**AngularJS Checklist**

* Keep all similar directive JS and HTML (template) files in single folder under newly created directive folder.
* All java backend interaction API endpoints must be specified only in separate JSON file. It will be easy way to maintain and re-use it.
* Follow camel-case with no-hyphens when naming file or angular components like controller, service, factory, filter, directive, etc. Also naming convention given at the bottom area in the code.
* Controller, directive file name should match exactly with the respective controller-name, directive-name until file contains multiple angular components.
* Avoid hardcoding constant values, instead of move the same to separate JSON (constant.json).
* Directive scope should be restricted to minimum scope if necessary.
* Append common services in separate file (angularServices.js).
* One should avoid using controllers to do some of the following:
  + Manipulate DOM
  + Format input
  + Filter output
  + Share code or state across different controllers
  + Manage the life-cycle of other components (for example, to create service instances)
* A controller should contain only the business logic needed for a single view. The functionality should rather be moved to services and these services should be injected into the controllers using dependency injection.
* The recommended way to declare the controller function is to use the array notation because, it protects against minification.
* Use ‘strict’ mode for all controllers.
* Use switch case instead of multiple if-else conditions
* Create directive/services/factories for reusability.
* Prefer using directives through tag name and attributes over comment and class names.
* While creating directives, it is recommended to prefix your own directive names to avoid collisions with future standard.
* Avoid using jQuery or any other framework code inside controllers
* Names of variables, properties and methods should not start with $. The $ prefix is reserved for AngularJS internals
* Use dependency injection to inject global wrappers.
* Put AngularJS specific directives after standard attributes. By using this method, it will be easier to read and maintain the code.
* Naming conventions:
  + **Modules**: lowerCamelCase; **Example**: myApp
  + **Controllers**: UpperCamelCase + Controller; **Example**: MyController
  + **Directives**: lowerCamelCase; **Example**: myDirective
  + **Filters/Factories**: lowerCamelCase; **Example**: myFilter
  + **Services**: lowerCamelCase; **Example**: myService
* Refer mentioned keywords with respective place when dynamic values are passed.
* ng-src **-->** src,
* ng-href **-->** href,
* ng-style **-->** style, etc.

**Reason**: They make sure markup remains valid, even when expression value is not available yet.

* Use $ services provided by AngularJS whenever possible. **Example**: $http and $timeout can be used rather than http and timeout.
* use $log service. use $log.debug() to write debug messages to the console. **Reason**: allows you to turn off debug logging in production.
* use ng-if instead of ng-show.

**Good Practice**

* **Good to follow:** I18n is a must for the application, so move all UI display labels to a separate JSON (resource.json). Don’t use hardcoded text in the application. **Sitecore point of view:** This is not possible for Sitecore based on projects, since the label content comes from them.
* **Good to follow:** Don’t bind all the variables to scope/this. Don’t leave undefined variables. But It’s not mandatory. Sometimes it’s good to bind all the variables.
* **Good to follow:** Write small modular functions. But sometimes it’s not feasible always to write it in small module, Depends on the context.