WEB DEVELOPMENT

Building a Web App From Scratch in AngularJS

by Leon Revill *27 Jun 2013* *84 Comments*

In a previous AngularJS tutorial I covered all the basics of how to get up and running with Angular in around 30 minutes. This tutorial will expand on what was covered there by creating a simple real world web application.

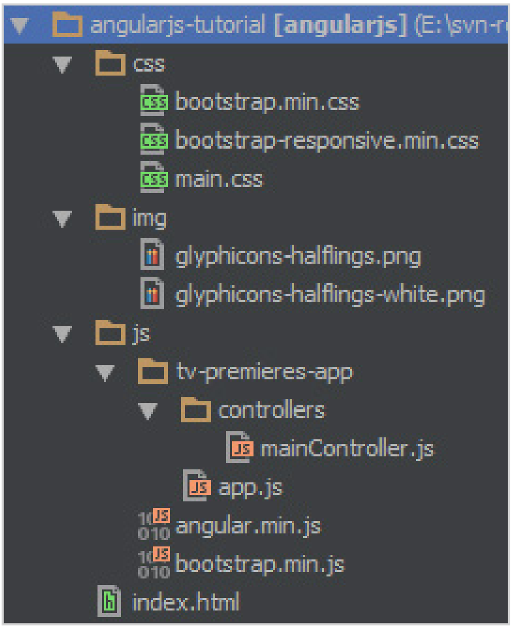
This simple web application will allow its users to view, search and filter TV Show Premieres for the next 30 days. As a keen series viewer, I am always looking for something new to watch when my favorite shows are off air, so I thought I would create an app to help me find what I am looking for.

Before we get started, you may want to take a look at the demo from above, to see what we will be creating in this tutorial.

Getting Started

To begin, we need a skeleton AngularJS application which already has all the required JavaScript and CSS to create the TV Show Premieres app. Go ahead and download this skeleton from the "download source files" button above.

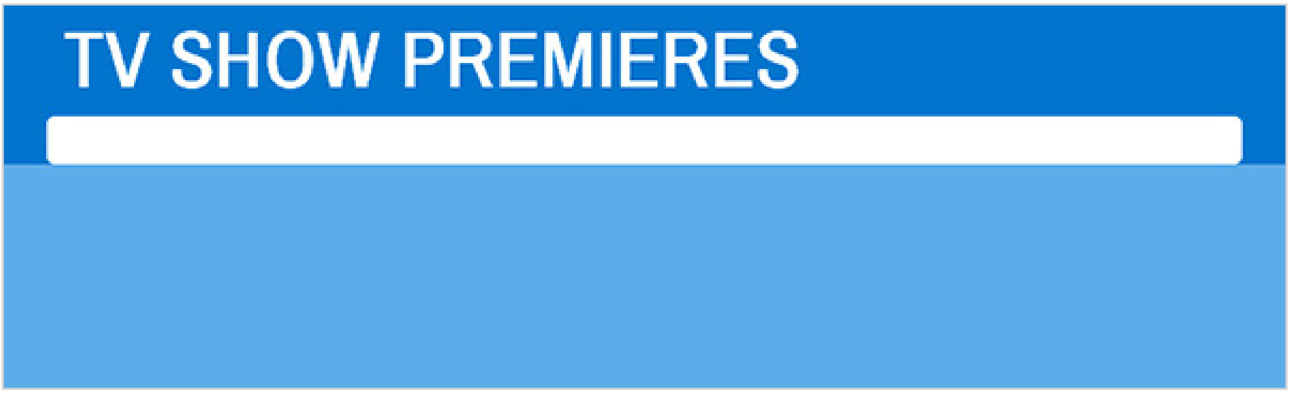
Once you have downloaded the files you should have a directory structure as shown below:



```
▼ angularjs-tutorial [angularjs] (E:\svn-r
  ▼ css
      bootstrap.min.css
      bootstrap-responsive.min.css
      main.css
  ▼ img
      glyphicons-halflings.png
      glyphicons-halflings-white.png
  ▼ js
    ▼ tv-premieres-app
      ▼ controllers
          mainController.js
        app.js
      angular.min.js
      bootstrap.min.js
    index.html
```

Looking at the directory structure and the included files you will see that we will be using Twitter Bootstrap to make our web app a little prettier, but this tutorial will not look at Twitter Bootstrap in any detail (learn more about Twitter Bootstrap). Additionally, this tutorial will not be covering how to setup a new AngularJS application as the aforementioned AngularJS tutorial already covers this in detail.

Upon opening `index.html`, with your browser of choice, you should see a very simple web page with just a title and some basic formatting as seen below:



Loading In Our Data

The first thing we are going to need to create our TV Show app, is information about TV shows. We are going to use an API provided by Trakt.tv. Before we can get started you are going to need an API key, you can register for one on their website.

Why use this API? Do I really have to register? We are using this API so our app will use real data and will actually provide some use once completed. Also, by using this API we do not need to go over any server side implementations within this tutorial and can focus completely on AngularJS. An extra couple of minutes to register for the API will be well worth it.

Now that you have your own API key, we can utilize the Trakt API to get some information on TV shows. We are going to use one of the available API calls for this tutorial, more information on this is available in the api docs. This API call will provide us with all the TV Show Premieres within a specified time frame.

Open `mainController.js` and modify it to match the below code:

```
app.controller("mainController", function($scope, $http){

    $scope.apiKey = "[YOUR API KEY HERE]";
    $scope.init = function() {
        //API requires a start date
        var today = new Date();
        //Create the date string and ensure leading zeros if required
        var apiDate = today.getFullYear() + ("0" + (today.getMonth() +
        $http.jsonp('http://api.trakt.tv/calendar/premieres.json/' + $s
            console.log(data);
        }).error(function(error) {

        });
    };

});
```

If you look within the `index.html` file, for the following line:

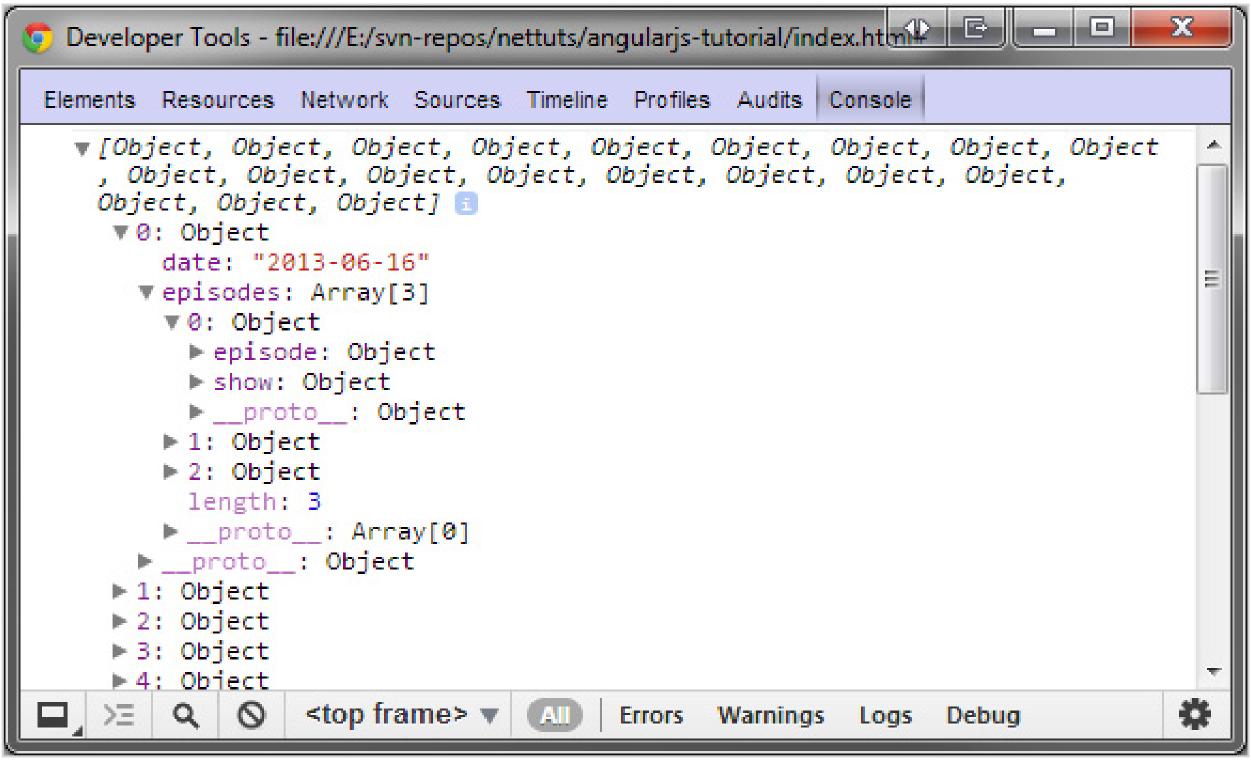
```
main-frame" ng-app="TVPremieresApp" ng-controller="mainController" ng-init="init()">
```

You will see that the `ng-init` method is calling the `init` function, this means that the `init()` function within our `mainController` will be called after the page has been loaded.

If you read the API documentation for the `calendar/premieres` method you will have seen that it takes three parameters, your API key, the start date (e.g. 20130616) and the number of days.

To provide all three parameters, we first need to get today's date using JavaScripts `Date()` method and format it to the API specified date format to create the `apiDate` string. Now that we have everything we need, we can create an `$http.jsonp` call to the API method. This will allow our web app to call a URL that is not within our own domain and receive some JSON data. Ensure that `?callback=JSON_CALLBACK` is prepended onto the request URI so that our attached `.success` callback function is called on response.

Within our `.success` function we then simply output the received data to the console. Open `index.html` within your browser and open the JavaScript console, you should see something like the following:


Developer Tools - file:///E:/svn-repos/nettuts/angularjs-tutorial/index.html
Elements Resources Network Sources Timeline Profiles Audits Console
<top frame> All Errors Warnings Logs Debug

```
▼[Object, Object, Object, Object, Object, Object, Object, Object, Object
 , Object, Object, Object, Object, Object, Object, Object, Object,
 Object, Object, Object]
  ▼0: Object
     date: "2013-06-16"
   ▼episodes: Array[3]
     ▼0: Object
       ▶episode: Object
       ▶show: Object
       ▶__proto__: Object
     ▶1: Object
     ▶2: Object
      length: 3
     ▶__proto__: Array[0]
   ▶__proto__: Object
  ▶1: Object
  ▶2: Object
  ▶3: Object
  ▶4: Object
```

This demonstrates that we are successfully performing a call to the Trakt API, authenticating with our API key and receiving some JSON data. Now that we have our TV show data, we can move on to the step.

Displaying Our Data

Processing the JSON Objects

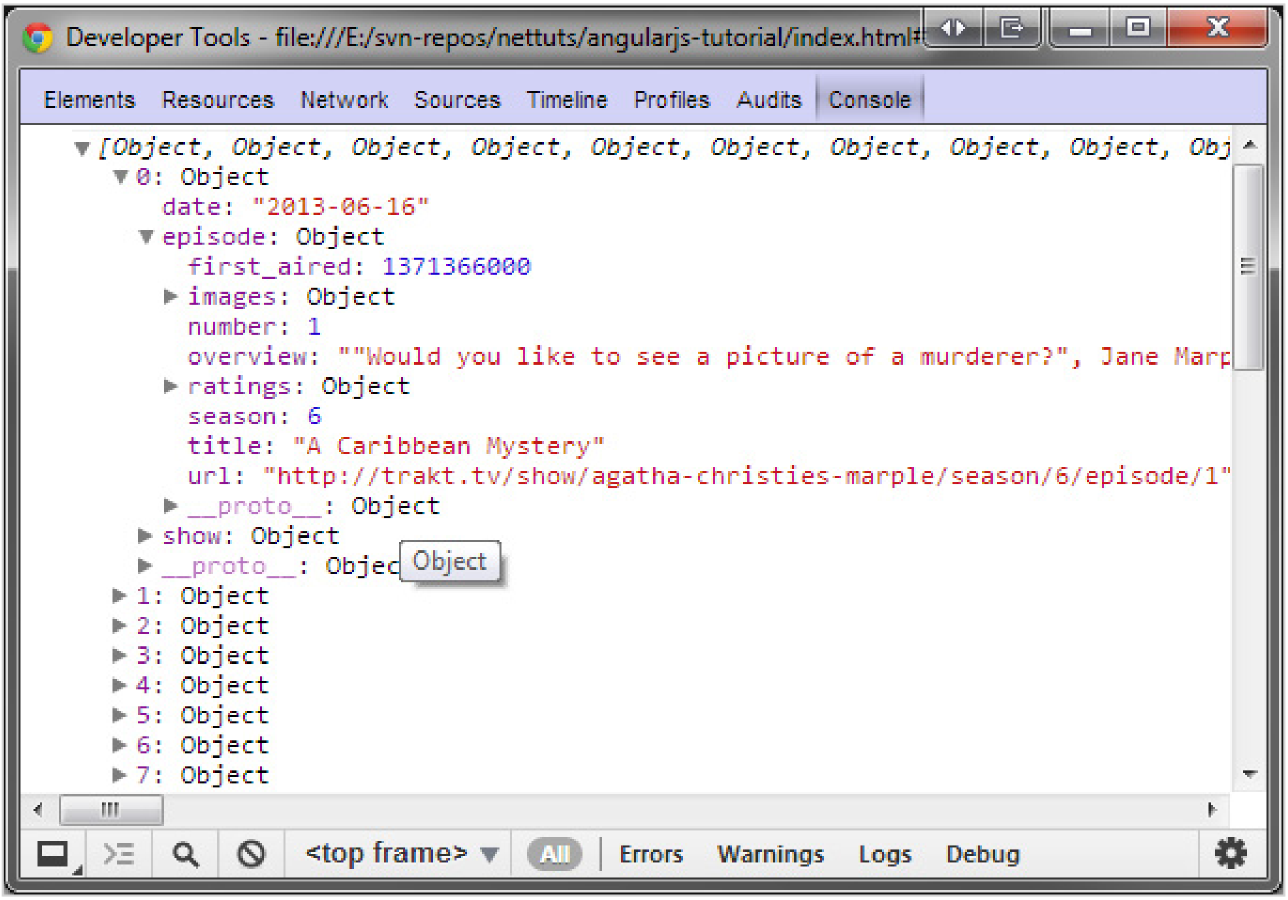
Before we can display our data, we need to process and store it. As the API returns the premiere episodes grouped by date, we want to remove this grouping and just create a single array with all the premiere episodes and their associated data. Modify `mainController.js` to be as follows:

```
app.controller("mainController", function($scope, $http){
    $scope.apiKey = "[YOUR API KEY]";
    $scope.results = [];
    $scope.init = function() {
        //API requires a start date
        var today = new Date();
        //Create the date string and ensure leading zeros if required
        var apiDate = today.getFullYear() + ("0" + (today.getMonth() +
        $http.jsonp('http://api.trakt.tv/calendar/premieres.json/' + $s
            //As we are getting our data from an external source, we ne
            //For each day, get all the episodes
            angular.forEach(data, function(value, index){
                //The API stores the full date separately from each epi
                var date = value.date;
                //For each episodes, add it to the results array
                angular.forEach(value.episodes, function(tvshow, index)
                    //Create a date string from the timestamp so we can
                    tvshow.date = date; //Attach the full date to each
                    $scope.results.push(tvshow);
                });
            });
        }).error(function(error) {

        });
    };
});
```

The above code is well commented and should be easy to follow, lets take a look at these changes. First, we declare a scope variable `$scope.results` as an array which will hold our processed results. We then use `angular.forEach` (which is similar to jQuery's `$.each` method for those who know it) to loop through each date group and store the date in a local `date` variable.

We then create another loop which loops through each of the TV shows within that date group, adds the locally stored date to the `tvshow` object and then finally adds each `tvshow` object to the `$scope.results` array. With all of this done, our `$scope.results` array will look like the following:



```
▼[Object, Object, Object, Object, Object, Object, Object, Object, Object, Obj
  ▼0: Object
     date: "2013-06-16"
    ▼episode: Object
       first_aired: 1371366000
      ▶images: Object
       number: 1
       overview: ""Would you like to see a picture of a murderer?", Jane Marp
      ▶ratings: Object
       season: 6
       title: "A Caribbean Mystery"
       url: "http://trakt.tv/show/agatha-christies-marple/season/6/episode/1"
      ▶__proto__: Object
    ▶show: Object
    ▶__proto__: Objec Object
  ▶1: Object
  ▶2: Object
  ▶3: Object
  ▶4: Object
  ▶5: Object
  ▶6: Object
  ▶7: Object
```

Creating the List HTML

We now have some data we wish to display within a list, on our page. We can create some HTML with `ng-repeat` to dynamically create the list elements based on the data within `$scope.results`. Add the following HTML code within the unordered list that has the `episode-list` class in `index.html`:

```
<li ng-repeat="tvshow in results">
    <div class="row-fluid">
        <div class="span3">
            <img src="{{tvshow.episode.images.screen}}" />
            <div class="ratings"><strong>Ratings:</strong> <span class="label"
        </div>
        <div class="span6">
            <h3>{{tvshow.show.title}}: {{tvshow.episode.title}}</h3>
            <p>{{tvshow.episode.overview}}</p>
        </div>
        <div class="span3">
            <div class="fulldate pull-right label label-info">{{tvshow.date}}</
            <ul class="show-info">

```

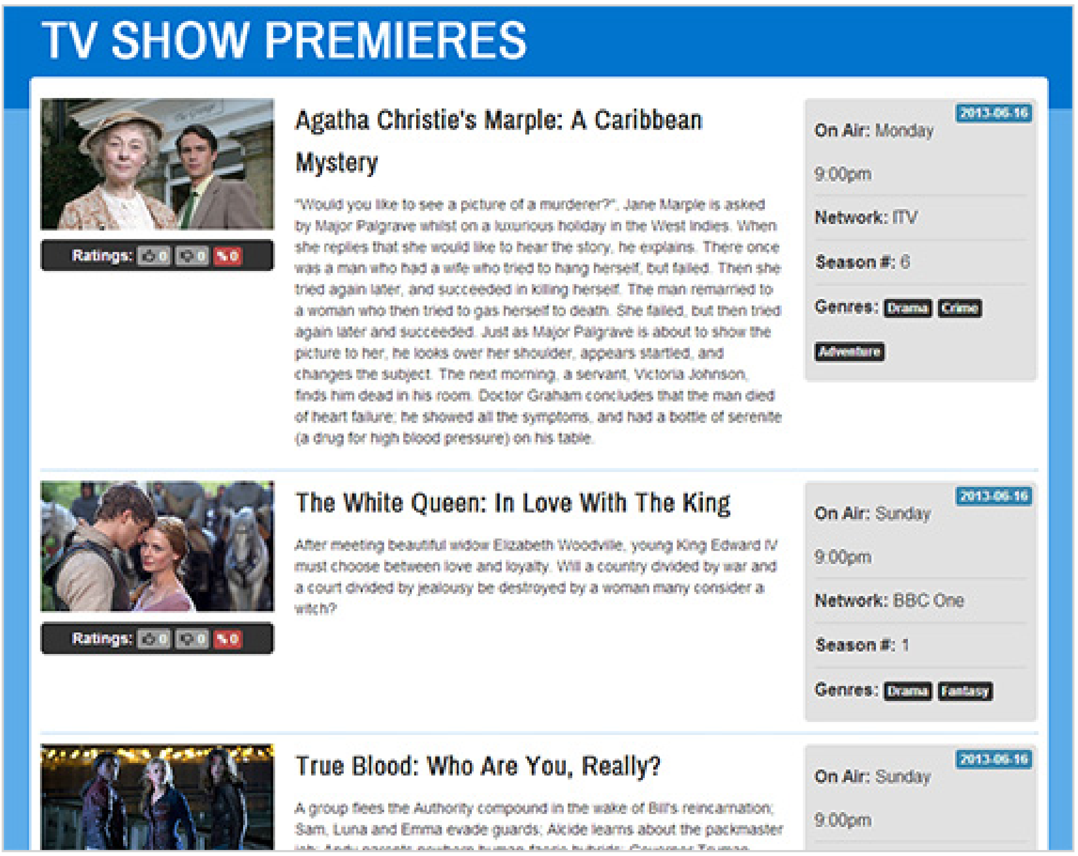
```
                <li><strong>On Air:</strong> {{tvshow.show.air_day}} {{tvshow.
                <li><strong>Network:</strong> {{tvshow.show.network}}</li>
                <li><strong>Season #:</strong> {{tvshow.episode.season}}</li>
                <li><strong>Genres:</strong> <span class="label label-inverse
            </ul>
        </div>
    </div>
</li>
```

This HTML is simply creating a single list element with `ng-repeat`. `ng-repeat="tvshow in results"` is telling angular to repeat this list element for each object within the `$scope.results` array. Remember that we do not need to include the `$scope`, as we are within an element with a specified controller (refer to the previous tutorial for more on this).

Inside the `li` element we can then reference `tvshow` as a variable which will hold all of the objects data for each of the TV shows within `$scope.results`. Below is an example of one of the objects within `$scope.results` so you can easily see how to reference each slice of data:

```
{
"show":{
    "title":"Agatha Christie's Marple",
    "year":2004,
    "url":"http://trakt.tv/show/agatha-christies-marple",
    "first_aired":1102838400,
    "country":"United Kingdom",
    "overview":"Miss Marple is an elderly spinster who lives in the vil
    "runtime":120,
    "network":"ITV",
    "air_day":"Monday",
    "air_time":"9:00pm",
    "certification":"TV-14",
    "imdb_id":"tt1734537",
    "tvdb_id":"78895",
    "tvrage_id":"2515",
    "images":{
        "poster":"http://slurm.trakt.us/images/posters/606.jpg",
        "fanart":"http://slurm.trakt.us/images/fanart/606.jpg",
        "banner":"http://slurm.trakt.us/images/banners/606.jpg"
    },
    "ratings":{
        "percentage":91,
        "votes":18,
        "loved":18,
        "hated":0
    },
    "genres":[
        "Drama",
        "Crime",
        "Adventure"
    ]
},
"episode":{
    "season":6,
    "number":1,
    "title":"A Caribbean Mystery",
    "overview":"\"Would you like to see a picture of a murderer?\", Jan
    "url":"http://trakt.tv/show/agatha-christies-marple/season/6/episod
    "first_aired":1371366000,
    "images":{
        "screen":"http://slurm.trakt.us/images/fanart/606-940.jpg"
    },
    "ratings":{
        "percentage":0,
        "votes":0,
        "loved":0,
        "hated":0
    }
},
"date":"2013-06-16"
}
```

As an example, within the `li` element, we can get the show title by referencing `tvshow.show.title` and wrapping it in double curly brackets: `{{ }}`. With this understanding, it should be easy to see what information will be displayed for each list element. Thanks to the CSS bundled with the skeleton structure, if you save these changes and open `index.html` within your browser, you should see a nicely formatted list of TV shows with the associated information and images. This is shown in the figure below:



TV SHOW PREMIERES

Agatha Christie's Marple: A Caribbean Mystery

"Would you like to see a picture of a murderer?". Jane Marple is asked by Major Palgrave whilst on a luxurious holiday in the West Indies. When she replies that she would like to hear the story, he explains. There once was a man who had a wife who tried to hang herself, but failed. Then she tried again later, and succeeded in killing herself. The man remarried to a woman who then tried to gas herself to death. She failed, but then tried again later and succeeded. Just as Major Palgrave is about to show the picture to her, he looks over her shoulder, appears startled, and changes the subject. The next morning, a servant, Victoria Johnson, finds him dead in his room. Doctor Graham concludes that the man died of heart failure; he showed all the symptoms, and had a bottle of serenite (a drug for high blood pressure) on his table.

Ratings: 0 0 % 0

2013-06-16
On Air: Monday
9:00pm
Network: ITV
Season #: 6
Genres: Drama Crime Adventure

The White Queen: In Love With The King

After meeting beautiful widow Elizabeth Woodville, young King Edward IV must choose between love and loyalty. Will a country divided by war and a court divided by jealousy be destroyed by a woman many consider a witch?

Ratings: 0 0 % 0

2013-06-16
On Air: Sunday
9:00pm
Network: BBC One
Season #: 1
Genres: Drama Fantasy

True Blood: Who Are You, Really?

A group flees the Authority compound in the wake of Bill's reincarnation; Sam, Luna and Emma evade guards; Alcide learns about the packmaster

2013-06-16
On Air: Sunday
9:00pm

Conditional Classes

You may or may not have noticed:

```
ng-class="{'label-success': tvshow.episode.ratings.percentage >= 50}"
```

...which is attached to one of the span elements, within the ratings section, in the above HTML. `ng-class` allows us to conditionally apply classes to HTML elements. This is particularly useful here, as we can then apply a different style to the percentage `span` element depending on whether the TV show rating percentage is high or not.

In the above HTML example, we want to apply the class `label-success`, which is a Twitter Bootstrap class, which will style the span to have a green background and white text. We only want to apply this class to the element if the rating percentage is greater than or equal to 50. We can do this as simply as `tvshow.episode.ratings.percentage >= 50`. Take a look down the list of formatted TV shows in your browser, if any of the percentages meet this condition, they should be displayed green.

Creating a Search Filter

We now have a list of upcoming TV show premieres, which is great, but it doesn't offer much in the way of functionality. We are now going to add a simple text search which will filter all of the objects within the results array.

Binding HTML Elements to Scope Variables

Firstly we need to declare a `$scope.filterText` variable within `mainController.js` as follows:

```
app.controller("mainController", function($scope, $http){
    $scope.apiKey = "[YOUR API KEY]";
    $scope.results = [];
    $scope.filterText = null;
    $scope.init = function() {
        //API requires a start date
        var today = new Date();
        //Create the date string and ensure leading zeros if required
        var apiDate = today.getFullYear() + ("0" + (today.getMonth() +
        $http.jsonp('http://api.trakt.tv/calendar/premieres.json/' + $s
            //As we are getting our data from an external source, we ne
            //For each day get all the episodes
            angular.forEach(data, function(value, index){
                //The API stores the full date separately from each epi
                var date = value.date;
                //For each episodes add it to the results array
                angular.forEach(value.episodes, function(tvshow, index)
                    //Create a date string from the timestamp so we can
                    tvshow.date = date; //Attach the full date to each
                    $scope.results.push(tvshow);
                });
            });
        }).error(function(error) {

        });
    };
});
```

Now we need to add a text input so that the user can actually input a search term. We then need to bind this input to the newly declared variable. Add the following HTML within the `div` which has the `search-box` class in `index.html`.

```
<label>Filter: </label>
<input type="text" ng-model="filterText"/>
```

Here we have used `ng-model` to bind this input to the `$scope.filterText` variable we declared within our scope. Now this variable will always equal what is inputted into this search input.

Enforcing Filtering On `ng-repeat` Output

Now that we have the text to filter on, we need to add the filtering functionality to `ng-repeat`. Thanks to the built-in filter functionality of AngularJS, we do not need to write any JavaScript to do this, just modify your `ng-repeat` as follows:

```
<li ng-repeat="tvshow in results | filter: filterText">
```

It's as simple as that! We are telling AngularJS - before we output the data using `ng-repeat`, we need to apply the filter based on the filterText variable. Open `index.html` in a browser and perform a search. Assuming you searched for something that exists, you should see a selection of the results.

Creating a Genre Custom Filter

So, our users can now search for whatever they are wanting to watch, which is better than just a static list of TV shows. But we can take our filter functionality a little further and create a custom filter that will allow the user to select a specific genre. Once a specific genre has been selected, the `ng-repeat` should only display TV shows with the chosen genre attached.

First of all, add the following HTML under the `filterText` input in `index.html` that we added previously.

```
<label>Genre: </label>
<select ng-model="genreFilter" ng-options="label for label in availableGenres">
    <option value="">All</option>

    </select>
```

You can see from the above HTML that we have created a select input bound to a model variable called `genreFilter`. Using `ng-options` we are able to dynamically populate this select input using an array called `availableGenres`.

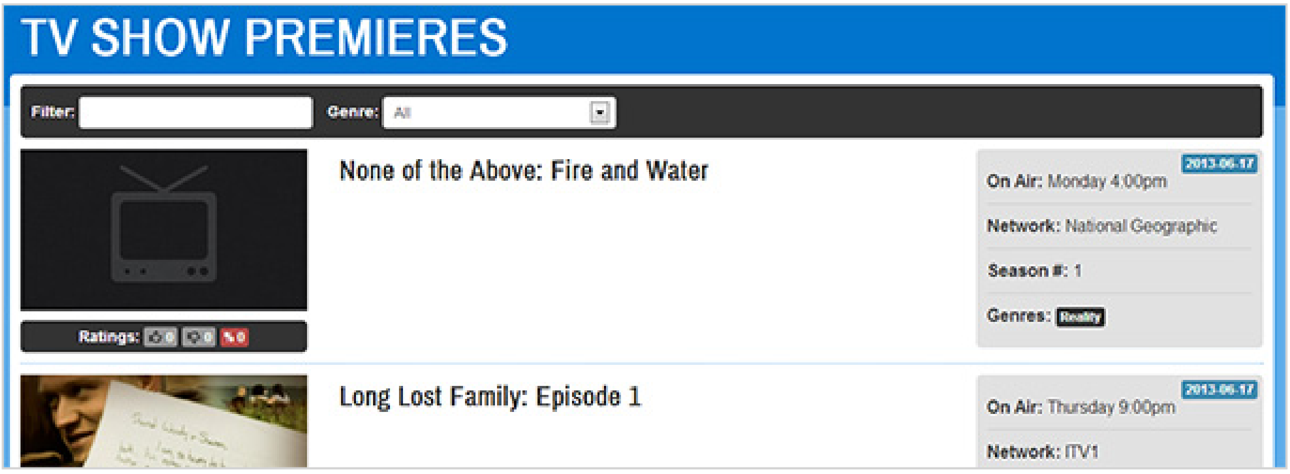
First of all, we need to declare these scope variables. Update your `mainController.js` file to be as follows:

```
app.controller("mainController", function($scope, $http){
    $scope.apiKey = "[YOUR API KEY HERE]";
    $scope.results = [];
    $scope.filterText = null;
    $scope.availableGenres = [];
    $scope.genreFilter = null;
    $scope.init = function() {
        //API requires a start date
        var today = new Date();
        //Create the date string and ensure leading zeros if required
        var apiDate = today.getFullYear() + ("0" + (today.getMonth() +
        $http.jsonp('http://api.trakt.tv/calendar/premieres.json/' + $s
            //As we are getting our data from an external source, we ne
            //For each day get all the episodes
            angular.forEach(data, function(value, index){
                //The API stores the full date separately from each epi
                var date = value.date;
                //For each episodes add it to the results array
                angular.forEach(value.episodes, function(tvshow, index)
                    //Create a date string from the timestamp so we can
                    tvshow.date = date; //Attach the full date to each
                    $scope.results.push(tvshow);
                    //Loop through each genre for this episode
                    angular.forEach(tvshow.show.genres, function(genre,
                        //Only add to the availableGenres array if it d
                        var exists = false;
                        angular.forEach($scope.availableGenres, functio
                            if (avGenre == genre) {
                                exists = true;
                            }
                        });
                        if (exists === false) {
                            $scope.availableGenres.push(genre);
                        }
                    });
                });
            });
        }).error(function(error) {

        });
    };
});
```

It is obvious that we have now declared both `genreFilter` and `availableGenres` which we saw referenced within our HTML. We have also added some JavaScript which will populate our `availableGenres` array. Within the `init()` function, while we are processing the JSON data returned from the API, we are now doing some additional processing and adding any genres that are not already within the `availableGenres` array to this array. This will then populate the select input with any available genres.

If you open `index.html` within your browser, you should see the genre select drop down populate as illustrated below:


TV SHOW PREMIERES

Filter: Genre: All

None of the Above: Fire and Water

Ratings: 0 0 0

2013-06-17
On Air: Monday 4:00pm
Network: National Geographic
Season #: 1
Genres: Reality

Long Lost Family: Episode 1

2013-06-17
On Air: Thursday 9:00pm
Network: ITV1

When the user chooses a genre, the `$scope.genreFilter` variable will be updated to equal the selected value.

Creating the Custom Filter

As we are wanting to filter on a specific part of the TV show objects, we are going to create a custom filter function and apply it alongside the AngularJS filter within the `ng-repeat`.

At the very bottom of `mainController.js`, after all of the other code, add the following JavaScript:

```
app.filter('isGenre', function() {
    return function(input, genre) {
        if (typeof genre == 'undefined' || genre == null) {
            return input;
        } else {
            var out = [];
            for (var a = 0; a < input.length; a++){
                for (var b = 0; b < input[a].show.genres.length; b++){
                    if(input[a].show.genres[b] == genre) {
                        out.push(input[a]);
                    }
                }
            }
            return out;
        }
    };
});
```

The above JavaScript declares a custom filter to our app called `isGenre`. The function within the filter takes two parameters, `input` and `genre`. `input` is provided by default (which we will see in a moment) and is all the data that the `ng-repeat` is processing. `genre` is a value we need to pass in. All this filter does, is take the specified genre and checks to see if each of the TV show objects within `input` have the specified genre attached to them. If an object has the specified genre, it adds it to the `out` array, which will then be returned to the `ng-repeat`. If this doesn't quite make sense, don't worry! It should shortly.

Applying the Custom Filter

Now that we have our customer filter available, we can add this additional filter to our ng-repeat. Modify your `ng-repeat` in `index.html` as follows:

```
<li ng-repeat="tvshow in results | filter: filterText | isGenre:genreFilter">
```

This simply chains another filter onto the `ng-repeat` output. Now the output will be processed by both filters before it is displayed on the screen. As you can see we have specified our custom filter as `isGenre:` and then we are passing the scope variable `genreFilter` as a parameter, which is how we provide our customer filter with the `genre` variable we talked about earlier. Remember that AngularJS is also providing our filter with the data that the `ng-repeat` is processing as the `input` variable.

OK, our custom genre filter is complete. Open `index.html` in a browser and test out the new functionality. With this filter in place, a user can easily filter out genres they are not interested in.

Calling Scope Functions

You may have noticed that each TV show listing also shows the genre itself. For some additional functionality, we are going to allow the user to click these genres, which will then automatically apply the genre filter for the genre they have clicked on. First of all, we need to create a scope function that the `ng-click` can call. Add the following code within the `mainController` on `mainController.js`:

```
$scope.setGenreFilter = function(genre) {
    $scope.genreFilter = genre;
}
```

In the above code, this function takes a genre value and then sets the `$scope.genreFilter` to the specified value. When this happens, the genre filter select box's value will update and the filter will be applied to the `ng-repeat` output. To trigger this function when the genre span elements are clicked, add an `ng-click` to the genre span elements within `index.html` as follows:

```
peat="genre in tvshow.show.genres" ng-click="setGenreFilter(genre)">{{genre}}</span>
```

The `ng-click` calls our previously created `setGenreFilter` function and specifies a

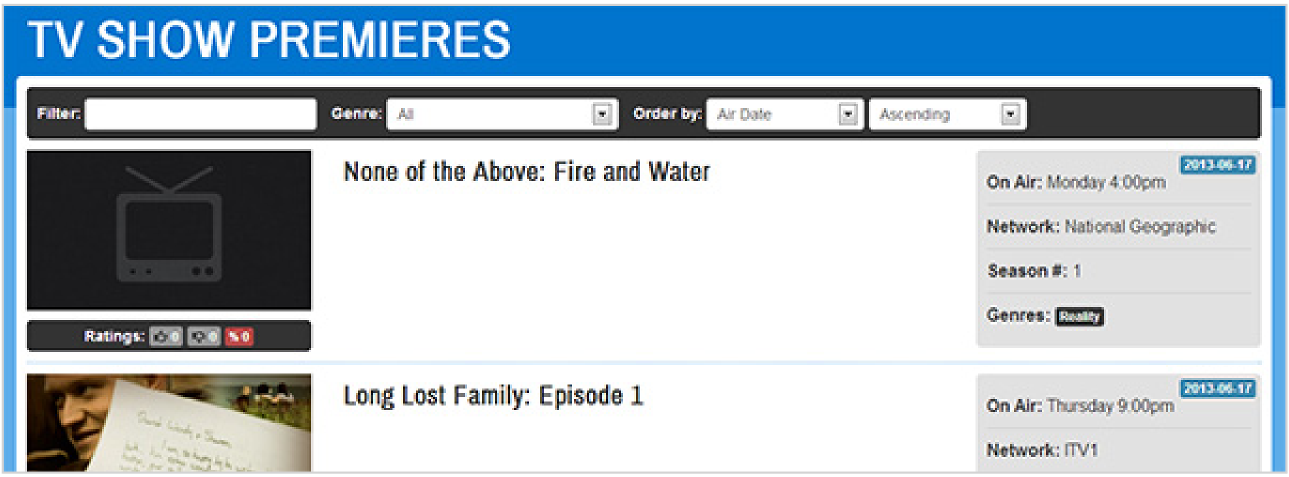
```html
<label>Order by: </label>
<select ng-model="orderField" ng-options="label for label in orderFields" class=
<select ng-model="orderReverse"class="input-medium">
    <option value="true">Descending</option>
    <option value="false">Ascending</option>
</select>
```

```
                return tvshow.episode.first_aired;
                break;
            case "Rating":
                return tvshow.episode.ratings.percentage;
                break;
        }
    };
```

We also need to declare some additional scope variables:

```
    $scope.orderFields = ["Air Date", "Rating"];
    $scope.orderDirections = ["Descending", "Ascending"];
    $scope.orderField = "Air Date"; //Default order field
    $scope.orderReverse = false;
```

If you now open `index.html` within your browser, you should see the added drop downs populated with **Air Date** already selected as the default order field. This is shown in the figure below:


TV SHOW PREMIERES

Filter: Genre: All Order by: Air Date Ascending

None of the Above: Fire and Water
2013-06-17
On Air: Monday 4:00pm
Network: National Geographic
Season #: 1
Genres: Reality
Ratings: 0 0 % 0

Long Lost Family: Episode 1
2013-06-17
On Air: Thursday 9:00pm
Network: ITV1

Finally, as we have done with our other filters, we are going to need to append this to our `ng-repeat`, update this as follows:

```
<li ng-repeat="tvshow in results | filter: filterText | isGenre:genreFilter | or
```

We are now applying an order-by-filter on our data in addition to the other filters. We are telling the order by to use our `customOrder` function and we are passing our `orderReverse` scope variable through as well. Open `index.html` in a browser and see the ordering in action.

Conclusion

AngularJS has allowed us to quickly create a detailed and functional web application with minimum effort. Utilizing AngularJS's built-in filter functions, alongside some of our own custom code, our web application allows our users to easily filter and search through the TV show premieres.

After reading this tutorial you should now be able to understand and use the following principles:

- Using `ng-repeat` to display information on screen.
- Binding to inputs, allowing users to search and filter `ng-repeat` output.
- Chaining filters on `ng-repeat` to perform multiple filtering functions.
- Custom ordering of data.
- Using events such as `ng-click` to respond to user interaction.
- Using `ng-class` to conditionally apply styling to page elements.

So in conclusion, the topics covered in this tutorial should give you a strong foundation and understanding of what you can achieve when creating rich web applications in AngularJS.

Difficulty:
Intermediate

Length:
Medium

Categories:

Web Development | JavaScript | AngularJS | Front-End | HTML | Single Page Web Apps

Translations:

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

Translate this post

Download Attachment

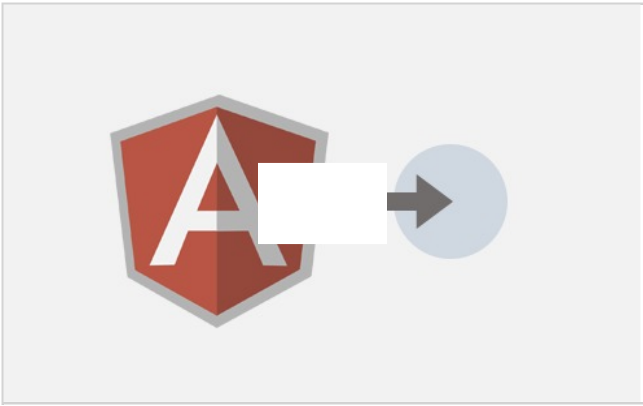
View Online Demo

About Leon Revill

I have a passion for JavaScript and am a huge fan of AngularJS and jQuery which I work with every day. I am always on the look out for new technologies to learn and when I do I aim to help the development community as much as I can by providing tutorials.

Advertisement

Suggested Tuts+ Course



Creating Angular Directives $15

Related Tutorials

Build an AngularJS App Powered by Python EVE: Part 2
Code

Build an AngularJS App From Scratch, Powered by Python EVE
Code

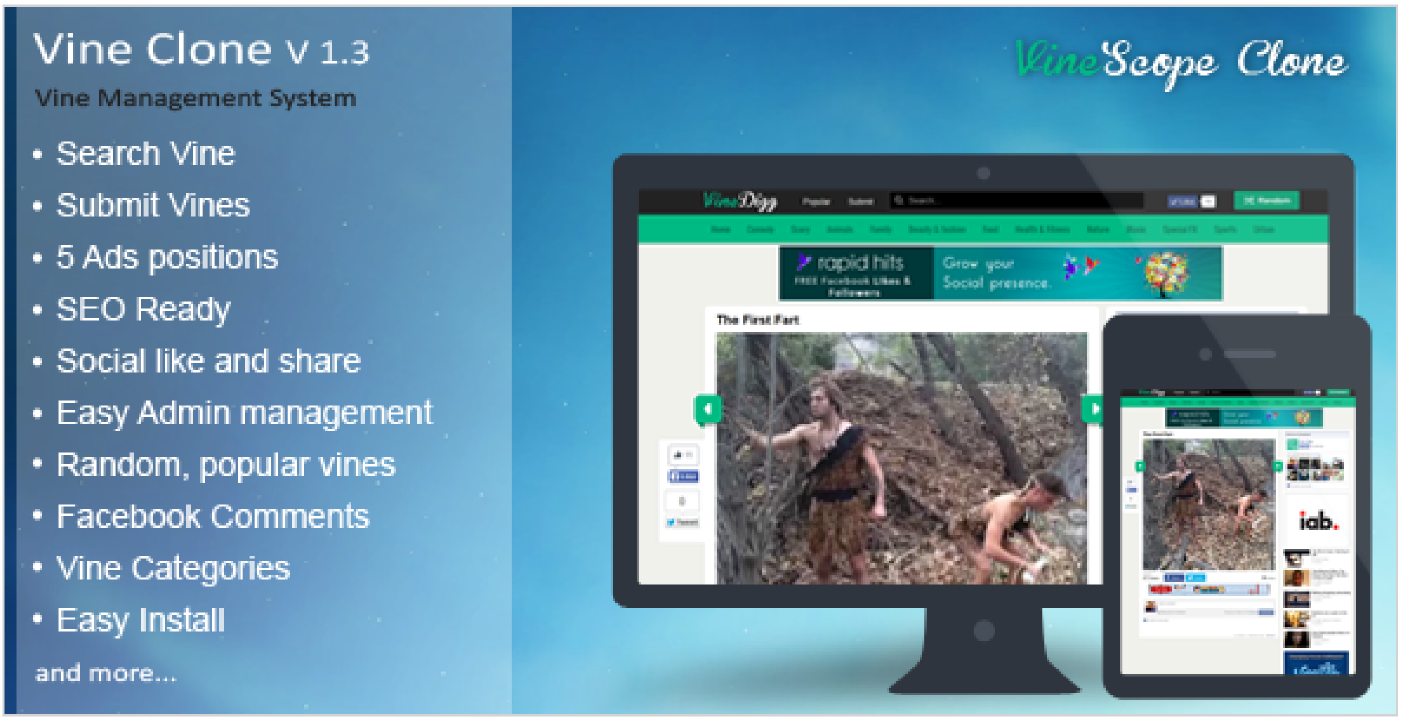
Create a Simple Shopping Cart Using AngularJS: Part 1
Code

Jobs

Senior QA Automation Engineer
at Vidstore in Denver, CO, USA

Front-End PHP Developer
at BrightLocal.com in Lewes, Lewes, East Sussex, UK

Envato Market Item



84 Comments Nettuts+ dongguangming

Recommended 10 Share Sort by Best

Join the discussion…

Adam Conrad • 2 years ago

A better AngularJS practice is instead of calling $http.jsonp to fetch your data, you should create a Service and wrap your shows in a TvShow Service object. Why did you choose this approach instead?

28 | • Reply • Share ›

revillwebdesign Adam Conrad • 2 years ago

Hi Adam. A service is of course the typical implementation as it allows you to access the same resource/http service from all controllers and keep you data in a single place. I simply chose this approach so I could focus on explaining other elements. You are right though, a service would have been better :)

Thanks for your feedback, I appreciate it.

10 | • Reply • Share ›

Victor Schelin revillwebdesign • 2 years ago

Great tutorial for someone who has not really played with AngularJS before! There are always smarter and more complicated patterns and best practises, but for these kind of tutorials, you have to narrow your education which I think you did. Would be great with a follow-up tutorial on using services though!

6 | • Reply • Share ›

axel cateland Victor Schelin • 2 years ago

Sorry but best practices must be enforced each time you can, else you will find out people following tutorial and keep producing horrendus code.
Plus i don't see any mention of testing your code in this tutorial wich saden me.

25 | • Reply • Share ›

reasonableman axel cateland • 2 years ago

Create your own tutorial then smarta$$.

73 | • Reply • Share ›

rossdavidh axel cateland • 2 years ago

The basic definition of a tutorial would be to teach one aspect, in a smaller-and-simpler-than-reality situation. Your assertion could not be more incorrect.

10 | • Reply • Share ›

asdas rossdavidh • a year ago

adasdasd

• Reply • Share ›

JJTB Somhorst axel cateland • a year ago

So you skipped the lines "now open up your browser" completely? Because, to be honost, in a little tutorial like this testing your code is overkill other then just opening the browser and see what happens..

And another note. Yes best practices should be enforced but what is the effect of enforcing best practices to someone who doesn't even know what you are talking about?

You most probably are scaring the person away!

3 • Reply • Share ›

Guest revillwebdesign • 2 years ago

Just a suggestion, instead of $http, using $resource can be a better approach as well, as it returns a resource object having action methods which provide high-level behaviours without the need to interact with the low level $http service. For more details on $resource, check http://docs.angularjs.org/api/...

• Reply • Share ›

BrianFrichette Guest • 2 years ago

No reason to use the $resource wrapper for simple GET requests. $resource makes sense for CRUD.

8 • Reply • Share ›

MPinteractiv Adam Conrad • 2 years ago

No need for another service since there is only one controller.using the $http (which is a service itself). I fail to see any reason to write services in the current tutorial ,there is only one controller using the result of the jsonp request.

4 • Reply • Share ›

Adam Conrad MPinteractiv • 2 years ago

Except you have to understand that tutorials exist to facilitate adoption into production code: rarely will you find such a small and isolated app in the wild. Inevitably people who see this tutorial will want to create something much larger (and if we were to expand on this tutorial into a multi-part series, we would quickly find this current method to be limiting). So while it works for this app, it doesn't work to expose people to building their own web app from scratch, with the idea of expanding that app into something larger.

12 • Reply • Share ›

MPinteractiv Adam Conrad • 2 years ago

In the Phone cat tutorial (the only official one) custom services are introduced only at the end of it.Is it an issue ? no.

How many custom services are created in the tutorials on AngularJS homepage ? none.

You get my point. People need to learn that complicated framework one step at the time. Like you did when you started with Angular. And that's not the last AngularJS tutorial on NetTuts anyway.

10 • Reply • Share ›

Javier Villanueva → MPinteractiv • 2 years ago

I agree, a hello world app would take hundred lines of code if we follow every single best practice for a simple tutorial. People need to understand that tutorials are introductions to a particular topic and not take them literally and simply copy/paste the code into their production apps.

5 • Reply • Share ›

codedungeon → Adam Conrad • 2 years ago

Hey Adam,

Do you have a course online where it shows best practice use of AngularJS and services. I would love to see it! I assume based on your comments that you have done lessons which show this?

2 • Reply • Share ›

Adam Conrad → codedungeon • 2 years ago

egghead.io is a great source

21 • Reply • Share ›

codedungeon → Adam Conrad • 2 years ago

I was more speaking to your comment about how his tutorial was *wrong* and that he should have done it a different way (ie your way). Instead of taking the course at face value (more content is good), bashing it as saying he did it wrong is not helpful (unless you have something you have done yourself which shows how to do it right).

9 • Reply • Share ›

faruzzy → Adam Conrad • 2 years ago

Adam could you share the service approach that you're suggesting somewhere ? I'm really interested in what you're saying! thanks

• Reply • Share ›

MPinteractiv → faruzzy • 2 years ago

Basically the idea is to declare everything that is related to the Track API inside a component called service : Then a controller would only need to reference theservice to use its attributes or methods:

(pseudo code,not tested)

```
// ShowService can be shared amongst controllers ,directives ,filters ,etc ...
app.factory("ShowService",function($http){
    return {
        availableGenres:[],
        results:[],
        url:"http://someapiurl",
        getGenres:function(){},
        getResults:function(){},
        init:function(callback){
            var that =this ;
            return $http.jsonp(this.url).success(
                function(data){
                    that.availableGenres = that.getGenres(data);
```

see more

8 ^ | ˅ • Reply • Share ›

Lasse Larsen ➦ MPinteractiv • 10 months ago

How would you mock the showservice? I'm still trying to wrap my head around that part in general and testing... Would you mind showing an example?

^ | ˅ • Reply • Share ›

Sunny ➦ MPinteractiv • a year ago

I loved your approach of adding value to a post instead of just criticizing for what somebody shared... nobody is perfect... nor every point could be made in a single post. Thanks for your value addition... I learnt something new!!

^ | ˅ • Reply • Share ›

faruzzy ➦ MPinteractiv • 2 years ago

great stuff! thanks!

^ | ˅ • Reply • Share ›

Happy • 2 years ago

Thank you for using Webstorm. I sometimes feel like I'm the only one! :(

13 ^ | ˅ • Reply • Share ›

dtouch ➦ Happy • 2 years ago

You are not alone :) Though I use phpstorm!

2 ^ | ˅ • Reply • Share ›

James ➦ Happy • a year ago

Visual Studio Express is free and alright, dunno why I'd pay about £40 for this?

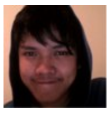
^ | ˅ • Reply • Share ›

Sowhatnow ➦ James • a year ago

Because Visual Studio only works on Windows machines?

I use WebStorm, XCode and AppCode on my Mac, Visual Studio 2013 Pro on my PC. All depends what platform I'm working on.

^ | ˅ • Reply • Share ›



David • 2 years ago

For anyone new to AngularJS try to spend your 60-ish minutes in this introductory video SO YOU WON'T GET LOST IN THE JUNGLE.

see more

12 ^ | ˅ • Reply • Share ›

MPinteractiv • 2 years ago

Great Tut as usual !

Another strategy that allows one not having to write a custom filter :

```
data-ng-repeat="tvshow in results|filter:filterText|filter:{show.genres:genreFilter}"
```

Another optimisation that makes $scope.customOrder method unnecessary:

```
// in the MainController controller function
$scope.orderFields:[{label:'Air Date',value:"episode.first_aired"},{label:'Rating',value:'episod
// in the select dealing with order by ratings and Air date
 data-ng-options="f.value as f.label for f in orderFields"
// in the html, with the data-ng-repeat dealing with the show list :
data-ng-repeat="tvshow in results|filter:filterText|filter:{show.genres:genreFilter}|orderBy:ord
```

8 ^ | ˅ • Reply • Share ›

8 • Reply • Share ›

Ville • 2 years ago

A small correction to tutorial: When using angular generated url:s in src-attribute, you should use ng-src instead. It loads the content afterwards, so it does not use something like "{{object.imageURL}}" as a url of image.

7 • Reply • Share ›

Jared Pavan • a year ago

I think the mainController.js code is being malformed by tutsplus. The longest line reads:

$http.jsonp('http://api.trakt.tv/calendar/p... $scope.apiKey + '/' + apiDate + '/' + 30 + '/?callback=JSON_CALLBACK').success(function(data) {

When it should read:

$http.jsonp('http://api.trakt.tv/calendar/p... + $scope.apiKey + '/' + apiDate + '/' + 30 + '/?callback=JSON_CALLBACK').success(function (data){

2 • Reply • Share ›

jamie piazza Jared Pavan • a year ago

you're right! thanks for pointing that out, jared! (i could tell the code highlighting looked wrong in my editor but was struggling to figure out where it was breaking - although now it seems so obvious!) you're comment is actually still breaking/truncating the beginning of that string concatenation because tuts/disqus is seeing it as a link - so i'll just post an image of what trakt expects and hopefully anyone else confused by this can figure out the missing ".../" + fix too!

• Reply • Share ›

David C. Grimm jamie piazza • 7 months ago

This seems like it could be an awesome tutorial, but after four hours, I simply cannot get past this first step of getting the info. from the trakt api. I have tried these two solutions, among many, many others, and continually get this error (or something similar when I
enter /date/days) in the console:

GET http://api.trakt.tv/calendar/p... api="" key="">/20141223/30?callback=angular.callbacks._0

with a list underneath of (anonymous function)
errors in red text with a corresponding line in the angularjs code.

Here is my code:

app.controller("mainController", function($scope, $http){

$scope.apiKey = "<my api="" key="">";
$scope.init = function() {

see more

• Reply • Share ›

David C. Grimm David C. Grimm • 7 months ago

As Jamie mentioned, Disqus treats everything that starts with http as a link and truncates it so I'll try to show what the two links above are here:

The error message:

GET http : // api . trakt . tv / calendar / premieres . json / apikey / date / days / <my api="" key=""> / 20141223 / 30?callback=angular.callbacks._0

My code:
http : // api . trakt . tv / calendar / premieres . json / apikey / date / days

Hope that's clear.

Thanks.

• Reply • Share ›

Ankush Chadda • 2 years ago

Wont putting API key in js , reveal the key, out in public. What could we do to make the app more secure?

2 • Reply • Share ›

Arif Majid Ankush Chadda • 2 years ago

Use a server side service to fetch ur data using CURL(PHP), http/https module for nodejs. This was ur apikey and api secret is hidden from external user...

Hope that helps.

2 • Reply • Share ›

OluwaseunLadeinde • 2 years ago

I am new to angularjs but haven read some of the comments, I decided to challenge myself and revamp the code. Pls note that it might be a little cranky and may be done better but this is what I came up with

maincontroller.js

app.factory("ShowService",function($http){

```
return {

availableGenres:[],
filterText:null,
availableGenres:[],
results:[],
url:"",
getURL:function(){
var that = this ;
//API Key
var apiKey = "[YOUR API KEY]";
//API requires a start date
```

see more

2 • Reply • Share ›

OluwaseunLadeinde OluwaseunLadeinde • 2 years ago

The code is not formatted or properly indented. But if you throw this on any javascript editor, it should work out of the box.

Reply • Share ›

roy OluwaseunLadeinde • 2 years ago

You might get people to try it out if you created a jsfiddle instead.

2 • Reply • Share ›

Kenya Sullivan roy • a year ago

Or codepen.io

Reply • Share ›

Kasper Thorø Plougmann • 6 months ago

This tut should either be marked as broken, or be updated.
The links to Trakt doesn't work, and Trakt isn't using API keys anymore.

Please, do update the tut to reflect the changes from Trakt

1 • Reply • Share ›

Javier Caballero Kasper Thorø Plougmann • a month ago

agree, i hope it will be updated

Reply • Share ›

miller3818 • a year ago

This is by far the best AngularJS tutorial I've come across. I love the data!

1 • Reply • Share ›

Dmitri • a year ago

Unfortunately the code is very hard to read due to very long lines that do not break to fit the windows. Would really help to have lines shorter or use a style that breaks lines to make them readable without scrolling back and forth?

1 • Reply • Share ›

Tony • 2 years ago

Leon- excellent tutorial, as usual (i really enjoyed the 30 min one). Thx!

Except that I got tripped on step 1: after I put my apiKey (in the exact code above), I never got the json back (error 400). If I just put my key with the url i get json back. So the issue must be in the code (i humbly presume). Would it be possible to include a finished app source-code as well (where one just needs to put his apiKey) so that I can compare and troubleshoot. With my modest understanding, I do not know what else I should do to get it to work??

Translate for Tuts+
Meetups

Email Newsletters

Get Tuts+ updates, news, surveys & offers.

Email Address

Subscribe

Privacy Policy