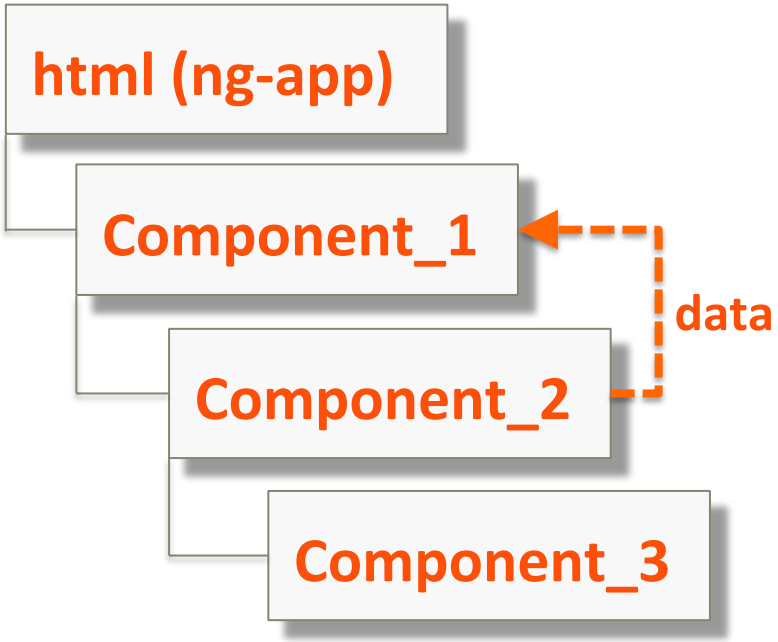
AngularJS Event System

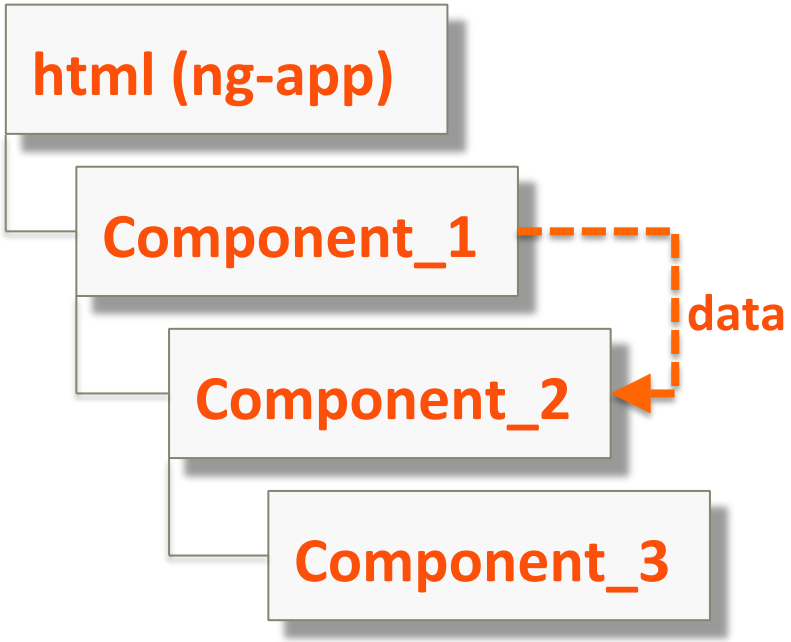
Communicating with Parent



Solution: Access Parent Scope

- $scope.$parent
- Use &method callback binding

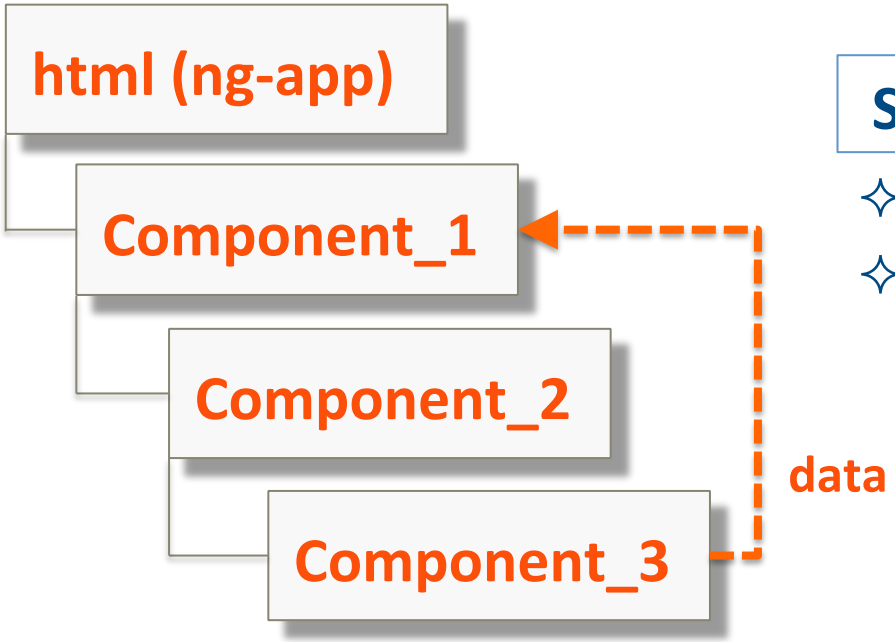
Communicating with Child



Solution: Provide Data Input

- **Send data into that component**

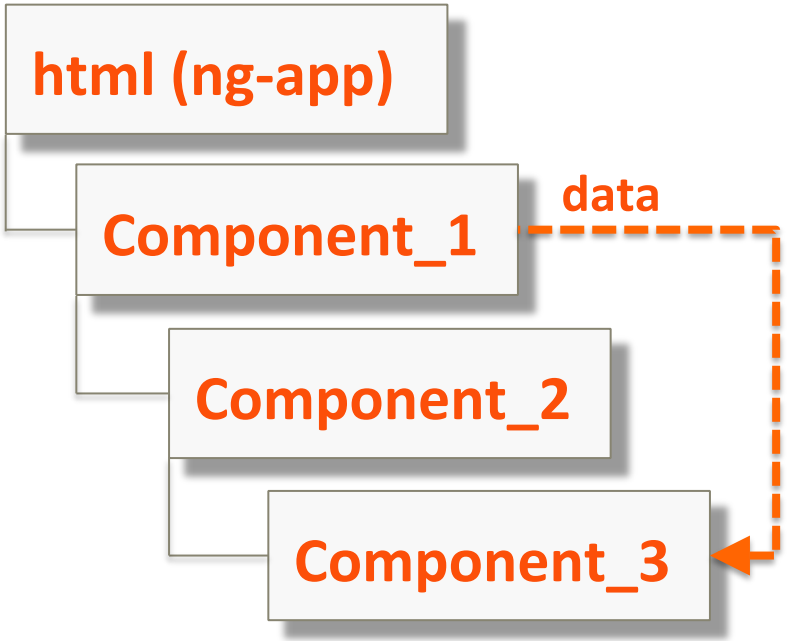
Communicating with Grandparent?



Solution: Access Parent's Parent?

- **$scope.$parent.$parent?**
- **User Service to share data?**
  - **Would need to set up a watch in component_1 to react to change**

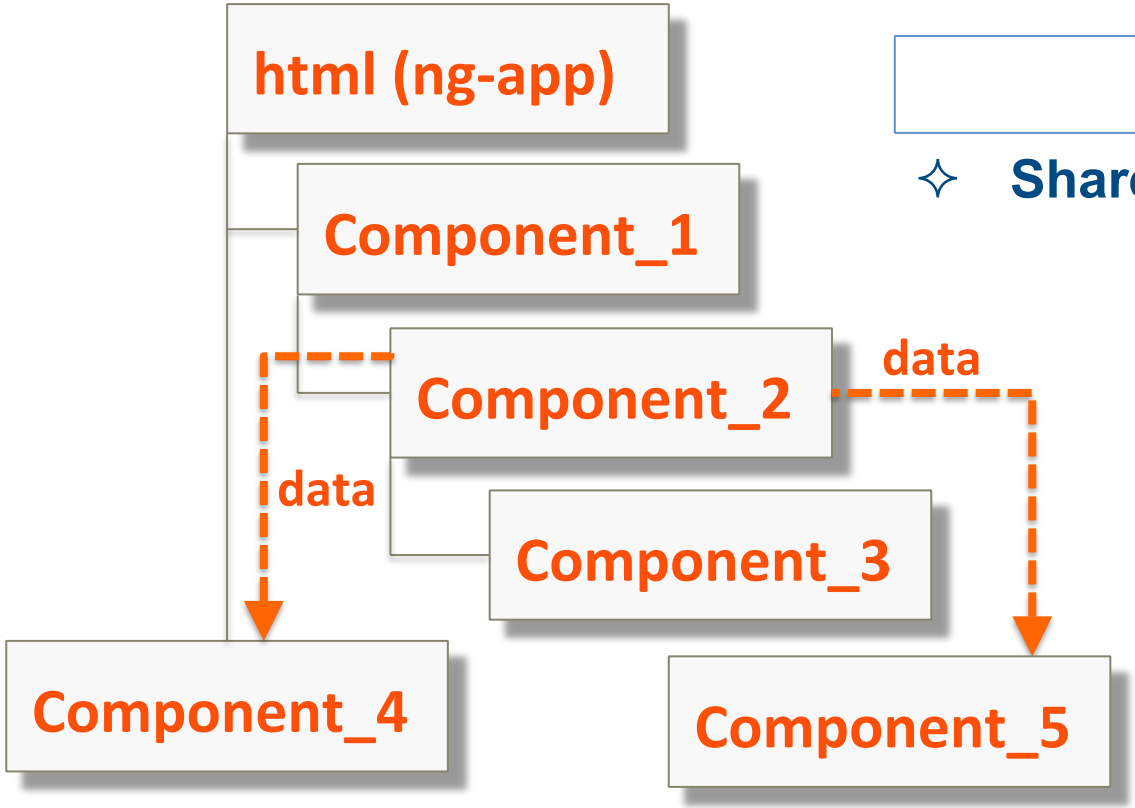
Communicating with Grandchild?



Solution: Access Child's Child?

- **Send data into component_2 and have compoenent_2 send data to component_1?**
- **Use Service to share data?**
  - **Would need to set up a watch in component_3 to react to change**

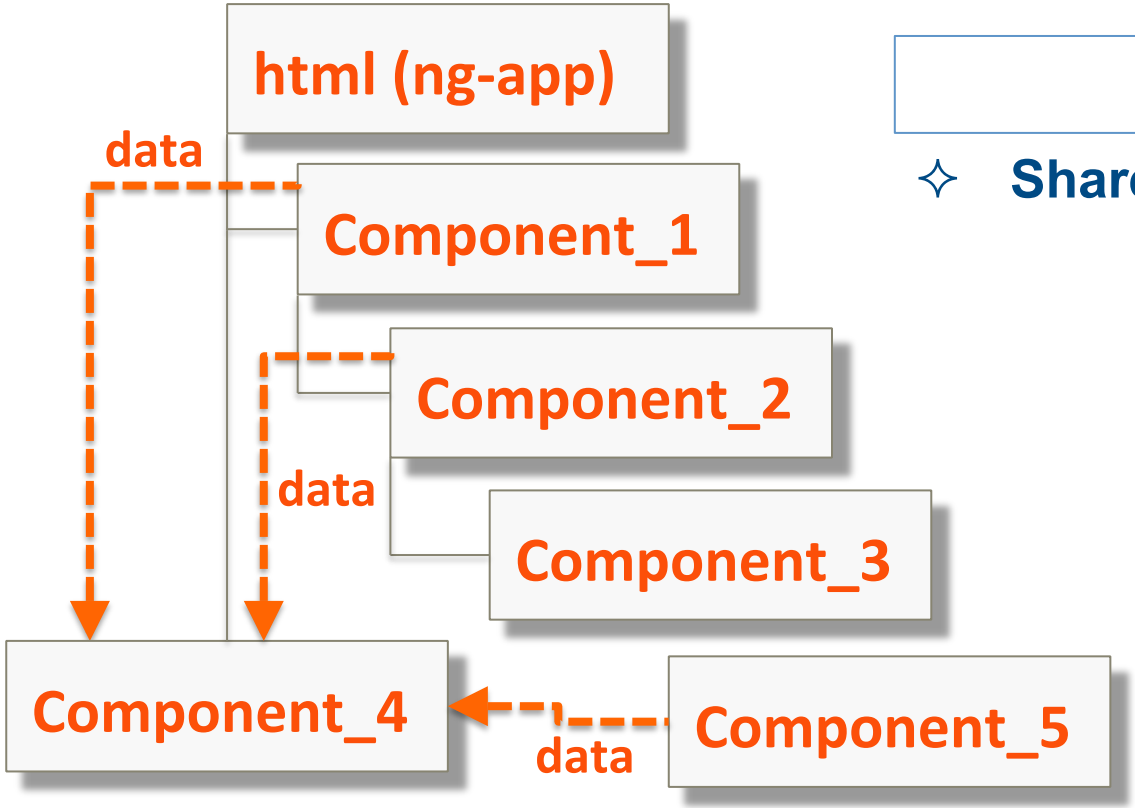
Communicating with Multiple Components?



Solution?

- **Shared service everywhere?**

Multiple Components Communicating with One?



Solution?

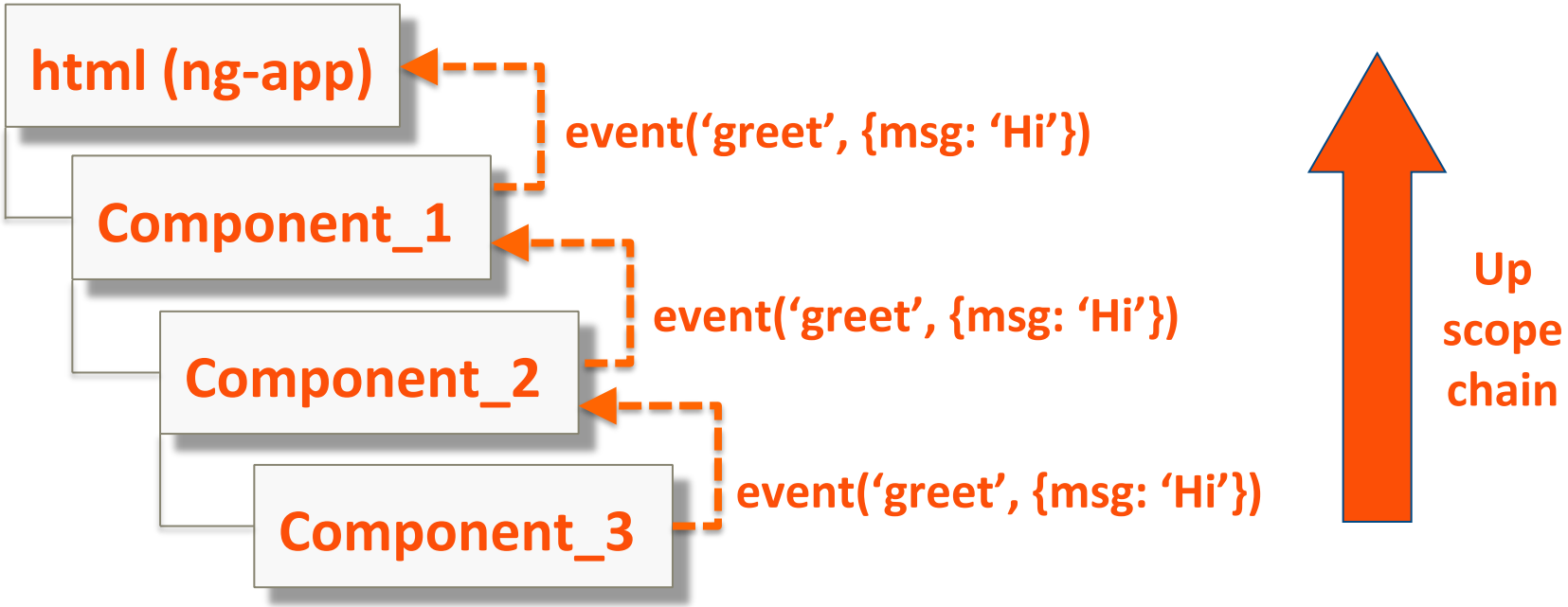
- ✧ **Shared service everywhere?**

publish–subscribe design pattern

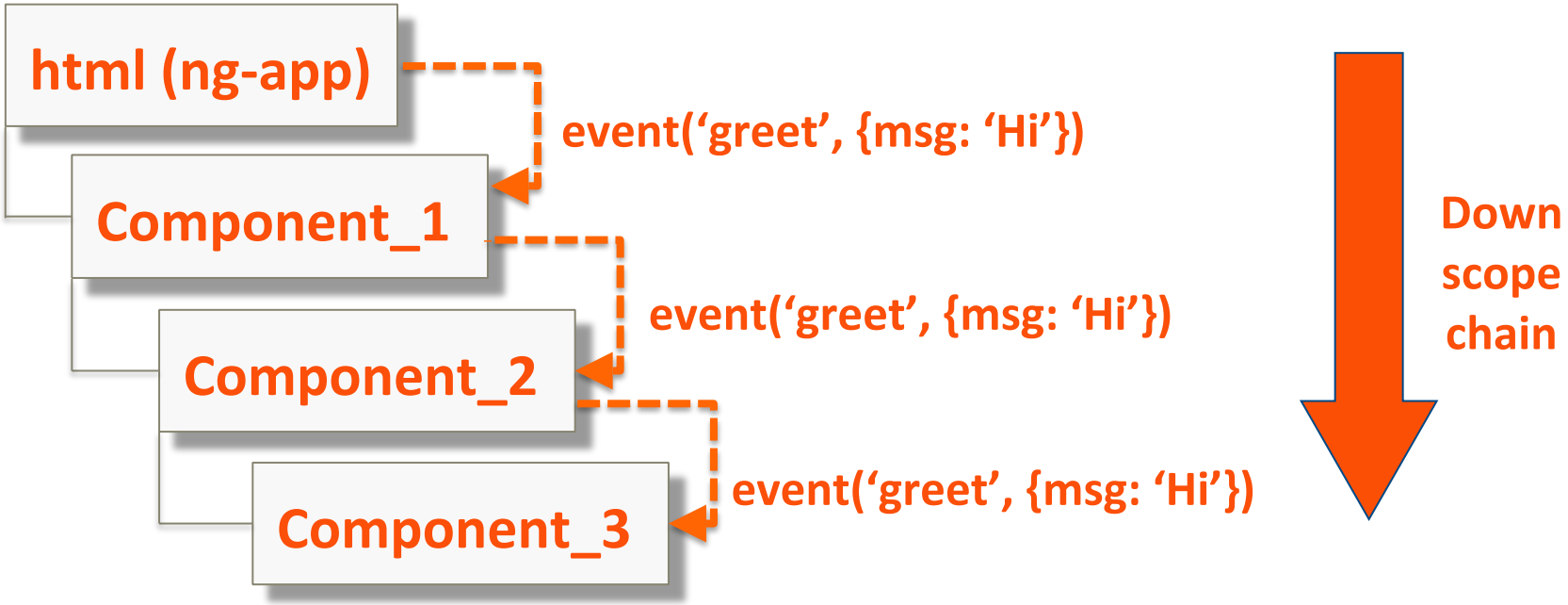
Publishers send messages to subscribers on a common channel

- Publishers:
  - Mark messages with a classification
  - Don’t know subscribers or if there are any
- Subscribers:
  - Sign up to listen for messages with a particular classification
  - Don’t know publishers or if there are any
- **In Angular, the common channel is scope**
  - **Messages are events that can hold data**

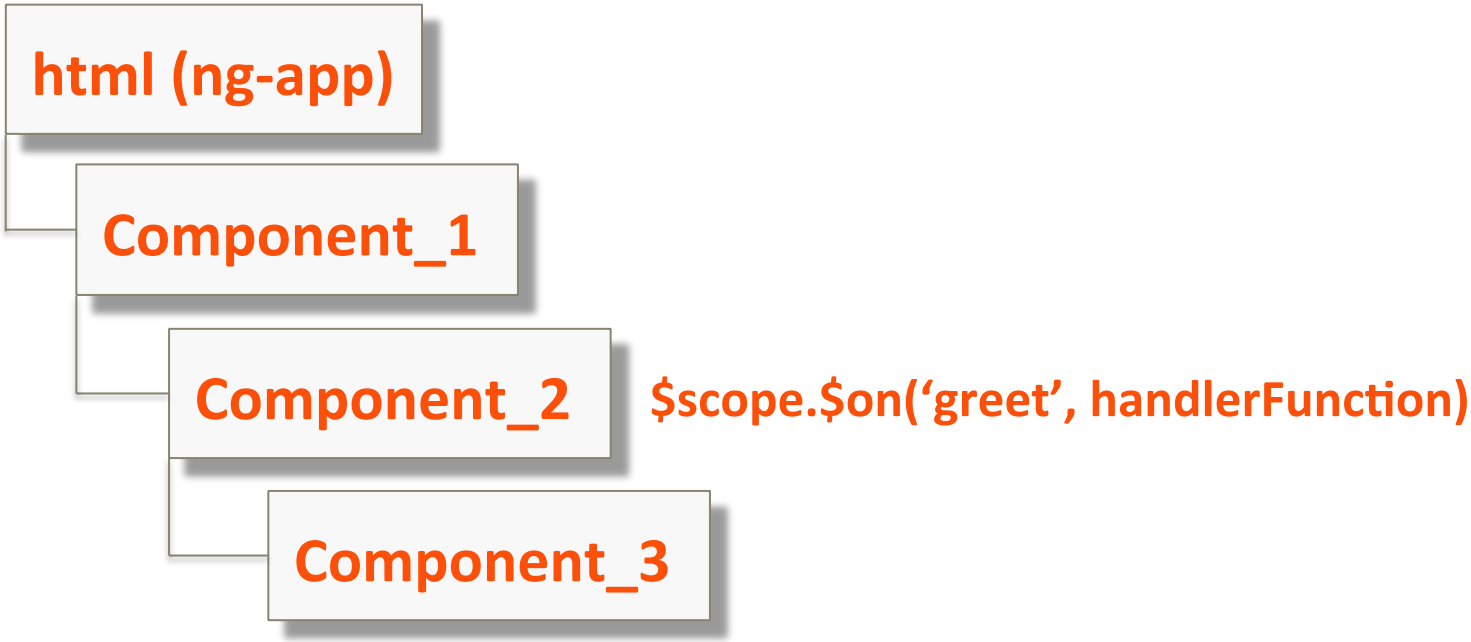
Publishing an Event: $scope.$emit



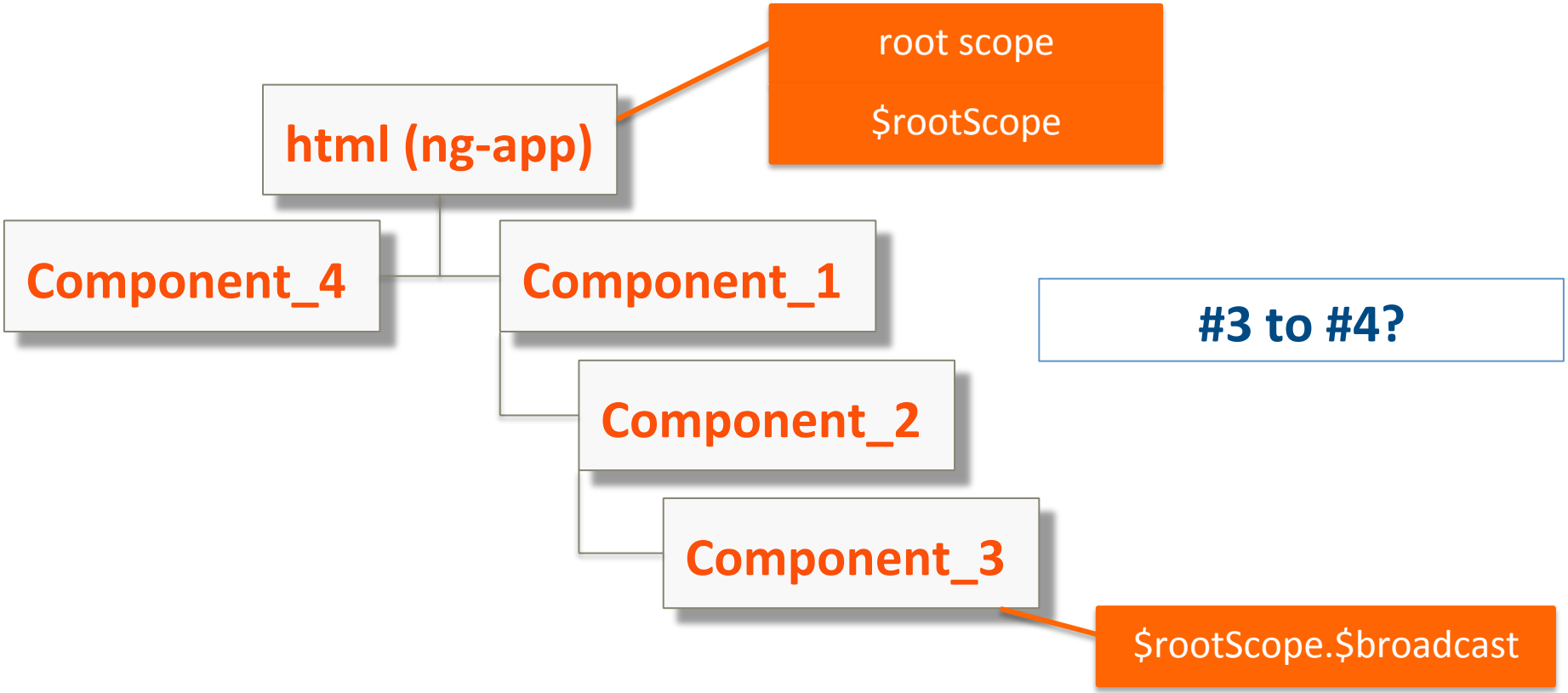
Publishing an Event: $scope.$broadcast



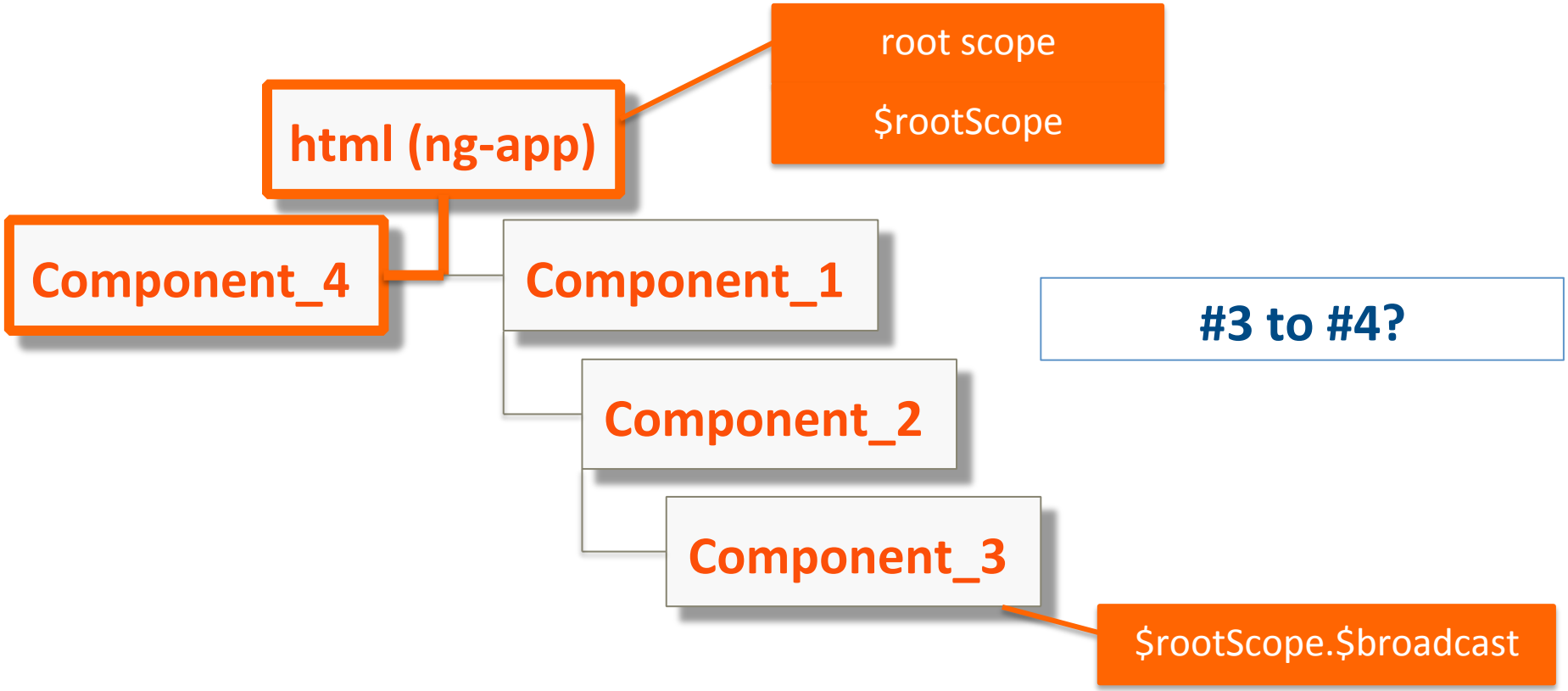
Listening for an Event: $scope.$on



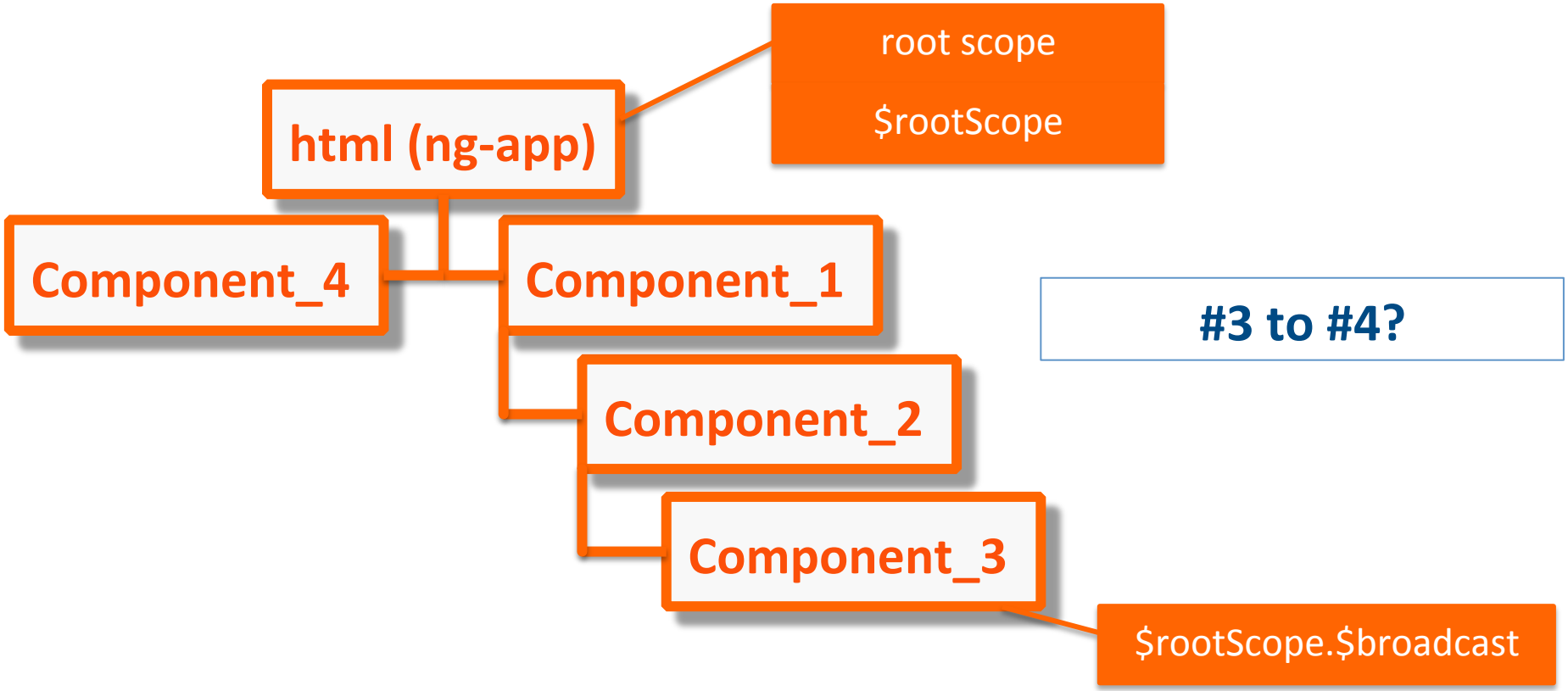
Publishing an Event: $rootScope.$broadcast



Publishing an Event: $scope.$broadcast



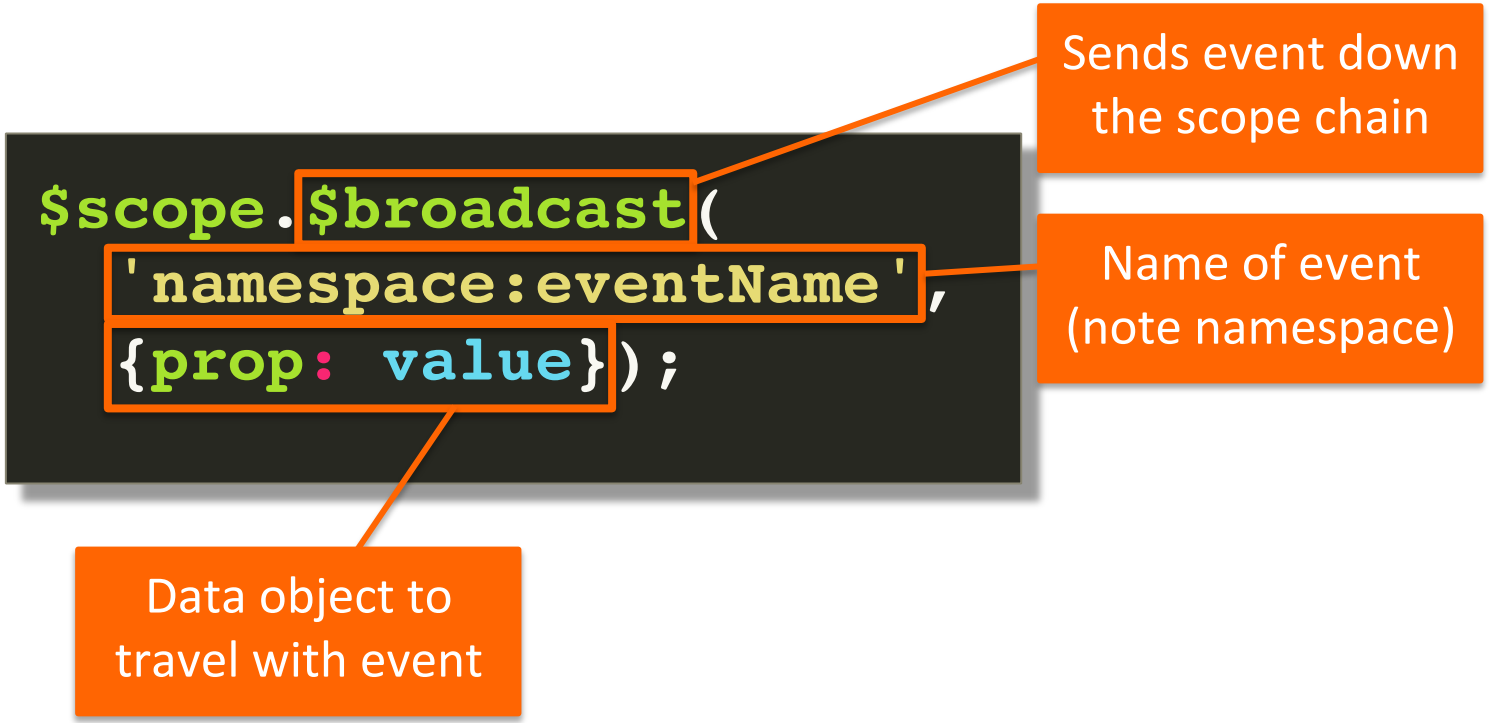
Publishing an Event: $scope.$broadcast



Step 1: Broadcast or Emit an Event

```
$scope.$emit(
   'namespace:eventName',
   {prop: value});
```

Step 1: Broadcast or Emit an Event



Step 2: Listen for & Handle the Event

```
$scope.$on('namespace:eventName',
           handler);

function handler(event, data) {
  if (data.prop === 'val1') {
    ...
  }
});
```

Same name as was emitted/broadcasted

Data that traveled with the event

Summary

- Publish-subscribe design pattern is implemented using the Angular Events system
- You can publish events from anywhere in the system and listen for those events anywhere in the system
- $scope.$emit sends the event up the scope chain
- $scope.$broadcast sends the event down the scope chain
- To broadcast to all nodes, use $rootScope.$broadcast
- To listen for event, use either $scope.$on or $rootScope.$on
- Deregister listener when using $rootScope.$on