Client Side Framework

Angular,React,Vue

Single Page Application

Angular

- Angular is an open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. It is a complete rewrite from the same team that built
- Angular is a component-based framework which uses declarative HTML templates. At build time, transparently to developers, the framework's compiler translates the templates to optimized JavaScript instructions.

Vue

- First released Vue in 2014, after working on and learning from the original AngularJS project. Vue is the youngest of the big four, but has enjoyed a recent uptick in popularity.
- Vue, like AngularJS, extends HTML with some of its own code. Apart from that, it mainly relies on modern, standard JavaScript.
- Now it started using Typescript

React

- originally developed and maintained by Facebook, React is an open-source, declarative, front-end, “learn once, write anywhere” JavaScript library aimed at helping web developers build user interfaces specifically for single-page or mobile apps. React is simple to use, scalable and fast.

CLI-Setup

- Pre-requisite for any kind of client side framework
  - Node Latest Version should be installed.
- Angular: npm install -g @angular/cli
- Vue: npm install -g @vue/cli
- React: npm install -g react-cli react

Npm

- npm stands for node package manager. It allows for seamless node.js package management. You can install, share and manage node.js packages.
- Package.json file in our project contains npm command to build and run the project

Project Creation & Run

Angular

```
ng new projectname
cd projectname
npm start
```

Vue

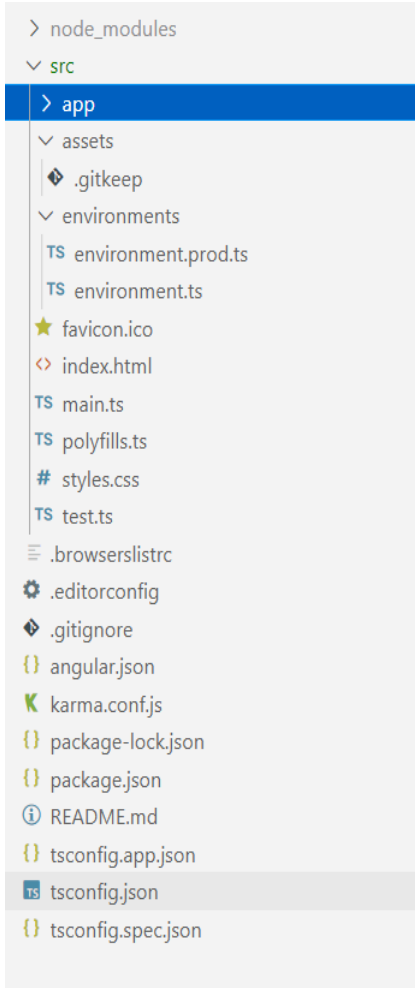
```
vue create projectname
cd projectname
npm run serve
```

React

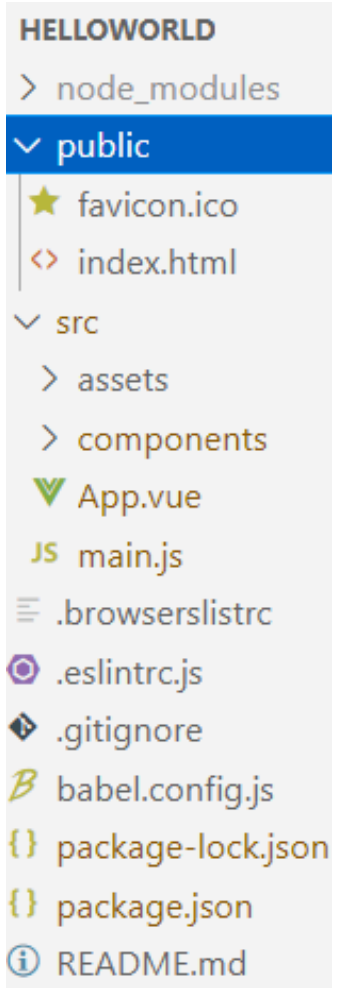
```
npx create-react-app projectname
cd projectname
npm start
```

Project Structure

Angular



Vue



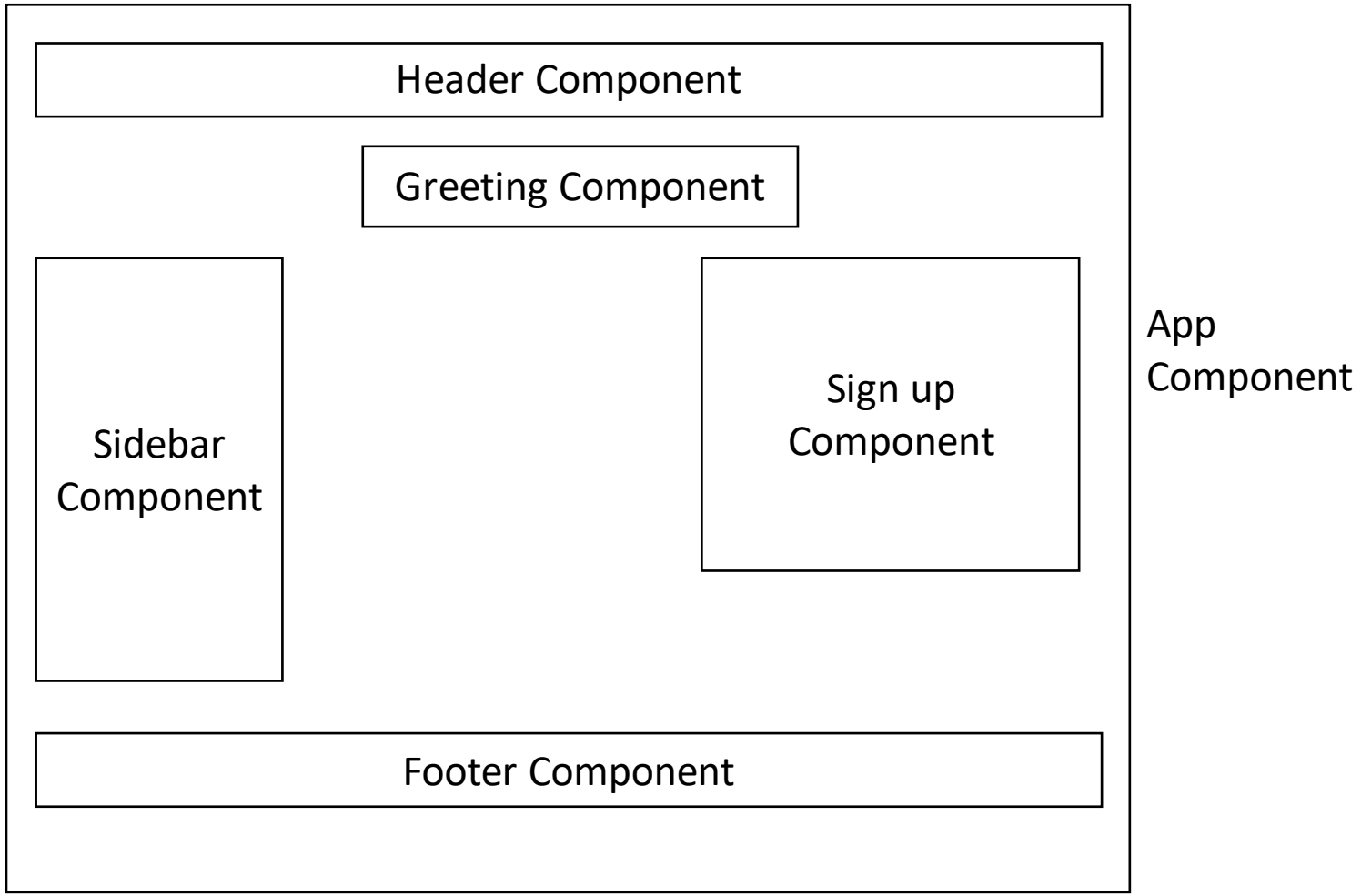
React

Component

Components are a logical piece of code

A component consists of three primary elements:

- The HTML template
- The logic
- The styling (CSS, Sass, Stylus, etc..)
- By Default App Component is created in (Angular,React,Vue)



Component

Angular

ng generate component greeting

```
<p>greeting works!</p>
<h1 class="blue">{{message}}</h1>
```

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-greeting',
  templateUrl: './greeting.component.html',
  styleUrls: ['./greeting.component.css']
})
export class GreetingComponent implements OnInit {

  message="Greeting of the Day "+ new Date().toDateSt
ring();

  constructor() { }

  ngOnInit(): void {
  }

}
```

```
.blue{color:blue;}
```

Vue

```
<template>
<h1 class="blue">{{message}}</h1>
</template>
<script>
export default {
  name: 'Greeting',
   data() {
    return {
      message:"Greeting of the Day "+ new Da
te().toDateString()
    }},
}
</script>

<!--
 Add "scoped" attribute to limit CSS to this
 component only -->
<style scoped>
.blue
{
  color:blue;
}
</style>
```

React

```
import './Greeting.css';

var message="Greeting of the Day "+ n
ew Date().toDateString();

function App() {
  return (
    <div>

      <p>greeting works!</p>
      <h1 class="blue">{message}</h1>

    </div>
  );
}

export default Greeting;
```

```
.blue{color:blue;}
```

Data Binding

Angular	Vue	React
• Interpolation {{}} • Property Binding[] • Two Way Binding [(ngModel)]	• Interpoloation{{}} • Two Way Data Binding(v-model)	• Interpoloation{} • Two Way Data Binding(State)

Conditional Constructs

Angular

```
currentHour=new Date().getHours();
```

```
<div *ngIf="currentHour>=5 && curren
tHour<12">
<h1>Good Morning</h1>
</div>
<div *ngIf="currentHour>=12 && curre
ntHour<17">
<h1>Good Afternoon</h1>
</div>
<div *ngIf="currentHour>=17 && curre
ntHour<=24">
<h1>Good Evening</h1>

</div>
```

Vue

```
 data() {
   return {
     message:"Greeting of the Day "+ new
Date().toDateString(),
     currentHour:new Date().getHours()
   }},
```

```
<template>
<div>
<h1 class="blue">{{message}}</h1>

<div v-if="currentHour>=5 && currentHour<12">
<h1>Good Morning</h1>
</div>
<div v-if="currentHour>=12 && currentHour<17">
<h1>Good Afternoon</h1>
</div>
<div v-if="currentHour>=17 && currentHour<=24">
<h1>Good Evening</h1>
</div>
</div>
</template>
```

React

```
function App() {
  const currentHour=new Date().getHours()
  return (

    <div>

    {(currentHour>17) && <div>Good Evening
</div>}
    {(currentHour>12) && <div>Good Afterno
on</div>}
    {(currentHour>=5) && <div>Good morning
</div>}


      </div>

  );
}
```

Iterating the collection

Angular

```
employeeList=[{ID:1,Name:"Reena",Address:"Ahmedabad"},{"ID":2,Name:"Meghna",Address:"Mumbai"},{ID:3,Name:"Ritu",Address:"Pune"}]
```

```
<table class="table">
     <tr *ngFor="let item Of employeeList">
             <th scope="row">{{item.ID}}</th>
              <td>{{item.Name}}</td>
              <td>{{item.Address}}</td>
       </tr>
</table>
```

Vue

```
data() {
    return {
      employeeList:[{ID:1,Name:"Reena",Address:"Ahmedabad"},{"ID":2,Name:"Meghna",Address:"Mumbai"},{ID:3,Name:"Ritu",Address:"Pune"}]
    }},
```

```
<table class="table">
          <tr v-
for="item in employeeList" :key="item.message"
>
                <th scope="row">{{item.ID}}</th>
                <td>{{item.Name}}</td>
                <td>{{item.Address}}</td>
              </tr>
    </table>
```

React

```
function App() {
   let employeeList=[{ID:1,Name:"Reena",Address:"Ahmedabad"},{"ID":2,Name:"Meghna",Address:"Mumbai"},{ID:3,Name:"Ritu",Address:"Pune"}]
    const list = employeeList;
    const listItems = list.map((listItem)
=>
    <tr  key={listItem.ID.toString()}>
    <td>{listItem.ID}</td>
    <td>{listItem.Name} </td>
    <td>{listItem.Address}</td></tr>

    );
  return (
          <table>{listItems}</table>

   );
}
export default App;
```

Passing Values from Parent Component to Child component

Angular	Vue	React
•@input	•props	•props

Example Angular

```
<!-- calling app-greeting component and passing UserName --
>
<input type="text" [(ngModel)]="Name">
<app-greeting [UserName]="Name"></app-greeting>

 //Greeting Component ts file
 import { Component, Input, OnInit } from '@angular/core';
 @Input() UserName:string=''

<!-- Greeting Component -->
<h1>Hello {{UserName}}</h1>
```

Example Vue

```
<!-- App Component-->
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <!--calling Hello World Component and passing name and msg property-->
    <HelloWorld v-bind:name="name" msg="Welcome to Your Vue.js App"/>
    <input type="text" v-model="name">
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
   data() {
    return { name: "megha" }
  }
  ,
  components: {
    //registring component
    HelloWorld
  }
}
</script>
```

```
<!-- Hello World Component-->
<template>
  <div class="hello">
  <!-- interpolate props -->
    <h1>{{ msg }} {{name}}</h1>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  //registring props
  props: {
    msg: String,
    name:String
  }
}
</script>
```

Example React

```
import './App.css';
//Hello world Component
function HelloWorld(props)
{
  return <h1>Hello, {props.name}</h1>
}

function App() {
  var name="megha"
   return (
     <div className="App">
       {/* invoking hello world component and passing
 name props*/}
       <HelloWorld name={name}></HelloWorld>
      </div>
  );
}

export default App;
```

Passing values from Child Component to Parent Component

Angular	Vue	React
• Output, EventEmitter	• $Emit	• Props-Parentcallback

Angular Example(passing values from child component to a parent component

Parent Component

```
<h1>Total Items in the cart {{cartitems.length}} </h1>
<!-- Event Binding Child Component will invoke the cartevent pass data to cartsevent function of parent component -->
<app-products (cartnotify)="cartsevent($event)"></app-products>

cartitems:Array<any>=[];
cartsevent(cartitems:Array<any>)
{
   this.cartitems=cartitems;
   // console.log(cartitems.length)
}
```

Angular Example(passing values from child component to a parent component

```
<table>
   <tr border="3" *ngFor="let item of productList">
       <td>{{item.ID}}</td>
       <td>{{item.Name}}</td>
       <td><img height="100" width="100" src="{{'../../assets/' + item.image}}"/></td>
       <td><button (click)=addtocart(item) value="Add to cart">Add to Cart</button></td>
    </tr>
</table>
```

```
//import output and eventemitter
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
 productList=[
{ID:1,Name:"keyboard",Price:600,image:"keyboard.jpg"},
{"ID":2,Name:"Mouse",Price:500,image:"mouse.jpg"},
{ID:3,Name:"HeadPhone",Price:700,image:"headphone.jpg"}]

  cartlist:Array<any>=[]

  @Output() cartnotify = new EventEmitter<any>();

addtocart(item:any)
  {
    this.cartlist.push(item);
    console.log("-----",this.cartlist);
    this.cartnotify.emit(this.cartlist);
  }
```

Vue Example(passing values from child component to a parent component

Parent Component

```
<!-- App Component-->
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <h1>Total Items in the Cart {{totalitems}}</h1>
    <!--
Product component fire an event cartnotify and pass
the data to carts event function-->
    <Products  @cartnotify="cartsevent($event)" >
</Products>
  </div>
</template>
```

```
<script>
//importing child component
import Products from './components/Products.vue'
export default {
  name: 'App',
   data() {
    return {cartlist:[] }
  },
   computed: {
    // a computed getter
    totalitems: function () {
       return this.cartlist.length;
    }
  },
  components: {
    //registring component
        Products
  },
  methods:{
    cartsevent(data)
    {
      this.cartlist=data;
    }
  }
}
</script>
```

Vue Example (Passing value from child component to a parent component

Child Component

```
<!-- Product Component-->
<template>
  <div class="hello">
<table>
   <tr border="3" v-
for="item in productList" :key="item.ID">
       <td>{{item.ID}}</td>
       <td>{{item.Name}}</td>
       <td>
<button  v-
on:click=addtocart(item) value="Add to cart"
>Add to Cart</button></td>
    </tr>
</table>
  </div>
</template>
```

```
<script>
export default {
  name: 'Products',
     data() {
    return { name: "megha",
   productList:[{ID:1,Name:"keyboard",Price:600},{"ID":2,Name:"Mouse",
Price:"700"},{ID:3,Name:"headphone",Price:500}],cartlist:[]}
    },
  methods:{
  addtocart(item)
  {
    this.cartlist.push(item);
    console.log("-----",this.cartlist);
    //raise an event cartnotify after appending element to cart list a
nd sending the same data to cartnotify
    this.$emit("cartnotify",this.cartlist);
  }
  }
}
</script>
```

React Example Passing values from child to parent

Parent Component

```
import './App.css';
import './Products'
import Products from './Products';
import React, { useState } from 'react';
function App() {
  const [childcartlist, setCartList] = useState([]);
  function handleCallback(cartlist)
  {
  setCartList(cartlist);
  }
  return (
    <div className="App">
      <h1>Total Items in Cart {childcartlist.length}</h1>
      <Products childnotify = {handleCallback}></Products>
   </div>
  );
}

export default App;
```

React Example Passing values from child to parent

```
import React, { useState } from 'react';
function Products(props) {
    const [cartlist, setCartList] = useState([]);
    let productList=[{ID:1,Name:"keyboard",Price:600},{ID:2,Name:"Mouse",Price:"700"},{ID:3,Name:"headphone",Price:500}]
    const listproduct =productList;
    const productlistItems =listproduct.map((listItem) =>
      <tr  key={listItem.ID.toString()}>
        <td>{listItem.ID}</td>
        <td>{listItem.Name} </td>
        <td>{listItem.Price}</td>
        <td> <button onClick={(e) => onTrigger(listItem, e)}>Add to cart</button></td>
      </tr>
    );
function onTrigger(data,event)  {
    event.preventDefault();
    setCartList([...cartlist, data]);
    console.log(cartlist);
    props.childnotify(cartlist);
}
    return (
        <div>
            <table><tr><td>ID</td><td>Name</td><td>Price</td></tr>{productlistItems}
     </table>
      </div>

    );
}
export default Products;
```

Forms(Handling User Input)

Angular	Vue	React
• Template Driven • Reactive Forms • Dynamic Forms	• V-model	• Controlled(State) • Uncontrolled(ref)

Reference

- **Angular**
  - Template Driven Forms
    - https://angular.io/guide/forms
  - Reactive Forms
    - https://angular.io/guide/reactive-forms
  - Dynamic Forms
    - https://angular.io/guide/dynamic-form
- Vue
  - https://vuejs.org/v2/guide/forms.html
- React
  - https://reactjs.org/docs/forms.html

Routing

- In a Single Page Application (SPA), all of your application's functions exist in a single HTML page. As users access your application's features, the browser needs to render only the parts that matter to the user, instead of loading a new page. This pattern can significantly improve your application's user experience.
- To define how users navigate through your application, you use routes. Add routes to define how users navigate from one part of your application to another. You can also configure routes to guard against unexpected or unauthorized behavior.

Routing

Routing Packages		
Technology	**Name**	**Remarks**
Angular	@angular/router	By Default included
Vue	vue-router	installation required
React	react-router-dom	Installation required

Routing Registration		
Technology	**Path**	**Place**
Angular	const routes: Routes = [{path:'/',component:ComponentName}];	@NgModule({ imports: [**RouterModule.forRoot**(routes)], exports: [RouterModule] })

Vue	const routes = [
	{ path: '/foo', component: Products },
	]
		`const router = new VueRouter({`
		`routes`
		`})`
		Vue.use(VueRouter);
		new Vue({
		render: h => h(App),
		router
		}).$mount('#app')

React	`<Route exact path='/' component={Home}></Route>`	`<Router>`
		`<div className="App">`
		`</div>`
		`</Router>`

Routing

Routing Directives		
Technology	**Name**	**Remarks**
Angular	Router-outlet	used by the router to mark where in a template, a matched component should be inserted
Vue	router-view	used by the router to mark where in a template, a matched component should be inserted
React	switch	To render a single component, wrap all the routes inside the Switch Component

Routing Navigation		
Technology	**Name**	**example**

Angular	routerLink	`<a [routerLink]="['/home']">Home</a>`
		`<a [routerLink]="['/Product', '2']">{{product.name}} </a>`
		`<a [routerLink]="['/Product', ['/Product', product.productID]"]>{{product.name}} </a>`

Vue	router-link	`<router-link to="/student">Go to Bar</router-link>`

React	Link	<Link to="/">Home</Link>
		or
		`<Route`
		`exact`
		`path="/course/:cname"`
		`render={(props) => (`
		`<Course`
		`{...props}`
		`setLoginStatus={setLoginStatus}`
		`loginStatus={loginStatus}`
		`/>`
		`)}`

Routing

Programmatically Routing Navigation		
Technology	**Name**	**example**
Angular	Inject router module in a specific component	`this.router.navigate(["/student",item.StudentID])`
Vue	push	`this.$router.push({ name: 'student', params: { item.StudentID } })`
React	useHistory Push Method	const history = useHistory(); history.push("/home");

Subscribing Routes

Subscribing Routes		
Technology	**Name**	**example**

Angular	Inject ActivatedRoute then subscribe params and data based on the requirement	`constructor(private route:ActivatedRoute){}`
		`ngOnInit() {`
		`this.route.params.subscribe(params => {`
		`this.route.data.subscribe(data => {`
		`console.log(params.id,data.name)`
		`})`
		`})`

Vue	$route	`this.$route.params.userId`
React	useLocation	`import { useLocation } from 'react-router-dom';` `const { search } = useLocation();`

Route Auth Guard

Route guards are interfaces which can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class which implements the given guard interface.

Route Guard		
Technology	**Name**	**example**
Angular	CanActivate from @angular/core	https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3
Vue	Router.beforeEach function	https://router.vuejs.org/guide/advanced/navigation-guards.html#global-before-guards `router.beforeEach((to, from, next) => { if (to.name !== 'Login' && !isAuthenticated) next({ name: 'Login' }) else next() })`
React	`react-router-guards`	GuardProvider, GuardedRoute https://www.npmjs.com/package/react-router-guards?activeTab=readme

Group a list of children without adding extra nodes to the DOM

DOM appending	
Technology	**Name**
Angular	<ng-container> </ng-container>
Vue	<Fragement> </Fragement>
React	<React.Fragement> </React.Fragement>

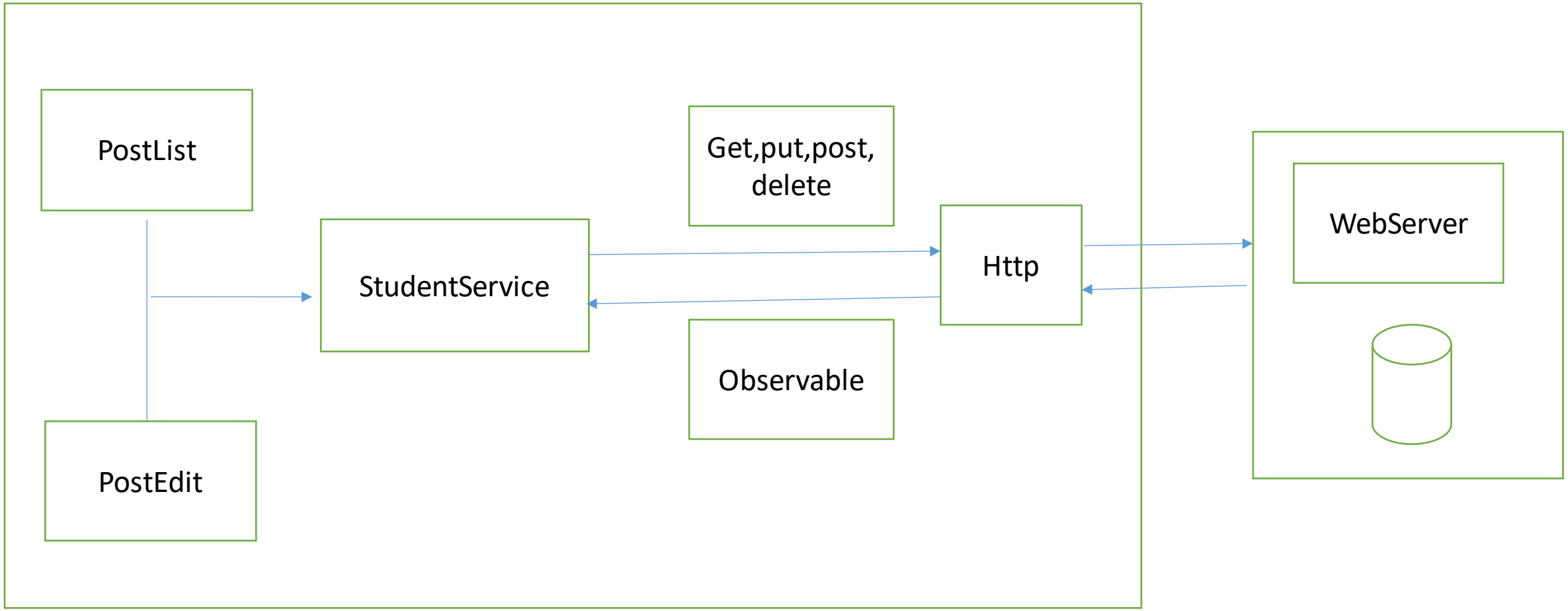
Pass down Html elements from one component to another

Technology	Name	Example	
Angular	<ng-content> </ng-content>	<!-- App component --> <app-hello-world> Hello World </app-hello-world>	<!-- Hello World component --> <ng-content></ng-content>
Vue	<slot></slot>	<!-- App component --> <template> <HelloWorld><**div**>**Welcome Everyone**</**div**></HelloWorld> </template>	<!-- Hello World component --> <template> <**slot**></**slot**> </template>

React	props.children	import React from 'react';
		import HelloWorld from './HelloWorld';
		export default function App() {
		return (
		
		<HelloWorld>
		<**h1**>**Welcome to the Application**</**h1**>
		</HelloWorld>
		
		);
		}
			<!-- Hello World component -->
			import React from 'react';
			
			export default function HelloWorld(props) {
			return {**props.children**} ;
			}

HTTP

Http Mechanism



Topics

- Introduction to HTTP requests.
- Using API to make request.
- Using HttpClient/axios API to make HTTP request

Introduction to HTTP requests

- An HTTP request is a packet of information which is transferred from source to destination and termed as Client-Server respectively. A client makes the HTTP request, and the server sends the information to the client. HTTP has multiple request methods to make the request and error to indicate the error.

HTTP Request Methods

- OPTIONS :his is used to check all metadata the server requires to ensure a secure request like CORS. It is called automatically before any HTTP request. You can read more about OPTIONS .
- GET: This is used to get data from the server.
- POST: This is used to send data to the server.
- PUT: This is used to update data.
- DELETE: This is used to delete data from the server

Response Status Codes

- 200: This is used to indicate the "OK" message when the HTTP request has completed successfully.
- 404: This is used to indicate a "Resource NOT Found" message when HTTP doesn't find the specified URL at the server.
- You can read more about HTTP Response codes and HTTP request methods .

Using APIs to Make an HTTP Request

- There are two main types of APIs used to make an HTTP request.
  - XMLHttpRequest(XHR)
  - Fetch

  Other libraries like Axios and HttpClient use one of the above internally and abstract the complexities around these APIs.

    - You can read more about XMLHttpRequest and Fetch.

Communicating with backend services			Reference
Technology	**PackageName**	**Install**	
Angular	@angular/common/http Module: HttpClientModule Inject Service: HttpClient	Yes	
Vue	axios	Yes	
React	axios	yes	

Life Cycle Hooks of Angular

- Every component in Angular has its own lifecycle events that occurs as the component gets created, renders, changes it's property values or gets destroyed. Angular invokes certain set of methods or we call them hooks, that gets executed as soon as those lifecycle events gets fired.
- Lifecycle hooks are wrapped in certain interfaces which are included in the angular core **'@angular/core'** library.

Life Cycle Events

- One thing to note that each of these interfaces includes one method whose name is same as of the interface name but it is just prefixed by **"ng"**. For example **OnInit** interface has one method **ngOnInit()**. The following lifecycle hooks are exposed by Angular corresponding to different lifecycle events.
- ngOnChanges()
- ngOnInit()
- ngDoCheck()
- ngAfterContentInit()
- ngAfterContentChecked()
- ngAfterViewInit()
- ngAfterViewChecked()
- ngOnDestroy()
- https://angular.io/guide/lifecycle-hooks