Event Binding in Angular 8

**Event Binding**

Event binding allows us to bind events such as keystroke, clicks, hover, touche, etc to a method in component. It is one way from view to component. By tracking the user events in the view and responding to it, we can keep our component in sync with the view. For Example, when the user changes to an input in a text box, we can update the model in the component, run some validations, etc. When the user submits the button, we can then save the model to the backend server.

**Syntax**

The Angular event binding consists of two parts

|  |
| --- |
| (target-event)="TemplateStatement" |

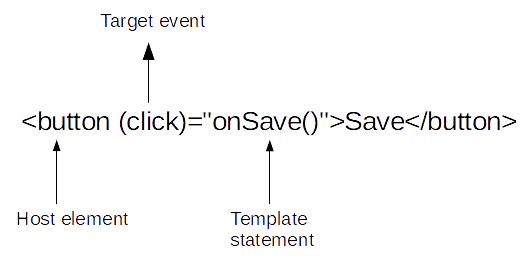
* We enclose the target event name in parentheses on the left side
* Assign it to a template statement within a quote on the right side

Angular event binding syntax consists of a target event name within parentheses on the left of an equal sign, and a quoted template statement on the right.

The following event binding listens for the button’s click events, calling the component’s onSave() method whenever a click occurs

|  |  |
| --- | --- |
|  | <button (click)="onSave()">Save</button> |

**For example:**



1. **<button** (click)="cookBacon()"**></button>**

n Angular 8, event binding is used to handle the events raised by the user actions like button click, mouse movement, keystrokes, etc. When the DOM event happens at an element(e.g. click, keydown, keyup), it calls the specified method in the particular component.

Using Event Binding we can bind data from DOM to the component and hence can use that data for further purposes.

**Syntax:**

< element (event) = function() >

**Approach:**

* Define a function in the app.component.ts file which will do the given task.
* In the app.component.html file, bind the function to the given event on the HTML element.

**Example 1:**Using click event on the input element.

**app.component.html**

# Event Binding in Angular 2 (v11)

Events are handled in Angular using the following special syntax.

(target event name) = "template statement"

Bind the target event name within parentheses on the left of an equal sign, and event handler method or statement on the right.

Example: Binding Button Click Event

 Copy

<button (click)="onShow()">Show</button>

Above, (click) binds the button click event and onShow() statement calls the onShow() method of a component.

Example: Handle Button Click Event in Component

 Copy

@Component({

selector: 'event-demo,

template: '<button (click)="onShow()" >Show</button>'

})

export class EventBindingDemoComponent implements OnInit {

constructor() { }

ngOnInit(): void {

}

onShow() {

alert('Show button clicked!');

}

}

Alternatively, use the on- prefix, known as the canonical form:

Example: on-event

<button on-click="onShow()" >Show</button>

By default, an event propagates up to the parent container event. In the following example, click event propagates to click of div and will call both the onShow() and onDivClick() methods.

Example: Event Bubbling

<div (click)="onDivClick()">

<button (click)="onShow()" >Show</button>

</div>

## $event

Mostly, when an event is raised, you may need to pass some value to the event handler function. This value can be number, string, or an object that contains information about an event.

You can pass the number or string value to the event handler function, as shown below.

Example: Passing Event Data

<button (click)="onShow(20)">Show</button>

Angular includes $event that contains the information about an event. The type of $event depends on the target event, e.g., if the target event is a native DOM element event, then it is an object.

Example: $event

<button (click)="onShow($event)">Show</button>

A component should define the onShow(event) method where the type of the parameter can be KeyboardEvent, MouseEvent, etc. If you don't know the exact event type, they use “any” type, as shown below.

Example: event Parameter

onShow(event:any) {

console.log(event);

}

If event is a native DOM element event then $event.target get DOM element reference using which you can access element's property e.g. $event.target.innerHTML returns the value of innerHTML property of a DOM element.

Example: Event Handling

<button (click)="onShow($event)">Show</button>

//component method

onShow(event:any) {

alert(event.target.innerHTML); // returns Show

}

You can use $event.target in the template statement. The following example binds a component property to $event.target.value of the input box on the input event without using ngModel.

Example: Bind Event without ngModel

<input type="text" (input)="userName=$event.target.value"><br/>

{{userName}}

* HTML

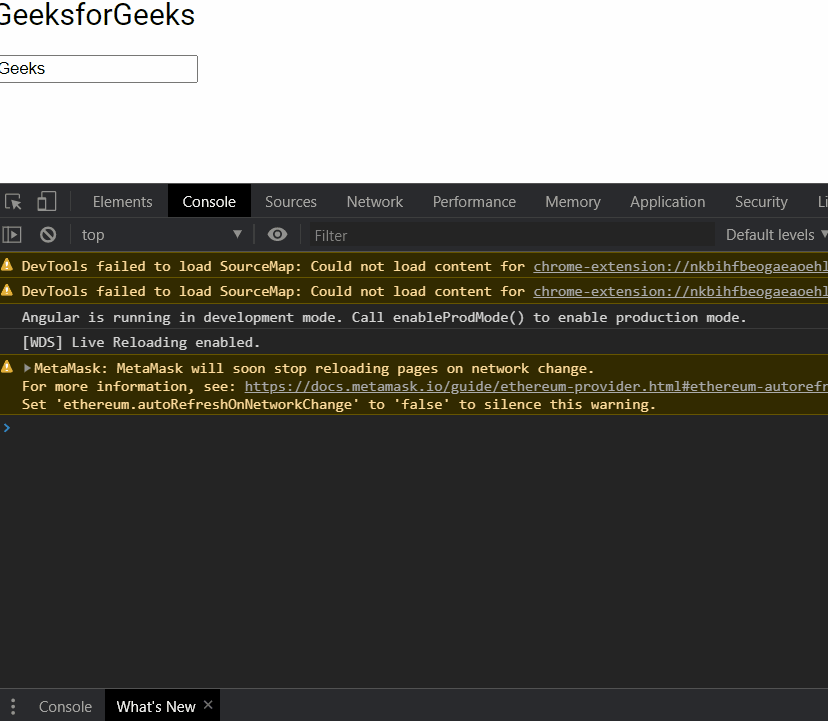
|  |
| --- |
| <h1>    GeeksforGeeks  </h1>  <input (click)="gfg($event)" value="Geeks"> |

**app.component.ts**

* App.component.ts

|  |
| --- |
| import { Component } from '@angular/core';  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    gfg(event) {      console.log(event.toElement.value);    }  } |

**Output:**



**Example 2:**Using keyup event on the input element.

**app.component.html**

* HTML

|  |
| --- |
| <!-- event is passed to function -->  <input (keyup)="onKeyUp($event)">      <p>{{text}}</p> |

**app.component.ts**

|  |
| --- |
| import { Component } from '@angular/core';  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    text = '';    onKeyUp(x) {        // Appending the updated value      // to the variable      this.text += x.target.value + ' | ';    }  } |

**Output:**

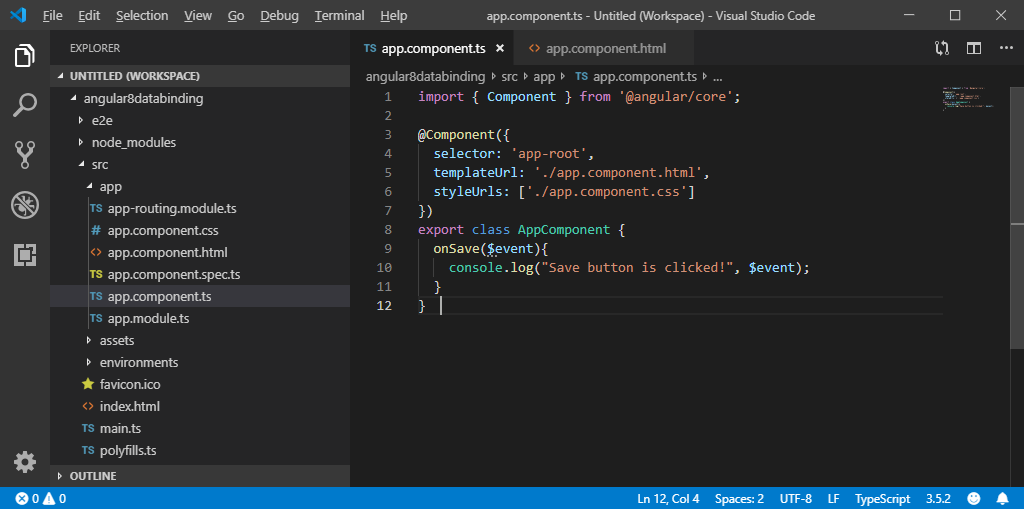
[](https://media.geeksforgeeks.org/wp-content/uploads/20200502124741/gfg-key.gif)

Event Binding Example

Let's take a button in the HTML template and handle the click event of this button. To implement event binding, we will bind click event of a button with a method of the component.

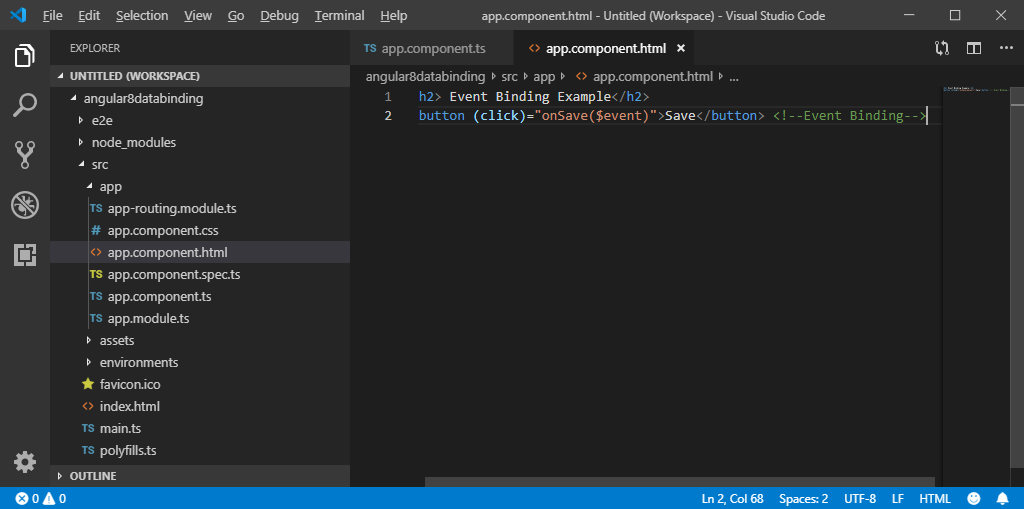
Now, open the **app.component.ts** file and use the following code:

1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export class AppComponent {
8. onSave($event){
9. console.log("Save button is clicked!", $event);
10. }
11. }

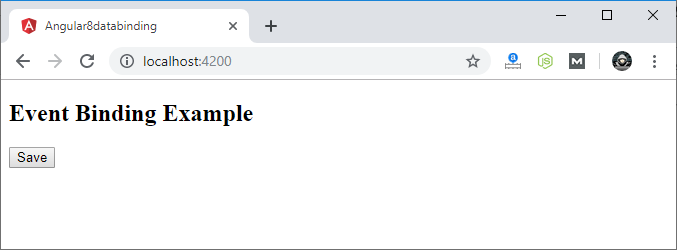


**app.component.html:**

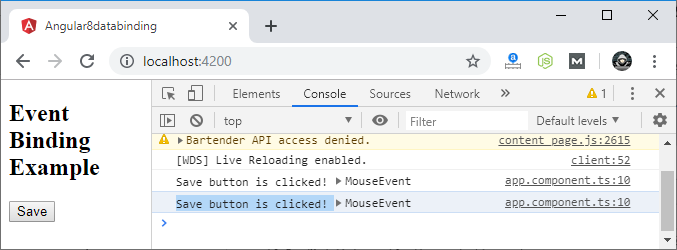
1. **<h2>** Event Binding Example**</h2>**
2. **<button** (click)="onSave($event)"**>**Save**</button>** <!--Event Binding-->



**Output:**



Click on the "Save" button and open console to see result.



Now, you can see that the "Save" button is clicked.

Event Bubbling

Event bubbling is used to specify an order in which event handlers are called when one element is nested inside a second element, and both elements have registered a listener for the same event (i.e. click).

Let's see the above button example. Here, I have used a div wrapper around the button in component HTML and div has also a click event handler. It is only to show some message if div is clicked.

Use the following code in **app.component.ts** file:

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  onSave($event){

    console.log("Save button is clicked!", $event);

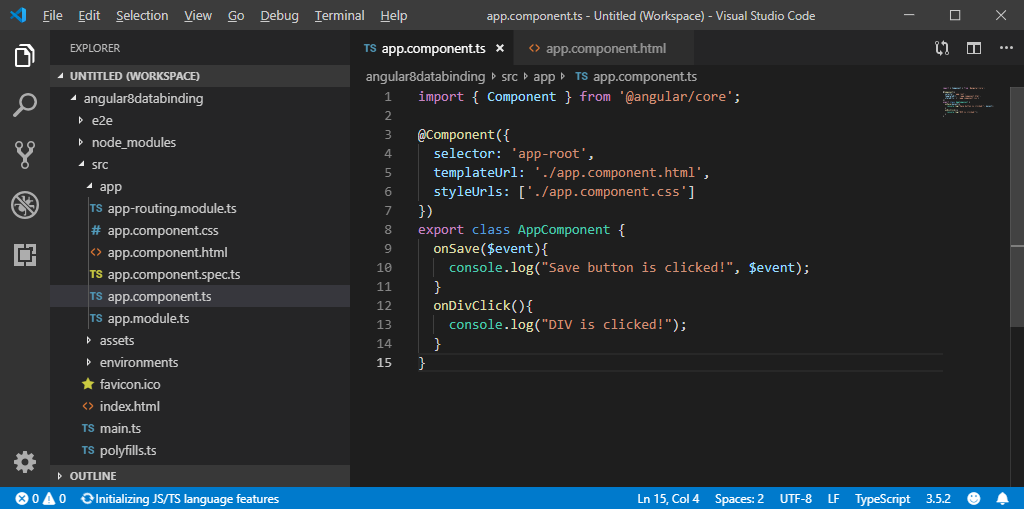
  }

  onDivClick(){

    console.log("DIV is clicked!");

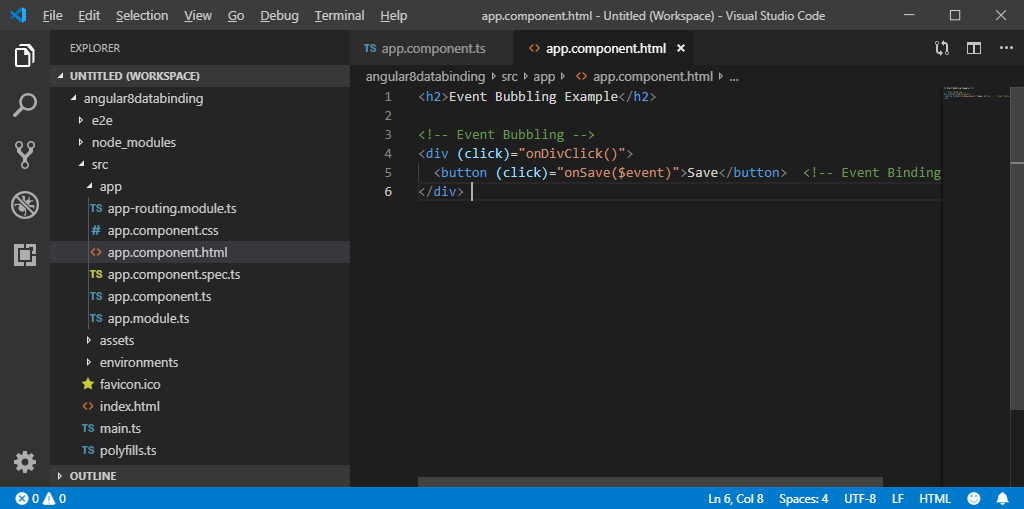
  }

}



**app.component.html:**

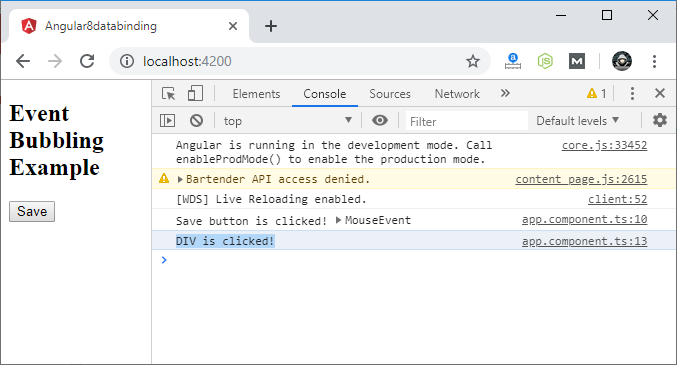
1. **<h2>**Event Bubbling Example**</h2>**
2. <!-- Event Bubbling -->
3. **<div** (click)="onDivClick()"**>**
4. **<button** (click)="onSave($event)"**>**Save**</button>**  <!-- Event Binding -->
5. **</div>**



**Output:**



Click on the "Save" button and open console to see result.



Here, you can see that your div message is also occurred. This is all due to event bubbling where you have specified onDivClick button

### Angular Even binding examples

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joi**Angular Even binding examples**

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joining Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example an**Angular Even binding examples**

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joining Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example and **Angular Even binding examples**

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joining Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example and run it, initial display should be as given below. **Angular Even binding examples**

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joining Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example and run it, initial display should be as given below.

**Angular Even binding examples**

1. First example shows how to disable a button after it is clicked. It uses both property and event binding. A boolean flag is bound to the “disabled” property of the button, when button is clicked using click event a method is called which toggles the boolean flag.

**Component (Typescript code)**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

flag : boolean = false;

// toggle the flag

onButtonClick(){

this.flag = !this.flag;

}

}

**Template (HTML file)**

<div class="container">

<div class="row">

<div class="col-md-12">

<label for="name" class="col-form-label">Name:</label>

<input class="form-control" placeholder="Enter name" id="name">

</div>

</div>

<div class="row mt-3">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onButtonClick()" [disabled]="flag">Submit</button>

</div>

</div>

</div>

In the button element, click event is bound to a method onButtonClick(). When the button is clicked onButtonClick() method is called on the component which toggles the boolean flag field. Initially value of flag is false thus in the method it becomes true.

2. Binding with input event of input element to get the value of input field.

<div class="container">

<div class="row">

<div class="col-sm-12">

<label for="name" class="col-form-label">Name:</label>

<input type="text" class="form-control" (input)="updateUserName($event)" id="name">

</div>

<p>{{ userName }}</p>

</div>

</div>

In the input element, input event is bound to a method updateUserName and $event is passed as an argument. $event denotes the data emitted with the event.

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

userName : String;

updateUserName(event : Event){

this.userName = (event.target as HTMLInputElement).value;

}

}

In the component there is a userName property, value from the passed $event argument is extracted in the method updateUserName() and assigned to the userName.

3. Using the mouseover event to display the selected item. In the example User records are displayed in a table and the user name over which mouse pointer is hovering is displayed in the separate <div> section using the mouseover event.

**User Model class**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**Component**

import {

Component

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

users: User[];

userName : String;

constructor(){

// Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

**Template**

<div class="container">

<div class="bg-warning my-4">

Selected User: {{userName || '(None)'}}

</div>

<table class="table table-sm table-bordered m-t-4">

<tr>

<th>Name</th>

<th>Age</th>

<th>Joining Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example and run it, initial display should be as given below.

With mouse over selection.

With mouse over selection.

With mouse over selection.

run it, initial display should be as given below.

With mouse over selection.

d run it, initial display should be as given below.

With mouse over selection.

ning Date</th>

</tr>

<tr \*ngFor="let user of users">

<td (mouseover)="userName=user.name">{{user.name}}</td>

<td>{{user.age}}</td>

<td>{{user.joinDate | date:'dd/MM/yyyy'}}</td>

</tr>

</table>

</div>

In the table (mouseover) event is used with the <td> element.

Build the example and run it, initial display should be as given below.

With mouse over selection.