

# **Unit 7**

## **Client-Side Development in ASP.NET Core**

# COMMON CLIENT-SIDE WEB TECHNOLOGIES

- ASP.NET Core applications are web applications and they typically rely on client-side web technologies like HTML, CSS, and JavaScript.
- By separating the content of the page (the HTML) from its layout and styling (the CSS), and its behavior (via JavaScript), complex web apps can leverage the Separation of Concerns principle.
- While HTML and CSS are relatively stable, JavaScript, by means of the application frameworks and utilities developers work with to build web-based applications.
- We will discuss on JavaScript, jQuery, Angular SPA, React, Vue.

# Javascript

- JavaScript is a dynamic, interpreted programming language of the web.
- Just like CSS, it's recommended to organize JavaScript into separate files, keeping it separated as much as possible from the HTML found on individual web pages or application views.
- With Javascript, we can perform following:
  - Selecting an HTML element and retrieving and/or updating its value.
  - Decision Making, complex calculations, Validate Data, Animate and Add Effects
  - Interaction with properties of page object
  - React to events
  - Querying a Web API for data.
  - Sending a command to a Web API (and responding to a callback with its result).
  - Performing validation.

# Quick Example Review on Javascript

Example

```
<HTML>  
  <TITLE> Displaying Text </TITLE>  
  <BODY>  
    <script>  
      document.write("<h1> Hello Good Day </H1>");  
      document.write("<H3> Best of Luck. </H3>");  
      alert("Hello");  
    </script>  
  </BODY>  
</HTML>
```

## Example2

```
<script type="text/javascript">
  s1=12;
  s2=28;
  sum=s1+s2;
  diff=s1-s2;
  mult=s1*s2;
  div=s1/s2;
  document.write("<br>Sum: "+sum);
  document.write("<br>Difference: "+diff);
  document.write("<br>Multiply: "+mult);
  document.write("<br>Division: "+div);
</script >
```

## Example3 – JavaScript Array

```
<script>
var sports = new Array( "Football", "Tennis", "Cycling");
document.write(sports[0]);
document.write(sports[1]);
document.write(sports[2]);
var count = sports.length;
// loop through array elements
for(i=0; i< count; i++)
{
    document.write("<br>Index " + i + " is " + sports[i]);
}
</script>
```

## Example 4 – JavaScript String

- Used for storing and manipulating text
- Zero or more characters within quotes.

```
/* String : J a v a s c r i p t
Index   : 0 1 2 3 4 5 6 7 8 9 */
var myText = "Javascript";
document.write("<br>" + myText.length);
document.write("<br>" + myText.charAt(4));
document.write("<br>" + myText.indexOf("va"));
document.write("<br>" + myText.substr(0,4));
document.write("<br>" + myText.toUpperCase());
document.write("<br>" + myText.toLowerCase());
```

## Example 5 – JavaScript Function

```
<script>
    // function defination
    function callme() {
        alert("Hello there");
    }
    function f3(n1, n2) {
        var sum = n1 + n2;
        return sum;
    }
    callme();    // calling a function
    callme();
    var returned_sum = f3(10, 20);
    document.write(returned_sum);
    document.write(f3(20, 30));
</script>
```



## Example 5 – JavaScript Date

```
<html>
<body>
  <h1>Demo: Current Date</h1>
  <p id="p1"></p>
  <p id="p2"></p>
  <script>
    document.getElementById("p1").innerHTML = Date();
    var currentDate = new Date();
    document.getElementById("p2").innerHTML = currentDate;
  </script>
</body>
</html>
```

# Javascript Events

- Interaction with HTML page and HTML elements is handled through events. Events can be page loads, button click, pressing a key, select data in form controls, focus on control, mouse over and mouse out on any element, etc.
- Events are a part of the Document Object Model (DOM) and every HTML element contains a set of events which can trigger JavaScript Code.
- We can categories Javascript events on:
  - Document Level Events - onload, onunload
  - Form Level Events - Onsubmit, Onreset, Onchange, onselect, onblur, onfocus
  - Keyboard Events - Onkeydown, onkeypress, onkeyup
  - Mouse Events - Onclick, ondblclick, onmouseover, onmouseout

## Events –Example

```
<html>
<head>
<script>
    function callme() {
        alert("Hello");
        document.write("Hello");
    }
</script>
</head>
<body>
    <form>
        <input type = "button" onclick = "callme()" value = "Click Me">
    </form>
</body>
</html>
```

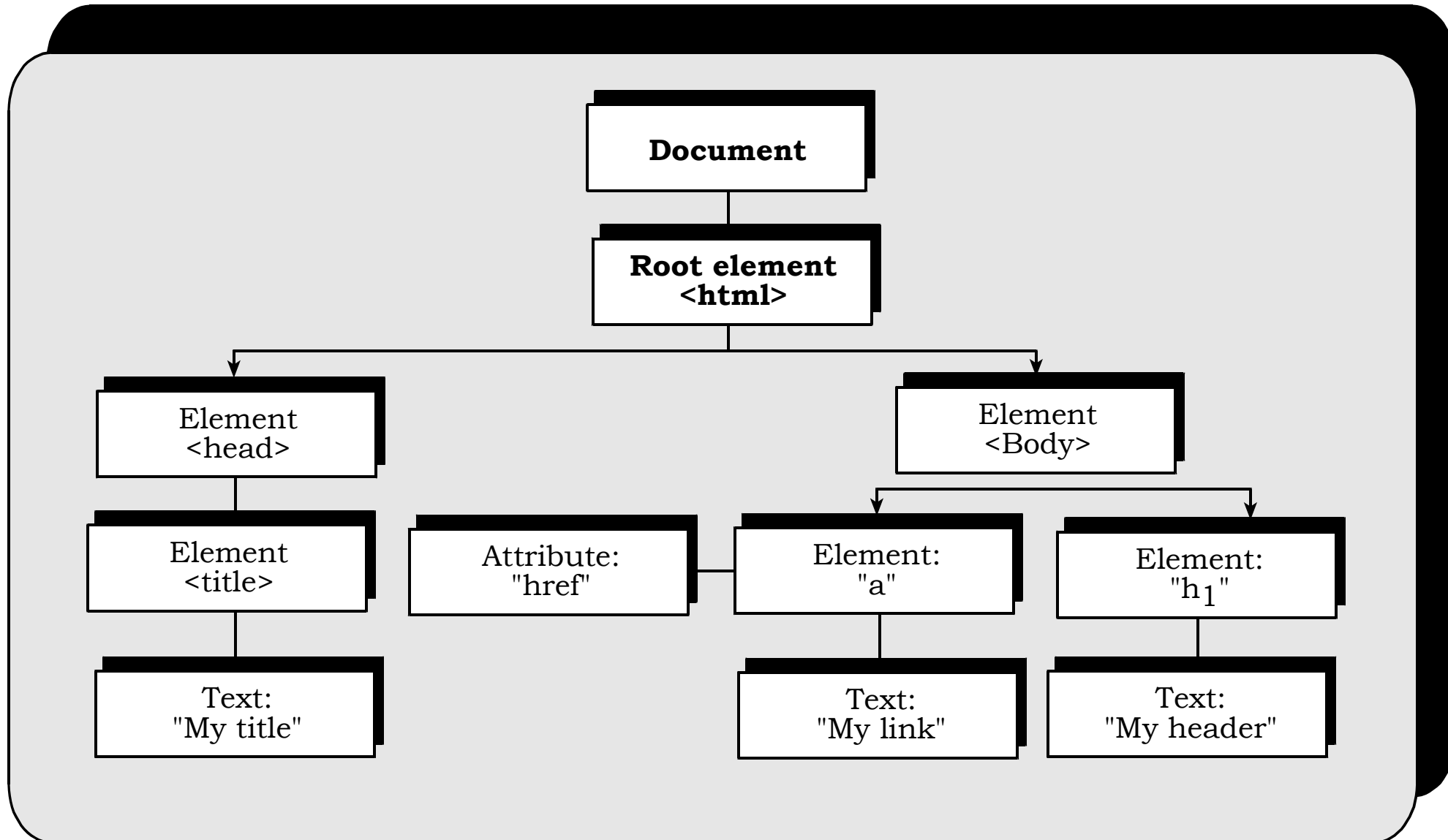
## Events –Example

```
<html>
<head>
  <script type="text/javascript">
    function over() {
      alert("Mouse Over");
    }
    function out() {
      alert("Mouse Out");
    }
  </script>
</head>
<body>
  <div onmouseover="over()" onmouseout="out()">
    <h2> This is inside the division </h2>
  </div>
</body>
</html>
```

# HTML DOM

- When a web page is loaded, browser creates a Document Object Model of the page. With the object model, JavaScript can do following:
  - modify all the HTML elements and attributes in the
  - change all the CSS styles in the page
  - Add remove existing HTML elements and attributes
  - add new HTML elements and attributes
  - react to all existing HTML events in the page
  - create new HTML events in the page

The HTML DOM model is constructed as a tree of Objects:



- In the DOM, all HTML elements are defined as objects. Below example changes the content (the innerHTML) of the <p> element with id="demo" and getElementById is a method and innerHTML is a property

```
<body>
```

```
  <p id="demo"></p>
```

```
  <script>
```

```
    document.getElementById("demo").innerHTML = "Hello World!";
```

```
  </script>
```

```
</body>
```

## Changing HTML Elements

Property	Description
<code>element.innerHTML = new htm content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style an HTML element
Method	Description
<code>element.setAttribute(attribute, value)</code>	Change the attribute of an HTML element

## Finding HTML Elements

Method	Description
<code>document.getElementById(id)</code>	Find an element by element id
<code>document.getElementsByTagName(name)</code>	Find elements by tag name
<code>document.getElementsByClassName(name)</code>	Find elements by class name



# Adding and Deleting Elements

Method	Description
<code>document.createElement(element)</code>	Create an HTML element
<code>document.removeChild(element)</code>	Remove an HTML element
<code>document.appendChild(element)</code>	Add an HTML element
<code>document.replaceChild(new, old)</code>	Replace an HTML element
<code>document.write(text)</code>	Write into the HTML output stream

## EXAMPLE

```
<html>
<head>
<script>
    var btn = document.querySelector('button');
    function random(number) {
        return Math.floor(Math.random() * (number+1));
    }
    function changeBgColor() {
        var rCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ') ';
        document.body.style.backgroundColor = rCol;
    }
</script>
</head>
<body>
    <button onclick= "changeBgColor()">Change color</button>
</body>
</html>
```

# Form Validation

- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.
- Form validation generally performs two functions.
  - Basic Validation – check all the mandatory fields are filled in.
  - Data Format Validation – data entered checked for correct form and value with appropriate logic to test correctness of data.

```

<html> <head> <title>Form Validation</title>
  <script type = "text/javascript">
    <!--      // Form validation code will come here.      //-->
  </script></head>
<body>
  <form action = "next_page" name = "myForm" onsubmit = "return(validate());">
    <table cellspacing = "2" cellpadding = "2" border = "1">
      <tr> <td align = "right">Name</td> <td><input type = "text" name = "Name" /></td> </tr>
      <tr> <td align = "right">EMail</td> <td><input type = "text" name = "EMail" /></td> </tr>
      <tr> <td align = "right">Zip Code</td> <td><input type = "text" name = "Zip" /></td> </tr>
      <tr> <td align = "right">Country</td> <td>
        <select name = "Country">
          <option value = "1">USA</option>
          <option value = "2">UK</option>
          <option value = "3">Nepal</option>
        </select>
      </td> </tr>
      <tr> <td align = "right"></td> <td><input type = "submit" value = "Submit" /></td> </tr>
    </table>
  </form> </body> </html>

```

Name	<input type="text"/>
EMail	<input type="text"/>
Zip Code	<input type="text"/>
Country	USA ▼
	<input type="submit" value="Submit"/>

```
<script type = "text/javascript">
```

```
function validate() {
```

```
    if( document.myForm.Name.value == "" ) {
```

```
        alert( "Please provide your name!" ); document.myForm.Name.focus() ; return false;
```

```
    }
```

```
    if( document.myForm.EMail.value == "" ) {
```

```
        alert( "Please provide your Email!" ); document.myForm.EMail.focus() ; return false;
```

```
    }
```

```
    if( document.myForm.Zip.value == "" || isNaN( document.myForm.Zip.value ) ||
```

```
        document.myForm.Zip.value.length != 5 ) {
```

```
        alert( "Please provide a zip in the format #####." ); document.myForm.Zip.focus() ; return false;
```

```
    }
```

```
    if( document.myForm.Country.value == "-1" ) {
```

```
        alert( "Please provide your country!" ); return false;
```

```
    }
```

```
    return( true );
```

```
}
```

```
</script>
```

# jQuery

- jQuery is a fast, small and feature-rich JavaScript library included in a single .js file.
- It provides many built-in functions using which developers can accomplish various tasks easily and quickly.
- Some of the jQuery important features are:
  - DOM Selection: jQuery provides Selectors to retrieve DOM element based on different criteria like tag name, id, css class name, attribute name, value, nth child in hierarchy etc.
  - DOM Manipulation: You can manipulate DOM elements using various built-in jQuery functions. For example, adding or removing elements, modifying html content, css class etc.

# jQuery

- Some of the jQuery important features are:
  - Special Effects: You can apply special effects to DOM elements like show or hide elements, fade-in or fade-out of visibility, sliding effect, animation etc.
  - Events: jQuery library includes functions which are equivalent to DOM events like click, dblclick, mouseenter, mouseleave, blur, keyup, keydown etc. These functions automatically handle cross-browser issues.
  - Ajax: jQuery also includes easy to use AJAX functions to load data from servers without reloading whole page.
  - Cross-browser support: jQuery library automatically handles cross-browser issues, so the user does not have to worry about it.

# Advantages of jQuery

- Easy to learn: jQuery is easy to learn because it supports same JavaScript style coding.
- Write less do more: jQuery provides a rich set of features that increase developers' productivity by writing less and readable code.
- Excellent API Documentation: jQuery provides excellent online API documentation.
- Cross-browser support: jQuery provides excellent cross-browser support without writing extra code.
- Unobtrusive: jQuery is unobtrusive which allows separation of concerns by separating html and jQuery code.



# Getting Started with jQuery

- You can start writing jQuery code on any of the editor like notepad, SublimeText, Visual Studio. it's time to use jQuery.
- There are several ways to start using jQuery on your web site. You can:
  - Download the jQuery library from [jQuery.com](http://jQuery.com)
  - Include jQuery from a CDN, like Google
- There are two versions of jQuery available for downloading:
  - Production version - this is for your live website because it has been minified and compressed
  - Development version - this is for testing and development (uncompressed and readable code)
- The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag (notice that the `<script>` tag should be inside the `<head>` section)

# Getting Started with jQuery

```
<head>
```

```
<script src="jquery-3.5.1.min.js"></script>
```

```
</head>
```

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
```

## jQuery Syntax

- The jQuery syntax is used to select HTML elements and perform some **action** on those element(s). Basic syntax is: `$(selector).action()`
  - A \$ sign to define/access jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)

## Examples:

`$(this).hide()` - hides the current element.

`$("p").hide()` - hides all `<p>` elements.

`$(".test").hide()` - hides all elements with `class="test"`.

`$("#test").hide()` - hides the element with `id="test"`.

## The Document Ready Event

All jQuery methods in are inside a document ready event.

```
$(document).ready(function(){  
    // jQuery methods go here...  
});
```

## Example: jQuery Element Selector to hide all paragraphs

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("p").hide();
    });
  });
</script>
</head>
<body>
  <h2>This is a heading</h2>
  <p>This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <button>Click me to hide paragraphs</button>
</body>
</html>
```

**Example: Using id Selector - with id=test will be hidden on button click**

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#test").hide();  
    });  
});
```

**Example: Using .class Selector with class=test will be hidden on button click**

```
$(document).ready(function(){  
    $("button").click(function(){  
        $(".test").hide();  
    });  
});
```

# More Examples of jQuery Selectors

Syntax	Description
<code>\$(this)</code>	Current HTML element
<code>\$("p")</code>	All <code>&lt;p&gt;</code> elements
<code>\$("p.intro")</code>	All <code>&lt;p&gt;</code> elements with <code>class="intro"</code>
<code>\$(".intro")</code>	All elements with <code>class="intro"</code>
<code>\$("#intro")</code>	The first element with <code>id="intro"</code>
<code>\$("ul li:first")</code>	The first <code>&lt;li&gt;</code> element of each <code>&lt;ul&gt;</code>
<code>\$("[href\$='.jpg']")</code>	All elements with an <code>href</code> attribute that ends with <code>".jpg"</code>
<code>\$("div#intro .head")</code>	All elements with <code>class="head"</code> inside a <code>&lt;div&gt;</code> element with <code>id="intro"</code>

# jQuery Events

- An event represents the precise moment when something happens.
- An event can be moving a mouse over an element, selecting a radio button, clicking on an element, etc
- The term "fires/fired" is often used with events. Example: "The keypress event is fired, the moment you press a key".
- Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
<b>click</b>	keypress	submit	load
<b>dblclick</b>	keydown	change	resize
<b>mousedown</b>	keyup	focus	scroll
<b>mouseleave</b>		blur	unload

# jQuery Syntax for Events

Event Method	Description
<code>\$(selector).click(function)</code>	Invokes a function when the selected elements are clicked
<code>\$(selector).dblclick(function)</code>	Invokes a function when the selected elements are double-clicked
<code>\$(selector).focus(function)</code>	Invokes a function when the selected elements receive the focus
<code>\$(selector).mouseover(function)</code> <code>)</code>	Invokes a function when the mouse is over the selected elements
<code>\$(selector).keypress(function)</code>	Invokes a function when a key is pressed inside the selected elements



# jQuery Event Methods

**Example:** jQuery Event Methods

```
$(document).ready(function(){
```

```
    $("p").click(function(){  
        $(this).hide();  
    });
```

```
    $("p").dblclick(function(){  
        $(this).hide();  
    });
```

```
    $("#id1").hover(function(){  
        alert("You hover on id1").  
    });
```

```
});
```

# jQuery Effects

Event Method	Description
<code>\$(selector).hide()</code>	Hide selected elements
<code>\$(selector).show()</code>	Show selected elements
<code>\$(selector).toggle()</code>	Toggle (between hide and show) selected elements
<code>\$(selector).slideDown()</code>	Slide-down (show) selected elements
<code>\$(selector).slideUp()</code>	Slide-up (hide) selected elements
<code>\$(selector).slideToggle()</code>	Toggle slide-up and slide-down of selected elements
<code>\$(selector).fadeIn()</code>	Fade in selected elements
<code>\$(selector).fadeOut()</code>	Fade out selected elements
<code>\$(selector).fadeTo()</code>	Fade out selected elements to a given opacity
<code>\$(selector).fadeToggle()</code>	Toggle between fade in and fade out

# jQuery Effects

**EX - Fade Paragraph id1 with 50% opacity when btnFade is clicked**

```
<script>
    $("#btnFade").click(function(){
        $("#id1").fadeTo("slow", 0.5);
    });
</script>
```

# LEGACY WEB APPS WITH JQUERY

- Although ancient by JavaScript framework standards, jQuery continues to be a commonly used library for working with HTML/CSS and building applications that make AJAX calls to web APIs.
- However, jQuery operates at the level of the browser document object model (DOM), and by default offers only an imperative, rather than declarative, model.
- For example, imagine that if a textbox's value exceeds 10, an element on the page should be made visible. In jQuery, this would typically be implemented by writing an event handler with code that would inspect the textbox's value and set the visibility of the target element.
- This is an imperative, code-based approach. Another framework might instead use databinding to bind the visibility of the element to the value of the textbox declaratively.
- As client-side behaviors grow more complex, data binding approaches frequently result in simpler solutions with less code and conditional complexity

# jQuery vs a SPA Framework

Factor	jQuery	Angular
Abstracts the DOM	Yes	Yes
AJAX Support	Yes	Yes
Declarative Data Binding	No	Yes
MVC-style Routing	No	Yes
Templating	No	Yes
Deep-Link Routing	No	Yes

**jQuery Vs a SPA Framework**

- Most of the features jQuery lacks intrinsically can be added with the addition of other libraries. SPA framework like Angular provides these features in a more integrated fashion, since it's been designed with all of them in mind from the start.
- Also, jQuery is an imperative library, meaning that you need to call jQuery functions in order to do anything with jQuery. Much of the work and functionality that SPA frameworks provide can be done declaratively, requiring no actual code to be written.
- Data binding is a great example of this. In jQuery, it usually only takes one line of code to get the value of a DOM element or to set an element's value. However, you have to write this code anytime you need to change the value of the element, and sometimes this will occur in multiple functions on a page.
- Another common example is element visibility. In jQuery, there might be many different places where you'd write code to control whether certain elements were visible. In each of these cases, when using data binding, no code would need to be written. You'd simply bind the value or visibility of the elements in question to a viewmodel on the page, and changes to that viewmodel would automatically be reflected in the bound elements.

# Angular SPAs

- Angular remains one of the world's most popular JavaScript frameworks. The redesigned Angular continues to be a robust framework for building Single Page Applications.
- Angular applications are built from components. Components combine HTML templates with special objects and control a portion of the page. A simple component from Angular's docs is shown here:

```
import{ Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: '<h1>Hello {{name}}</h1>'
})

export class AppComponent{ name = 'Angular'; }
```

# Angular SPAs

- Components are defined using the `@Component` decorator function, which takes in metadata about the component. The `selector` property identifies the ID of the element on the page where this component will be displayed.
- The `template` property is a simple HTML template that includes a placeholder that corresponds to the component's `name` property, defined on the last line.
  - `Import { Component } from '@angular/core';`
- By working with components and templates, instead of DOM elements, Angular apps can operate at a higher level of abstraction and with less overall code than apps written using just JavaScript (also called "vanilla JS") or with jQuery.
- Angular also imposes some order on how you organize your client-side script files.



# Getting Started with Angular

- AngularJS is a client side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework. AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.
- AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create custom attributes using simple JavaScript.
- We need the following tools to setup a development environment for AngularJS:
  1. AngularJS Library – download from [angularjs.org](http://angularjs.org)
  2. Editor/IDE – notepad++, SublimeText, Visual Studio & others
  3. Web server – IIS, Apache, etc
  4. Browser

# Advantages of AngularJS

- Open source JavaScript MVC framework.
- Supported by Google
- No need to learn another scripting language. It's just pure JavaScript and HTML.
- Supports separation of concerns by using MVC design pattern.
- Built-in attributes (directives) makes HTML dynamic.
- Easy to extend and customize.
- Supports Single Page Application.
- Uses Dependency Injection.
- Easy to Unit test.
- REST friendly.

## See this example with jQuery

```
<!DOCTYPE html>
<html>
<head>
  <script src="~/Scripts/jquery-1.10.2.min.js"></script>
</head>
<body>
  Enter Your Name: <input type="text" id="txtName" /> <br />
  Hello <label id="lblName"></label>

  <script>
    $(document).ready( function () {
      $('#txtName').keyup(function () {
        $('#lblName').text($('#txtName').val());
      });
    });
  </script>
</body>
</html>
```

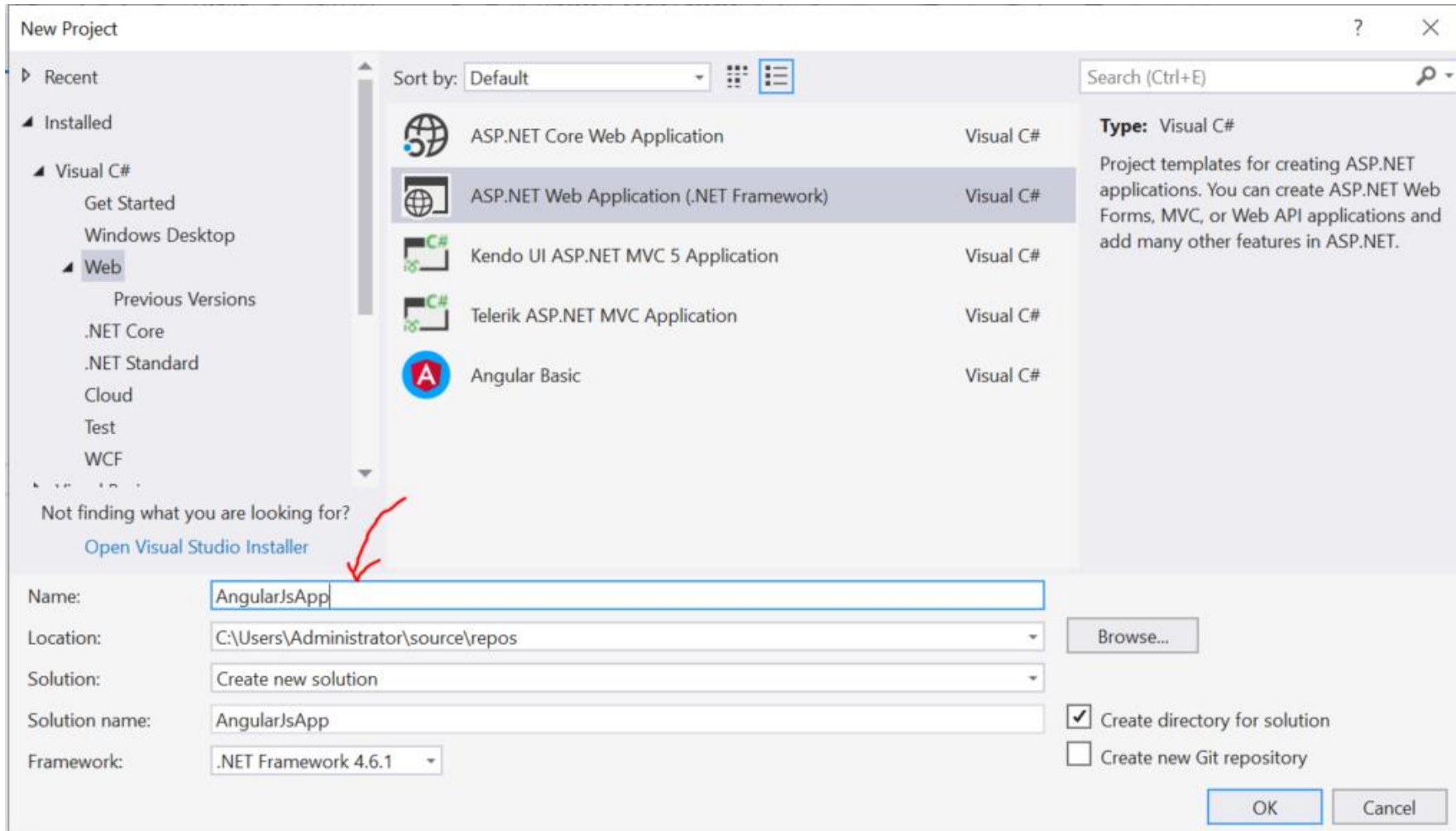
## Example

- Conversion of above jQuery program to Angular Code to shows plain HTML code with couple of AngularJS directives (attributes) such as ng-app, ng-model, and ng-bind.

```
<!DOCTYPE html>
<html>
<head>
  <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
  Enter Your Name: <input type="text" ng-model="name" /> <br />
  Hello <label ng-bind="name"></label>
</body>
</html>
```

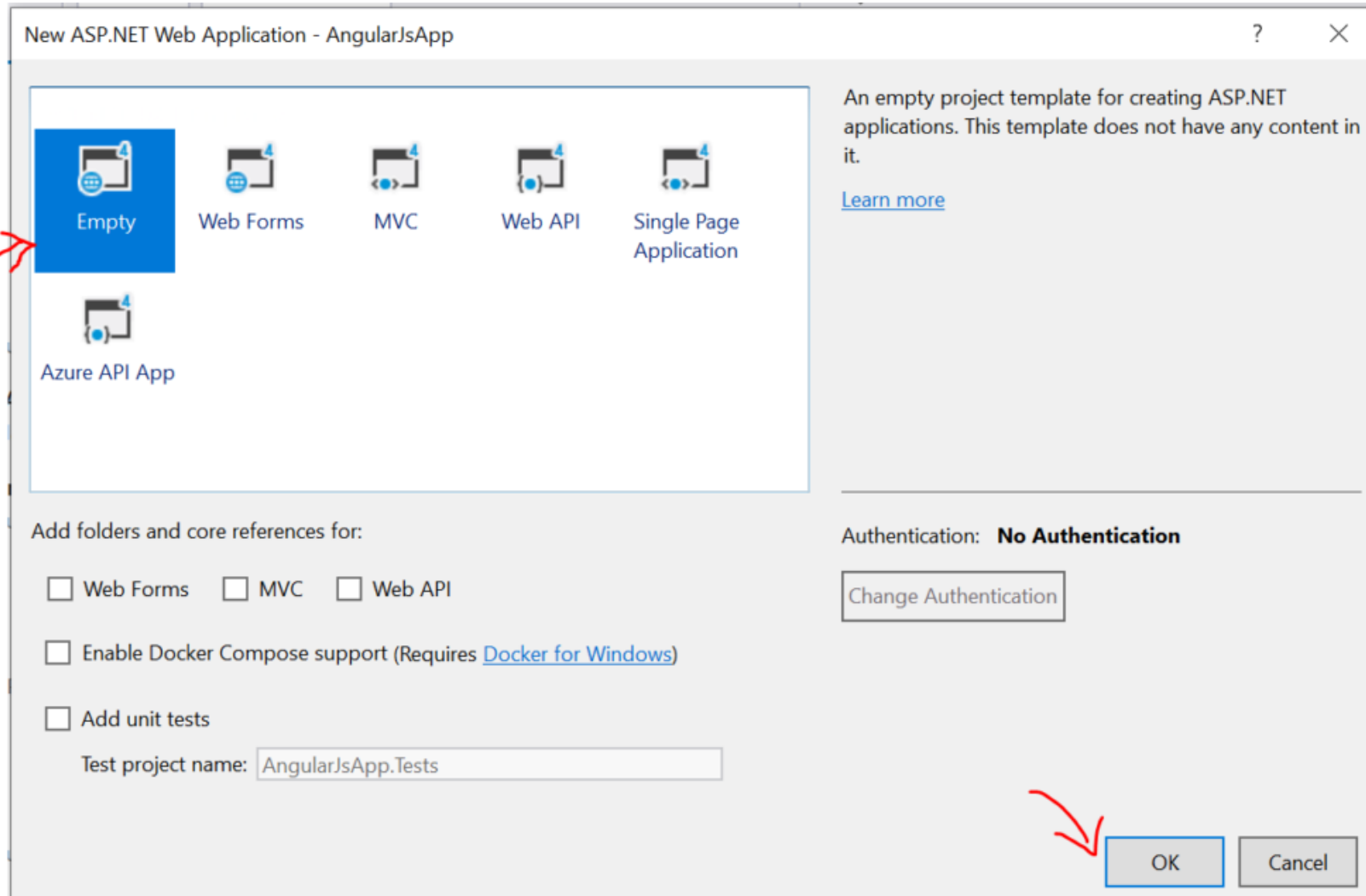
# Setup angularjs application in Visual Studio 2019

- Open visual studio create angularjs project name like “angularJsApp”



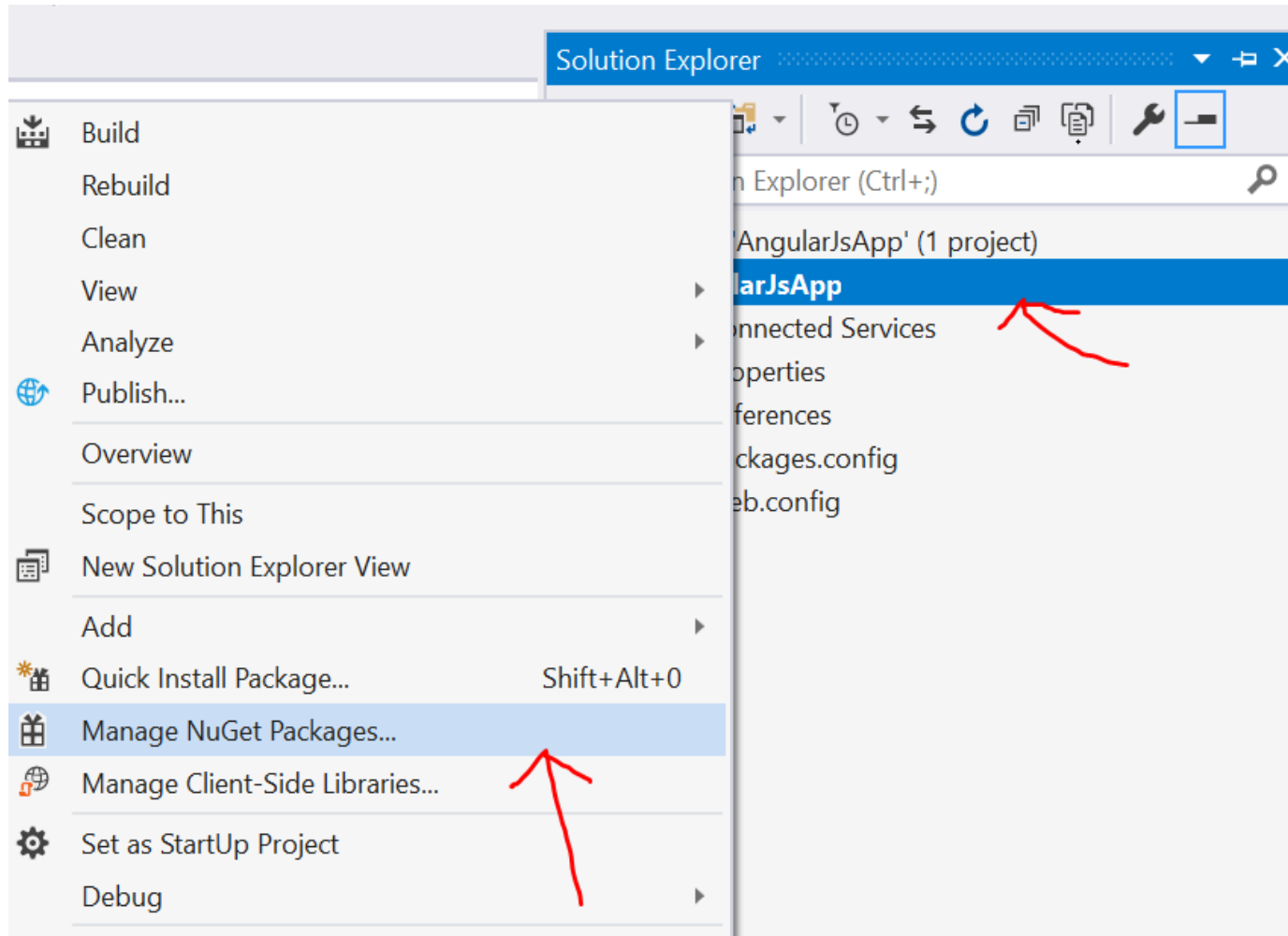
# Setup angularjs application in Visual Studio 2019

- Select an empty project and then click on ok button



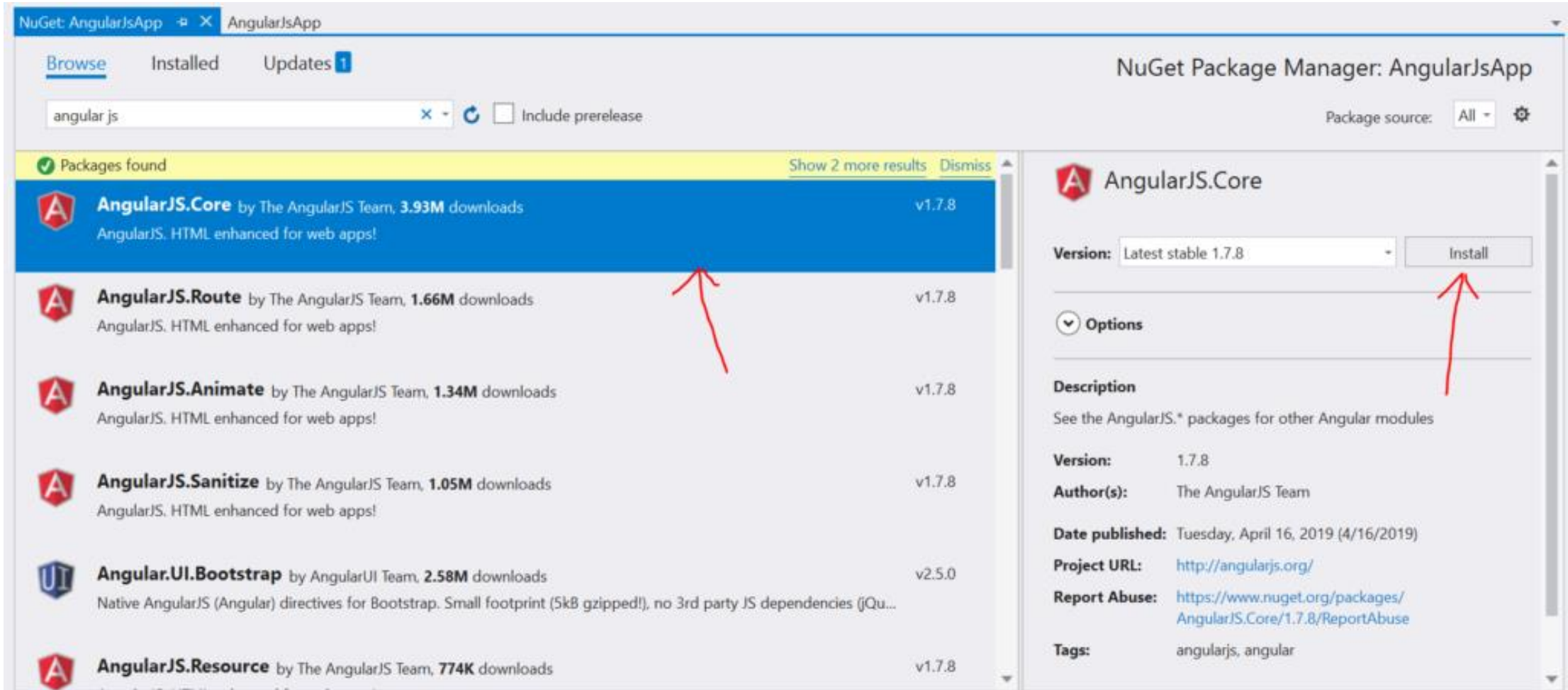
# Setup angularjs application in Visual Studio 2019

- Right-click on your project select Manage NuGet packages



# Setup angularjs application in Visual Studio 2019

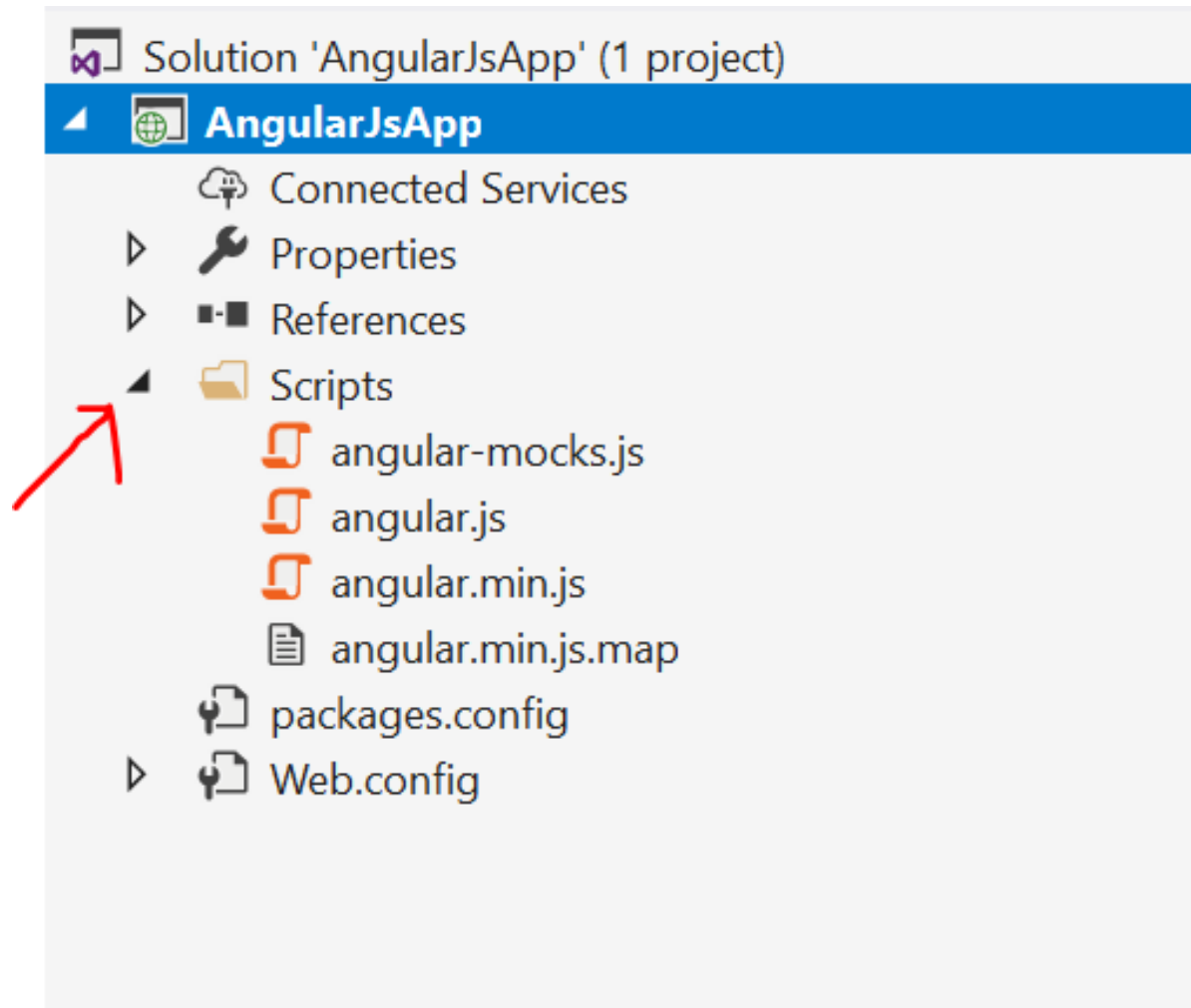
- Browse the angularjs.core and install





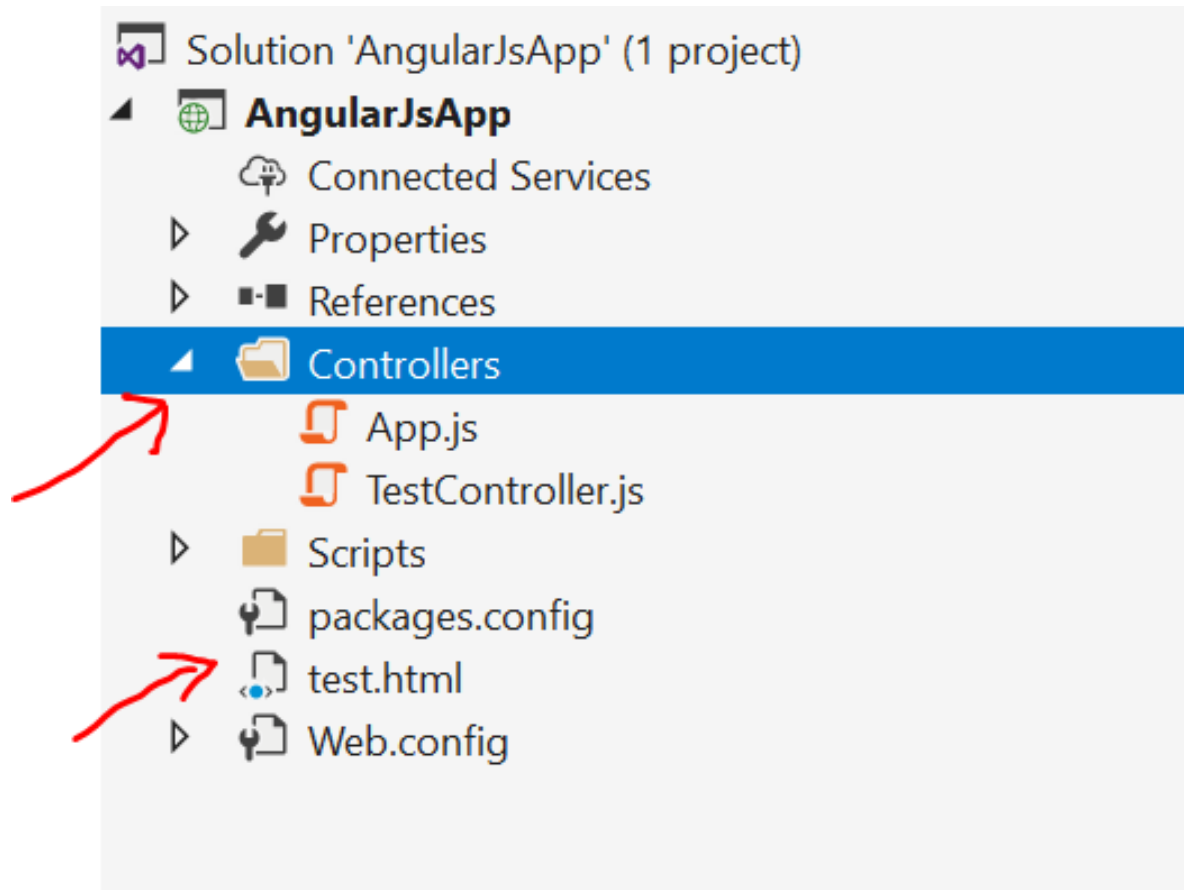
# Setup angularjs application in Visual Studio 2019

- Once installed the angularjs you will have js file in a script folder



# Setup angularjs application in Visual Studio 2019

- Setup is done. Now let's test by using one sample example. Create a directory structure for angularJs application following the below image.



**App.js** `var app = angular.module('myapp', []);`

**TestController.js** `app.controller('TestController', function ($scope) {  
 $scope.testmessage = "Setup angularjs application on visual studio 2019";  
});`

**Test.html**

```
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="utf-8" />  
    <title></title>  
    <script src="Scripts/angular.js"></script>  
    <script src="Controllers/App.js"></script>  
    <script src="Controllers/TestController.js"></script>  
</head>  
<body ng-app="myapp">  
    <div ng-controller="TestController">  
        <h2>{{testmessage}}</h2>  
    </div>  
</body>  
</html>
```

# REACT

- Unlike Angular, which offers a full Model-View-Controller pattern implementation, React is only concerned with views. It's not a framework, just a library. There are a number of libraries that are designed to be used with React to produce rich single page applications.
- One of React's most important features is its use of a virtual DOM. The virtual DOM provides React with several advantages, including performance (the virtual DOM can optimize which parts of the actual DOM need to be updated) and testability (no need to have a browser to test React and its interactions with its virtual DOM).
- Rather than having a strict separation between code and markup (with references to JavaScript appearing in HTML attributes perhaps), React adds HTML directly within its JavaScript code as JSX. JSX is HTML-like syntax that can compile down to pure JavaScript.

# REACT

- For Example

```
<ul>
  { authors.map(author =>
    <li key={author.id}>{author.name}</li>
  )}
</ul>
```





- Because React isn't a full framework, you'll typically want other libraries to handle things like routing, web API calls, and dependency management. The nice thing is, you can pick the best library for each of these.

# Create a Node.js and React app in Visual Studio

## Prerequisites

- Visual Studio installed and the Node.js development workload.
- If you need to install the workload but already have Visual Studio, go to Tools > Get Tools and Features..., which opens the Visual Studio Installer. Choose the Node.js development workload, then choose Modify.

### Web & Cloud (7)

 <p>ASP.NET and web development Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and container development tools.</p> <input checked="" type="checkbox"/>	 <p>Azure development Azure SDK, tools, and projects for developing cloud apps and creating resources.</p> <input checked="" type="checkbox"/>
 <p>Python development Editing, debugging, interactive development and source control for Python.</p> <input type="checkbox"/>	 <p>Node.js development Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.</p> <input checked="" type="checkbox"/>

# Create a Node.js and React app in Visual Studio

## Create a Node.js Web Application Project

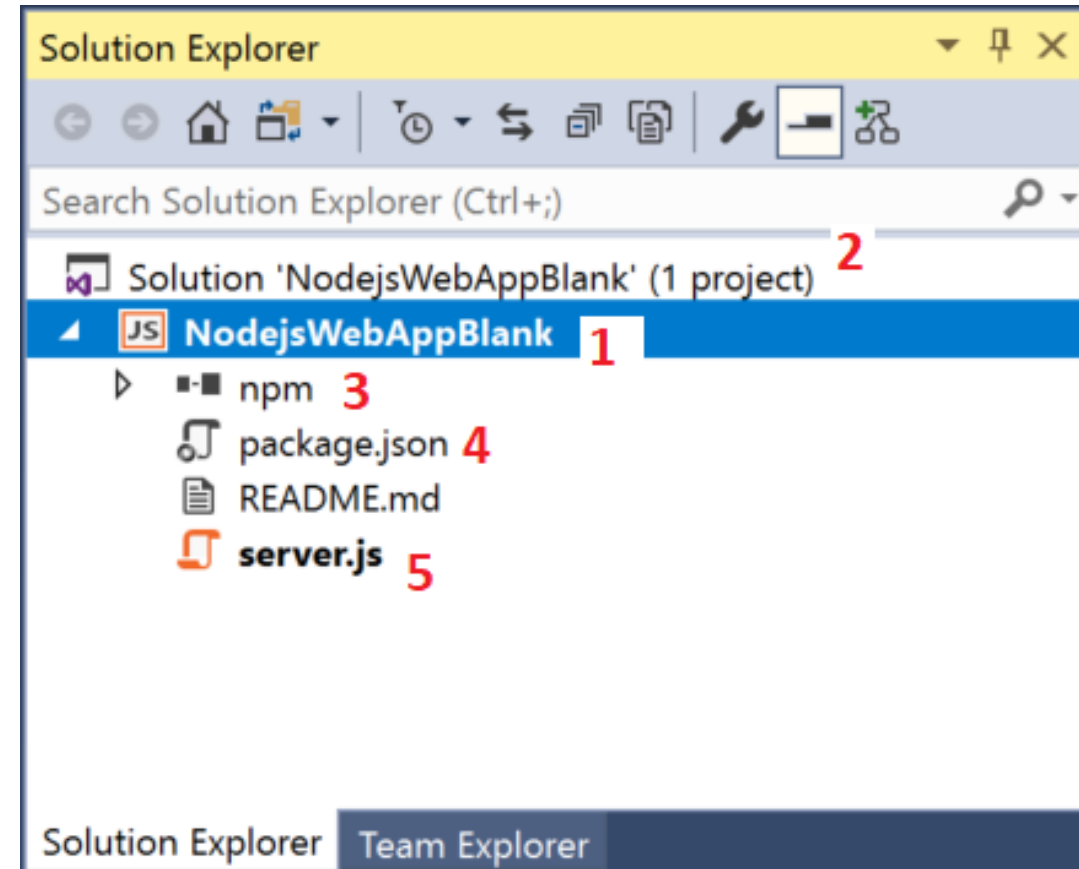
1. Open Visual Studio.
2. Create a new project.
  - Press Esc to close the start window. Type Ctrl + Q to open the search box, type Node.js, then choose Blank Node.js Web Application - JavaScript
  - In the dialog box that appears, choose Create.
  - If you don't see the Blank Node.js Web Application project template, you must add the Node.js development workload.
  - Visual Studio creates the new solution and opens your project.

# Create a Node.js and React app in Visual Studio

## Add npm packages

This app requires a number of npm modules to run correctly.

- react
- react-dom
- express
- path
- ts-loader
- typescript
- webpack
- webpack-cli

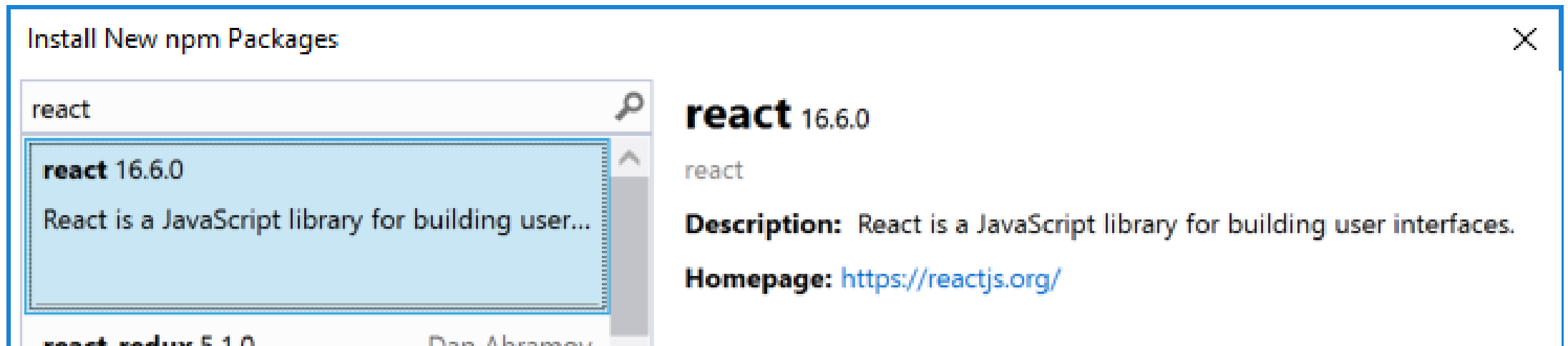




# Create a Node.js and React app in Visual Studio

## Add npm packages

1. In Solution Explorer (right pane), right-click the npm node in the project and choose Install New npm Packages. In the Install New npm Packages dialog box, you can choose to install the most current package version or specify a version.
2. In the Install New npm Packages dialog box, search for the react package, and select Install Package to install it.



# Create a Node.js and React app in Visual Studio

When installed, the package appears under the npm node.

- The project's package.json file is updated with the new package information including the package version.
- Instead of using the UI to search for and add the rest of the packages one at a time, paste the following code into package.json.

To do this, add a dependencies section with this code:

JSON

```
"dependencies": {  
  "express": "~4.17.1",  
  "path": "~0.12.7",  
  "react": "~16.13.1",  
  "react-dom": "~16.13.1",  
  "ts-loader": "~7.0.1",  
  "typescript": "~3.8.3",  
  "webpack": "~4.42.1",  
  "webpack-cli": "~3.3.11"  
}
```

# CHOOSING A SPA FRAMEWORK

- When considering which JavaScript framework will work best to support your SPA, keep in mind the following considerations:
  - Is your team familiar with the framework and its dependencies (including TypeScript in some cases)?
  - How opinionated is the framework, and do you agree with its default way of doing things?
  - Does it (or a companion library) include all of the features your app requires?
  - Is it well documented?
  - How active is its community? Are new projects being built with it?
  - How active is its core team? Are issues being resolved and new versions shipped regularly?
- JavaScript frameworks continue to evolve with breakneck speed. Use the considerations listed above to help mitigate the risk of choosing a framework you'll later regret being dependent upon

# Discussion Exercise

1. Write about the State Management Strategies.
2. What is Session State? Show with an example to manage session state in ASP.NET Core.
3. Show the difference between TempData and Using HttpContext with suitable example.
4. How do you manage to handle state with client side strategies?