Date | | | | Mo Tu We Th Fr Sa Su

Typescript

What is Typescript?

i) Typescript is a syntactic superset of Javascript which adds static typing.

ii) This basically means that Typescript adds syntax on top of Javascript, allowing developers to add types.

iii) Ex: Typescript will report an error when passing a string into a function that expects a number. Javascript will not.

iv) Using Typescript is to use the official Typescript compiler, which transpiles Typescript code into Javascript.

Installing Typescript Compiler.

npm install typescript --save-dev

∴ The compiler is installed the node_modules directory and can be run with:

npx tsc

Configuring the compiler.

tsconfig.json file

npx tsc --init.

Date

* Type Assignment:
When creating a variable, there are two main ways Typescript assigns a type.
- Explicit.
- Implicit.

* Explicit Type:
let firstname: string = "Uday";
∴ Type assignment are easier to read & more conventional.

* Implicit Type:
let firstname = "Uday";
∴ Typescript will "guess" the type, based on the assigned value.

* Diff blw Javascript vs Typescript.

Javascript	Typescript
i) It's a plain vanila javascript	i) It's a top of javascript
ii) Ex: It's a like small kid	ii) It's a like small with discipline.
	iii) Static typing, code completion, & Shorthand notation
iii) Small applicat, we can use Javascript	iii) Big Application.

Date

Statically-Typed Lang

i) C, C++, Java.

ii) If we declare variable as Integer, we have to allign only Integer value.

iii) Ex: int number = 10;
number = "lucky";
↳ Error.

Dynamically-Typed Lang

i) Javascript, Ruby, & Python.

ii) Type of variable is a given at time, we can also change the value.

iii) Ex: let number = 10;
number = "aaa";

~~iv) Dis~~

iv) We have a problem if you mentioned
let number = 100;
number = "lucky";
Math.round(number);
↓
Here we will faced, Application will not work as expected.
Due to number storing value.

Date

Mo Tu We Th Fr Sa Su

4. Statically-Typed Lang | Dynamically-Typed-Lang

i) C, C++, Java.

ii) If we declare variable as Integer, we have to allign only Integer value.

iii) Ex int number = 10;
number = "lucky";
↳ Error.

i) Javascript, Ruby, & Python.

ii) Type of variable is a run time. we can also change the value.

iii) Ex let number = 10;
number = "aaa";

~~iv) Dis~~

iv) we have a problem. if you mentioned
let number = 100;
number = "lucky";
Math.round(number);

Here we will faced, Application will not work as expected.
Due to number storing value.

Date

* Typescript is developed by Microsoft.

* For Typescript, we can use wherever we are using javascript.

* Typescript Drawbacks:

i) compilation.

Browser will not understood our typescript code, so we need to translate from .ts → .js using compiler.

.ts ⟶ Compiler ⟶ .js.

ii) Discipline in coding.

* If you want to use medium to Large project we can use Typescript.

* Small project, we can use javascript, If will have big project, facing unexpected bugs or errors.

Typescript setup:

i) npm i typescript.

ii) tsc -v → version will display.

iii) index.ts (code)

tsc index.ts ⟶ It will display index.js & index.ts

Date

Mo Tu We Th Fr Sa Su

* Typescript is a superset of Javascript. Means whatever we have in Javascript features, we can use all & some additional features.

* First declaration.

- let age: number = 27; ✓
  age = "vinay"
  ↳ It will give error.

* Configuring Typescript Compiler:

↳ tsc --init

* Javascript & Typescript Built in Types:

Javascript	Typescript
number, string	any
boolean, null	unknown, never,
undefined, Object.	enum, tuple
Array.	

If any:
If we declare variable, & don't initialize any values. Assumes the variable any.

Ex: let level;
level = 1;
level = 'a';

Note: Best practice to avoid any type.

Date

ii) array:

```
let numbers = [],
let nos: number[] = []
nos[0] = 1;
nos[1] = 2;
```

iii) tuple:

* Tuple is a typed array, with a pre-defined length and types for each index.

Ex. `let tupleRes: [number, boolean, string]`
↳ // define our Tuple.

```
tupleRes = [27, true, "Uday"];
```

* Destructuring tuple:

```
const graph: [number, number] = [100, 200]
const [x, y] : graph;
```

iv) ~~Func~~ Enum:

```
const enum Size { small = 1, med, large }
let tShirt: Size = Size.med;
console.log(tShirt);
```

Date

v) Functions

```
function render(income:number):number
{
  return 0;
}
render(100);
```

vi) object:-

```
let emp: { id:number,
           name: string
         } = { id: 1,
               name: "Ilary"
             };
```

YAML Course

Date 30/08/2023.

Mo Tu We Th Fr Sa Su

Introduction.

i) YAML -> YAML ain't markup language.
ii) data format used to exchange data.
iii) Similar to XML & JSON
iv) In YAML, we can store only data, not commands.

-> Data serialization:-

Object -> file.
Ex:- YAML, JSON, XML

-> used to where:-

* configuration files
  L> Docker / kubernates, etc.
* logs, caches, etc.

-> Benefits:-

* Easy to read & simple.
* It has a strict syntax.
  L> Indentation is imp.
* Easily convertable to JSON, XML
* most language use YAML.

Date

Ex: hello.yml.

"apple": "It's a fruit"
1: "This is available in India"

-apple
- mango → list type document.
- banana

---

cities:
- newdelhi → Array type values.
- Bangalore

---

cities: [newdelhi, Bangalore]

→ JAML flow with CI/CD

Stages:
↳ Jobs
↳ steps

Ex stages:
- stage: Build
jobs:
- job: Build package.
steps:
- Build
- Package
- Publish

Mo Tu We Th Fr Sa Su

Date

Array Loops (Javascript)

1. map():

* The map() method creates a new array by performing a function on each element of an array.
* map() method does not change the original array.

Ex: const numbers1 = [45, 4, 9, 16, 25];
const res = numbers1.map(myFunction);

function myFunction(value, index, array)
{
return value * 2;
}

O/P
90, 8, 18, 32, 50.

2. forEach():

* forEach() method calls a function (a callback function) once for each array element.

Ex:- const num = [45, 9, 16, 25, 36];
num.forEach(myFunction);

function myFunction(val, index, arr)
{
console.log(val);
}

O/P
displays array elements

Date

Mo Tu We Th Fr Sa

3. flatMap():

* First map all elements of an array and then creates a new array by flattening the array

Ex:

```
Const arr = [1,2,3,4,5,6,7]
const newArr = arr.flatMap((x) => x*2);
```

Arrow function

o/p

2, 4, 6, 8, 10, 12, 14.

4. filter():

* The filter() method creates a new array with array elements that pass the condition.

```
const numbers = [45, 4, 9, 16, 25];
const res = numbers.filter(myFunction);

function myFunction(val, index, arr)
{
  return val > 18;
}
```

o/p

45, 25

Mo Tu We Th Fr Sa Su

Date

5. reduce():-

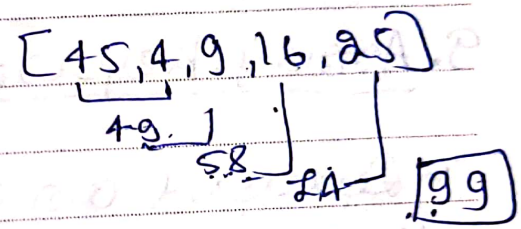
- The reduce method runs a function on each array elements to produce a single value.
- It will execute left-to-right.

Ex: const numbers = [45, 4, 9, 16, 25];

const res = numbers.reduce(myFunction);

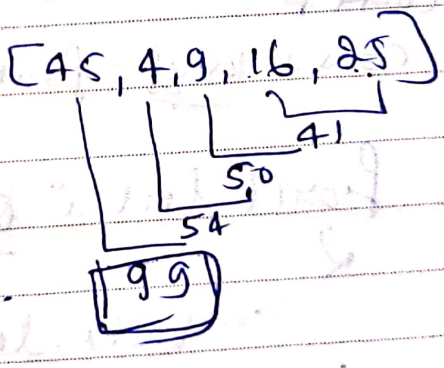
function myFunction(preVal, val, index, array)
{
  return preVal + val;
}

console.log(sum of array)



6. reduceRight():-

* Same feature of reduce.
* It will start execution from Right (top of the array).



7. find()
   ↳ Returns the value of the first array element that passes the condition.

ex. find((item) => { return item > 10 });

8. findIndex():
It will return the index value.

Date

8. Array.from():-

* Returns an array object from any object with a length property or any iterable object.
* create Array from string

let str = "ABCDE";

let arr = Array.from(str);

console.log(arr); // ['A','B','C','D','E']

9. Array.keys():-

* Returns an array Iterator object with keys of of an array.

Eg: ("Mango","Apple","Banana");

const fruits:

const keys = fruits.keys();

for(let i of keys)

{

console.log(i);

}

10. Array.entries():-

Create an array Iterator, and then iterate over the key/value pairs.

fruits.entries();

Mo Tu We Th Fr Sa Su

Date

Prototype Inheritance:

* All Javascript objects inherit properties and methods from a prototype.
  i) Date objects inherits from Date.prototype.
  ii) Array objects inherits " Array.prototype.
  iii) Person objects " " Person.prototype.

∴ Date, Array, Person objects inherit from Object.prototype.

* Adding Properties & Methods to objects:

i) If you want to add new property / methods to existing object.
ii) add new prop/methods into constructor functin.

↳ Prototype Property:

adding new property or method to constructor functions.

Ex
```
function myFunc(first, last, salary)
{
    this.first = first;
    this.last = last;
    this.salary = salary;
}
```
(method added here)

```
myFunc.nationality = 'India'  // not working

myFunc.prototype.nationality = "India"  // working
```

Date

HTML DOM (Document object Model)

* It's a manipulating the DOM.

.HTML Page.

```
<!DOCTYPE html>
<html>
<head>
  <title> First DOM </title>
  <script> script file </script>
                  > src = "./index.js"
</head>
<body>
  <div> ——— </div>
</body>
</html>
```

index.js (javascript)

i) create an element.

```
const createEle = document.createElement("div")
document.body.append(createEle)
```

index.html o/p

 ——

* For adding any text / sentance into an html element we can use these 2 ways.

i) createEle.textContent = "Hi buddy, How are you";

ii) createEle.innerText = "Hello Men";

Mo Tu We Th Fr Sa Su

Date

* innerText:-
Returns the content of the element & all its children, without CSS hidden text spacing & tags, except <Script> & style tags.

* ~~Inner~~ textContent
Returns the text content of the element & all descendants, with spacing and CSS hidden text, but without tags.

* innerHTML:-
If you want to add any html tags inside the text, It should render & displayed on web page.

Ex:-
```
const createEle = document.createElement("div")
createEle.innerHTML = "<h1>Hello Uma</h1>"
document.append(createEle);
```

O/P
Hello Uma.

Ex O/P & Hello World.
```
const el = document.createElement('div');
const boldTxt = document.createElement('b');
boldTxt.innerHTML = 'Hello World';
document.append(boldTxt);
document.append(el);
```

Date

HTML5 Tags:

→ HTML tags:

i) <nav> → Represents a navigation menu or links to other pages.

ii) <header>
It contains headings, logos, & navigation elements.

iii) <footer>
It contains the metadata, copyright information, or links to related documents.

iv) <marks>
Highlighting the content.

v) <section>, <progress> <article>

→ HTML5 properties:

i) placeholder:-
Short hint that describes the expected value.

ii) required:-
Input field must be filled out before submitting form.

iii) autofocus:
Input element should automatically focus.

Mo Tu We Th Fr Sa Su

Date

Next-JS:-

* It's a framework that make it easier to build web application.

* **Server-Side-Rendering (SSR):-**
It means that page can be rendered on the server before being sent to the client. this can improve initial page load performance.

* **Static Site Generation (SSG):-**
where page can be pre-built at build time.

* **Automatic Code Splitting:-**
Next.js automatically splits your javascript bundles to only send the necessary code for each page. It increases the page loads.

* Automatic refresh the code.

* **API Routes:-**
Easy to create API routes within the application, allowing you to handle server-side logic & data fetching.

* **File System Routing:-**
If you have created page like homepage, Routing will be /homepage.

* Typescript support.