AngularJS Events

Topics : AngularJS
Written on January 09, 2024

In AngularJS, events are an integral part of building dynamic and interactive applications. AngularJS provides a set of directives and services to handle events efficiently. Here's an overview of how events are managed in AngularJS:

1. ng-click Directive:

The `ng-click` directive is one of the simplest ways to handle click events on elements. It allows you to specify a function to be executed when the element is clicked.

```
<button ng-click="handleClick()">Click me</button>

angular.module('myApp').controller('MyController', function($scope) {
$scope.handleClick = function() {
alert('Button clicked!');
};
});
```

2. ng-change Directive:

The `ng-change` directive is used to execute a function when the input value changes. It is commonly used with input elements like `<input>` and `<select>`.

```
<input ng-model="inputValue" ng-change="handleChange()">

angular.module('myApp').controller('MyController', function($scope) {
$scope.inputValue = '';

$scope.handleChange = function() {
console.log('Input value changed:', $scope.inputValue);
};
});
```

3. ng-submit Directive:

The `ng-submit` directive is used to specify a function to be called when a form is submitted.

```
<form ng-submit="submitForm()">
<!-- Form content here -->
```

```
<button type="submit">Submit</button>
</form>

angular.module('myApp').controller('MyController', function($scope) {
$scope.submitForm = function() {
alert('Form submitted!');
};
});
```

4. $broadcast, $emit, and $on:

AngularJS provides the `$broadcast` and `$emit` methods to broadcast events and the `$on` method to listen for events. These methods can be useful for communication between different components, such as controllers.

```
angular.module('myApp').controller('ParentController', function($scope) {
$scope.$on('customEvent', function(event, data) {
console.log('ParentController received:', data);
});

$scope.triggerEvent = function() {
$scope.$broadcast('customEvent', 'Hello from ParentController');
};
});

angular.module('myApp').controller('ChildController', function($scope) {
$scope.$on('customEvent', function(event, data) {
console.log('ChildController received:', data);
});
});
```

In this example, the `ParentController` broadcasts a custom event, and the `ChildController` listens for that event. The `$broadcast` method sends the event downwards to child scopes, and the `$on` method listens for the event.

5. $timeout Service:

AngularJS provides the `$timeout` service, which is a wrapper for the `window.setTimeout` function. It allows you to delay the execution of a function.

```
angular.module('myApp').controller('MyController', function($scope, $timeout) {
$scope.delayedFunction = function() {
console.log('Delayed function executed');
};

// Delay the execution of delayedFunction by 2 seconds
$timeout($scope.delayedFunction, 2000);
});
```

6. Custom Directives:

Custom directives in AngularJS can also handle events. Directives can define their own behavior and

trigger events based on user interactions.

```
angular.module('myApp').directive('customDirective', function() {
return {
restrict: 'A',
link: function(scope, element) {
element.on('click', function() {
scope.$emit('customDirectiveClick', 'Click event from custom directive');
});
}
};
});

angular.module('myApp').controller('MyController', function($scope) {
$scope.$on('customDirectiveClick', function(event, data) {
console.log('Custom directive click event:', data);
});
});
```

In this example, the `customDirective` triggers a custom event when clicked, and the controller listens for that event.