# Angular JS 学习笔记

Angular Intro

1. AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.
2. AngularJS extends HTML with **ng-directives**.
3. The **ng-app** directive defines an AngularJS application.
4. The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
5. The **ng-bind** directive binds application data to the HTML view.
6. AngularJS expressions are written inside double braces: **{{ expression }}**.
7. AngularJS expressions bind AngularJS data to HTML the same way as the **ng-bind** directive.
8. AngularJS **modules** define AngularJS applications.

AngularJS **controllers** control AngularJS applications.

The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

AngularJS Expression

1. AngularJS expressions can be written inside double braces: {{ *expression* }} or inside a directive: ng-bind="*expression*".
2. AngularJS Expressions vs. JavaScript Expressions
3. Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
4. Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
5. AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
6. AngularJS expressions support filters, while JavaScript expressions do not.

AngularJS Modules

1. An AngularJS module defines an application.
2. The module is a container for the different parts of an application.
3. The module is a container for the application controllers.
4. Controllers always belong to a module
5. The [] parameter in the module definition can be used to define dependent modules. Without the [] parameter, you are not creating a new module, but retrieving an existing one.

AngularJS Directives

1. AngularJS lets you extend HTML with new attributes called **Directives**.
2. AngularJS has a set of built-in directives which offers functionality to your applications.
3. AngularJS also lets you define your own directives.

The ng-app Directive:

1. The ng-app directive defines the **root element** of an AngularJS application.
2. The ng-app directive will **auto-bootstrap** (automatically initialize) the application when a web page is loaded.

The ng-init Directive:

The ng-init directive defines **initial values** for an AngularJS application.

Normally, you will not use ng-init. You will use a controller or module instead.

The ng-model Directive:

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

The ng-model directive can also:

* Provide type validation for application data (number, email, required).
* Provide status for application data (invalid, dirty, touched, error).
* Provide CSS classes for HTML elements.
* Bind HTML elements to HTML forms.

AngularJS ng-model Directive

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

AngularJS Data Binding

1. You can use the ng-bind directive, which will bind the innerHTML of the element to the specified model property:

**Example:** <p ng-bind="firstname"></p>

1. You can also use double braces {{ }} to display content from the model:

**Example:** <p>First name: {{firstname}}</p>

1. Use the ng-model directive to bind data from the model to the view on HTML controls (input, select, textarea)

**Example:** <input ng-model="firstname">

AngularJS Controllers

1. AngularJS controllers **control the data** of AngularJS applications.
2. The **ng-controller** directive defines the application controller.
3. A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

AngularJS Scope

1. The scope is the binding part between the HTML (view) and the JavaScript (controller).
2. The scope is an object with the available properties and methods.
3. The scope is available for both the view and the controller.
4. If we consider an AngularJS application to consist of:

* View, which is the HTML.
* Model, which is the data available for the current view.
* Controller, which is the JavaScript function that makes/changes/removes/controls the data.

Then the scope is the Model.

The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

1. Root Scope

* All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.
* The rootScope is available in the entire application.
* If a variable has the same name in both the current scope and in the rootScope, the application use the one in the current scope.

AngularJS Filters

AngularJS provides filters to transform data:

* currency Format a number to a currency format.
* date Format a date to a specified format.
* filter Select a subset of items from an array.
* json Format an object to a JSON string.
* limitTo Limits an array/string, into a specified number of elements/characters.
* lowercase Format a string to lower case.
* number Format a number to a string.
* orderBy Orders an array by an expression.
* uppercase Format a string to upper case.

AngularJS Services

What is a Service?

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application. AngularJS has about 30 built-in services. One of them is the $location service. The $location service has methods which return information about the location of the current web page

Why use Services?

For many services, like the $location service, it seems like you could use objects that are already in the DOM, like the window.location object, and you could, but it would have some limitations, at least for your AngularJS application.

AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the $location service instead of the window.location object.

AngularJS AJAX - $http

$http is an AngularJS service for reading data from remote servers.

Methods:

The .get method is a shortcut method of the $http service. There are several shortcut methods:

* .delete()
* .get()
* .head()
* .jsonp()
* .patch()
* .post()
* .put()

The methods above are all shortcuts of calling the $http service:

Properties:

The response from the server is an object with these properties:

.config the object used to generate the request.

.data a string, or an object, carrying the response from the server.

.headers a function to use to get header information.

.status a number defining the HTTP status.

.statusText a string defining the HTTP status.

AngularJS Select Boxes

ng-options vs ng-repeat:

Because the ng-repeat directive repeats a block of HTML code for each item in an array, it can be used to create options in a dropdown list, but the ng-options directive was made especially for filling a dropdown list with options, and has at least one important advantage:

Dropdowns made with ng-options allows the selected value to be an object, while dropdowns made from ng-repeat has to be a string.

AngularJS Events

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

* ng-blur
* ng-change
* ng-click
* ng-copy
* ng-cut
* ng-dblclick
* ng-focus
* ng-keydown
* ng-keypress
* ng-keyup
* ng-mousedown
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-mouseup
* ng-paste

An AngularJS event will not overwrite an HTML event, both events will be executed.

AngularJS Forms

Input controls are the HTML input elements:

* input elements
* select elements
* button elements
* textarea elements

AngularJS Form Validation

1. AngularJS offers client-side form validation.
2. AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
3. AngularJS also holds information about whether they have been touched, or modified, or not.
4. You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

Client-side validation cannot alone secure user input. Server side validation is also necessary.

1. Form State and Input State:

AngularJS is constantly updating the state of both the form and the input fields.

**Input fields have the following states:**

* $untouched The field has not been touched yet
* $touched The field has been touched
* $pristine The field has not been modified yet
* $dirty The field has been modified
* $invalid The field content is not valid
* $valid The field content is valid

They are all properties of the input field, and are either true or false.

**Forms have the following states:**

* $pristine No fields have been modified yet
* $dirty One or more have been modified
* $invalid The form content is not valid
* $valid The form content is valid
* $submitted The form is submitted

They are all properties of the form, and are either true or false.

1. CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

* ng-untouched The field has not been touched yet
* ng-touched The field has been touched
* ng-pristine The field has not been modified yet
* ng-dirty The field has been modified
* ng-valid The field content is valid
* ng-invalid The field content is not valid
* ng-valid-*key* One *key* for each validation.

Example: ng-valid-required, useful when there are more than one thing that must be validated

* ng-invalid-*key* Example: ng-invalid-required

The following classes are added to, or removed from, forms:

* ng-pristine No fields has not been modified yet
* ng-dirty One or more fields has been modified
* ng-valid The form content is valid
* ng-invalid The form content is not valid
* ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-*key* Example: ng-invalid-required

The classes are removed if the value they represent is false.