ANGULAR JS

FETCHING DATA FROM API

Safvan P
safvanparangodath12@gmail.com

CONTENT

INTRODUCTION

Angular JS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology. AngularJs is a JavaScript open source front-end framework that is mainly used to develop single page web applications (SPAs).It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML .It's features like dynamic binding and dependency injection eliminates the need of code that we have to write otherwise. AngularJs is rapidly growing and because of this reason we have different versions of AngularJs with the latest stable being 1.7.7. It is also important to note that Angular is different from AngularJs. It is an open source project which can be freely used and changed by anyone .It extends HTML attributes with Directives, and data is binded with HTML.

.

PURPOSE

The Fetch API makes it easier to make asynchronous requests and handle responses better than using an XMLHttpRequest. Fetch allows us to create a better API for the simple things, using modern JavaScript features like promises. The Fetch API is basically a modern replacement for XHR; it was introduced in browsers recently to make asynchronous HTTP requests easier to do in JavaScript, both for developers and other APIs that build on top of Fetch.

Fetch has three major benefits over the XMLHttpRequest API:

1. Ease of development. It's much easier to read and write.
2. Implements the use of promises. In short, promises allow us to control the flow of asynchronous JavaScript and dictate the order our functions run in.
3. The XMLHttpRequest API works with both the request and response to the network. Fetch exclusively works with the response so this makes things more straight forward.

DESCRIPTION

Project code is intended to manipulate information stored in JSON file.When a client request something from the server ,the might couldn't understand the request. Because the front and backend might be different languages in most cases .In this kind of situation ,the api need to work .API translates the client request into a state representational form .Then this representational data is transferred into server side and fetch the according to the state and behavior of the data .Then it transmit to client in a server side language so in between api again convert it into state and behavior format.

That format is either in XML or JSON data format .In this project we transmitting that JSON file into client side scripting language which is html.

FUNCTIONAL REQUIREMENTS

This section gives the list of Functional and nonfunctional requirements which are applicable to the Library Management System.

1.Interface Requirements

This section describes how the software interfaces with other software products or users for input or output.

2. Hardware and Software

HARDWARE SPECIFICATION:

- Processor :Intel(R) Core(TM) i3 CPU
- RAM : 1 GB
- Hard Disk : 2 GB

3.SOFTWARE SPECIFICATION:

- Operating System :Windows/ Linux
- Web Browsers : IE, Firefox, Chrome, Mozilla, Opera
- Web Server :Apache 2.0
- Front End :PHP,HTML5,JQUARY,JAVASCRIPT,AJAX,CSS3

4.SOFTWARE TOOLS USED

- HTML5
- CSS3
- BOOTSTRAP
- ANGULAR
- JAVASCRIPT

CODE

1.INDEX

Index.php

```
<html>
    <head>
        <title>Angular JS</title>
        <style>
            table, th, td {
                border: 1px solid #000000;
                border-collapse: collapse;
                padding: 5px;
            }

                table tr:nth-child(odd) {
                    background-color: #00ff00;
                }

                table tr:nth-child(even) {
                    background-color: #a5a5a5;
                }
        </style>
    </head>
    <body>
        <h2>Vehicles and Company</h2>
        <div ng-app="" ng-controller="categoryController">
            <table>
                <tr>
                    <th>ID</th>
                    <th>Vehicle Name</th>
                    <th>Company</th>

                </tr>
                <tr ng-repeat="category in categories">
                    <td>{{ category.ID }}</td>
                    <td>{{ category.Name }}</td>
                    <td>{{ category.Type }}</td>
                </tr>
            </table>
        </div>
        <script>
            function categoryController($scope, $http) {
                var url = 'sample.json';
                $http.get(url).success(function (response) {
```

```
                    $scope.categories = response;
                });
            }
        </script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular
.min.js"></script>
    </body>
    </html>
```

2.JSON

Sample.json

```
[
    {
       "ID": "001",
       "Name": "Etios",
       "Type": "Toyoto"
    },
    {
        "ID": "002",
        "Name": "Amaze",
        "Type": "Honda"
    },
    {
        "ID": "003",
        "Name": "City",
        "Type": "Honda"
    },
    {
        "ID": "004",
        "Name": "Innovo Crysta",
        "Type": "Toyoto"
    },
    {
        "ID": "005",
        "Name": "Creta",
        "Type": "Hyundai"
    },
    {
        "ID": "006",
        "Name": "i20",
        "Type": "Hyundai"
    },
    {
        "ID": "007",
        "Name": "Activa",
        "Type": "Honda"
    },
```

```
{
    "ID": "008",
    "Name": "DUKE",
    "Type": "KTM"
},
{
    "ID": "009",
    "Name": "DIO",
    "Type": "Honda"
},
{
    "ID": "010",
    "Name": "RC 390",
    "Type": "KTM"
}
]
```

3. OUTPUT

Vehicles and Company

ID	Vehicle Name	Company
001	Etios	Toyoto
002	Amaze	Honda
003	City	Honda
004	Innovo Crysta	Toyoto
005	Creta	Hyundai
006	i20	Hyundai
007	Activa	Honda
008	DUKE	KTM
009	DIO	Honda
010	RC 390	KTM