Question 1 - Part A (5%): Please explain Interpolation in the context of Angular. Please provide a functional example to elaborate on your explanation.

**Answer:**

Interpolation offers us a means to incorporate calculated strings into the text between HTML element tags and within attribute assignment.

Example:

In the foo.component.ts, we have a string variables called studentName, with value “Jason Bourne” initialized.

*export* class FooComponent implements OnInit {

studentName: string = "Jason Bourne";

In the foo.component.html, when we want to use this variable, we can use interpolation here. The double curly bracket with variable name will make “Jason Bourne” displayed.

<h1>Student Name: {{ studentName }}</h1>

Question 1 - Part B (10%): Please compare Interpolation with PropertyBinding. What are the differences between the two techniques. Please provide a functional example to elaborate on your explanation.

Answer:

In some cases, Interpolation and Property Binding are the same thing. They can pass the value from the .ts file to .html file. And Interpolation is better in terms of readability. However, the Property Binding will allow property passed from parent components to child components via square brackets. And when a property is a non-string data value, we must use Property Binding.

Example:

In the parent.component.ts file, we have a property called herName, with string “Lucy” initialized.

*export* class ParentComponent implements OnInit {

public herName:string = "Lucy";

In the parent.component.html file, we pass the property herName to the child FooComponent, with the property called name in the square bracket.

<app-foo [photos]="studentPhotos" [name]="herName" (childEvent)="message=$event"></app-foo>

Then in the foo.component.ts, we use @Input to receive the name property

*export* class FooComponent implements OnInit {

@Input() name:string;

So, in the foo.component.html, we can use this property from the parent component.

<h1>Name: {{name}}</h1>

Question 2 (20%): With a fully functional example, please explain Emitting Events in the context of Angular. Please explain how the same is done in React with a small example.

Answer:

Emitting Event will make the data passing from child components to parent component.

Example:

In the child component, we create an Output for a childEvent. We initialize the EventEmitter firstly here.

*export* class FooComponent implements OnInit {

@Output() public childEvent = new EventEmitter();

In the child html file, we have a button with click event binding with an action. The purpose of this action is to send info back to the parent component.

<button (click)="fireEvent()">Send Event</button>

In the child ts file, we have this fireEvent, which will emit a string to the parent component.

fireEvent(){

this.childEvent.emit("Hi! Good Evening!")

}

In the parent html file, we can receive the childEvent info, and save it to a variable called message.

<app-foo [photos]="studentPhotos" [name]="herName" (childEvent)="message=$event"></app-foo>

Of course, we need to firstly create the message variable in the parent component.

*export* class ParentComponent implements OnInit {

public message:string = "";

So, in the parent html file, we can receive the message info and display it when we click the button in the child component.

<h1>{{message}}</h1>

In react world, we can achieve the same effect via useState.

The data can also be passed from Child to Parent easily.

In the parent component, App.js, we have message state, which can be set in Child component.

function App() {

const [message,setMessage] = useState("");

*return* (

<div >

<h1>{message}</h1>

<Child setMessage={setMessage} />

</div>

);

}

In the Child component, we have a button. Once the button has been clicked, the message will be set as “This is a message from child”. The parent component will render it.

function Child({setMessage}) {

const handleClick = ()=>{

setMessage("This is a message from child");

}

*return* (

<div>

<button onClick={handleClick} >Set Message</button>

</div>

)

}