120A3051
Shreya Idate
Batch: E3

EXPERIMENT 3

Aim: Write Type Script program -Hello World and Calculator.

Theory:

TypeScript

TypeScript is an open-source, object-oriented programing language, which is developed and maintained by Microsoft under the Apache 2 license. It was introduced by Anders Hejlsberg, a core member of the development team of C# language.

TypeScript is a strongly typed superset of JavaScript which compiles to plain JavaScript. It is a language for application-scale JavaScript development, which can be executed on any browser, any Host, and any Operating System.

TypeScript is not directly run on the browser. It needs a compiler to compile and generate in JavaScript file. It contains all elements of the JavaScript.

Need of TypeScript

- TypeScript supports Static typing, Strongly type, Modules, Optional Parameters, etc.
- TypeScript supports object-oriented programming features such as classes, interfaces, inheritance, generics, etc.
- TypeScript is fast, simple, and most importantly, easy to learn.
- TypeScript provides the error-checking feature at compilation time. It will compile the code, and if any error found, then it highlighted the mistakes before the script is run.
- TypeScript supports all JavaScript libraries because it is the superset of JavaScript.
- TypeScript support reusability because of the inheritance.
- TypeScript gives all the benefits of ES6 plus more productivity.

<u>TypeScript Built-in Types</u>

Data type	Keyword	Description
Number	number	Double precision 64-bit floating point values. It can be used to represent both, integers and fractions.
String	string	Represents a sequence of Unicode characters
Boolean	boolean	Represents logical values, true and false
Void	void	Used on function return types to represent non-returning functions
Null	null	Represents an intentional absence of an object value.
Undefined	undefined	Denotes value given to all uninitialized variables
Data type	Keyword	Description

<u>Typescript arrays:</u>
An array is a user-defined data type. An array is a homogeneous collection of similar types of elements that have a contiguous memory location and which can store multiple values of different data types.
An array is a type of data structure that stores the elements of similar data type and consider it as an object too. We can store only a fixed set of elements and can’t expand its size, once its size is declared.

Typescript array methods:

1.	concat() Returns a new array comprised of this array joined with other array(s) and/or value(s).
2.	every() Returns true if every element in this array satisfies the provided testing function.
3.	map() Creates a new array with the results of calling a provided function on every element in this array.
4.	pop() Removes the last element from an array and returns that element.
5.	push() Adds one or more elements to the end of an array and returns the new length of the array.
6.	reduce() Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
7.	reverse() Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
8.	slice() Extracts a section of an array and returns a new array.
9.	sort() Sorts the elements of an array.

Typescript Tuples:
TypeScript introduced a new data type called Tuple. Tuple can contain two values of different data types.
A tuple has two operations:

- Push()
- Pop()

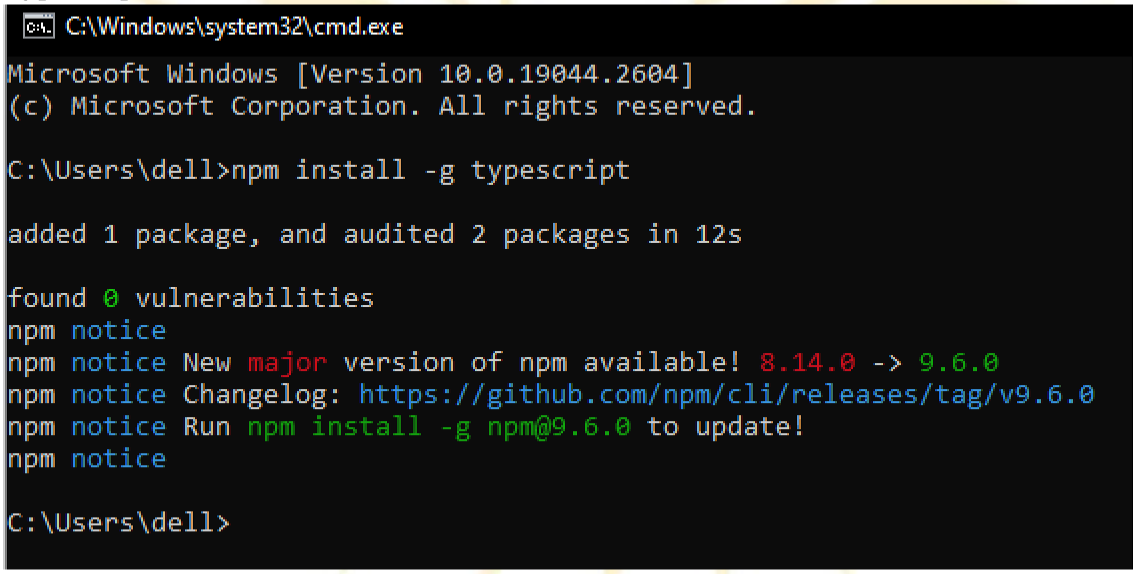
Push()
The push operation is used to add an element to the tuple.
Pop()
The pop operation is used to remove an element from the tuple

Typescript environment setup:
Installation on Windows:
Step 1 − Download and run the .msi installer for Node.
Step 2 − To verify if the installation was successful, enter the command node –v in the terminal window.
Step 3 − Type the following command in the terminal window to install TypeScript. npm install -g typescript The development environment used here is Visual Studio Code (Windows platform). VScode is available at − https://code.visualstudio.com/

The TypeScript Compiler
The TypeScript compiler is itself a .ts file compiled down to JavaScript (.js) file. The TSC (TypeScript Compiler) is a source-to-source compiler (transcompiler / transpiler).

Output:

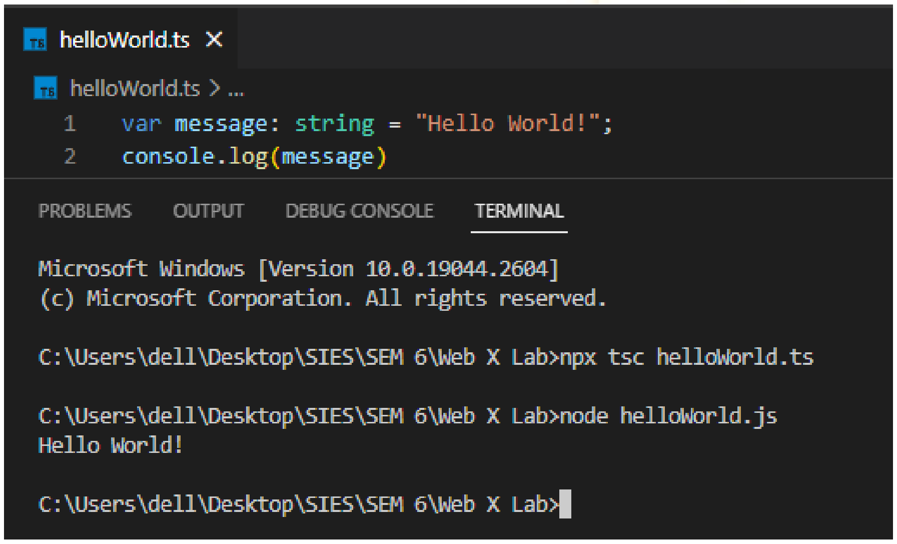
TypeScript installation



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell>npm install -g typescript

added 1 package, and audited 2 packages in 12s

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.14.0 -> 9.6.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.0
npm notice Run npm install -g npm@9.6.0 to update!
npm notice

C:\Users\dell>
```

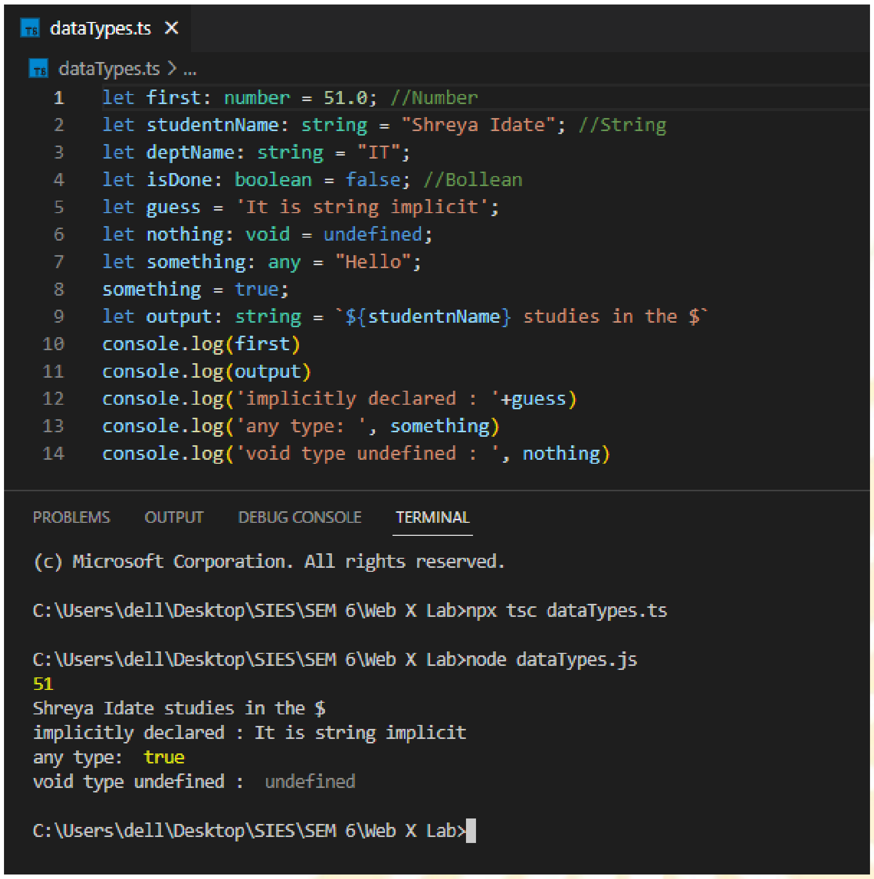
Hello World



```
var message: string = "Hello World!";
console.log(message)
```

```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>npx tsc helloWorld.ts

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>node helloWorld.js
Hello World!

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>
```

DataTypes:

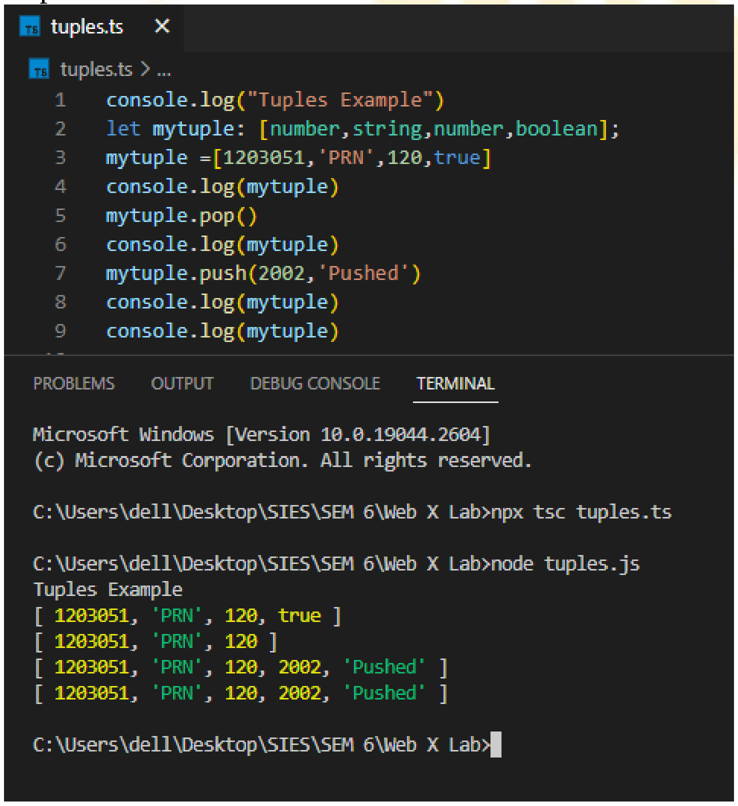


dataTypes.ts

```
let first: number = 51.0; //Number
let studentnName: string = "Shreya Idate"; //String
let deptName: string = "IT";
let isDone: boolean = false; //Bollean
let guess = 'It is string implicit';
let nothing: void = undefined;
let something: any = "Hello";
something = true;
let output: string = `${studentnName} studies in the $`
console.log(first)
console.log(output)
console.log('implicitly declared : '+guess)
console.log('any type: ', something)
console.log('void type undefined : ', nothing)
```

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>npx tsc dataTypes.ts

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>node dataTypes.js
51
Shreya Idate studies in the $
implicitly declared : It is string implicit
any type:  true
void type undefined :  undefined

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>
```

Tuples:

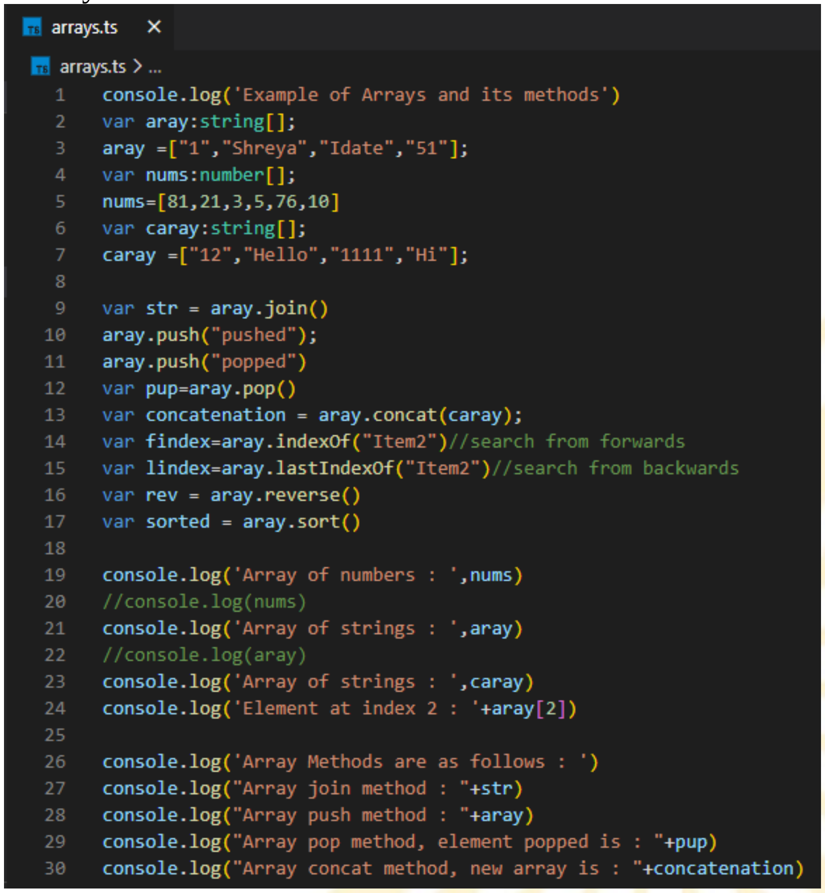


tuples.ts

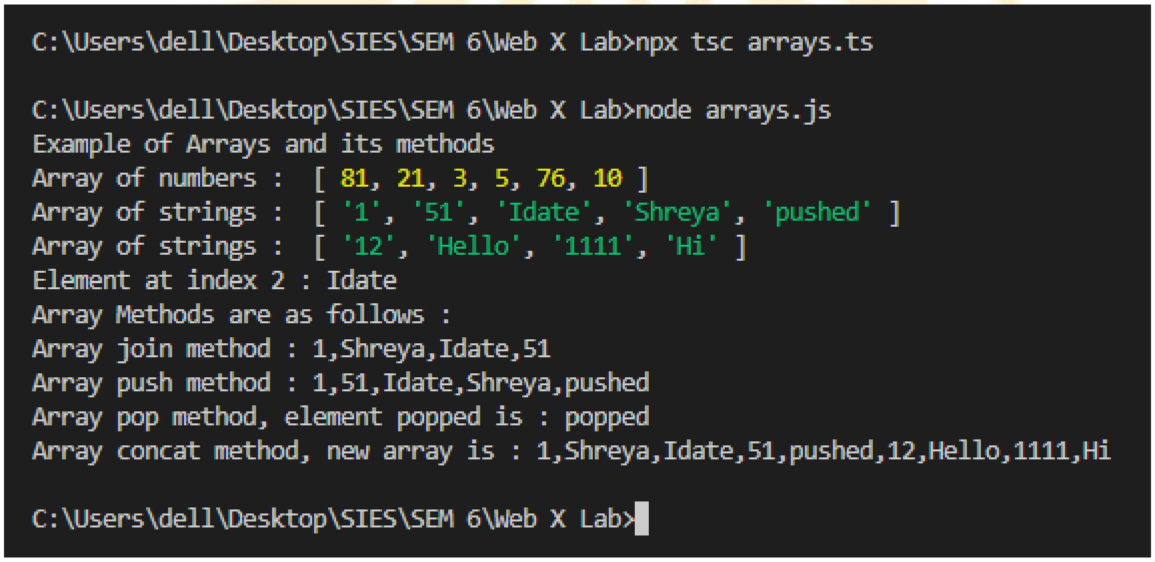
```
console.log("Tuples Example")
let mytuple: [number,string,number,boolean];
mytuple =[1203051,'PRN',120,true]
console.log(mytuple)
mytuple.pop()
console.log(mytuple)
mytuple.push(2002,'Pushed')
console.log(mytuple)
console.log(mytuple)
```

```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>npx tsc tuples.ts

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>node tuples.js
Tuples Example
[ 1203051, 'PRN', 120, true ]
[ 1203051, 'PRN', 120 ]
[ 1203051, 'PRN', 120, 2002, 'Pushed' ]
[ 1203051, 'PRN', 120, 2002, 'Pushed' ]

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>
```

Arrays and Methods:



```
arrays.ts ×
arrays.ts > ...
console.log('Example of Arrays and its methods')
var aray:string[];
aray =["1","Shreya","Idate","51"];
var nums:number[];
nums=[81,21,3,5,76,10]
var caray:string[];
caray =["12","Hello","1111","Hi"];

var str = aray.join()
aray.push("pushed");
aray.push("popped")
var pup=aray.pop()
var concatenation = aray.concat(caray);
var findex=aray.indexOf("Item2")//search from forwards
var lindex=aray.lastIndexOf("Item2")//search from backwards
var rev = aray.reverse()
var sorted = aray.sort()

console.log('Array of numbers : ',nums)
//console.log(nums)
console.log('Array of strings : ',aray)
//console.log(aray)
console.log('Array of strings : ',caray)
console.log('Element at index 2 : '+aray[2])

console.log('Array Methods are as follows : ')
console.log("Array join method : "+str)
console.log("Array push method : "+aray)
console.log("Array pop method, element popped is : "+pup)
console.log("Array concat method, new array is : "+concatenation)
```



```
C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>npx tsc arrays.ts

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>node arrays.js
Example of Arrays and its methods
Array of numbers :  [ 81, 21, 3, 5, 76, 10 ]
Array of strings :  [ '1', '51', 'Idate', 'Shreya', 'pushed' ]
Array of strings :  [ '12', 'Hello', '1111', 'Hi' ]
Element at index 2 : Idate
Array Methods are as follows :
Array join method : 1,Shreya,Idate,51
Array push method : 1,51,Idate,Shreya,pushed
Array pop method, element popped is : popped
Array concat method, new array is : 1,Shreya,Idate,51,pushed,12,Hello,1111,Hi

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab>
```

Calculator

HTML:

```
<!DOCTYPE html>
<html lang="zh">
  <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>Typescript Calculator</title>
    <link rel="stylesheet" href="./calculator.css" />
  </head>
  <body>
    <script src="./calculator.js"></script>
    <script>
      const root = document.body;
      const calculator = new Calculator(root);
      calculator.create();
    </script>
  </body>
</html>
```

CSS File

```
html,
body {
  margin: 0;
  padding: 0;
}

body {
  width: 100vw;
  height: 100vh;
  display: flex;
  user-select: none;
  align-items: center;
  justify-content: center;
  background-color: #eee;
}

.calculator {
  width: 232px;
  height: 320px;
  overflow: hidden;
  border-radius: 4px;
  border: 0.5px solid black;
  box-shadow: 0px 10px 30px 0px rgba(125, 125, 125, 0.8);
}
```

```
.decorator {
  height: 20px;
  display: flex;
  align-items: center;
  background-color: rgb(88, 84, 85);
}

.dot {
  width: 12px;
  height: 12px;
  margin-top: 3px;
  margin-left: 28px;
  position: relative;
  border-radius: 50%;
  background-color: rgb(255, 190, 46);
}

.dot::after,
.dot::before {
  top: 0;
  width: 12px;
  height: 12px;
  content: '';
  display: block;
  position: absolute;
  border-radius: 50%;
}

.dot::before {
  right: calc(11px + 8px);
  background-color: rgb(255, 97, 89);
}

.dot::after {
  left: calc(11px + 8px);
  background-color: rgb(84, 194, 43);
}

.display {
  height: 60px;
  display: flex;
  align-items: center;
  justify-content: flex-end;
  background-color: rgb(88, 84, 85);
}

.result {
  color: white;
  height: 40px;
  font-size: 50px;
  font-weight: 200;
  line-height: 40px;
  margin-right: 15px;
}

.row {
  display: flex;
  height: 48px;
  color: white;
  background-color: rgb(131, 127, 127);
}

.button {
  display: flex;
  width: 57px;
  font-size: 23px;
  font-weight: 300;
  align-items: center;
  justify-content: center;
  border-right: 1px solid rgb(94, 89, 89);
  border-bottom: 1px solid rgb(94, 89, 89);
}

.button:active {
  background-color: rgb(166, 163, 163);
}

.row:nth-child(3) .button {
  border-top: none;
}

.row:last-child .button {
  border-bottom: none;
}

.row .button:last-child {
  border-right: none;
  background-color: rgb(255, 159, 8);
}

.row .button:last-child:active {
  background-color: rgb(203, 125, 5);
}

.row:nth-child(3) .button:first-child {
  width: 173px;
  background: rgb(106, 102, 102);
```

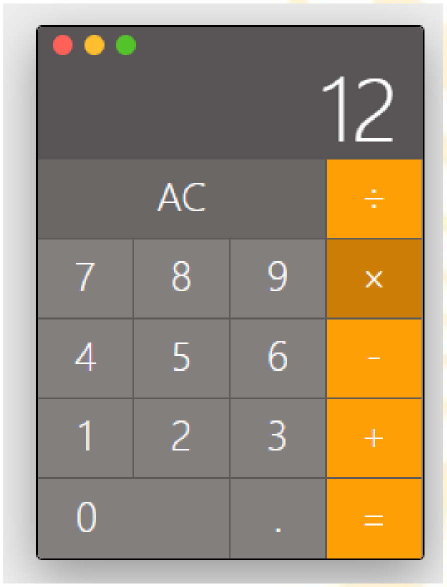
```
}

.row:nth-child(3) .button:first-child:active {
  background-color: rgb(131, 127, 127);
}

.row:last-child .button:first-child {
  width: 92px;
  padding-left: 23px;
  justify-content: flex-start;
}
```

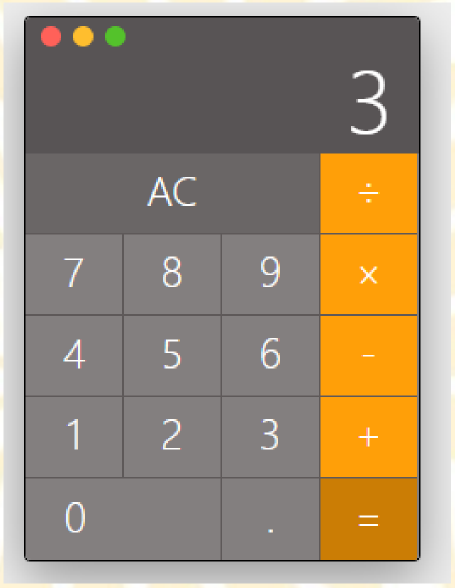
TS file

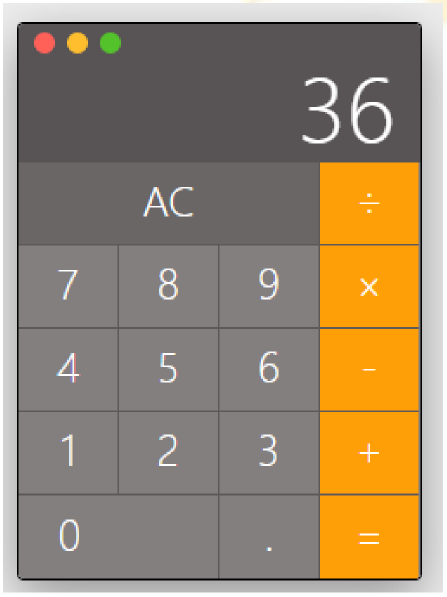
```
class Calculator {
    constructor(private rootElement:HTMLElement) {}
    private readonly KEYS: Array<Array<number | string>> = [
      ['AC', '÷'],
      [7, 8, 9, '×'],
      [4, 5, 6, '-'],
      [1, 2, 3, '+'],
      [0, '.', '=']
    ];
    private readonly NUMBERS: Array<string> = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.'];
    private readonly OPERATORS: Array<string> = ['÷', '×', '+', '-'];
    private readonly EXECUTE_FLAG: string = '=';
    private readonly CLEAR_FLAG: string = 'AC';
    private calculator: HTMLElement;
    private displayContainer: HTMLElement;
    private resultElement: HTMLElement;
    private x: string = '';
    private y: string = '';
    private operator: string = '';
    private result: string = '';

    public create(): void {
      this.createCalculatorContainer();
      this.createDecorator();
      this.createResultDisplayContainer();
      this.createResultDisplayElement();
```

```
  this.createButtons();
  this.addEventListener();
}
private createCalculatorContainer(): void {
  this.calculator = this.createElement('div');
  this.addClass(this.calculator, 'calculator');
  this.rootElement.prepend(this.calculator);
}
private createDecorator(): void {
  const dot: HTMLElement = this.createElement('div');
  const container: HTMLElement = this.createElement('div');
  this.addClass(dot, 'dot');
  this.addClass(container, 'decorator');
  container.appendChild(dot);
  this.calculator.appendChild(container);
}
private createResultDisplayContainer(): void {
  this.displayContainer = this.createElement('div');
  this.addClass(this.displayContainer, 'display');
  this.calculator.appendChild(this.displayContainer);
}
private createResultDisplayElement(): void {
  this.resultElement = this.createElement('div');
  this.addClass(this.resultElement, 'result');
  this.resultElement.textContent = '0';
  this.displayContainer.appendChild(this.resultElement);
}
private createButtons(): void {
  this.KEYS.forEach((rowKeys: Array<string | number>) => {
    const row: HTMLElement = this.createElement('div');
    this.addClass(row, 'row');
    this.calculator.appendChild(row);
    rowKeys.forEach((key: string | number) => {
      const button: HTMLElement = this.createElement('div');
      this.addClass(button, 'button');
      button.textContent = `${key}`;
      row.appendChild(button);
    });
  });
}
private addEventListener(): void {
  this.calculator.addEventListener('click', (event: MouseEvent) => {
    const target = event.target as HTMLElement;
    const { className } = target;
    if (className === 'button') {
      const key: string = ""+target.textContent;
      if (this.NUMBERS.indexOf(key) > -1) {
        if (!this.operator) {
          this.x += key;
          this.updateResult(this.x);
        } else {
          this.y += key;
          this.updateResult(this.y);
        }
      } else if (this.OPERATORS.indexOf(key) > -1) {
        if (this.x === '' && this.y === '') {
          this.x = '0';
          this.operator = key;
        } else if (this.x !== '' && this.y === '') {
          this.operator = key;
        } else if (this.x !== '' && this.y !== '') {
          this.result = this.excuteAlgorithm();
          this.updateResult(this.result);
          this.x = this.result;
          this.y = '';
          this.operator = key;
        }
      } else if (this.EXECUTE_FLAG === key) {
        if (this.x !== '' && this.y === '') {
          this.result = this.x;
          this.updateResult(this.result);
        } else if (this.x === '' && this.y === '') {
          this.result = '0';
          this.updateResult(this.result);
        } else if (this.x !== '' && this.y !== '') {
          this.result = this.excuteAlgorithm();
          this.updateResult(this.result);
          this.x = this.result;
          this.y = '';
          this.operator = '';
        }
      } else if (this.CLEAR_FLAG === key) {
        this.x = '';
        this.y = '';
        this.operator = '';
        this.result = '';
        this.updateResult('0');
      }
    }
  });
}
private updateResult(result: string): void {
  this.resultElement.textContent = result;
}
private excuteAlgorithm(): string {
  switch (this.operator) {
    case '+':
      return `${Number(this.x) + Number(this.y)}`;
    case '-':
      return `${Number(this.x) - Number(this.y)}`;
    case '×':
      return `${Number(this.x) * Number(this.y)}`;
    case '÷':
      return `${Number(this.x) / Number(this.y)}`;
  }
```

```
      return`error`;
    }
    private createElement(tag: string): HTMLElement {
      return document.createElement(tag);
    }
    private addClass(target: HTMLElement, className: string): void {
      target.classList.add(className);
    }
  }
```







Conclusion:

Successfully executed basic typescript programs, Hello world and calculator.