Router State Transition Events

$stateChangeStart

fires when state change transition begins

```
$rootScope.$on('$stateChangeStart',
   function(event, toState, toParams,
              fromState, fromParams, options)
   { ... });
```

$stateChangeSuccess

fired once the state transition is complete

```
$rootScope.$on('$stateChangeSuccess',
   function(event, toState, toParams,
             fromState, fromParams)
   { ... });
```

$stateChangeError

fires when an error occurs during transition

```
$rootScope.$on('$stateChangeError',
   function(event, toState, toParams,
             fromState, fromParams, error)
   { ... });
```

$stateChangeStart

Use `event.preventDefault()` to prevent the transition from occurring

ui-router State Change Events

- ui-router exposes numerous state change events that our code is able to listen for
- All ui-router events are fired on the $rootScope
- $stateChangeStart – starts the state transition
  - Call event.preventDefault() to prevent the transition
- $stateChangeSuccess indicates a successful transition end
- $stateChangeError indicates that the transition failed, including having errors in the resolve
  - Listen for this event to catch ALL errors during state changes

Summary

- Nested states allow us to logically represent nested views
- Parent state template has a ui-view in its template for the child state’s template to insert its HTML
- Child state name is usually declared with syntax ‘parent.child’
- The optionally declared url of the child gets concatenated to the declared url of the parent
- The parent’s resolve property is inherited by the child and is injectable directly into the child’s controller