Angular-Component-Interaction

A. Binding

- Interpolation{{}} : `<h3>{{Pagetitle}} </h3>`
- Property Binding [] : `<img [src]="imgUrl" alt="Image">`
- Event Binding() : `<button (click)="incrementCounter()"> Click!</button>`
- Two -way Binding{()

B. Split Two way Binding

```
<div>
  <input #nameRef type="text" [(ngModel)]='username'
(ngModelChange)='greetMe($event)'>
  <p>Welcome {{ username }}</p>
</div>
```

C. Getters & Setters

```
  get customerName(): string {
    return this.CustomerName;
  }

  set customerName(value: string) {
    if (value === 'Emu') {
      alert('Hello Emu!');
    }
    this.CustomerName = value;
  }
  <div>
  <input type="text" [(ngModel)]='customerName'>
  <p>Welcome {{ customerName }}</p>
  </div>
```

D. View Child Decorator

```
import { Component, AfterViewInit, ViewChild, ElementRef } from
'@angular/core';
```

```
export class AppComponent implements AfterViewInit {

  Pagetitle = 'Angular Component Interaction';
  imgUrl = 'https://picsum.photos/200';
  count = 0;
  name: string;
  username: string;
  private CustomerName: string;
  @ViewChild('nameRef') nameElementRef: ElementRef;

  ngAfterViewInit(): void {
    this.nameElementRef.nativeElement.focus();
    console.log(this.nameElementRef);
  }
```

<< solution: Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser

What a memory !! thousand problems! But 1 Line solution!!!!!>>

Inter Component Interaction

➔ Input() Decorator

```
@Input() loggedIn: boolean
```

➔ Getters and Setters

```
get loggedIn(): boolean {
   return this.LoggedIn;
 }

 @Input()
 set loggedIn(value: boolean) {
   this.LoggedIn = value;
   if (value === true) {
```

```
      this.message = 'Welcome back Emu!';
    } else {
      this.message = 'Please log in';
    }
  }
```

➔ ngOnChanges()

```
export class ChildComponent implements OnChanges {
  @Input() loggedIn: boolean;
  message: string;

  ngOnChanges( changes: SimpleChanges) {
    console.log(changes);
    const loggedInValue = changes.loggedIn;
    if (loggedInValue.currentValue === true) {
      this.message = 'Welcome back Emu!!';
    } else {
      this.message = 'Please log in';
    }
  }
```

➔ Template Reference Variables

Useful in Forms and nested properties.

```
<app-child #child [loggedIn]="userLoggedIn"></app-child>
<div>
  {{child.name}}
  <button (click)="child.greetMe()">Greet</button>
</div>
```

➔ ViewChild()

```
export class AppComponent implements AfterViewInit{
  userLoggedIn = true;
  @ViewChild(ChildComponent, { static: true })
childComponentRef: ChildComponent;

  ngAfterViewInit() {
```

```
    this.childComponentRef.message = 'Message from parent
component';
  }
}
```

➔ Emitting Events

```
 @Output() greetEvent = new EventEmitter();
 name: 'emu';

  constructor() { }

  ngOnInit() {
  }

  callParentGreet() {
    this.greetEvent.emit(this.name);
  }
```

➔ Services

```
  constructor( private interactionService: InteractionService)
{}
```

```
constructor(private interactionService: InteractionService) {
}

  ngOnInit() {
    this.interactionService.teacherGreeted$.subscribe(
      message => {
        if (message === 'Good Morning') {
          alert('Good Morning Teacher!');
        } else if (message === 'Well Done') {
          alert('Thank you Teacher!');
        }
      }
    );
  }
```