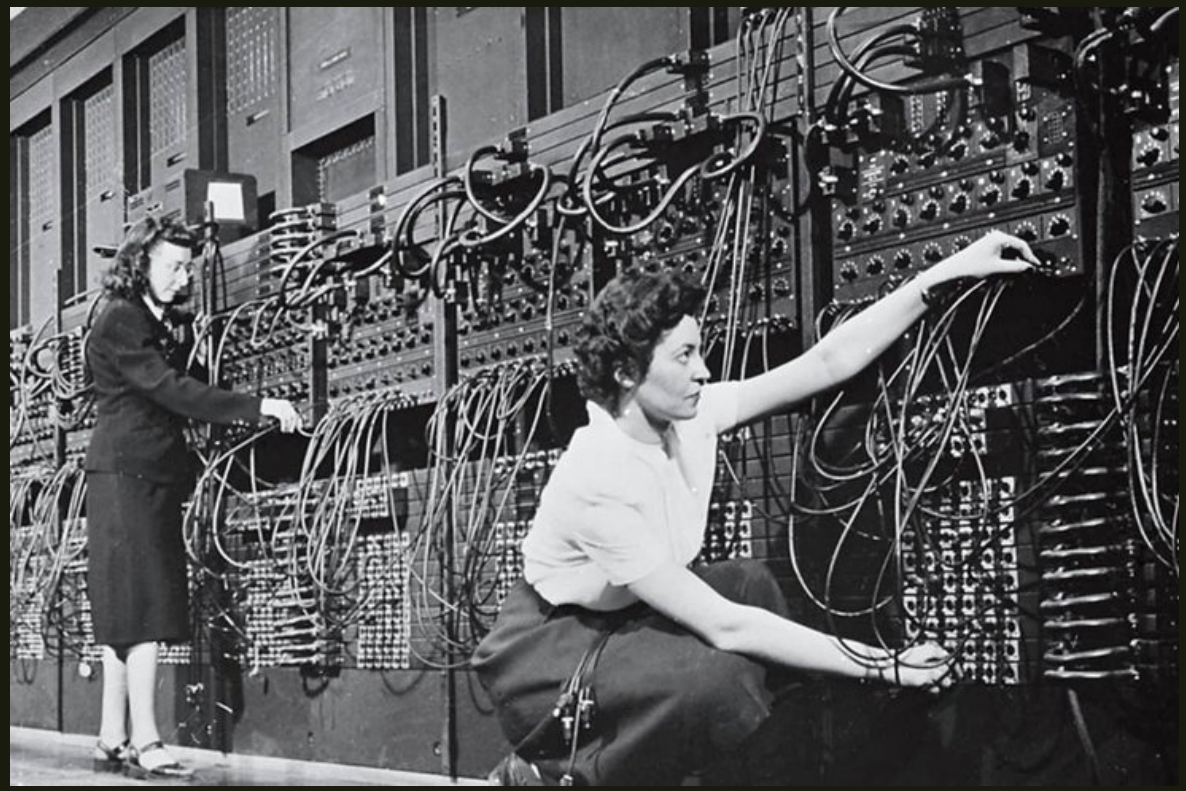
ANGULAR WEEK 3

Hack Your Future

Agenda

- Component bindings
- URL Routing
- Filters



COMPONENT BINDINGS

Component Bindings

- Maps HTML **attributes** to **properties** on the component controller, as specified in the component definition
- Values passed through attributes are bound to the component controller (`$ctrl`)

```
angular.module('app.people')
    .component('peopleItem', {
        template: '<h4>{{ $ctrl.person.name }}</h4>',
        bindings: {
            person: '<'
        }
    })
```

- Parent component HTML:

```
<people-item ng-repeat="person in $ctrl.persons"
    person="person"></people-item>
```

Input Bindings: '<', '@'

- Passing data down from parent component to the child component
- One-way binding: '<'
- String literal binding: '@'

```
templateUrl: '......',
bindings: {
    person: '<',            // one-way binding
    comment: '@',           // special case: string literals
    myTitle: '<'            // alternative: double quoted strings
})
```

- Parent component HTML:

```
<people-detail person="person"
    comment="Just some commment"
    my-title="'Mentor'"></people-detail>
```

Output Bindings: '&'

- Passing events up from the child component to the parent component
- Beware: pass event parameters as JS object (name/value pairs)!

```
angular.module('app.people')
    .component('peopleList', {
       templateUrl: '...',
       controller: peopleListController
       bindings: {
           onClick: '&'
       }
   })

<li ng-repeat="person in $ctrl.people">
    {{ person.name }}
    <button ng-click="$ctrl.onClick({person: person})">see
</button></li>
```

- Parent component HTML:

```
<people-list on-click="$ctrl.onClick(person)"></people-list>
```

Two-way Bindings: '='

- Two-way sharing of data between child and parent component
- Changes in child component are reflected in the parent component
- … and vice versa
- Do not use without good reason: performance-wise expensive (watchers)

```
templateUrl: '......',
bindings: {
    item: '='
})
```

URL ROUTING

Hack Your Future

URL Routing

- Swapping views in the DOM by changing the URL
- No **`ng-show`**, **`ng-hide`** or **`ng-if`** involved
- Angular 1.4 router **`ngRoute`**: no further development
- Third-party router AngularUI router (**`ui.router`**): popular and advanced alternative
  - *Inserts active view inside* **`ui-view`** *directive*
- Example **`index.html`**:

```
...
<body ng-app='app'>
    <ui-view></ui-view>
</body>
...
```

`ui.router` Configuration

- Route to component

```
angular.module('app', ['ui.router', 'app.people'])
```

```
angular.module('app.people')
    .config(routing)

function routing($stateProvider) {
    $stateProvider
        .state('list', {
            url: '/',
            component: 'peopleList'
        })
        .state('person', {
            url: '/person/:id',
            component: 'personDetail'
        })
}
```

Changing Router State

- For example, to URL: ‘/person/:id’

```
<button ui-sref="person({id: person.id})">see</button>
```

```
<a ui-sref="person({id: person.id})">see</a>
```

```
<a ng-href="person/{{id}}">see</a>
```

```
function myController($state) {
    var vm = this
    vm.onClick = function(id) {
        $state.go('person', {id: id})
    }
}
```

Accessing State Parameters

- Use the `$stateParams` service from `ui.router`

```
angular.module('app.people')
    .component('personDetail', {
        templateUrl: 'app/people/detail/detail.component.html',
        controller: peopleDetailController
    })

function peopleDetailController($stateParams, peopleService) {
    var vm = this

    peopleService.getById($stateParams.id)
        .then(function(person) {
            vm.person = person
        })
}
```

URL Routing and Deep Linking

- Each route is responsible for its own data
- Make no assumptions on how the route was reached
- Users may deep-link to your route or use the browser’s navigation buttons or the browser’s history function

ANGULAR FILTERS

Hack Your Future

Angular Filters

- Angular filters transform data before it is processed by a component
- They can help to reduce the logic in controllers
- Most often used in HTML, but can also be used in code (**`$filter`** service)
- You can roll your own

Examples of standard Angular Filters

```
<ul>
  <li ng-repeat="person in $ctrl.people | limitTo:$ctrl.limit">
    {{ person.name }}
  </li>
</ul>
```

```
<li ng-repeat="person in $ctrl.people | orderBy:'name'">
```

```
<li ng-repeat="person in $ctrl.people | orderBy:'name' | limitTo:5">
```

```
<input type="text" ng-model="$ctrl.filterText">
<li ng-repeat="person in $ctrl.people | filter:{$: $ctrl.filterText}">
```

Available Filters

- Standard Angular filters
  - *currency*
  - *date*
  - *filter*
  - *limitTo*
  - *lowercase*
  - *number*
  - *orderBy*
  - *uppercase*

- Third-party library **`angular-filter`**
  - *Collection filters (after, afterWhere, …)*
  - *String filters (endsWith, repeat, …)*
  - *Math filters (min, max, …)*
  - *Boolean filters (isNull, isDefined, …)*

`https://github.com/a8m/angular-filter`