5 Awesome AngularJS Features

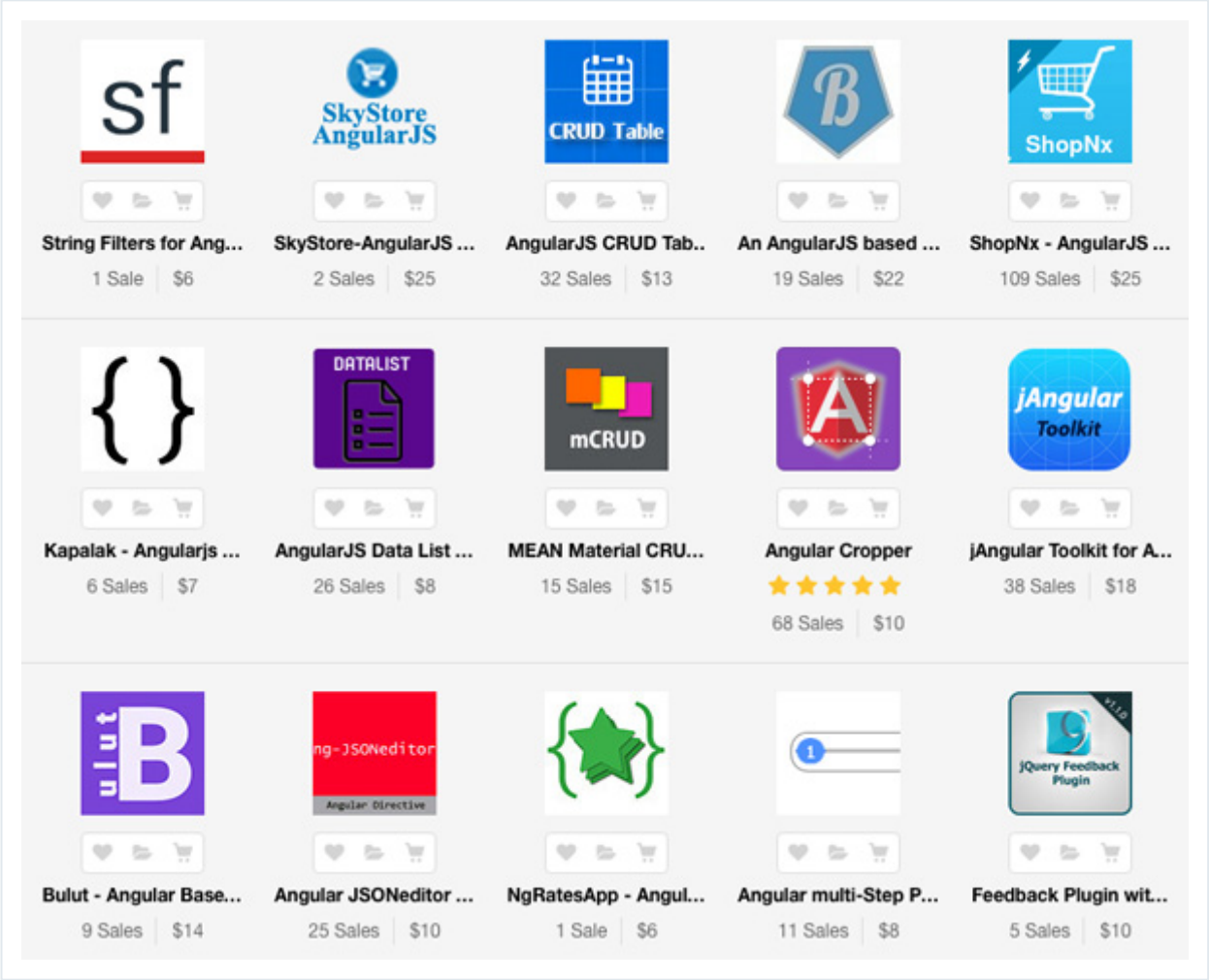
by Lukas Ruebbelke 5 Jul 2012

Difficulty: Intermediate Length: Quick Languages: English

Web Development JavaScript AngularJS

AngularJS is a great JavaScript framework that has some very compelling features for not only developers, but designers as well! In this tutorial, we will cover what I consider to be the most essential features, and how they can help make your next web application awesome.

To get an idea of what you can do with AngularJS, check out the range of AngularJS items on Envato Market. You can find an image cropper, an eCommerce web application, a JSON editor, and much more.



AngularJS items on Envato Market

The Elevator Pitch: AngularJS in a Nutshell

> *AngularJS is a new, powerful, client-side technology that provides a way of accomplishing really powerful things in a way that embraces and extends HTML, CSS and JavaScript, while shoring up some of its glaring deficiencies. It is what HTML would have been, had it been built for dynamic content.*

In this article, we will cover a few of the most important AngularJS concepts to get the "big picture." It is my goal that, after seeing some of these features, you will be excited enough to go and build something fun with AngularJS.

Feature 1: Two Way Data-Binding

> *Think of your model as the single-source-of-truth for your application. Your model is where you go to to read or update anything in your application.*

Data-binding is probably the coolest and most useful feature in AngularJS. It will save you from writing a considerable amount of boilerplate code. A typical web application may contain up to 80% of its code base, dedicated to traversing, manipulating, and listening to the DOM. Data-binding makes this code disappear, so you can focus on your application.

Think of your model as the single-source-of-truth for your application. Your model is where you go to to read or update anything in your application. The data-binding directives provide a projection of your model to the application view. This projection is seamless, and occurs without any effort from you.

Traditionally, when the model changes, the developer is responsible for manually manipulating the DOM elements and attributes to reflect these changes. This is a two-way street. In one direction, the model changes drive change in DOM elements. In the other, DOM element changes necessitate changes in the model. This is further complicated by user interaction, since the developer is then responsible for interpreting the interactions, merging them into a model, and updating the view. This is a very manual and cumbersome process, which becomes difficult to control, as an application grows in size and complexity.

There must be a better way! AngularJS' two-way data-binding handles the synchronization between the DOM and the model, and vice versa.

Here is a simple example, which demonstrates how to bind an input value to an `<h1>` element.

```
<!doctype html>
<html ng-app>
  <head>
    <script src="http://code.angularjs.org/angular-1.0.0rc10.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
```

```
10        <hr>
11        <h1>Hello, {{yourName}}!</h1>
12      </div>
13    </body>
14  </html>
```

This is extremely simple to set up, and almost magical…

Feature 2: Templates

> *It's important to realize that at no point does AngularJS manipulate the template as strings. It's all the browser DOM.*

In AngularJS, a template is just plain-old-HTML. The HTML vocabulary is extended, to contain instructions on how the model should be projected into the view.

The HTML templates are parsed by the browser into the DOM. The DOM then becomes the input to the AngularJS compiler. AngularJS traverses the DOM template for rendering instructions, which are called directives. Collectively, the directives are responsible for setting up the data-binding for your application view.

It is important to realize that at no point does AngularJS manipulate the template as strings. The input to AngularJS is browser DOM and not an HTML string. The data-bindings are DOM transformations, not string concatenations or `innerHTML` changes. Using the DOM as the input, rather than strings, is the biggest differentiation AngularJS has from its sibling frameworks. Using the DOM is what allows you to extend the directive vocabulary and build your own directives, or even abstract them into reusable components!

One of the greatest advantages to this approach is that it creates a tight workflow between designers and developers. Designers can mark up their HTML as they normally would, and then developers take the baton and hook in functionality, via bindings with very little effort.

Here is an example where I am using the `ng-repeat` directive to loop over the `images` array and populate what is essentially an `img` template.

```
function AlbumCtrl($scope) {
        scope.images = [
                {"thumbnail":"img/image_01.png", "description":"Image 01
                {"thumbnail":"img/image_02.png", "description":"Image 02
                {"thumbnail":"img/image_03.png", "description":"Image 03
                {"thumbnail":"img/image_04.png", "description":"Image 04
                {"thumbnail":"img/image_05.png", "description":"Image 05
        ];
}
```

```
<div ng-controller="AlbumCtrl">
  <ul>
    <li ng-repeat="image in images">
      <img ng-src="{{image.thumbnail}}" alt="{{image.description}}">
    </li>
  </ul>
</div>
```

It is also worth mentioning, as a side note, that AngularJS does not force you to learn a new syntax or extract your templates from your application.

Feature 3: MVC

AngularJS incorporates the basic principles behind the original MVC software design pattern into how it builds client-side web applications.

The MVC or Model-View-Controller pattern means a lot of different things to different people. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel).

The Model

The *model* is simply the data in the application. The *model* is just plain old JavaScript objects. There is no need to inherit from framework classes, wrap it in proxy objects, or use special getter/setter methods to access it. The fact that we are dealing with vanilla JavaScript is a really nice feature, which cuts down on the application boilerplate.

The ViewModel

A *viewmodel* is an object that provides specific data and methods to maintain specific views.

The *viewmodel* is the `$scope` object that lives within the AngularJS application. `$scope` is just a simple JavaScript object with a small API designed to detect and broadcast changes to its state.

The Controller

The *controller* is responsible for setting initial state and augmenting `$scope` with methods to control behavior. It is worth noting that the *controller* does not store state and does not interact with remote services.

The View

The *view* is the HTML that exists after AngularJS has parsed and compiled the HTML to include rendered markup and bindings.

This division creates a solid foundation to architect your application. The `$scope` has a reference to the data, the *controller* defines behavior, and the *view* handles the layout and handing off interaction to the *controller* to respond accordingly.

Feature 4: Dependency Injection

AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

Dependency Injection (DI) allows you to ask for your dependencies, rather than having to go look for them or make them yourself. Think of it as a way of saying "Hey I need X', and the DI is responsible for creating and providing it for you.

To gain access to core AngularJS services, it is simply a matter of adding that service as a parameter; AngularJS will detect that you need that service and provide an instance for you.

```
function EditCtrl($scope, $location, $routeParams) {
     // Something clever here...
}
```

You are also able to define your own custom services and make those available for injection as well.

```
angular.
    module('MyServiceModule', []).
    factory('notify', ['$window', function (win) {
    return function (msg) {
        win.alert(msg);
    };
}]);

function myController(scope, notifyService) {
    scope.callNotify = function (msg) {
        notifyService(msg);
    };
}

myController.$inject = ['$scope', 'notify'];
```

Feature 5: Directives

Directives are my personal favorite feature of AngularJS. Have you ever wished that your browser would do new tricks for you? Well, now it can! This is one of my favorite parts of AngularJS. It is also probably the most challenging aspect of AngularJS.

Directives can be used to create custom HTML tags that serve as new, custom widgets. They can also be used to "decorate" elements with behavior and manipulate DOM attributes in interesting ways.

Here is a simple example of a directive that listens for an event and updates its `$scope`, accordingly.

```
myModule.directive('myComponent', function(mySharedService) {
    return {
        restrict: 'E',
        controller: function($scope, $attrs, mySharedService) {
            $scope.$on('handleBroadcast', function() {
                $scope.message = 'Directive: ' + mySharedService.messag
            });
        },
        replace: true,
        template: '<input>'
    };
});
```

Then, you can use this custom directive, like so.

```
1 <my-component ng-model="message"></my-component>
```

Creating your application as a composition of discrete components makes it incredibly easy to add, update or delete functionality as needed.

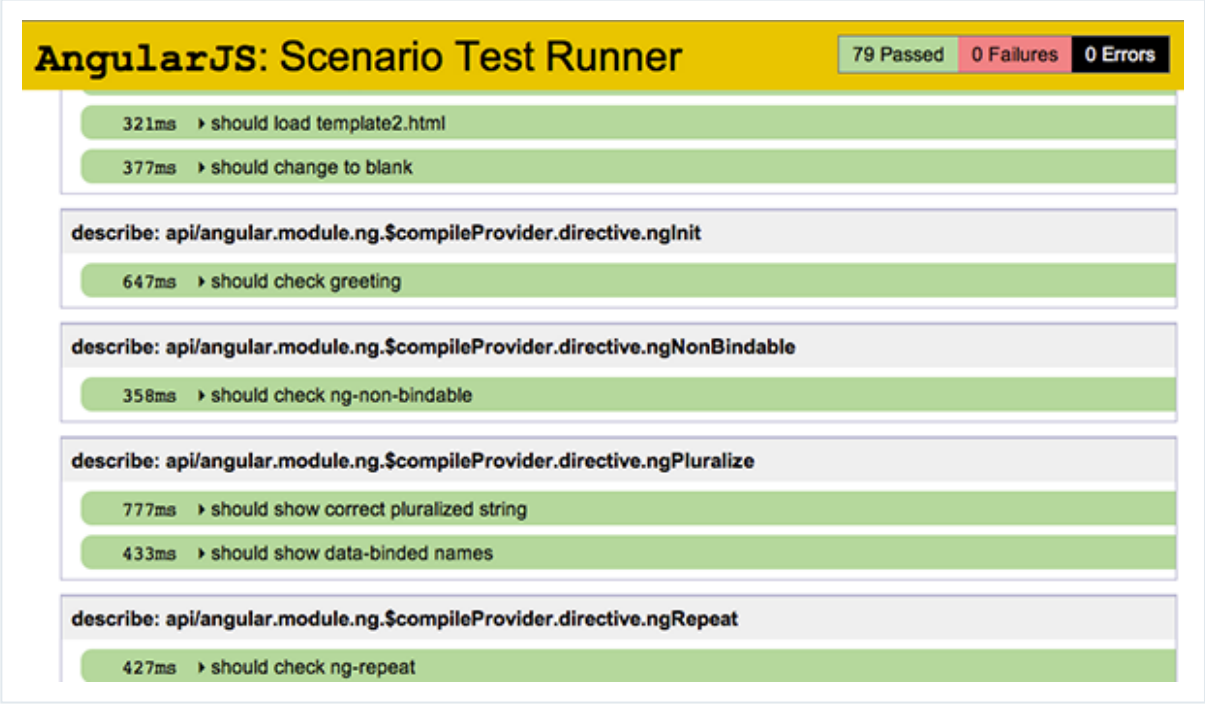
Bonus Feature: Testing

The AngularJS team feels very strongly that any code written in JavaScript needs to come with a strong set of tests. They have designed AngularJS with testability in mind, so that it makes testing your AngularJS applications as easy as possible. So there's no excuse for not doing it.

Given the fact that JavaScript is dynamic and interpreted, rather than compiled, it is extremely important for developers to adopt a disciplined mindset for writing tests.

AngularJS is written entirely from the ground up to be testable. It even comes with an end-to-end and unit test runner setup. If you would like to see this in action, go check out the angular-seed project at https://github.com/angular/angular-seed.

Once you have the seed project, it's a cinch to run the tests against it. Here is what the output looks like:



The API documentation is full of end-to-end tests that do an incredible job of illustrating how a certain part of the framework should work. After a while, I found myself going straight to the tests to see how something worked, and then maybe reading the rest of the documentation to figure something out.

Conclusion

We have covered six of the many AngularJS features that I love. I believe that these six features are essential for not only getting up and running with AngularJS, but also putting your application together in a way that is maintainable and extensible.

The website for AngularJS, http://angularjs.org, has plenty of working examples and plenty of excellent documentation. I also recommend checking out the amazing community on the AngularJS mailing list.

Let me know what you think!

Lukas Ruebbelke

N/A

Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by native

Join the discussion…

Guest • 3 years ago

To those wondering, "Should I use AngularJS or not?"

From a guy whose been in this industry 30 years, with the last 15 dedicated to web application development:

1) Stop looking for the perfect technology stack that you will be able to use from here until you retire. It doesn't exist. I've been through 4 generations of technology stacks over the past 15 years as the Internet, languages, tools, hardware, etc. have all evolved, they have rendered my previous architectural decisions out-dated.
2) You don't have to choose just one JS framework. There's no genie here saying you get one wish and one wish only. Develop your next project with AngularJS. If you like it, keep developing with it. If you don't, try another. There is nothing wrong with having each project built with a different set of frameworks. Fact of that matter, that's going to happen anyway (see my point #1 as to why).
3) Use AngularJS, knowing full well that, in all likelihood, something better will evolve out of it and, in 5 years, AngularJS will be passé and you'll abandon it in favor of an even better framework. That's a good thing - not a bad thing. The use of these frameworks expose their weakness, which allows AngularJS to improve upon itself, or someone else will take the lessons learned from it and build a framework twice as good.
4) Finally, the more frameworks you use, the greater your perspective and the more flexible your mind becomes, making it quicker and easier to move from one technology stack to another without fear or intimidation.

140 ▲ ▼ • Reply • Share ›

Luis Alberto Romero Calderon ➚ Guest • 3 years ago

:') nice words

5 ▲ ▼ • Reply • Share ›

Lê Võ Hữu Tài ➚ Guest • 3 years ago

Very nice words it solves all my "... or ..., ... vs ..." question

3 ▲ ▼ • Reply • Share ›

ño ➚ Guest • a year ago

100% agree. And I'd add that you should stay closer to Angular than others right now because it's on Google hands and they're ahead of the curve in web innovations these days. Not that you should only focus on it, but stay aware.

1 ▲ ▼ • Reply • Share ›

vuthy ➚ Guest • a year ago

great idea, i like it.

1 ▲ ▼ • Reply • Share ›

Kyle Harrison ➚ Guest • 2 years ago

REALLY couldn't agree more

1 ^ | v • Reply • Share ›

Ali Baba • 4 years ago

Looks like there is new JS library comes every week.

The Nutshell can be the same for all of them:

BlahBlahJS is a new, powerful, client-side technology that provides a way of accomplishing really powerful things in a way that embraces and extends blah blah blah blah blah blah blah blah blah blah blah blah

9 ^ | v • Reply • Share ›

Tad Christiansen ➚ Ali Baba • 4 years ago

Ali Baba

Blah blah blah, some new JS ilbrary...innovation stinks I still want to write all my code with punch cards. :)

6 ^ | v • Reply • Share ›

Box Office Buz ➚ Tad Christiansen • 2 years ago

You should have given him this.

4 • Reply • Share ›

28inch ➚ Ali Baba • 4 years ago

And every framework post gets a comment BITCHING about new frameworks.

"Looks like there is a new JS library comes every week blah blah blah blah"

I. You don`t have to read it.
II. You don`t have to use it.

Is it so hard to believe, that this one might have some features that others don`t? I`m not even mentioning impelemnting some of the features to your choice of library/ workflow.

3 • Reply • Share ›

[rb] ➚ Ali Baba • 4 years ago

Yeah and they usually have a whole heap of crap like all the ng-whatever attributes on html elements.
I dont want to pollute html with made up attributes and haveing handlebars style place holders like {{whatever}} in html code for replacement is all fine until js is disabled and the whole thing falls apart and you view all the placeholders.

1 • Reply • Share ›

Guest ➚ [rb] • 3 years ago

That's about as stupid as someone saying, "I'm not using the Java Spring framework. What the hell is up with all this IoC and DI crap they added into the language?" If it was up to guys like you, languages (and HTML) would never evolve because everyone should stick to the core library and never extend or improve it. Oddly enough, it wasn't until years later, after all the IoC containers cropped up, that Sun decided it was a good idea and came up with CDI. the problem is, if you stick with the core of anything, you stick to one company's set of ideas, which limits innovation.

3 • Reply • Share ›

Jimish Fotariya ➚ Guest • a year ago

Right buddy ! but core is best !

• Reply • Share ›

MySchizoBuddy ➚ [rb] • 4 years ago

This is true for EVERY SINGLE js framework. If you disable JS then the thing falls apart.
JS frameworks depend on JS being enabled.

jquery is mostly used for show and hide dom elements it isn't for making web apps. So disabling js in jquery web pages just disable the lame animations you added that's all.

Angular is completely different beast than jquery

3 • Reply • Share ›

Tad Christiansen ➚ MySchizoBuddy • 4 years ago

I know its been 5 months, but I just have clarify this.

"jquery is mostly used for show and hide dom elements it isn't for making web apps"

I totally disagree. JQuery does so much more than showing and hiding dom elements. It makes selecting and decorating elements so much easier. Not to mention all the useful ajax functionality.

"Angular is completely different beast than jquery"

True, but under the hood even Angular uses a light version of jquery.

5 ^ | v • Reply • Share ›

Alchemication ➚ Ali Baba • 4 years ago

True, I know the feeling...but you have to able to distinguish yourself - if it's worth it or not... Is Angular worth it? Yes for me, but is it for you?

Btw: Do you understand a concept of data binding? No offense, just a plain question...
Btw2: It's out there for 2 years (like somebody else already said) ...

1 ^ | v • Reply • Share ›

AGDM ➚ Ali Baba • 4 years ago

How can I vote this up? Basically I feel the same way.

1 ^ | v • Reply • Share ›

Jesus Bejarano ➚ AGDM • 4 years ago

I am in this group too sorry :(. but nevertheless it's look interesting.

1 ^ | v • Reply • Share ›

Tom Conlon ➚ Ali Baba • 4 years ago

@ali baba, @agdm, @Jesus Bejarano, @shane, @rory, @28inch

Angular has been out for two years now: see http://stackoverflow.com/tags/...

So, no, this doesn't come under the newbie JS library heading but is a solid and powerful offering.
Tom

^ | v • Reply • Share ›

Guest ➚ Tom Conlon • 4 years ago

Says here right in the header of the page "AngularJS is a new, powerful... "

^ | v • Reply • Share ›

Jesus Bejarano ➚ Tom Conlon • 4 years ago

The "new" label has nothing to do with the time that it have been created, is more about

the time that is now ganing awerness/popularity.

Reply • Share ›

Rory ➚ Ali Baba • 4 years ago

Haha. I was thinking the same thing.

It's got to the point where I just skim over these "new js library!" posts.

"Oh look. Another one"

Reply • Share ›

Shane ➚ Ali Baba • 4 years ago

I agree :)

Reply • Share ›

Tom ➚ Ali Baba • 4 years ago

So true...

Reply • Share ›

Pride.Chung • 4 years ago

LOL

So it's not just me have this felling. Recently I read a post that the author evaluated 12 JavaScript MVC frameworks and decided to choose Ember.js . I'm just thinking he only covered half of these frameworks .

Did we invent too much wheels ? Now I can't even decide which one to try, too many options is worse than no option.

8 Reply • Share ›

narkoz ➚ Pride.Chung • 4 years ago

You can't decide because you're ignorant.

11 Reply • Share ›

Guest ➚ Pride.Chung • 3 years ago

You're probably an iPhone user. They keep it simple: one size, and you can choose between black or white.

6 Reply • Share ›

Muhammad Azamuddin ➚ Pride.Chung • 3 years ago

Previously I'm thinking backbone as the most important javascript framework to learn, then I saw some of positive feeling about angular JS, most of the people who have that feeling comes from backbone and move to angular, so I think I have to learn angular. this article is helpful.

3 Reply • Share ›

zerocool81 ➚ Pride.Chung • 4 years ago

zeroc0ol81 ➚ Pride.Chung • 4 years ago

I'd suggest looking at http://addyosmani.github.com/t... that compare the same project written using the different frameworks.

2 ▲ | ▼ • Reply • Share ›

David Mair Spiess ➚ Pride.Chung • 4 years ago

Haha the same thing happend to me :)

▲ | ▼ • Reply • Share ›

Sam • 4 years ago

In the second example, scope.images needs to be $scope.images

Just in case someone is actually trying these out. =)

```
function AlbumCtrl($scope) {
$scope.images = [
{"thumbnail":"img/image_01.png", "description":"Image 01 description"},
{"thumbnail":"img/image_02.png", "description":"Image 02 description"},
{"thumbnail":"img/image_03.png", "description":"Image 03 description"},
{"thumbnail":"img/image_04.png", "description":"Image 04 description"},
{"thumbnail":"img/image_05.png", "description":"Image 05 description"}
];
}
```

10 ▲ | ▼ • Reply • Share ›

David Mair Spiess • 4 years ago

Seems to be less boilerplate code than in backbone. Two way data binding is awesome! ember.js seems also promising, but lacks of solid documentation, so i will give angularJS a try. thanks for the introduction.
its quite difficult to choose the right one for the job as there exist so many :)

4 ▲ | ▼ • Reply • Share ›

sadfsdaf • 2 years ago

fdsf ds fsdf
dsfsdf sadf
sda fsd
f sdf asd

2 ▲ | ▼ • Reply • Share ›

Phil Colson • 3 years ago

LOL at that elevator pitch! I have the exact same thing word for word to describe jQuery! "jQuery is a new, powerful, client-side technology that provides a way of accomplishing really powerful things in a way that embraces and extends HTML, CSS and JavaScript, while shoring up some of its glaring deficiencies. It is what HTML would have been, had it been built for dynamic content."

2 ▲ | ▼ • Reply • Share ›

Nasa Nguyen • 2 years ago

Full course: Single Page Applications with jQuery or Angular JS

1 ^ | v • Reply • Share ›

Mark Foote • 4 years ago

For those who are not quite as experienced writing with Angular, a demo page for your examples would really help out. I realize you might have to add just a bit more to make it meaningful, that would be good too! For example, if I understand correctly, your examples for dependency injection and directives are all about setting up an ability to invoke an alert?

Thanks.

1 ^ | v • Reply • Share ›

christ almighty • 4 years ago

Correct me if I'm wrong – disclaimer: I'm not wrong, so don't you dare f**king correct me, I'm christ almighty b*tch! – but the first 3 features are all part of MVC, so there's not 5 features there's 3.

Also, feature 5 is just a combination of features 2 and 4; possibly 1 and 3 as well, depending on your directive.

Let's get naked and crucified all up in this motherf**k!

2 ^ | v • Reply • Share ›

Guest ➚ christ almighty • 3 years ago

Try laying off the meth before you post.

6 ^ | v • Reply • Share ›

Don • 24 days ago

Indeed I find two way data-binding extremely helpful, thus, my favorite of angularJS. Regarding your writing on both #4 and #5, it would be very helpful if you can use two problem cases and address each of them with #4 and #5 respectively, thus, for angularJS beginners like me, the learning would be more practical. Thanks.

^ | v • Reply • Share ›

Harry Martin • a year ago

Excellent features mentioned here. I would also recommend you to make use of this advanced resource for learning AngularJS from scratch. Its interesting.

AngularJS tutorials

^ | v • Reply • Share ›

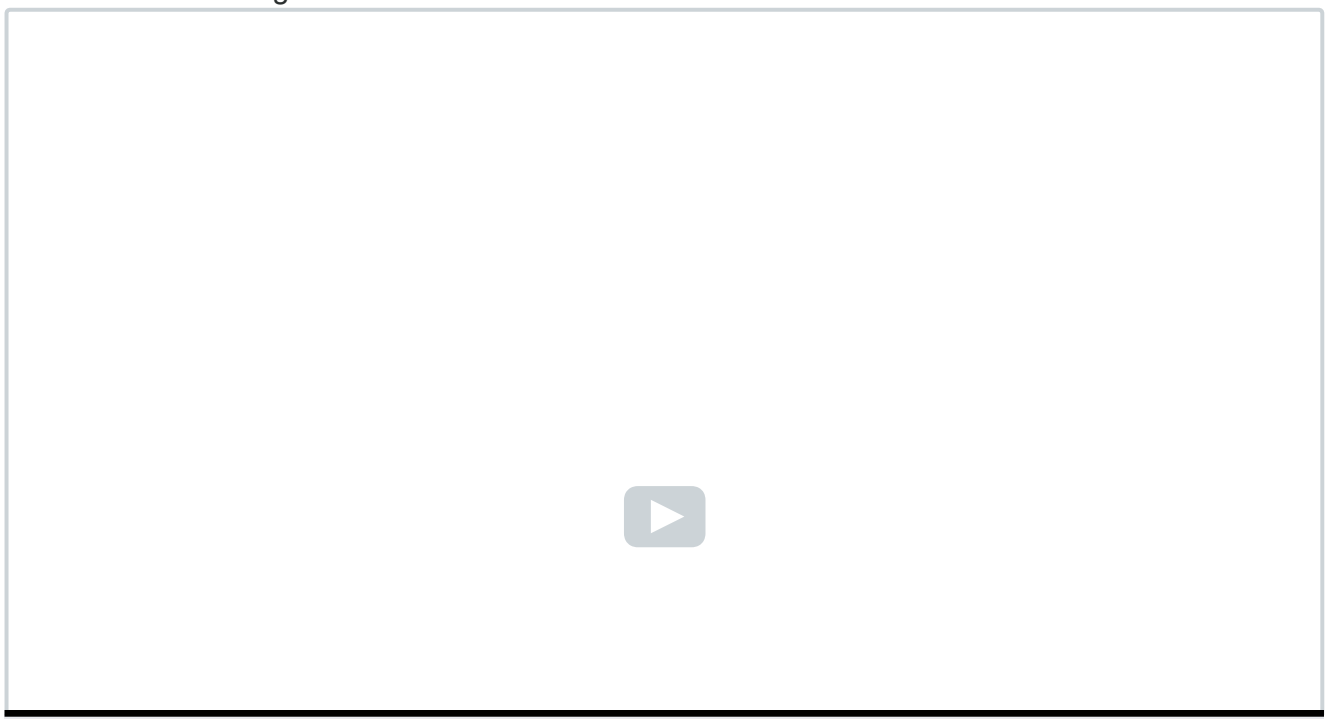
yangbai • a year ago

good!

^ | v • Reply • Share ›

Nasa Nguyen • a year ago

Zero to Hero with AngularJS at 2015



see more

^ | v • Reply • Share ›

Neha Nguyen • 2 years ago

AngularJS Custom Widgets and Architecture Best Practices

• Reply • Share ›

Gaurav Ashara • 2 years ago

Hello,
I am looking for angularjs with web component development in which wanted to build several component that work in front and also work in background. Also want to load components dynamically and manage events for that (plugable component) .

Follow Envato Tuts+