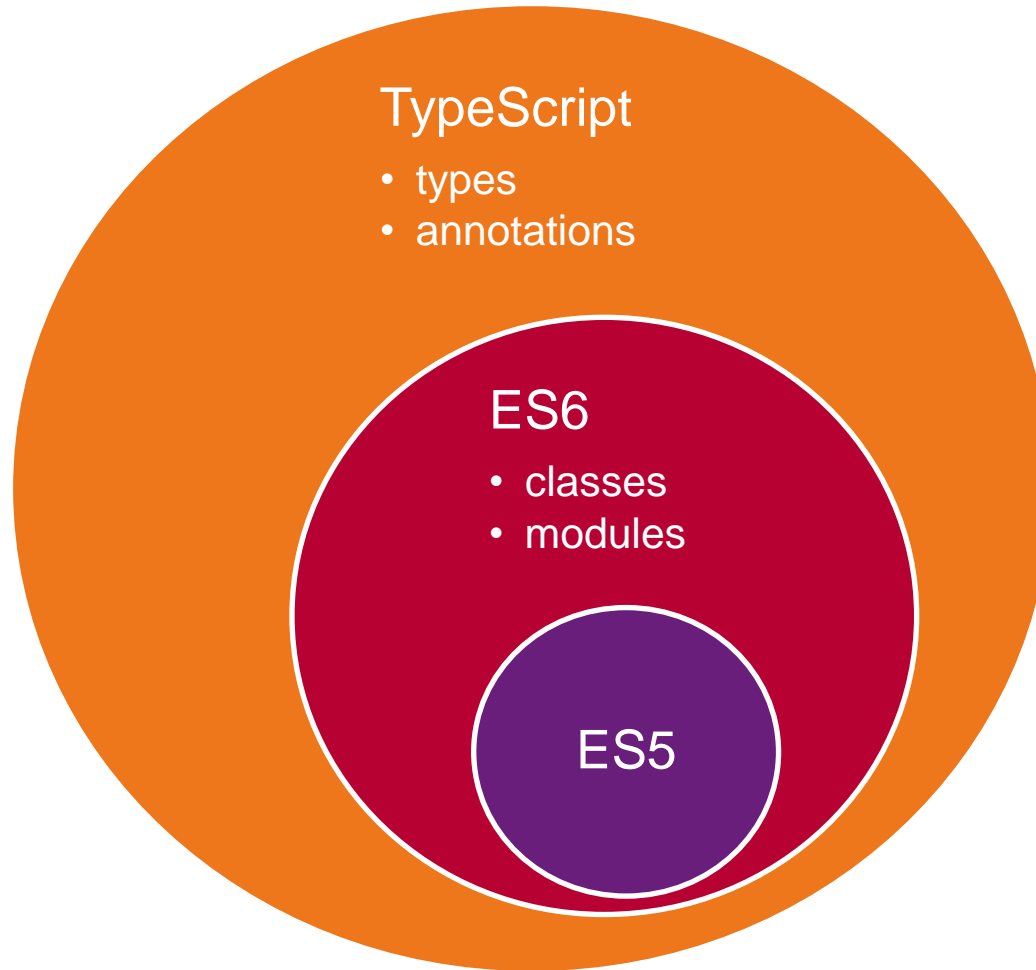


Angular 2

TypeScript Introduction

TypeScript



ANGULAR 2 IS BUILT IN TYPESCRIPT

Why TypeScript

- TypeScript can be used for cross browser development and is an open source project
- TypeScript compiles code which has syntax similar to ES6 to simple JavaScript code which runs on any desktop browsers, mobiles, servers / runtimes, or in any other ES3-compatible environment
- Using TypeScript developers can apply class-based approach, compile them down to JavaScript without waiting for the next version of JavaScript.
- With TypeScript existing JavaScript code can be easily incorporated with popular JavaScript libraries like jQuery, Backbone, Angular and so on.
- Enable scalable application development with optional Static types, classes and modules. Static types completely disappear at runtime.
- TypeScript converts JavaScript Programming from loosely typed to strongly typed.

Getting Started with TypeScript

- Visual Studio includes TypeScript in the box, starting with Visual Studio 2013 Update 2 and in Visual Studio 2015.
- Text editors like WebStorm, Atom, SublimeText, Eclipse and Brackets(freeware) can be used.
- TypeScript compiler can be installed for Node Environment by installing it as a node package.
 - `npm install -g typescript` (installing typescript as a global module)
 - `tsc <filename.ts>` (To compile TypeScript from command line).
- In the webpages, we need to refer JavaScript code file (*.js) and not the TypeScript code file (*.ts)
- Microsoft provides an online editor, where we can compile TypeScript Code to Native JavaScript.
 - Link : <http://www.typescriptlang.org/Playground>

TypeScript Playground

The screenshot shows the TypeScript Playground interface in a web browser. The browser's address bar displays `www.typescriptlang.org/play/`. The page header features the TypeScript logo and a notification: "TypeScript 1.8 is now available. [Download](#) our latest version today!".

Below the header, there is a control bar with a dropdown menu set to "Using Classes", and four buttons: "TypeScript", "Share", "Run", and "JavaScript".

The "TypeScript" panel on the left contains the following code:

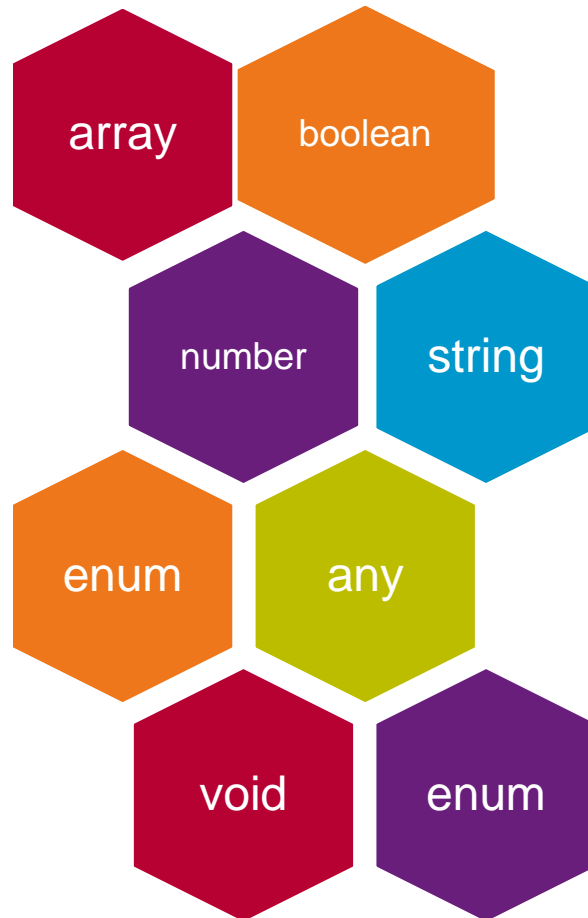
```
1 class Employee{
2     private _id:number;
3     constructor(id:number){
4         this._id = id;
5     }
6 }
```

The "JavaScript" panel on the right contains the equivalent function syntax code:

```
1 var Employee = (function () {
2     function Employee(id) {
3         this._id = id;
4     }
5     return Employee;
6 }());
7
```

Basic Types

- TypeScript supports the following Basic Types



Classes in TypeScript

- Traditional JavaScript focuses on functions and prototype-based inheritance, it is very difficult to build application using object-oriented approach.
- Starting with ECMAScript 6 (the next version of JavaScript), JavaScript programmers can build their applications using this object-oriented class-based approach.
- TypeScript supports public, private and protected access modifiers. Members of a class are public by default.

```
class Employee{  
    private _id:number;  
    constructor(id:number){  
        this._id = id;  
    }  
}
```

Inheritance

- TypeScript allows us to extend existing classes to create new ones using inheritance.
- 'extends' keyword is used to create a subclass.
- 'super()' method is used to call the base constructor inside the subclass constructor.

```
class Foo {  
    constructor() {  
        console.log("Foo constructor Invoked");  
    }  
}  
  
class Baz extends Foo {  
    constructor() {  
        super();  
        console.log("Baz constructor Invoked");  
    }  
}  
  
new Baz();
```


Inheritance

```
class Foo {
    private _privateSample: string;
    public publicSample: string;
    protected protectedSample: string;
    constructor() {
        this._privateSample = "Only in Foo";
        this.publicSample = "Foo-Derived-Instance";
        this.protectedSample = "Foo-Derived";
    }
}
class Baz extends Foo {
    constructor() {
        super();
        this.protectedSample = "Changed Protected in Baz";
        this.publicSample = "Changed Public in Baz";
    }
    print(): void {
        console.log(this.protectedSample);
        console.log(this.publicSample);
    }
}
var bazObj: Baz;
bazObj = new Baz();
bazObj.publicSample = "Changed Public via Instance";
bazObj.print();
```

Accessors

TypeScript supports getters/setters as a way of intercepting accesses to a member of an object.

It gives way of having finer-grained control over how a member is accessed on each object.

```
class Foo {  
    private _name: string;  
    get name():string{  
        return this._name;  
    }  
    set name(name){  
        this._name=name;  
    }  
}  
  
var fooObj = new Foo();  
fooObj.name = "Karthik";  
fooObj.name;
```

Static Property

- In TypeScript we can also create static members of a class, those that are visible on the class itself rather than on the instances.

```
class Foo {  
    static staticVariable:string;  
    instanceVariable:string  
    constructor(instanceVariable:string){  
        this.instanceVariable = instanceVariable;  
    }  
    static staticMethod(){  
        return Foo.staticVariable;  
    }  
}  
  
var fooObj = new Foo("Instance");  
console.log(fooObj.instanceVariable);  
Foo.staticVariable = "Static"  
Foo.staticMethod();
```

Optional, Default & Rest Parameters

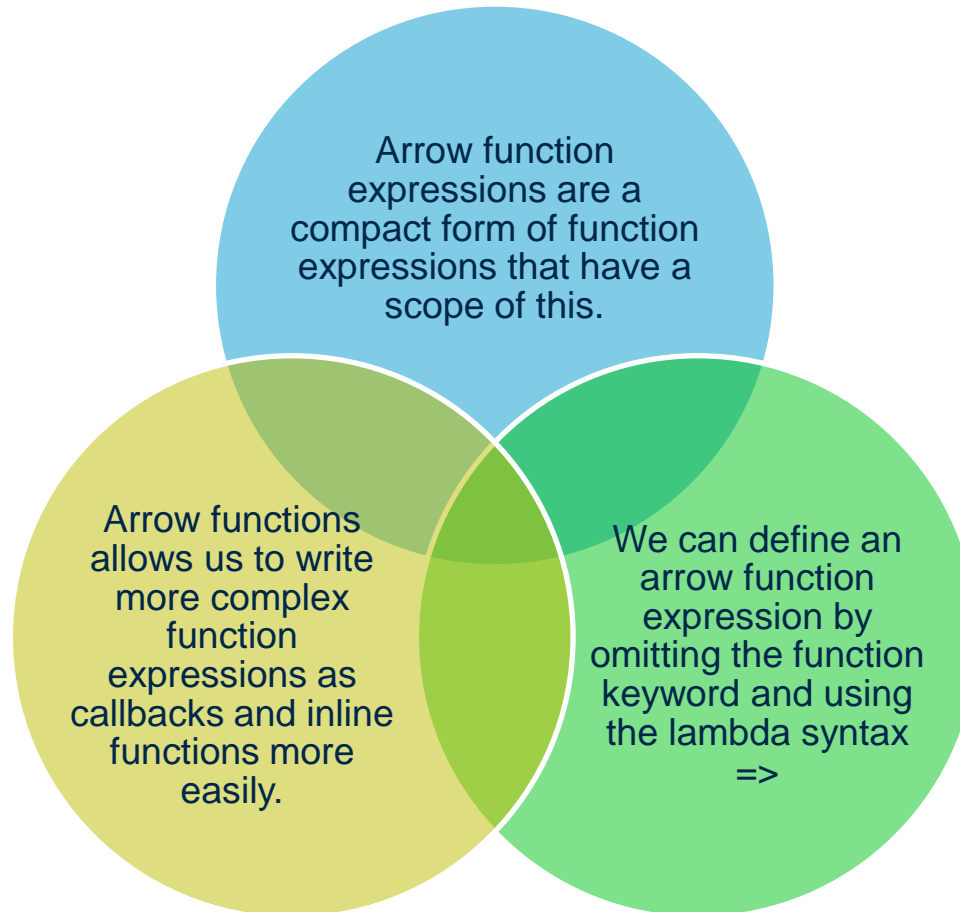
- In JavaScript, every parameter is considered optional, we can get this functionality in TypeScript by using the '?' beside parameters we want optional. Optional parameters must follow required parameters.
 - *function test(firstName:string, lastName?:string){ }*
- In TypeScript, we can also set up a value that an optional parameter will have if the user does not provide one. These are called default parameters.
 - *function test(firstName:string, lastName = "Gupta"){ }*
- To pass multiple parameters as a group or when we don't know how many parameters a function take we can use parameter followed by (...)
 - *function test(firstName:string, ...others:string[]){ }.*

Template Strings

- In ES6 new template strings were introduced.
- The two salient features of template strings are
 - Variables within strings (without being forced to concatenate with +)
 - Multi-line strings (using backticks `)

```
let firstName = "Karthik";
let lastName = "Muthukrishnan";
// interpolate a string
let greeting = `Hello ${firstName} ${lastName}`;
let multiLine = ` Multi
                  Line
                  String`;
console.log(greeting);
console.log(multiLine);
```

Arrow Functions



```
var addNumbers = (firstNumber:number, secondNumber:number) => firstNumber + secondNumber;  
addNumbers(5, 6);
```

Function Overload

- JavaScript is inherently a very dynamic language.
- JavaScript functions can be overloaded and return different types of objects based on the arguments passed in.

```
function pickData(data:string): string;
function pickData(data:number): number;

function pickData(data): any {
    if (typeof data === "string")
        return data.toUpperCase();
    if (typeof data === "number")
        return data * data;
}

pickData("Karthik");
pickData(5);
```

^ pickData(data: string): string
1/2
v

pickData()

Interfaces

- Interfaces plays many roles in TypeScript code.
 - Describing an Object : If function take lot of members but you're likely to pass a few of those, we can describe an object using interface with optional parameters
 - Describing an Indexable Object : Using TypeScript interfaces we can represent the expected type of an indexing operation.
 - Ensuring Class Instance Shape: Interface can be used in the same way which we use in traditional OOP languages like C# and JAVA.

```
interface testable{  
    name:string;  
}  
function test(args:testable){  
    console.log(args.name);  
}
```

```
interface CarPart {  
    [name: string]: string;  
}  
class Test{  
    part:CarPart={};  
}  
var testObj = new Test();  
testObj["frame"] = "Frame";
```

```
interface Greetable {  
    greet(message: string): void;  
    language: string;  
}  
  
class Greet implements Greetable{  
    language = 'English';  
    greet(message: string) {  
        console.log(message);  
    }  
}
```

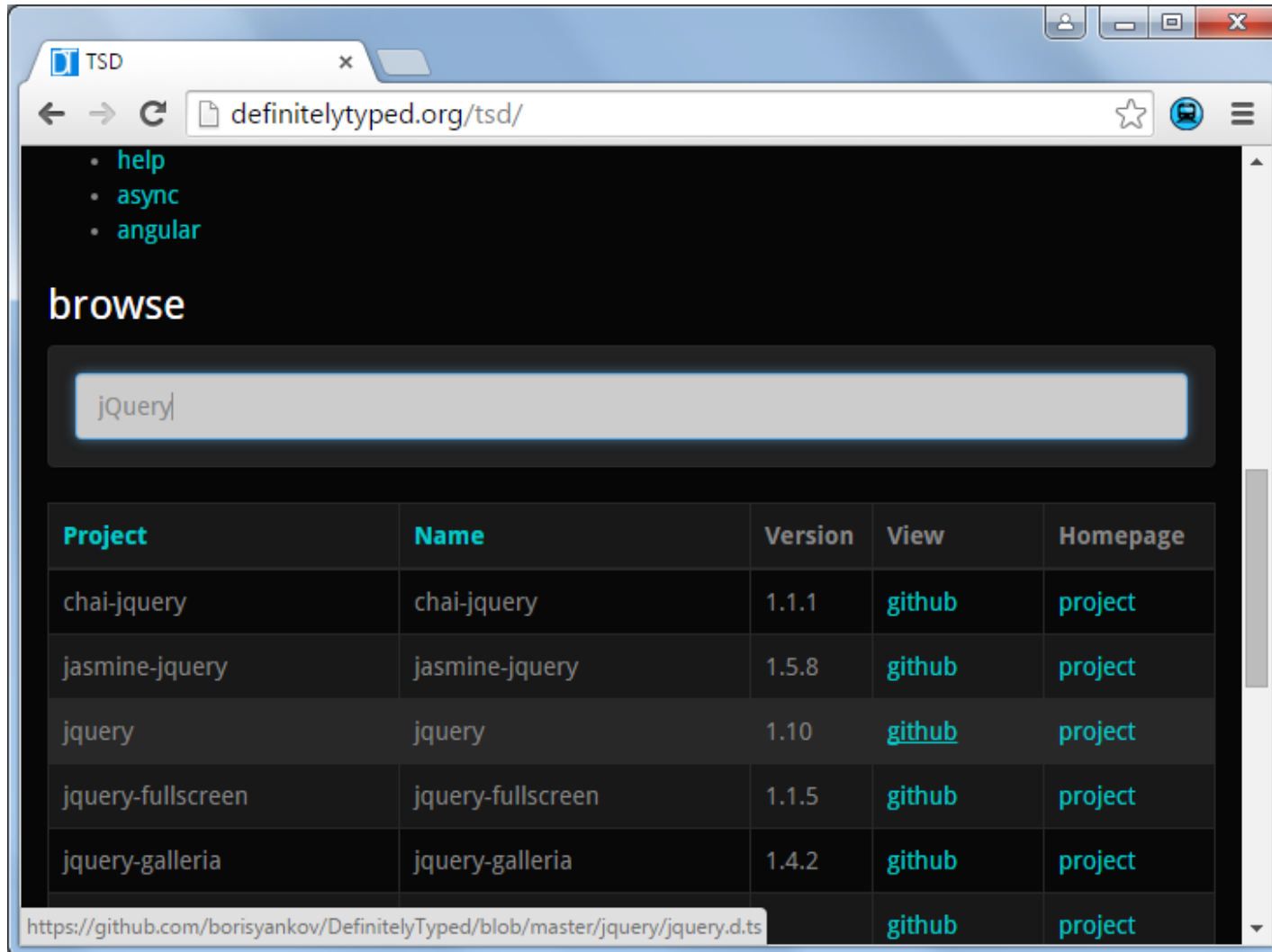

Modules

- A module is the overall container which is used to structure and organize code.
- It avoids collision of the methods/variables with other global APIs.
- TypeScript support modules to organize the code.
- 'export' keyword is used before the classes/ inner module to make it visible outside the module.
- We can refer the multi file internal module using the following line and compile multiple internal file into a single JS file using typescript compiler
 - `/// <reference path="<typescriptfilename>" />`
- External modules can be imported using 'import' keyword
 - `import module1 = require('./Module1');`

Declaration file (.d.ts)

- Documentation helps developers to understand the implementation details of JavaScript libraries.
- It is very much important to describe the shape of an external JavaScript library or new host API, so that it will make developer's life easier rather reading and understanding the JS code.
- Using a declaration file (.d.ts) in TypeScript, we can describe the shape and provide the documentation for the JavaScript library.
 - To generate declaration file use the command : `tsc --declaration <tsfilename>`
- definitelytyped.org is the repository for high quality TypeScript type definitions for popular JavaScript Libraries like jQuery, Angular, Backbone etc.

definitelytyped.org



The screenshot shows a web browser window with the address bar displaying `definitelytyped.org/tsd/`. The page has a dark theme and includes a sidebar with links to `help`, `async`, and `angular`. The main content area is titled "browse" and features a search input field containing the text "jQuery". Below the search field is a table listing various jQuery-related projects.

Project	Name	Version	View	Homepage
chai-jquery	chai-jquery	1.1.1	github	project
jasmine-jquery	jasmine-jquery	1.5.8	github	project
jquery	jquery	1.10	github	project
jquery-fullscreen	jquery-fullscreen	1.1.5	github	project
jquery-galleria	jquery-galleria	1.4.2	github	project
https://github.com/borisyankov/DefinitelyTyped/blob/master/jquery/jquery.d.ts			github	project

Person.ts

```
class Person {  
    private _age: number;  
    private _name: string;  
  
    constructor(name?: string, age?: number) {  
        this._age = age;  
        this._name = name;  
    }  
  
    get age(): number {  
        return this._age;  
    }  
  
    get name(): string {  
        return this._name;  
    }  
  
    getPersonInformation(): string {  
        return "Name : " + this._name + " Age: " + this._age;  
    }  
}
```

Person.d.ts

```
declare class Person {  
    private _age;  
    private _name;  
    /**  
     * Default constructor to create a person Instance  
     */  
    constructor();  
    /**  
     * constructor takes name and age for a person Instance  
     * @param name string type used to assign the name for the person  
     * @param age number type used to assign age for the person  
     */  
    constructor(name: string, age: number);  
    /**  
     * get the age for person  
     */  
    age: number;  
    /**  
     * get the name for person  
     *  
     */  
    name: string;  
    /**  
     * get the person information  
     * returns the name and age as string  
     */  
    getPersonInformation(): string;  
}
```

```
/// <reference path="person.d.ts" />  
var p = new Person("Ganesh",
```

▲ 2 of 2 ▼ Person(name: string, age: number): Person
constructor takes name and age for a person Instance
age: number type used to assign age for the person

Summary

- TypeScript is an open source project maintained by Microsoft.
- TypeScript generates plain JavaScript code which can be used with any browser.
- TypeScript offers many features of object oriented programming languages such as classes, interfaces, inheritance, overloading and modules, some of which are proposed features of ECMA Script 6.
- TypeScript is a promising language that can certainly help in writing neat code and organize JavaScript code making it more maintainable and extensible.
- Angular 2 is built in typescript

