**Week 4**

**HTML**

* HTML stands for Hyper Text Markup Language
* HTML is the standard markup language for Web pages
* HTML **elements** are the building blocks of HTML pages
* HTML elements are represented by **<> tags**

**CSS**

* CSS stands for Cascading Style Sheets
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media
* CSS saves a lot of work. It can control the layout of multiple web pages all at once
* External style sheets are stored in CSS files

# CSS Syntax



.

<html><head>

<style>

body {

background-color: lightblue;

}

h1 {

color: white;

text-align: center;

}

p {

font-family: verdana;

font-size: 20px;

}

</style>

</head>

<body>

<h1>My First CSS Example</h1>

<p>This is a paragraph.</p>

</body>

</html>

# CSS Selectors

A CSS selector selects the HTML element(s) you want to style.

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)

## The CSS element Selector

The element selector selects HTML elements based on the element name.

<html><head>

<style>

p {

text-align: center;

color: red;

}

</style></head>

<body>

<p>Every paragraph will be affected by the style.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body></html>

## The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

<html><head>

<style>

#para1 {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body></html>

## The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

<html><head><style>

.center {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1 class="center">Red and center-aligned heading</h1>

<p class="center">Red and center-aligned paragraph.</p>

</body></html>

## The CSS Universal Selector

The universal selector (\*) selects all HTML elements on the page.

<!DOCTYPE html>

<html><head>

<style>

\* {

text-align: center;

color: blue;

}

</style></head>

<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body></html>

## The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

<html>

<head>

<style>

h1, h2, p {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<h2>Smaller heading!</h2>

<p>This is a paragraph.</p>

</body>

</html>

**Three Ways to Insert CSS**

There are three ways of inserting a style sheet:

* External CSS
* Internal CSS
* Inline CSS

**External CSS**

With an external style sheet, you can change the look of an entire website by changing just one file

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

<html>

<head>

<link href="mystyle.css">

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

**mystyle.css**

body {  
  background-color: lightblue;  
}  
h1 {  
  color: navy;  
  margin-left: 20px;  
}

## Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

<html>

<head>

<style>

body {

background-color: linen;

}

h1 {

color: maroon;

margin-left: 40px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

## Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

<html><body>

<h1 style="text-align:center;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>

</body>

</html>

**What is JavaScript?**

JavaScript is the world's most popular programming language.

JavaScript is the programming language of the Web.

JavaScript is easy to learn.

JavaScript programs can be inserted almost anywhere into an HTML document using the <script> tag.

<html>

<body>

<p>Before the script...</p>

<script>

alert( 'Hello, world!' );

</script>

<p>...After the script.</p>

</body>

</html>

**JavaScript Can Change HTML Content**

One of many JavaScript HTML methods is getElementById().

<html>

<body>

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

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>

</html>

**JavaScript Statements**

* In a programming language, the list of programming instructions are called **statements**.
* **A JavaScript program** is a list of programming **statements.**
* JavaScript statements are composed of: Values, Operators, Expressions, Keywords, and Comments.

**Comments**

Single line comments start with //

Multi-line comments start with /\* and end with \*/

**Variables**

Variables are containers for storing data (storing data values).

In this example, x, y, and z, are variables, declared with the var keyword:

var x = 5;

var y = 6;

### Declaring a JavaScript Variable

* Using var
* Using let
* Using const

**The <script> Tag**

In HTML, JavaScript code is inserted between n <script> and </script> tags.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

<html>

<body>

<script>

var x = 5;

var y = 6;

var z = x + y;

document.write(z)

</script>

</body>

</html>

## [alert](https://javascript.info/alert-prompt-confirm" \l "alert)

## This one we’ve seen already. It shows a message and waits for the user to press “OK”

<script>

alert('HI there'); // with specified content

alert(); // without any specified content

</script>

**Types of JavaScript Operators**

There are different types of JavaScript operators:

* Arithmetic Operators
* Assignment Operators
* Comparison Operators
* Logical Operators
* Conditional Operators
* Type Operators

Comparisons

| **Operator** | **Meaning** |
| --- | --- |
| < | less than |
| > | greater than |
| <= | less than or equal to |
| >= | greater than or equal to |
| == | equal to |
| != | not equal to |

**Functions**

A JavaScript function is a block of code designed to perform a particular task.

<html>

<body>

<script>

function msg()

{

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="clickme"/>

</body>

</html>

<html>

<body>

<script>

function add()

{

Let x=4,y=8;

Let z=x+y;

Document.write(z);

}

</script>

<form>

<input type="button" value="click" onclick="add()"/>

</form>

</body>

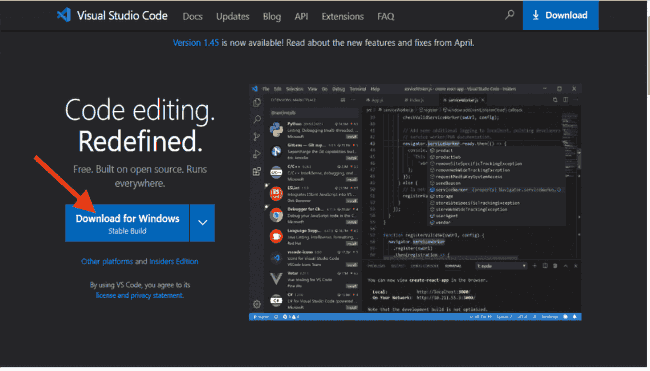
</html>

**Setting Up the Environment and Tools for front end development**

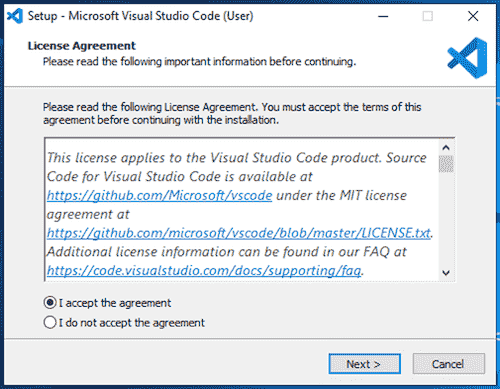
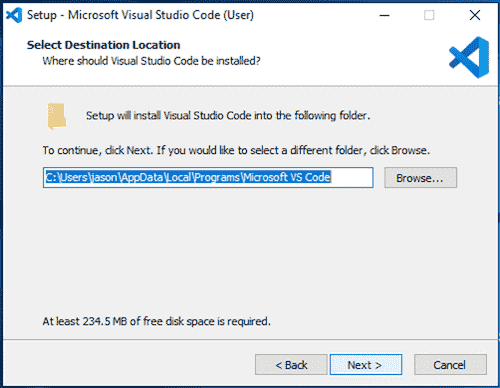
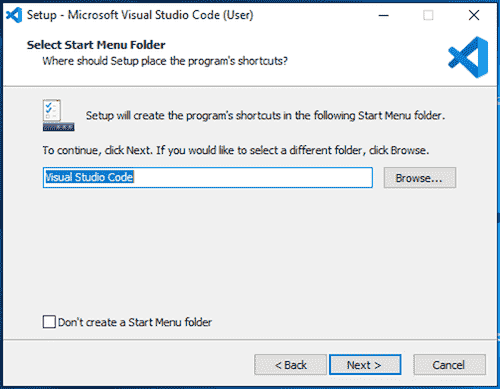
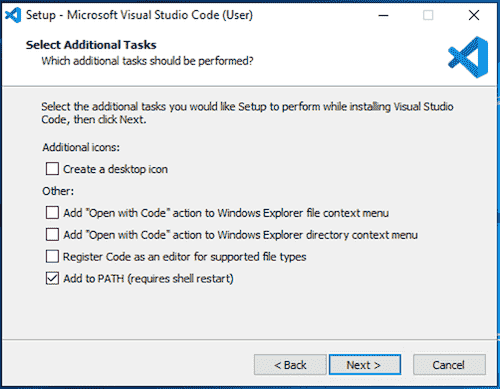
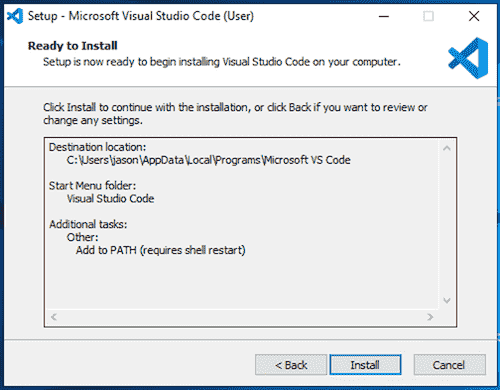
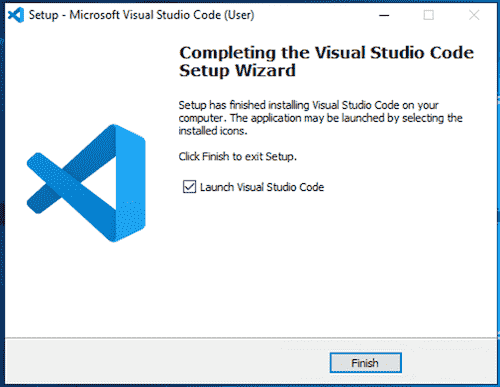
**Installing VS Code**

VS Code is a free code editor that runs on Windows, Mac and Linux.

Download VS Code from <https://code.visualstudio.com/>.



Install Visual Studio Code by opening the downloaded setup file and following the prompts.

**What is meant by TypeScript?**

* TypeScript is a syntactic superset of JavaScript which adds static typing.

**Why TypeScript?**

* TypeScript uses compile time type checking. Which means it checks if the specified types match **before** running the code, not **while** running the code
* JavaScript is better suited for small-scale applications, while TypeScript is better for larger applications.
* TypeScript is better than JavaScript in terms of language features, reference validation, project scalability, collaboration within and between teams, developer experience, and code maintainability.

### Setting up development environment for TypeScript

### Pre-requisite to install TypeScript

1. Text Editor or IDE
2. Node.js Package Manager (npm)
3. The TypeScript compiler

### Ways to install TypeScript

There are two ways to install TypeScript:

1. Install TypeScript using Node.js Package Manager (npm).
2. Install the TypeScript plug-in in your IDE (Integrated Development Environment).

### Install TypeScript using Node.js Package Manager (npm)

**Step-1** Install Node.js. It is used to setup TypeScript on our local computer.

To install Node.js on Windows, go to the following link: [**https://www.javatpoint.com/install-nodejs**](https://www.javatpoint.com/install-nodejs)

To verify the installation was successful, enter the following command in the Terminal Window.

node -v

npm -v

**Step-2** Install TypeScript. To install TypeScript, enter the following command in the Terminal Window.

* npm install typescript --save-dev         //As dev dependency
* npm install typescript -g                      //Install as a global module

or

* npm install -g typescript
* npm install typescript@latest -g          //Install latest if you have an older version

**Step-3** To verify the installation was successful, enter the command **$ tsc -v** in the Terminal Window.

**Install Live server**

npm install -g live-server

**Create and run first program in TypeScript**

* open visual studio code
* file-open folder-choose typescript folder
* create new file- save it as types.ts(any name.ts)
* Write the below code and save it
* console.log("Hello World");
* go to command prompt and compile the program
* tsc types.ts
* run the program
* node types.js
* Observe the output

**Basic Types**

| Built-in Data Type | keyword | Description |
| --- | --- | --- |
| Number | number | It is used to represent both Integer as well as Floating-Point numbers |
| Boolean | boolean | Represents true and false |
| String | string | It is used to represent a sequence of characters |
| Void | void | Generally used on function return-types |
| Null | null | It is used when an object does not have any value |
| Undefined | undefined | Denotes value given to uninitialized variable |
| Any | any | If variable is declared with any data-type then any type of value can be assigned to that variable |

**Example**

**let a: null = null;**

**let b: number = 123;**

**let c: number = 123.456;**

**let d: string = ‘Geeks’;**

**let e: undefined = undefined;**

**let f: boolean = true;**

console.log(a);

console.log(b);

console.log(c);

console.log(d);

console.log(e);

console.log(f);

**let a: any = null;**

**let b: any =123;**

**let c: any = 123.456;**

**let d: any = ‘Geeks’;**

**let e: any = undefined;**

**let f: any = true;**

**Control flow statement**

TypeScript control statements:

1. If Statement  
2. If else statement  
3. if else if statement

**TypeScript If Statement:**

If statement is used to execute a block of statements if specified condition is true.

if(condition){

*//Block of TypeScript statements.*

}

**TypeScript If Else Statement:**

If else statement is used to execute either of two block of statements depends upon the condition. If condition is true then if block will execute otherwise else block will execute.

if(condition){

*//Block of TypeScript statements1.*

}else{

*//Block of TypeScript statements2.*

}

**TypeScript If Else If Statement:**

If else statement is used to execute one block of statements from many depends upon the condition. If condition1 is true then block of statements1 will be executed, else if condition2 is true block of statements2 is executed and so on. If no condition is true, then else block of statements will be executed.

if(condition1){

*//Block of TypeScript statements1.*

}else if(condition2){

*//Block of TypeScript statements2.*

} . . . else if(conditionn){

*//Block of TypeScript statementsn.*

}else{

*//Block of TypeScript statements.*

}

**Example**

var num:number = 2;

if(num==1){

console.log("TypeScript Statement 1");

}

else if(num==2){

console.log("TypeScript Statement 2");

}

else if(num==3){

console.log("TypeScript Statement 3");

}

else{

console.log("TypeScript Statement n");

}

# TypeScript - for Loops

1. for loop
2. for..of loop
3. for..in loop

## for Loop

The for loop is used to execute a block of code a given number of times, which is specified by a condition.

for (first expression; second expression; third expression ) {

// statements to be executed repeatedly

}

for (let i = 0; i < 3; i++) {

console.log ("Block statement execution no." + i);

}

**for...of Loop**

TypeScript includes the for...of loop to iterate and access elements of an array, list, or tuple collection.

let arr = [10, 20, 30, 40];

for (var val of arr) {

console.log(val); // prints values: 10, 20, 30, 40

}

**for...in Loop**

Another form of the for loop is for...in. This can be used with an array, list, or tuple.

let arr = [10, 20, 30, 40];

for (var index in arr) {

console.log(index); // prints indexes: 0, 1, 2, 3

console.log(arr[index]); // prints elements: 10, 20, 30, 40

}

# TypeScript - while Loop

The while loop is another type of loop that checks for a specified condition before beginning to execute the block of statements. The loop runs until the condition value is met.

while (condition expression) {

// code block to be executed

}

let i: number = 2;

while (i < 4) {

console.log( "Block statement execution no." + i )

i++;

}

**do..while loop**

The do..while loop is similar to the while loop, except that the condition is given at the end of the loop. The do..while loop runs the block of code at least once before checking for the specified condition.

do {

// code block to be executed

}

while (condition expression);

let i: number = 2;

do {

console.log("Block statement execution no." + i )

i++;

} while ( i < 4)

**Functions**

In TypeScript, functions can be of two types: named and anonymous.

**Named Functions**

A named function is one where you declare and call a function by its given name.

function display() {

console.log("Hello TypeScript!");

}

display(); //Output: Hello TypeScript