Machine generated alternative text:
Open-source programming language from Microsoft 
Typed superset of JavaScript 
Compiles down to plain JavaScript 
What & Why? 
Relation to JavaScript 
Optional static typing and type inference 
4) 
IDE support 
Rapid growth and use 
3:47 / 48:10 

**Important point**

By adding import or export statement typescript treat as module instead of script.

A module has own scope whereas script share as global scope

export {}

let message='welcome back';

console.log(message);

Typescript file compile to javascript file which is then used in our application

To compile: Tsc main.ts

To run :node main.js

**NOTE : compile automatically**

To watch :Tsc main --watch

NOTE: javascript have global scope and function scope there is no block level scope

Point 2) **variable declaration**

Let and const keyword

Const: always assigned a value and never reassigned again.

Let :- value can be reassigned again

Point 3: variable type

Machine generated alternative text:
let 
let 
let 
let 
isBeginner: boolean 
true; 
total: number e; 
name: string 
'Vishwas ' 
sentence: string 
'My name is $1 name) 
I am a beginner in Typescript' ; 
console . log (sentence) ; 
let 
let 
let 
let 
n: null null; 
u: undefined undefined; 
isNew: 
myName : 
boolean 
- nun; 
string - 
undefined; 

Advantage:---type checking

Notes: null and undefined considered as subtypes of type in typescript

**Syntax of declaring and array type**

**Point 1:** Declaration of array of same type

Machine generated alternative text:
let listl 
number 
let list2 

**Point 2:** Declaration of array of mix type

We can create a tuple that will contain a string and number

**NOTE:** the number of element in the array is fixed

let person1:[string,number]=['saurabh',31];

Declaring a enum

Machine generated alternative text:
num Colorf'red' 
et c : Color—Color. green; 
onsole. 

Answer is 1:) since enum value is start from zero

Machine generated alternative text:
enum Color (Red 5, 
Green, Blue); 
let c: Color Color . Green; 
console . log(c); 

**Any type : we can assign any value to variable**

let random:any="saurabh";

random=1;

console.log(random);

**Union type**

Machine generated alternative text:
let a; 
a true; 
let b 
let multi Type: number I boolean; 
multi Type 
multi Type true; 

**Function**

Machine generated alternative text:
function 
add(numl 
return numl + 
console. 
number , num2 
n um2; 
number) : number 

We can make parameter as optional in function by putting ? After parameter but condition is optional parameter must be declared after required parameter

Machine generated alternative text:
function 
add(numl 
return numl; 
console. 
number , num2 ? 
number) : 
number 

We can also have default value in parameter

Machine generated alternative text:
function 
add(numl 
number , num2 
return numl + num2; 
console. 
number-ID : 
number 

Default parameter are like optional parameter with a set value instead of undefined

**Interface in typescript**

**It is also possible to define object in function**

**Case 1:--**

**Object having 2 property**

Machine generated alternative text:
function 
test (object : ffirstname : string, lastname : string)) 
console . log( M object . firstname) $(object . lastnameY ) 
let c 
firstname: ' saurabh ' 
lastname: 'kesarwani ' 
test(cbl; 

**Case 2 : object having more property**

**NOTE:** same interface can be use as a type in multiple function

Machine generated alternative text:
interface Person 
firstName: string; 
lastName: string; 
function fullName( rson: Person 
console . log( $(person . firstName) $1 person. lastNameY ) ; 
let p 
firstName : 
lastName : 
• Bruce ' 
• Wayne ' 

Note: interface parameter can be optional

**Class**

In javascript there is no concept of classes

Machine generated alternative text:
class Employee 
empname : string; 
constructor(name:string) 
this . empname:name ; 
greet ( ) 
console. employee name is $fthis .empname 
let emp:new Employee( ' saurabh ' ) ; 
emp. greet(); 