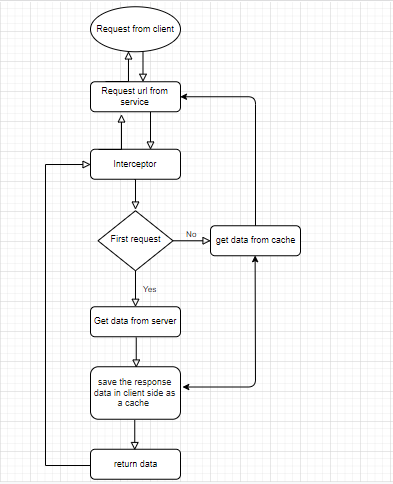
**Caching in angular**

Caching is a very handy feature to reduce the server-side network traffic. Caching is a temporary storage and can reduce the server requests by saving and retrieving the data in client-side itself. Hence, we will get more performance in client application.

Different ways to achieve Caching in angular,

**1. Achieve the client-side caching through HTTP interceptors.**

HTTP interceptor in Angular application and intercept all the requests to server. When we request to server first time, interceptor will save the response data in client side as a cache and when user requests the same URL again, it will not go to server again. Instead, the client data will be served from cache. But when we modify the data, interceptor will invalidate the client-side cache and send request to server and response data will be saved to cache again.



**Data bindings in angular**

Data Binding is the important concept of Angular. It Allow us to define the communication between component and view.Data Binding is passed from component to view and from view to the component.

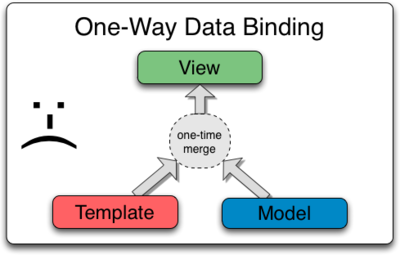
**Types:**

1. String Interpolation
2. Property Binding,
3. Event Binding,
4. Two Way Data Binding.

**1.String Interpolation:**

This data binding is one-way in the sense that, the Model’s property can only updated from the Model side and the View is updated or the Model’s variables are kept up to date from the View. The data flow is in one direction either from Model-to-View or View-to-Model. One-way data binding is unidirectional. It is used to pass data from model class to the template. syntax :**{{propertyname}}.**

export class AppComponent {  
 items= {  
 title: 'Foot Ball',  
 price: 700  
 };  
 }<h1>Product</h1>  
 <h2>Title : **{{**product.title**}}**</h2>  
 <h2>Price : **{{**product.price**}}**</h2>



2.Property Data Binding:

Interpolation is a special syntax that Angular converts into property binding [pair of square bracket]. It’s alternative to property binding.

Property binding is which will help to bind values to the properties of HTML elements.

Syntax: property[value]

@Component({

selector: 'app-example',  
 template: `  
 <div>  
 <input [value]='myBlog'></span>   
 </div>`

});

export class AppComponent {  
 myBlog: string = "My First Angular Blog";  
}

## 3.Event Data Binding:

A user expects a UI to respond to her/his actions on the page. Every such action would trigger an event on the page and the page has to respond by listening to these events like clicks, keystrokes, change events, etc.

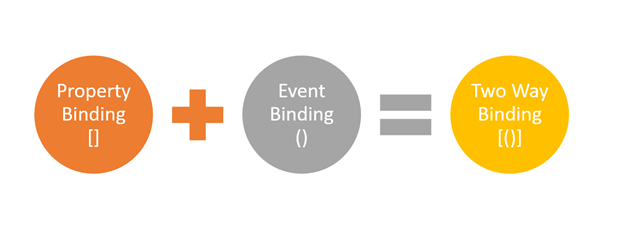
Angular gives you third type of binding to capture events raised on template in a component class. For (e.g), there’s a button on the component template and, on click of the button, you want to call a function in component class. You can do this using Event Binding.

Events are actions which occur as a result of the user or another source.

**syntax:(eventname)**

export class AppComponent {  
   
 addProduct() {  
 console.log('add items');  
 }  
   
 }<h1>Product</h1>  
<button (click)='additems()'>  
 Add Product  
</button>

4.Two way data binding:



The combination of property binding and the event binding is called the two way data binding. two-way databinding, automatic synchronization of data happens between the Model and the View.change is reflected in the both components.Whenever you make changes in the model it will reflected immedietly in the view component.

Angular provides us a directive, **ngModel,** to achieve two-way data binding.

## syntax: [(ngModel)] = “[property of your component]”

Everyone probably familiar with [(ngModel)] and its world-famous “banana in a box” .

Two-way data binding is mainly used in data entry forms where the user changes the view and makes changes in the model with the view data and vice-versa.

One important thing is NgModel is not a part of Angular's code library, it is defined in the forms module library so you need to import the FormsModule library in your app.module.ts file.

Two-Way Data binding example...  
import {Component} from '@angular/core';

@Component({

moduleId:module.id,

selector:'my-app',

template:`<div class="container"><input [(ngModel)]='name' /><br/><h1>Hello {{name}}</h1></div>

})

export class AppComponent{}

**Caching in angular implementation**

We have created all the components.

Angular service

**export** **class** SamleService {

**private** sampleUrl = 'http://localhost:3333/api/sampledata';

  constructor(**private** http: HttpClient) { }

  getAllSampledata(): Observable<Employee[]> {

**return** **this**.http.get<sample[]>(**this**.sampleUrl).pipe(

);

}

We can create a HttpCache service to implement client-side caching.

**export** **class** HttpCacheService {

put(url: string, response: HttpResponse<any>): **void** {

**this**.requests[url] = response;

 }

get(url: string): HttpResponse<any> | undefined {

**return** **this**.requests[url];

 }

invalidateCache(): **void** {

**this**.requests = { };

}

}

added “put”, “get”, and “invalidateCache” methods inside above service. “put” method will add http response to a local variable. “get” method will get the previously saved response value from cache. “invalidateCache” method will destroy the cache value from variable. We will call this cache service from http interceptor.

We can create the interceptor manually CacheInterceptor .

@Injectable()

**export** **class** CacheInterceptor **implements** HttpInterceptor {

}

We have implemented “HttpInterceptor” interface inside this interceptor class. This interface has a single “intercept” method. This will be used to check each request from browser. If the request is not a Get method, this will invalidate the existing cache. When we request the same URL again, interceptor will get the value from cache and return to response. This way, we can reduce the request to server.

// pass along non-cacheable requests and invalidate cache

**if**(req.method !== 'GET') {

     console.log(`Invalidating cache: ${req.method} ${req.url}`);

**this**.cacheService.invalidateCache();

**return** next.handle(req);

  }

// attempt to retrieve a cached response

**const** cachedResponse: HttpResponse<any> = **this**.cacheService.get(req.url);

// return cached response

**if** (cachedResponse) {

     console.log(`Returning a cached response: ${cachedResponse.url}`);

   console.log(cachedResponse);

**return** of(cachedResponse);

   }

// send request to server and add response to cache

**return** next.handle(req).pipe(

     tap(event => {

**if** (event **instanceof** HttpResponse) {

console.log(`Adding item to cache: ${req.url}`);

**this**.cacheService.put(req.url, event);

 }

 })

);

Html

Component

Cache Interceptor

Service

Checking httprequest type

Other than Get request

Response from server

Is exist url cache data

Get

False

Saving data in cache

API Server

Clear all cache data

Cached data

True