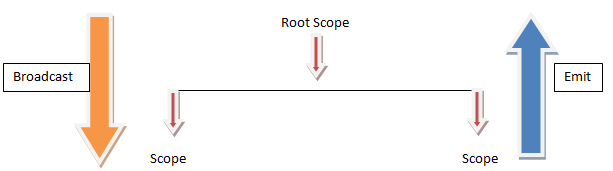
AngularJS provides $on, $emit, and $broadcast services for event-based communication between controllers.



$emit

It dispatches an event name upwards through the scope hierarchy and notify to the registered $rootScope.Scope listeners. The event life cycle starts at the scope on which $emit was called. The event traverses upwards toward the root scope and calls all registered listeners along the way. The event will stop propagating if one of the listeners cancels it.

//Define Module

var app = angular.module('IG', []);

//Define Controller

app.controller("firstCtrl", function ($scope) {

$scope.$on('ShowMe', function (event, args) {

$scope.message = args.message;

console.log($scope.message);

});

});

app.controller("secondCtrl", function ($scope) {

$scope.handleClick = function (msg) {

$scope.$emit('ShowMe', { message: msg });

};

});

<div ng-controller="firstCtrl">

<h1>Parent Controller</h1>

<p>Emit Message : {{message}}</p>

<br />

<div ng-controller="secondCtrl">

<h1>Child Controller</h1>

<input ng-model="msg">

<button ng-click="ShowMe(msg);">Emit</button>

</div>

</div>

## $broadcast

It dispatches an event name downwards to all child scopes (and their children) and notify to the registered $rootScope.Scope listeners. The event life cycle starts at the scope on which $broadcast was called. All listeners for the event on this scope get notified. Afterwards, the event traverses downwards toward the child scopes and calls all registered listeners along the way. The event cannot be canceled.

//Define Module

var app = angular.module('IG', []);

//Define Controller

app.controller("firstCtrl", function ($scope) {

$scope.handleClick = function (msg) {

$scope.$broadcast('ShowMe', { message: msg });

};

});

app.controller("secondCtrl", function ($scope) {

$scope.$on('ShowMe', function (event, args) {

$scope.message = args.message;

console.log($scope.message);

});

});

<div ng-controller="firstCtrl">

<h1>Parent Controller</h1>

<input ng-model="msg">

<button ng-click="handleClick(msg);">Broadcast</button>

<br /><br />

<div ng-controller="secondCtrl">

<h1>Child Controller</h1>

<p>Broadcast Message : {{message}}</p>

</div>

</div>

## $on

It listens on events of a given type. It can catch the event dispatched by $broadcast and $emit.

#### Note

1. If there is no parent-child relation between your scopes you can inject $rootScope into the controller and broadcast the event to all child scopes but you cannot emit your event.
2. You can emit your event only when you have parent-child relation and event propagation is initiated by child. However, $emit can fire an event only for all $rootScope.$on listeners.

# Communicating b/w Controllers using broadcast

<div ng-controller="Controller1">

<input ng-model="message" >

<button ng-click="handleClick(message);">LOG</button>

</div>

<div ng-controller="Controller2">

<input ng-model="message" >

</div>

<div ng-controller="Controller3">

<input ng-model="message" >

</div>

var app = angular.module('IG', []);

app.factory('mySharedService', function ($rootScope) {

var sharedService = {};

sharedService.message = '';

sharedService.prepForBroadcast = function (msg) {

this.message = msg;

this.broadcastItem();

};

sharedService.broadcastItem = function () {

$rootScope.$broadcast('handleBroadcast');

};

return sharedService;

});

function Controller1($scope, sharedService) {

$scope.handleClick = function (msg) {

sharedService.prepForBroadcast(msg);

};

$scope.$on('handleBroadcast', function () {

$scope.message = sharedService.message;

});

}

function Controller2($scope, sharedService) {

$scope.$on('handleBroadcast', function () {

$scope.message = 'two: ' + sharedService.message;

});

}

function Controller3($scope, sharedService) {

$scope.$on('handleBroadcast', function () {

$scope.message = 'three: ' + sharedService.message;

});

}

Controller1.$inject = ['$scope', 'mySharedService'];

Controller2.$inject = ['$scope', 'mySharedService'];

Controller3.$inject = ['$scope', 'mySharedService'];

Architecture

