**Difference between TypeScript and JavaScript:**

* TypesScript is known as Object oriented programming language whereas JavaScript is a scripting language.
* TypeScript has a feature known as Static typing but JavaScript does not have this feature.
* TypeScript gives support for modules whereas JavaScript does not support modules.
* TypeScript has Interface but JavaScript does not have Interface.
* TypeScript support optional parameter function but JavaScript does not support optional parameter function.

**Advantages of using TypeScript over JavaScript**

* TypeScript always point out the compilation errors at the time of development only. Because of this at the run-time the chance of getting errors are very less whereas JavaScript is an interpreted language.
* TypeScript has a feature which is strongly-typed or supports static typing. That means Static typing allows for checking type correctness at compile time. This is not available in JavaScript.
* TypeScript is nothing but JavaScript and some additional features i.e. ES6 features. It may not be supported in your target browser but TypeScript compiler can compile the **.ts** files into ES3,ES4 and ES5 also

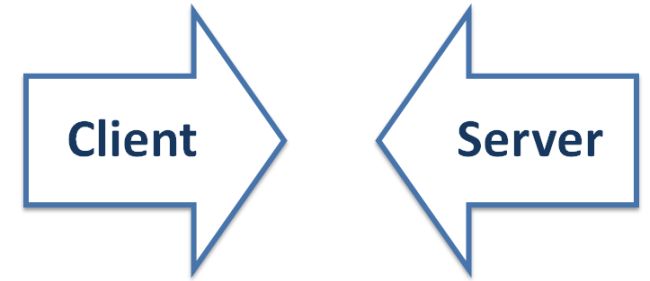
**Disadvantages of using TypeScript over JavaScript**

* Generally TypeScript takes time to compile the code.

Basic Background

Web development is all about communication and data exchange. This communication takes place via two parties over the HTTP protocol.

These parties are:



Server

The Server is responsible for serving the web pages depending on the client/end-user requirement. It can be either static or dynamic.

Client

A client is a party that requests pages from the server and displays them to the end-user. In general a client program is a web browser.

**Example | Working**

We can explain this entire mechanism using the following:

* The user opens his web browser (client)
* The user starts browsing

(for example [http://c-sharpcorner.com](http://c-sharpcorner.com/))

* The client forwards this request to the server, for accessing their web page.
* The server then acknowledges the request and replies back to the client program.

(An access link to that web page)

* The client then receives the page source and renders it.

(Into a viewable/under a stable website)

* Now the user types into the search bar
* The client then submits data to the server
* The server processes the data and replies back with a related search result
* The client again renders it back for the user's view
* The user gets access to the requested link.

Server-side Programming

Server-side programming can be explained as:

It is the general name for the kind of program that runs directly on the server.

Or we can say that server-side programming must deal with dynamic content. It runs on the server. Most web pages are not static since they deal with searching databases.

Server-side Uses

* It processes the user input
* Displays the requested pages
* Structure of web applications
* Interaction with servers/storages
* Interaction with databases
* Querying the database
* Encoding of data into HTML
* Operations over databases like delete, update.

Server-side Languages Example

There are several languages that can be used for server-side programming:

* PHP
* ASP.NET (C# OR Visual Basic)
* C++
* Java and JSP
* Python
* Ruby on Rails and so on.

Client-side Programming

client-side programming is also the name of the entire program that runs on the client.

Or we can say that client-side programming mostly deals with the user interface with which the user interacts in the web. It is mostly a browser, in the user's machine, that runs the code and is mainly done in any scripting language like JavaScript (or we can use Flash instead of JavaScript or VNScript).

Client-side Uses

* Makes interactive web pages
* Make stuff work dynamically
* Interact with temporary storage
* Works as an interface between user and server
* Sends requests to the server
* Retrieval of data from Server
* Interact with local storage
* Provides remote access for client-server program

Client-side Languages Example

There are many client-side scripting languages too.

* JavaScript
* VBScript
* HTML (Structure)
* CSS (Designing)
* AJAX
* jQuery etc.

####The **difference between** these two approaches is as follows: **Synchronous** way: It waits for each operation to complete, after that only it executes the next operation. ... **Asynchronous** way: It never waits for each operation to complete, rather it executes all operations **in the** first GO only.

####**dynamically**-**typed languages** perform **type** checking at runtime, while statically **typed languages** perform **type** checking at compile time

####**TRANSPILER**- It's a source-to-source compiler, it translates source code from one language to another (or to another version of the same language).  
For example GWT has a transpiler translating parts of your Java code into JavaScript.  
There's also Babel that converts JavaScript code in ES6 into ES5 so that it runs in environment where ES6 doesn't work properly.

### DEFER TAG

### Example

A script that will not run until after the page has loaded:

<script src="demo\_defer.js" defer></script>

## Definition and Usage

The defer attribute is a boolean attribute.

When present, it specifies that the script is executed when the page has finished parsing.

**Note:** The defer attribute is only for external scripts (should only be used if the src attribute is present).

####DOCUMENT.WRITE()

Write HTML elements with text directly to the HTML document:

<!DOCTYPE html>

<tml>

<body>

<script>

document.write("<h1>Hello World!</h1><p>Have a nice day!</p>");

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<script src="script.js" defer></script>

</head>

<body>

<h1>HIIIIIIIIIIIIIIIIIII</h1>

<button type="button" onclick="myFunction()">Click Me</button>

<br />

<script>

document.write("In HTML")

</script>

</body>

</html>

document.write("<h1>Hello World!</h1><p>Have a nice day!</p>");

document.write(Date());

function myFunction() {

console.log("CONSOLE")

document.write("In JS")

document.write(Date());

document.open();

document.write("<h1>Hello World</h1>");

document.close();

}

####Open a new window called "MsgWindow", and write some text into it:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<script src="script.js" defer></script>

</head>

<body>

<p>Click the button to open a new window called "MsgWindow" with some text.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

var myWindow = window.open("", "MsgWindow", "width=200,height=100");

myWindow.document.write("<p>This is 'MsgWindow'. I am 200px wide and 100px tall!</p>");

}

</script>

</body>

</html>

Use of write and writeln

<!DOCTYPE html>

<html>

<body>

<p>Note that write() does NOT add a new line after each statement:</p>

<pre>

<script>

document.write("Hello World!");

document.write("Have a nice day!");

</script>

</pre>

<p>Note that writeln() add a new line after each statement:</p>

<pre>

<script>

document.writeln("Hello World!");

document.writeln("Have a nice day!");

</script>

</pre>

</body>

</html>

The you don’t want to use the pre tag then the solution is

<script>

document.write("Hello World! <br>");

document.write("Have a nice day!");

</script

Get the first element in the document with class="example":

## Definition and Usage

The querySelector() method returns the first element that matches a specified CSS selector(s) in the document.

**Note:** The querySelector() method only returns the first element that matches the specified selectors. To return all the matches, use the [querySelectorAll()](https://www.w3schools.com/jsref/met_document_queryselectorall.asp) method instead.

If the selector matches an ID in document that is used several times (Note that an "id" should be unique within a page and should not be used more than once), it returns the first matching element.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

<script src="script.js" defer></script>

</head>

<body>

<h2 class="example">A heading with class="example"</h2>

<p class="example">A paragraph with class="example".</p>

<p>Click the button to add a background color to the first element in the document with class="example".</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.querySelector(".example").style.backgroundColor = "red";

}

</script>

</body>

</html>

The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object.

document.getElementById

<!DOCTYPE html>

<html>

<body>

<p id="demo">Click the button to change the text in this paragraph.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Hello World";

}

</script>

</body>

</html>

## Definition and Usage

The getElementById() method returns the element that has the ID attribute with the specified value.

This method is one of the most common methods in the HTML DOM, and is used almost every time you want to manipulate, or get info from, an element on your document.

Returns null if no elements with the specified ID exists.

An ID should be unique within a page. However, if more than one element with the specified ID exists, the getElementById() method returns the first element in the source code.

.INNERHTML

<!DOCTYPE html>

<html>

<body>

<p id="demo" onclick="myFunction()">Click me to change my HTML content (innerHTML).</p>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed!";

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<p id="myP">I am a paragraph.</p>

<p>Click the button get the HTML content of the p element.</p>

<button onclick="myFunction()">Try it</button>

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

<script>

function myFunction() {

var x = document.getElementById("myP").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<ul id="myList">

<li>Coffee</li>

<li>Tea</li>

</ul>

<p>Click the button get the HTML content of the ul element.</p>

<button onclick="myFunction()">Try it</button>

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

<script>

function myFunction() {

var x = document.getElementById("myList").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h1>My Web Page</h1>

<p id="myP">This is a p element.</p>

<div id="myDIV">This is a div element.</div>

<button onclick="fun1()">click</button>

<script>

function fun1(){

document.getElementById("myP").innerHTML = "Hello Dolly.";

document.getElementById("myDIV").innerHTML = "How are you?";

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<p id="demo">Click the button to alert the text of this paragraph.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

alert(document.getElementById("demo").innerHTML);

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<p id="demo">Click the button to delete my HTML content (innerHTML).</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "";

}

</script>

</body>

</html>

Change the HTML content, URL, and target of a link:

<!DOCTYPE html>

<html>

<body>

<a id="myAnchor" href="https://www.google.com/" target="\_blank">Google</a>

<button onclick="myFunction()">Change link</button>

<script>

function myFunction() {

document.getElementById("myAnchor").innerHTML = "W3Schools";

document.getElementById("myAnchor").href = "https://www.w3schools.com";

document.getElementById("myAnchor").target = "\_blank";

}

</script>

</body>

</html>