DATA BINDING IN ANGULARJS

FROM MODEL TO VIEW

Thomas Roch, Formedix Ltd

TYPES OF DATA BINDING

- One way (model to view)
- Two-way data binding

FROM VIEW TO MODEL

- Javascript DOM **Event** model: Event, KeyboardEvent, MouseEvent...
- Easy to **bind** model to view in Angular directives

```
app.directive('input', function () {
    return {
        restrict: 'E',
        scope: false,
        link: function (scope, element, attrs) {
            element.bind('input', function (event) {
                scope.$apply(function () {
                    scope.model = element.val();
                });
            });
        }
    };
});
```

FROM MODEL TO VIEW

- No events or notifications fired when a value changes in EcmaScript 5
- Observers are for the future (EcmaScript 6)

```
Object.observe(myObj, function observer(changes) {
    /* Do something */
    changes.forEach(function (change) {
        console.log(change.name, change.type, change.oldValue);
    });
});
```

A WAY TO OVERCOME THIS: DIRTY CHECKING

WATCHERS

```
// Object.observe(myObj, function observer(changes) {
//     changes.forEach(function (change) {
//         console.log(change.name, change.type, change.oldValue);
//     });
// });

$scope.$watch(watchExpression, function listener(newValue, oldValue) {
    // Do something
    console.log(newValue, oldValue);
});
```

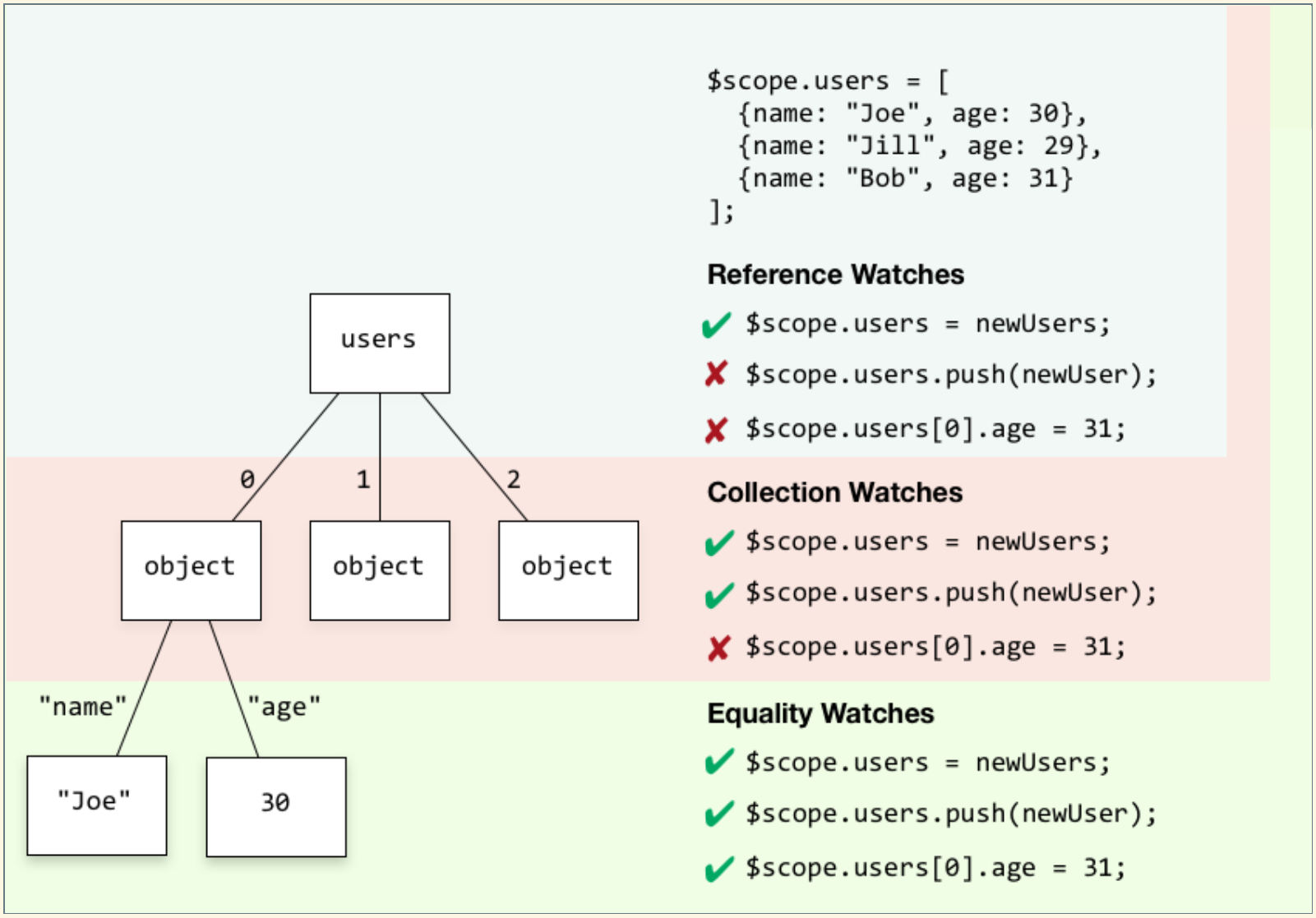
- **watchExpression** can be an expression (string)
  It is evaluated in the context of the $scope using $parse
- Or it can be a function returning a value
- Watched values can be primitives, array or objects
- The **listener** function is executed if the **watchExpression** result has changed since the last check.
- $scope.$watch() returns a deregistration function

WATCHING DEPTHS

- **$scope.$watch(watchExpr, listener)**
  for watching primitives or object references

- **$scope.$watchCollection(watchExpr, listener)**
  for watching additions and deletions in Arrays

- **$scope.$watch(watchExpr, listener, true)**
  for objects and arrays equality (using angular.equals)



Credits: Tero Parviainen, http://teropa.info/blog/2014/01/26/the-three-watch-depths-of-angularjs.html

WHICH DIRECTIVES ADD WATCHERS?

- **$watch:**
  {{ }}, ngBind, ngBindTemplate, ngModel
  ngShow, ngHide, ngIf, ngSwitch
  ngSelected, ngChecked, ngDisabled, ngRequired, etc...
- **$watchCollection:**
  ngRepeat
- **$watch with object equality:**
  ngClass, ngStyle

THE DIGEST PROCESS

- $scope.$digest() digests a scope and its children
- Cannot run concurently
- Evaluates all watch expressions and functions
- If at least one listener is fired, it will re-evaluate all watch expressions
- The digest process stops when no more listeners have been fired
- Stops at 10 times and throws an infinite digest loop

WHAT TRIGGERS A DIGEST?

- $digest usually not called directly

```
// Apply changes: execute function and call $digest
$scope.$apply(function () {
    $scope.myModel = 'Hello';
});
```

- Services: $http, $timeout, $interval
- Directives: ngClick, ngChange, ngFocus, ngPaste, etc...
- When inside a watch listener, no need to trigger a digest!

ISSUE #1: PERFORMANCE

To measure performance improvements we have created a crude benchmark of a table of 20 columns (a repeater) and 100 rows (another repeater) and placed a binding in each cell for a total of 2000 cells. We then proceeded to change a small portion of the model to see the performance characteristics of the update. The result is summarized below.

- Dirty checking: 40ms
- Observers: 1-2ms

ISSUE #2: SUSTAINABILITY

- The more the watchers, the more processing required
- The more processing, the more time it takes

SOLUTION

- Limiting the number of watchers!

SCOPES AND DATA

- A variable can be available when the scope is instanciated: use of resolve, or synchronous initialisation
- A variable can be available after the scope is instanciated: fetching data asynchronously
- A variable can be changing multiple times during a scope's life

THE THREE CASES OF WATCHING DATA

A watcher is watching...

- data available at scope creation which will never change
- data available after scope creation which will never change thereafter
- data available immediately or not, which will change once or more

DEMO

QUESTIONS