**Common folder**

- User-data.interface.ts
- Initial-data.service.ts

**A folder**

- A.component.html
- A.component.ts
- A.component.css or scss depending on your gulp/webpack preprocessing step setup
- A.service.ts

**B folder**

- B.component.html
- B.component.ts
- B.component.css or scss depending on your gulp/webpack preprocessing step setup
- B.service.ts

**A.component.scss**

```
.wh-fnd-a-component {
    display: flex;
   h5 {
      color: blue;
   }
   .wh-fnd-b-as-child-of-a {
      background-color: green;

   }
}
```

**A.component.html**

```
<div class="wh-fnd-a-component">
        <h5>This is parent component A</h5>
        <span class="header header-col-3">{{ x }}</span>
        <div class="header header-col-3" [innerText]="x"></div>
        <section class="header header-col-3" [innerHTML]="x"></section>
        <div class="child-holder">
           <wh-fnd-b class="col-xs-6 wh-fnd-b-as-child-of-a"
                     [input1]="5"
                     [input2]="'abcde'"
                     [input3]="address"
                     [input4]="computeMailingAdress()"
                     (onUserSignupComplete)="handleNewUser($event)">
           </wh-fnd-b>
        </div>
        <div class="child-holder">
```

```
            <c></c>
        </div>
</div>
```

**A.component.ts**

```
import { UserData } from '../common/user-data.interface';
import { AService } from './A.service';

@Component({
   selector: 'wh-fnd-a',
   templateUrl: './A.component.html',
   providers: [ AService ]
})
export class A {
   private x: number;
   private y: string;
   private z: any;
   private address: string;

   constructor(private aSer: AService) {
      this.x = 0;
      this.y = '';
      this.z = null;
      this.address = 'hsrlayout';
      this.doDataInit();
   }

   doDataInit(): void {
      this.aSer
         .downloadDataNeededByAComponent()
         .subscribe( (uData: UserData) => {
            this.x = uData.phoneNum;
            this.y = uData.lastName;
            this.address = uData.mailingAddress;
         } );
   }

   computeMailingAdress(): string {
      return this.x.toString() + this.y + this.z.toString() + this.address;
   }

   handleNewUser(incomingData: UserData) {
      console.info('inside parent component A: handleNewUser() method: child B has sent
me this data: ', incomingData);
   }
```

```
}
```

**A.service.ts**

```
@Injectable()
export class AService {
   constructor(private http: Http) { // ng5 varaku Http, ng6 onwards HttpClient
   }

   downloadDataNeededByAComponent(): Observable<UserData> {
      return this.http.post('url',
         {name: 'username', cookie: 'timestamp'});
   }

   downloadInitialData(): Observable<SomeOtherData | any> {
      return this.http.get('urlwithparams');
   }
}
```

———

**B.component.html**

```
<h5>This is child component B</h5>
<input type="text" name="firstName" #firstName />
<button (click)="handleUserClick($event)">sign up</button>
```

**B.component.ts**

```
import { EventEmitter, Component, Input, Output } from '@angular/core';

@Component({
   selector: 'wh-fnd-b',
   templateUrl: './B.component.html'
})
class B {
   // declarations of all data members
   @Input private input1: number;
   @Input private input2: string;
   @Input private input3: string;
   @Input private input4: string;

   @Output private onUserSignupComplete: EventEmitter<UserData> = new
EventEmitter<UserData>();

   @ViewChild('firstName') private firstNameInputField;
```

```
    // method members (declarations + definitions)
    constructor() {
        console.log('I am inside child component B: ', this.input1, this.input2, this.input3);
        // all are undefined, unless defined at place of declaration
    }

    ngOnInit() {
        console.log('I am inside child component B: ', this.input1, this.input2, this.input3);
        // 5, 'abcde', 'hsrlayout'
    }

    handleUserClick(event: MouseEvent) {
        let newlyCreatedUserData: UserData = new UserData();
        newlyCreatedUserData.firstName = this.firstNameInputField.value;
        this.onUserSignupComplete.emit(newlyCreatedUserData);
    }
}
```

**Parent to Child communication -> one way -> @Input**

Reverse lo Child to Parent communication -> one way, but in reverse order -> @Output

---

**user.data.ts**

```
interface UserData {
    private firstName: string;
    private middleNames: Array<string>;
    private lastName: string;

    private phoneNum: number;

    private zipCode?: string;

    private mailingAddress: string;

    private permanentAddress: string;
}

{
    firstName: 'jag',
    middleNames: ['j', 'r'],
    lastName: 'rp',
    phoneNum: +91839257982,
    ....
}
```

**Gmail app to be implemented:**

Header bar
Tabs-> Primary: list of mail -> max. 2 mails per thread
Basic Chat box,
Inbox, important, starred