

- The charAt() method returns the character at a specified index in a string.

- The index of the first character is 0, the second character is 1, and so on.

Example:

```
<html>
        <head>
                <script>
                        var s="very good morning";
                        document.write(s.charAt(3));
                </script>
        </head>
</html>
```

3.6.3 Javascript Events

- HTML events are "things" that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can "react" on these events.

- An HTML event can be something the browser does, or something a user does.
- Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked

3.6.3.1 Mouse Events

- onclick()
- ondbclick()
- onmousemove()
- onmouseover()
- onmouseup()
- onmousedown()

- **onclick()**
  - Fires on a mouse click on the element
- **ondbclick()**
  - Fires on a mouse double-click on the element
- **onmousemove()**
  - Fires when the mouse pointer is moving while it is over an element
- **onmouseover()**
  - Fires when the mouse pointer moves over an element
- **onmouseup()**
  - Fires when a mouse button is released over an element
- **onmousedown**()
  - Fires when a mouse button is pressed down on an element

3.6.3.2 Keyboard Events

- onkeydown()
- onkeyup()
- onkeypress()

- **onkeydown()**
  - Fires when a user is pressing a key
- **onkeyup()**
  - Fires when a user releases a key
- **onkeypress()**
  - Fires when a user presses a key

3.6.3.3 Form Events

- onsubmit()
- onreset()
- onblur()
- onfocus()
- onchange()
- onselect()

- **onsubmit()**
  - Fires when a form is submitted
- **onreset()**
  - Fires when the Reset button in a form is clicked
- **onblur()**
  - Fires the moment that the element loses focus
- **onfocus()**
  - Fires the moment when the element gets focus
- **onchange()**
  - Fires the moment when the value of the element is changed
- **onselect()**
  - Fires after some text has been selected in an elemenT