**Hands-On: Stage 3 - Angular SPA-001 to SPA-011 - Day 68 – Hands on 1**

**Session - 1**

1. **Explain the characteristics of a Single Page Application (SPA)?**

* A single-page application (SPA) is an application that interrelates with users by vigorously rewriting the existing web pages with novel data from the web server, instead of the default technique of the browser running a completely new page. The objective is to have quicker transitions that can make the site feel more like an inherent application. Angular was invented by Google in 2010 and is a JavaScript framework. Out of these three options, Angular is the oldest. It is based on TypeScript. Owing to TypeScript, Angular is a great choice to utilize by huge teams of developers and some businesses already employ this technology in their other available products.
* It is more mature as compared to other frameworks and has a great number of contributors on GitHub. Among the consumers that exercise Angular for their single-page applications are Google and their products like Google Drive and Gmail.

1. **Explain the need and benefits of Angular?**

* AngularJS code is written in Javascript. Angular code is written in typescript.
* AngularJS code is not mobile friendly. Angular developed applications are mobile browser friendly.
* AngularJS project is difficult to manage with increasing size of the source code. Angular code is better structured, is easy to create and manage bigger applications.

1. **Demonstrate usage of JavaScript Object Notation (JSON)?**

**i] JavaScript Object - Define JavaScript object for employee**

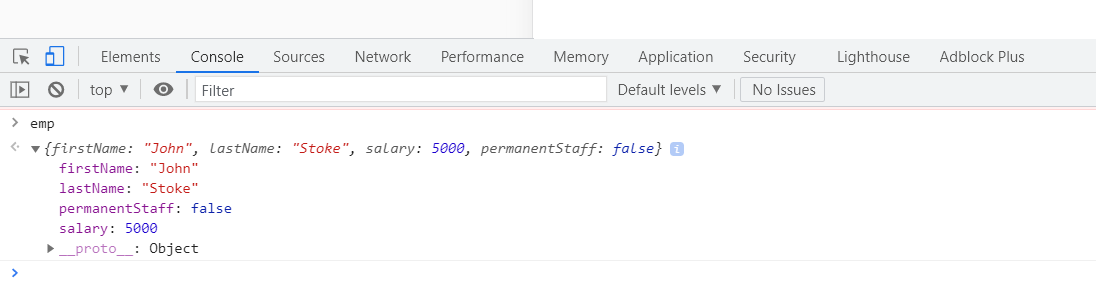
<script>

var emp = {firstName:"John",lastName:"Stoke", salary:5000, permanentStaff:false};

console.log(emp);

</script>

**OUTPUT:**



**ii] JSON - Define JSON for employee details and parse**

<script>

var emp = JSON.parse('{"firstName":"John", "lastName":"Stoke", "salary":"5000","permanentStaff":"false"}');

console.log("firstName : "+emp.firstName);

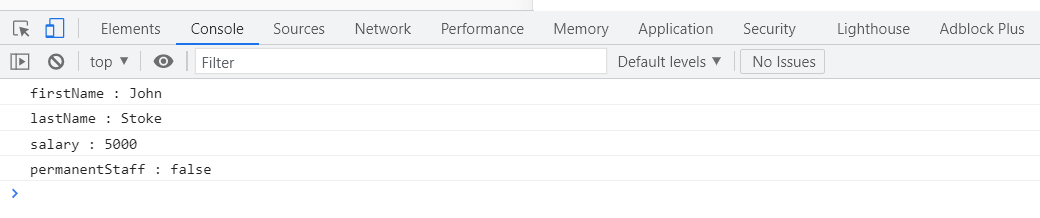
console.log("lastName : "+emp.lastName);

console.log("salary : "+emp.salary);

console.log("permanentStaff : "+emp.permanentStaff);

</script>

**OUTPUT:**



**iii] JSON - Add department details to employee**

<script>

const data = '{"firstName":"John", "lastName":"Stoke", "salary":"5000", "permanentStaff":"false", "Department":[{"id":"3","name":"payroll"}]}';

var emp = JSON.parse(data);

console.log("firstName : "+emp.firstName);

console.log("lastName : "+emp.lastName);

console.log("salary : "+emp.salary);

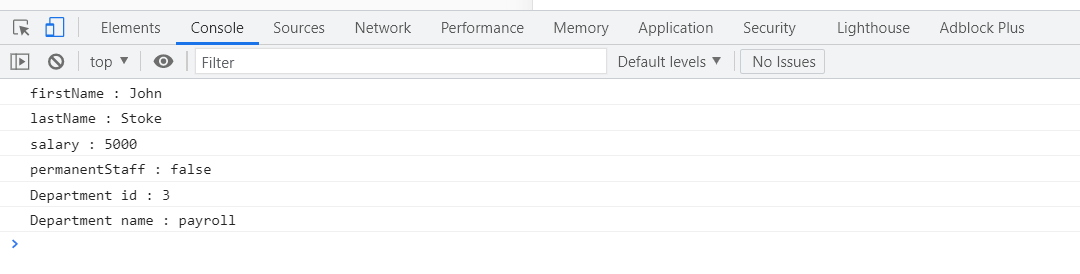
console.log("permanentStaff : "+emp.permanentStaff);

console.log("Department id : "+emp.Department[0].id);

console.log("Department name : "+emp.Department[0].name);

</script>

**OUTPUT:**



**iv] JSON - Add skill details to employee**

<script>

const data = '{"firstName":"John", "lastName":"Stoke", "salary":"5000", "permanentStaff":"false", "Department":[{"id":"3","name":"payroll"}], "skill":[{"id":"1", "value":"HTML"},{"id":"2", "value":"CSS"},{"id":"2", "value":"Javascript"}]}';

var emp = JSON.parse(data);

console.log("firstName : "+emp.firstName);

console.log("lastName : "+emp.lastName);

console.log("salary : "+emp.salary);

console.log("permanentStaff : "+emp.permanentStaff)

console.log("Department id : "+emp.Department[0].id);

console.log("Department name : "+emp.Department[0].name);

for(var i=0; i<emp.skill.length; i++)

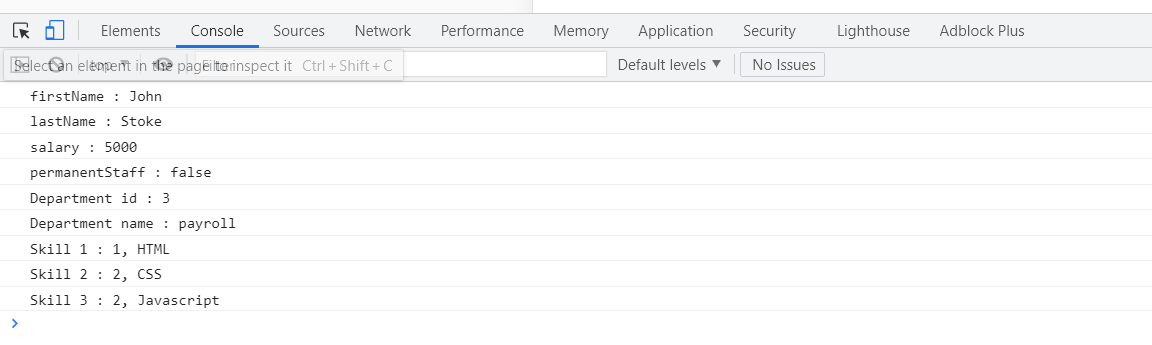
{

console.log("Skill "+(i+1)+" : "+emp.skill[i].id+", "+emp.skill[i].value);

}

</script>

**OUTPUT:**



**Session - 2**

**i] TypeScript data types and interface – Define Employee interface and display**

**Employee.ts**

export interface Employee {

id:number;

name:string;

salary:number;

permanent:boolean;

}

**Employee-test.ts**

import { Employee } from './Employee';

var emp:Employee = {

id:3,

name:"John",

salary:5000,

permanent:true,

}

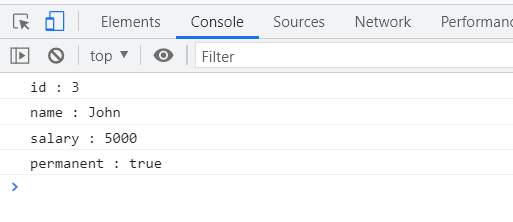
console.log("id : "+emp.id);

console.log("name : "+emp.name);

console.log("salary : "+emp.salary);

console.log("permanent : "+emp.permanent);

**OUTPUT:**



**ii] TypeScript data types and interface – Include Department details in Employee and display**

**Department.ts**

export interface Department{

    id:number;

    name:string;

    }

**Employee.ts**

import { Department } from './Department';

export class Employee implements Department{

id:number;

name:string;

salary:number;

permanent:boolean;

constructor(id:number, name:string, salary:number, permanent:boolean){

this.id = id;

this.name = name;

this.salary = salary;

this.permanent = permanent;

}

}

**Employee-test.ts**

import { Department } from './Department';

import { Employee } from './Employee';

var emp1 = new Employee(123, "John", 5000, true);

console.log("id: "+emp1.id);

console.log("name: "+emp1.name);

console.log("salary: "+emp1.salary);

console.log("permanent: "+emp1.permanent);

var dept:Department = {

id:1,

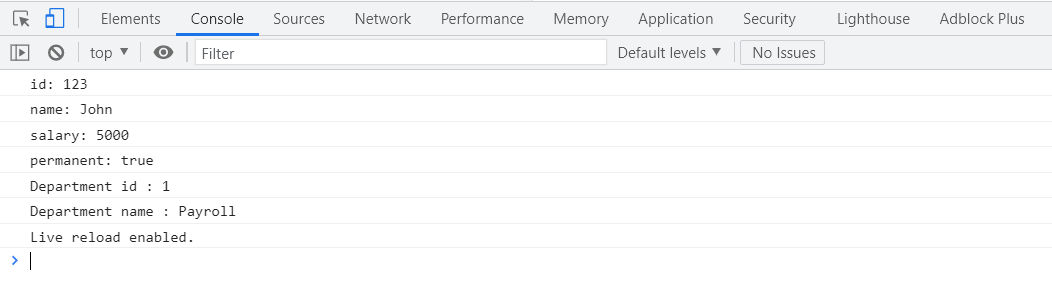
name:"Payroll",

}

console.log("Department id : "+dept.id);

console.log("Department name : "+dept.name);

**OUTPUT:**



**iii] TypeScript data types and interface – Define Employee interface and display**

**Department.ts**

export interface Department{

    id:number;

    name:string;

    }

**Skill.ts**

export interface Skill{

    id:number;

    name:string;

    }

**Employee.ts**

import { Employee } from './Employee';

import { Skill } from './Skill';

import { Department } from './Department';

var emp1 = new Employee(123, "John", 10000, true);

console.log("id: "+emp1.id);

console.log("name: "+emp1.name);

console.log("salary: "+emp1.salary);

console.log("permanent: "+emp1.permanent);

var dept:Department = {

    id:1,

    name:"Payroll",

    }

console.log("Department id : "+dept.id);

console.log("Department name : "+dept.name);

var skill1:Skill = {

    id:1,

    name:"HTML",

    }

var skill2:Skill = {

    id:2,

    name:"CSS",

    }

var skill3:Skill = {

    id:3,

    name:"JavaScript",

    }

console.log("Skill[0]:"+skill1.id+", "+skill1.name);

console.log("Skill[1]:"+skill2.id+", "+skill2.name);

console.log("Skill[2]:"+skill3.id+", "+skill3.name);

**Employee-test.ts**

import { Employee } from './Employee';

var emp1 = new Employee(1, "HTML");

var emp2 = new Employee(2, "CSS");

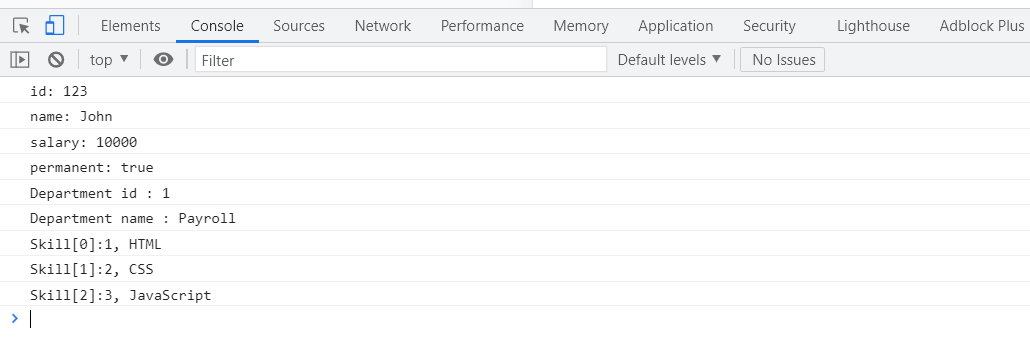
var emp3 = new Employee(3, "Javascript");

console.log("Skill 1: "+emp1.id+", "+emp1.name);

console.log("Skill 2: "+emp2.id+", "+emp2.name);

console.log("Skill 3: "+emp3.id+", "+emp3.name);

**OUTPUT:**



**iv]**

interface User {

name: string;

id: number;

}

class UserAccount {

name: string;

id: number;

constructor(name: string, id: number) {

this.name = name;

this.id = id;

}

}

const user: User = new UserAccount("Murphy", 1);