Directives:
Dynamic HTML

“Why Angular?” In Terms Of Directives

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

“Why Angular?” In Terms Of Directives

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

Typical Angular HTML

```
<ol>
  <list-item ng-repeat="item in list.items">
  </list-item>
</ol>
```

Directive

Marker on a DOM element that tells Angular's HTML compiler to attach a specified behavior to that DOM element

- The compiler can even transform/change the DOM elements and its children
- A marker can be attribute, element name, comment or CSS class

Step 1: Register Directive

```
angular.module('app', [])
.controller('MyCtrl', MyCtrl)
.directive('myTag', MyTag);
```

Normalized name
that will appear in HTML

Factory function – returns DDO:
Directive **D**efinition **O**bject

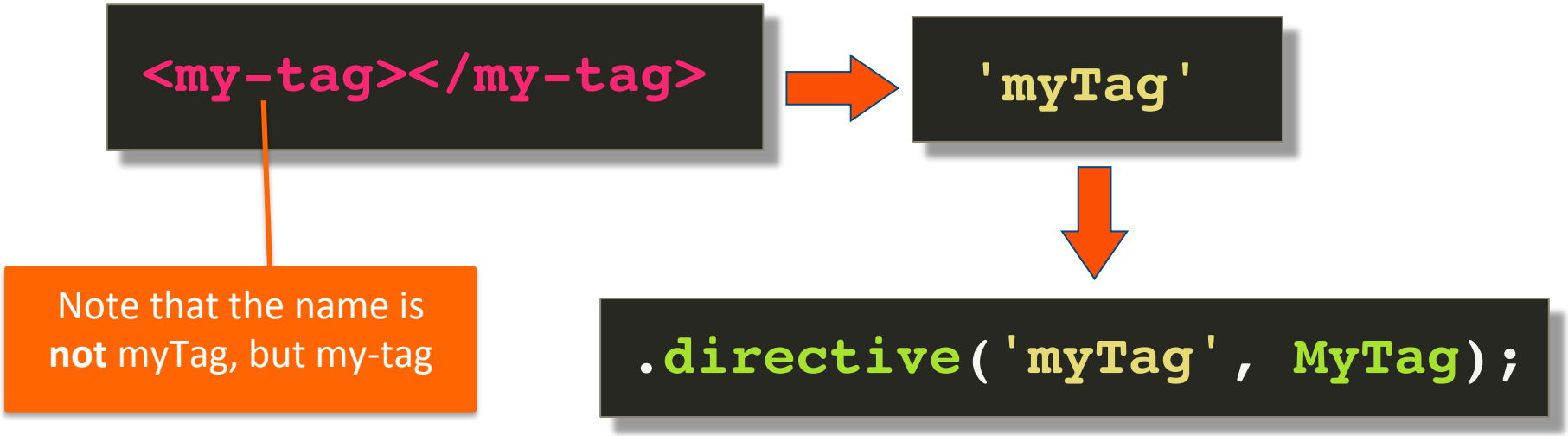
Step 2: Define Factory Function

```
MyTag.$inject = […]
function MyTag(…) {
  var ddo = {
    template: 'Hello World!'
    ...
  };

  return ddo;
}
```

Inject other services, controller, etc. as usual

Step 3: Use In HTML



Summary

- Directive is a marker in HTML that Angular compiles into some behavior
  - It can also change the HTML elements themselves
- Register name of directive using (normalized) camelCase
- Registered factory function must return a DDO
  - The factory function gets invoked only once
- With custom directives, our HTML coding becomes
  - Reusable
  - Semantically relevant to the actual web app we're building!