MERN stack

Day1

FrontEnd Technology / UI / Client Side Web applications

HTML

- "what" content you want to place on your webpage
- responsible to define the "structure" of a webpage

CSS

- "How" content shoud appear on a webpage
- responsible to define 'style rules' for webpage

JavaScript

- to provide functionalities on the webpage
- it is a scripting language
- the JS code executes only inside the browser (** using nodejs, we can execute js code outside browser)
- **- every browser has a javascript engine, which is actually responsible to execute js code
  - chrome - V8
  - firefox - spidermonkey
  - IE - chakra
  - ....
- using javascript
  - business logic
  - handle events
  - dom manipulation
  - css manipulation
  - ajax calls
  - ....

JQuery

- it is a JS library to simplify javascript programming
- light weight / fast / cross-browser / open source

Bootstrap / SemanticUI / ...

- it is HTML+CSS+JQUERY Library
- it is used to create responsive web application

Angular / React / Vue / Ember / ...

- framework

language	library /package	framework
javascript	jquery, bootstrap, react, etc.	Angular, Ember, etc

java io, collection, jdbc Spring
external libraries
c# / VB ..... .net framework
python ..... dgango
| | |
provide simplifies the standardize the process
programming tasks
capabilities

JavaScript

- 1997, JS introduced
- 1997, it was submitted to ECMA for standardization
- since then, ECMA keeps publishing 'Specifications / Standards' for JS Engines
- every browser has to implement these 'Specifications / Standards'
  - the implementation of these Specifications is called 'ECMAScript'
- 1997 = ES1
- 1998 = ES2
- 1999 = ES3
- 2009 = ES5 ----> JavaScript
- 2015 = ES6 -----> got popularity
- 2016 = ES7
- 2017 = ES8
....
- 2020 = ES11

ES6 / ECMAScript 2015
- it is specification / standard that needs to be implemented by browsers
- it introduced lot of new features
- bcaz of new features it is considered as "Modern JavaScript"

**till today many browsers have not yet even implemented ES6 completely except Chrome

ES6 Features

- let, const keyword
- arrow function
- classes & objects
- module system
- spread operators
- object destructuring
- promise, async, await
- generators
- array/string method improvisation
- .....

JavaScript tools

- Transpilers
  - i.e. babel
  - converts modern js into legacy js

- Bundlers
  - i.e. webpack
  - bundle several file together in a single file
- package managers
  - i.e. npm
  - manages the dependencies

JavaScript

- it is interpreted language
- can be executed directly inside browser

```
var x = 10;
x = 'welcome'
```

TypeScript
- superset of javascript (ES6) + some additional features i.e. types, decorators, generics etc.
- it is compiled language
- cannot be executed in browser directly, we have to compile
  - TS code gets converted into JS code
- it provides type safety

Type System

- primitive types
  1. string
  2. number
  3. boolean

```
let variableName:Type
  let x: number          //explicit type declaration i.e. number
  let x = 10             //implicit type inference i.e number
  let x;                 //default type is 'any'
```

- arrays

```
let names: string[]        //explicit type declaration
let names = ['aa','bb','cc','ee']
let nums: number[]
let nums = [1,2,3,4,5]     //implicit type inference
let nums: Array<number>;
```

- any
  - it is a special type in typescript

- union types

```
let x: string | number;
```

- object type

```
let p1 = {
    fname : 'Shubham',
```

```
        lname : 'Rai'
    }
    - custom object type
    ----------------------
        1. using 'type' keyword
            type Person = {fname : string, lname : string, age?: number}
            let p1:Person = {
               fname : 'Shubham',
               lname : 'Rai'
            }
        2. using 'interface'
            interface Person {
               fname : string
               lname : string
            }
            let p1:Person = {
               fname : 'Shubham',
               lname : 'Rai'
            }
        3. using class
            class Person{
               public fname: string
               public lname: string
               constructor(fname:string, lname:string){
                  this.fname = fname
                  this.lname = lname
               }
            }
            let p1:Person = {
               fname : 'Shubham',
               lname : 'Rai'
            }
    - null & undefined
    - Enum
       - set of named constants
       enum WEEKDAY{
          MON, TUE, WED, THUR, FRI, SAT, SUN
       }
    - tuple
       enum PlayerType {
          Bowler,
          Batsman
       }
       let player1 = ['Shubham', PlayerType.Batsman, 24 ]

- Classes & objects
----------------------
```

Angular 1.x / AngularJS / Angular 1

- based on javascript

Angular 2 / 4 / 5 / 6 / 7 / 8/ .....

- based on typescript

Environment setup

---

1. download & install Nodejs
2. download & install code editor (any)
    - VS Code
    - brackets
    - atom
    - webstorm
    - ..etc.
3. install angular-cli
    - angular-cli is a command line tool for angular projects

    > npm install -g @angular/cli

Introduction to Angular

- helloworld program
- directory structure
- how angular works

Angular Project directory structure

```
|
|-->e2e
|   |-> store end-to-end test cases written using protractor
|
|-> node_modules
|   |-> stores all the dependencies downloaded
|
|-> src
|   |->app
|   |   |->app.module.ts
|   |   |->app.component.ts
|   |   |->app.component.html
|   |
|   |->assets
|   |   |-images, pdf, videos, other files
|   |
|   |->environments
|   |   |-> store env specific properties
```

```
|   |
|   |->index.html
|   |   |-> this file gets serve first when the app starts
|   |
|   |->main.ts
|   |   |->entry point of your application
|   |
|   |->styles.css
|
|-> package.json
|   |-> lists all the dependencies & start up scripts
|
|-> angular.json
|   |->stores configuration related to angular application
|
|-> tsconfig.json
|   |-> stores configuration for typescript
|
|-> tslint.json
|   |-> stores configuration related to tslint
```