Classes & Interfaces.

{Objects}

To store data, manipulate data & to execute methods

Object is an instances of classes.

Classes

Are blue-prints of objects

Defines how objects look like, which properties and methods they have.

To speed up objects.

Allow creation of multiple similar objects.

Ex. // class looks like

```
class Classname {                    // first CAPITAL as of classes.
    name: string;                    // It is called field.
}
```

Special Funct. thing in class which is called method for class i.e constructor method

```
constructor () {}
```

for Ex.

```
constructor (n: string) {
    this.name = n;                   // sets name fieldor
}                                    // property to the
                                     // n i.e input value.

// department Object
new Department ();                   // To call the class
                                     // It will initiate the
                                     // constructor which takes
                                     // no parameter.
```

so

for Ex

new Department('Abc');

Sep 17, 2022, 12:32

Then we can manipulate new object by storing it or rendering further

Ex.

```
const checkClass = new Department('Abc');
cg(checkClass);
```

- The same class in JS will look like.

```
class Department {
    constructor(n) {
        this.name = n;
    }
}
const checkJSClass = new Department('Abc');
clg('checkJSClass);
```

Other methods in class {}

```
describe() {}
```

Ex.

```
describe() {
    clg('Dept' + this.name) // this. will check for variable fields in class
    // clg('Dept' + name)
    // this will look for Global name.
}
```

And to call describe

```
accounting.describe(); // accounting is object we created.
```

Sep 17, 2022, 12:32

The 'this' keyword :

for Ex.

dummy object.

const accountCopy = { describe : account.describe };

accountCopy.describe();

// "this" is typically responsible for calling

// a method.

// To come through this issue by

Ex.

describe(this: ~~string~~ Department) {

cl(this.name);

} // Now this will only type to class Dep-

// to call it now value of n is

// compulsive in params.

Ex.

const accountingCopy = { name: 's',

describe: accounting.describe };

accountingCopy.describe();

This issue:

Sep 17, 2022, 12:32

Public & private modifiers.

Ex.
```
class Department {
    name: string;
    employees: string[] = []; // empty array.

    constructor (n:string) {
        this.name = n;
    // method 1
    addEmployee (employee: string) {
        this.employees.push (employee); // Adds empl.
    // method 2                          //in the string
    printEmployee () {
        log (this.employee.length);
        log (this.employees)
    }

const accounting = new Department('Abc');
accounting.addEmployee ('mayank');
accounting.addEmployee ('Tomas');

accounting.printEmployee ();
// we can also add another employee from
outside like:
accounting.employee[2] = 'Ana';
```

Sep 17, 2022, 12:32

To get through this issue by adding private to methods or fields etc.
Ex. private employee: string[] = [];
// Now only accessed from the class Depart

• So:

CSS

CSS is used to provide styles to the webpages
Style include

i) layout:
This is related to positioning of elements.

ii) Design:
(colors, fonts) bg's, etc.

iii) Responsiveness:
Screen size or device friendliness.

(link) tag:

<link rel="stylesheet" type="text/css" href="./src/x.css">

Priority.

{ Inline > head or internal styling. }

* { Internal > External } (may vary on position of tag.
{ External > Internal }