TypeScript

Hello!

I am Hiten Pratap Singh

I am here because I love to dive into new interesting things.

You can find me at:
https://github.com/hitenpratap/
https://hprog99.wordpress.com/
hiten@nexthoughts.com

Agenda

- Introduction
- Installation
- Why TypeScript?
- Features
- Comparison With TypeScript Alternatives
- Who Uses TypeScript?
- Conclusion

1.
Introduction

JavaScript that scales.

TypeScript lets you write JavaScript the way you really want to.
TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
Any browser. Any host. Any OS. Open Source.

Overview

- Syntax based on ECMAScript 4 & ECMASCript 6 proposals
- TS is first and foremost a superset of JS
- Any regular Javascript is valid TypeScript Code

“Microsoft's TypeScript may be the best of the many JavaScript front ends. It seems to generate the most attractive code.”

- Douglas Crockford

"CoffeeScript is to Ruby as TypeScript is to Java/C#/C++."

- Luke Hoban

2.
Installation

How to get it setup?

How To Install

For more details check: https://www.typescriptlang.org/docs/tutorial.html

Via npm (the Node.js package manager)

```
$ npm install -g typescript
$ npm view typescript version
npm http GET https://registry.npmjs.org/typescript
npm http 304 https://registry.npmjs.org/typescript
0.8.1-1
```

By installing TypeScript's Visual Studio plugins

Just download any appropriate Visual Studio from Microsoft site and you are all set to go.

You can also try Visual Studio Code IDE from https://code.visualstudio.com/ which is free, open source and available for multi-platform.

3.
Why TypeScript?

Why everyone is using it more and more.

Main Goals of TypeScript

- Provide an optional type system for JavaScript.

```
1  var foo = 123;
2  foo = '456'; // Error: cannot assign 'string' to 'number'
3
4  var foo: number = 123;
5  var foo: number = '123'; // Error: cannot assign a 'string' to a 'number'
```

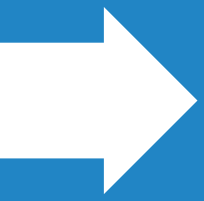
- Provide planned features from future JavaScript editions to current JavaScript engines
- Modular Development

4.
Features

What makes TypeScript super awesome

TypeScript Features

- Data Types Supported
- Optional Static Type Annotation
- Classes
- Interface
- Modules
- Arrow Expressions
- Type Assertions
- Ambient Declarations
- Source File Dependencies



Data Types

- Any
- Primitive
    - Number
    - Boolean
    - String
    - Void
    - Null
    - Undefined - Same as JS
- Array
- Enum

Any

Any is used when it's impossible to determine the type

```
// When it's impossible to know, there is the "Any" type
var notSure: any = 4;
notSure = "maybe a string instead";
notSure = false; // okay, definitely a boolean
```

Primitive

- Doesn't have separate integers and float/double type. These all are floating point values and get the type 'number'
- boolean - true/false value
- string - both single/double quote can be used
- No separate char type
- void - is used in function type returning nothing
- null and undefined - functions as usual

```
var isDone: boolean = false;
var lines: number = 42;
var name: string = "Hello World";

function bigHorribleAlert(): void {
  alert("I'm a little annoying box!");
}
```

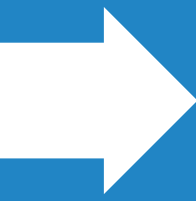
Array

```
var cities:string[] = ["Berlin","Quebec","New York"]
var primes:number[] = [1,3,5,7,11,13]
var bools:boolean[] = [true,false,false,true]

// Alternatively, using the generic array type
var list: Array<number> = [1, 2, 3];
```

Enum

By default, enums begin numbering their members starting at 0. You can change this by manually setting the value of one its members.

```
// For enumerations:
enum Color {Red, Green, Blue};
var c: Color = Color.Green;
enum Color {Red = 0, Green, Blue};
enum Color {Red = 3, Green, Blue};
```



Optional Types

Type Annotations/Checking

JavaScript

```
var a = 987;
a.trim();
//JavaScript error: TypeError: a.trim is not a function on line 5
```

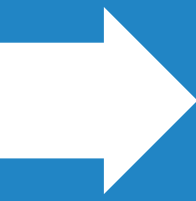
TypeScript

```
var a = 987;
a.trim();
//Property 'trim' doesn't exist on 'number'
var a:string = 123
a.trim()
//Cannot convert 'number' to 'string'
```

Type Inference

- **TypeScript tries to infer types**
- **Four ways to variable declaration -**
  - Type and Value in one statement
  - Type but no Value then Value will be undefined
  - Value but on Type then the it will be of Any type but maybe be inferred based on its value.
  - Neither Value nor Type then Type will be Any and Value will be undefined.

```
var message1:string = "Hello World";
var message2:string;
var message3 = "Hello World";
var message4;
```



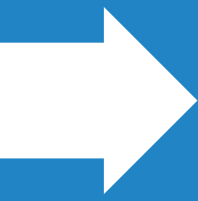
Classes

TypeScript Classes

- Can implement interfaces
- Inheritance
- Instance methods/members
- Static methods/members
- Single constructor
- Default/Optional parameter
- ES6 class syntax

TypeScript Classes Example

```
// Classes - members are public by default
class Point {
  // Properties
  x: number;
  constructor(x: number, public y: number = 0) {
    this.x = x;
  }
  // Functions
  dist() { return Math.sqrt(this.x * this.x + this.y * this.y); }
  // Static members
  static origin = new Point(0, 0);
}

var p1 = new Point(10 ,20);
var p2 = new Point(25); //y will be 0

// Inheritance
class Point3D extends Point {
  constructor(x: number, y: number, public z: number = 0) {
    super(x, y); // Explicit call to the super class constructor is mandatory
  }
  // Overwrite
  dist() {
    var d = super.dist();
    return Math.sqrt(d * d + this.z * this.z);
  }
}
```



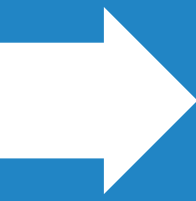
Inheritances

TypeScript Interfaces

- Declared using interface keyword
- Like other TS features it's design time features i.e. no extra code would be emitted to resultant JS file
- Errors being shown when interface signature and implementation doesn't match.

TypeScript Interfaces Example

```
interface Employee{
  firstName:string;
  lastName?:string; //optional member
  age:number;
}

function showEmployeeDetails(emp:Employee){
  console.log('Hello '+emp.firstName+' '+emp.lastName+'. Your age is '+emp.age);
}

var emp:Employee = {firstName:"Test",lastName:"Name",age:23};
var emp1 = "Hello Employee2";
var emp2:Employee = {firstName:"Test Emp#1",age:23};

showEmployeeDetails(emp);//Works as expected
showEmployeeDetails(emp1);//Not an Employee Type
showEmployeeDetails(emp2);//Works as expected as well
```



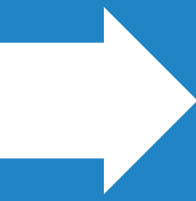
Modules

TypeScript Modules

- Modules can be defined using module keyword
- A module can contains sub-modules, class, enums or interfaces. But can’t directly contains functions.
- Modules can be nested(sub-modules).
- Classes and Interfaces can be exposed using export keyword.

TypeScript Modules Example

```
module Geometry {
  export class Square {
    constructor(public sideLength: number = 0) {
    }
    area() {
      return Math.pow(this.sideLength, 2);
    }
  }
}

var s1 = new Geometry.Square(5);

// Local alias for referencing a module
import G = Geometry;

var s2 = new G.Square(10);
```



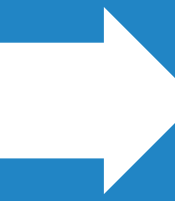
Arrow Expressions

Arrow Expressions

- Implicit return
- No braces for single expression
- Part of ES6
- Lexically scoped this
- You don't need to keep typing function
- It lexically captures the meaning of arguments

Arrow Expressions Example

```
function(arg){
  return arg.toLowerCase();
}

(arg) => arg.toLowerCase();
```



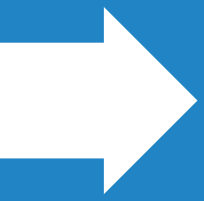
Type Assertions

Type Assertions

TypeScript's type assertion are purely you telling the compiler that you know about the types better than it does, and that it should not second guess you.

Type Assertions Example

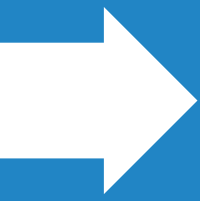
```
var foo = {};
foo.bar = 123; // error : property 'bar' does not exist on '{}'
foo.bas = 'hello'; // error : property 'bas' does not exist on '{}'

interface Foo {
    bar: number;
    bas: string;
}
var foo = {} as Foo;
foo.bar = 123;
foo.bas = 'hello';
```



Ambient Declarations

Ambient Declarations

A major design goal of TypeScript was to make it possible for you to safely and easily use existing JavaScript libraries in TypeScript. TypeScript does this by means of declaration.



Source File Dependencies

Source File Dependencies

- Can be done using reference keyword
- Must be the first statement of file
- Paths are relative to the current file
- Can also be done using tsconfig file

Source File Dependencies Example

```
/// <reference path="../typings/jquery.d.ts"/>
/// <reference path="components/someclass.ts"/>
class Foo { }
```

5.
Comparison with TS Alternative

TypeScript VS ES6 Harmony

- Complete language + Runtime overhaul
- More features: generators, comprehensions, object literals etc
- Will take years before widely deployed
- No typing as of now(Maybe ES7)

TypeScript VS CoffeeScript

- Also a superset to JavaScript
- More syntactic sugar, still dynamically typed
- Unlike TypeScript, JS is not valid CoffeeScript code
- It doesn't track ECMAScript 6

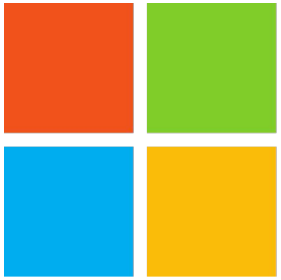
TypeScript VS DART

- Optionally typed
- A native VM
- Operator overloading
- ECMAScript Dart spec
- Completely different syntax and semantics than JS

6.
Who Uses TypeScript?

Companies use TypeScript

Adobe



7.
Conclusion

Conclusion

Pros:

- High value, low cost improvement over JavaScript
- Safer and more modular
- Solid path to ECMAScript 6

Cons:

- Still need to know some JS quirks
- Current compiler slowish(Faster one is in development)

Thanks!

Any questions?

You can find me at:
https://github.com/hitenpratap/
https://hprog99.wordpress.com/
hiten@nexthoughts.com

References

- https://www.typescriptlang.org/docs/tutorial.html
- https://learnxinyminutes.com/docs/typescript/
- https://basarat.gitbooks.io/typescript/content/
- http://www.slideshare.net/SanderMak/typescript-coding-javascript-without-the-pain
- http://www.slideshare.net/aniruddha.chakrabarti/typescript-44668095