

ANGULAR BASICS



PROGRAMMING
FOR BEGINNERS

J KING

HTML BASICS



PROGRAMMING
FOR BEGINNERS

J KING

ANGULAR BASIC



HTML AND ANGULAR BASICS

PROGRAMMING FOR BEGINNERS
J KING

HTML INTRODUCTION

HTML - BASICS

HTML - PARAGRAPH

HTML - TEXT FORMATTING

HTML - QUOTATIONS

HTML - TABLES

HTML - LISTS

HTML - ATTRIBUTES

HTML - SPELL CHECK

HTML - COLOR STYLES AND HSL

HTML - LAYOUT

INTRODUCTION TO CSS

ANGULARJS

ANGULARJS FIRST EXAMPLE

ANGULARJS DATA BINDING

ANGULARJS EXPRESSIONS

ANGULARJS DIRECTIVES

ANGULARJS CONTROLLERS

ANGULAR 8 TUTORIAL

ANGULAR 8 INTRODUCTION

ANGULAR 8 INSTALLATION

ANGULAR 8 FIRST APP

HOW AN ANGULAR'S APP GET LOADED AND STARTED

ANGULAR 8 ARCHITECTURE

ANGULAR 8 DIRECTIVES

ANGULAR 8 NGIF DIRECTIVE

ANGULAR 8 *NGFOR DIRECTIVE

ANGULAR 8 NGSWITCH DIRECTIVE

DATA BINDING IN ANGULAR 8

PROPERTY BINDING IN ANGULAR 8

HTML BASICS

PROGRAMMING FOR BEGINNERS
J KING

HTML INTRODUCTION

HTML stands for hypertext markup language .

The web pages are designed using markup language.

HTML is a combination of the Hypertext and Markup language.

Hypertext defines the web pages link .

Markup language is used to define the text document that defines the structure of the web pages within the tag.

This language is used to annotate (make notes for the computer) text so that it can be understood and manipulated by a machine.

Most markup languages (e.g. HTML) are human readable.

Language uses tags to define which manipulation is required on the text.

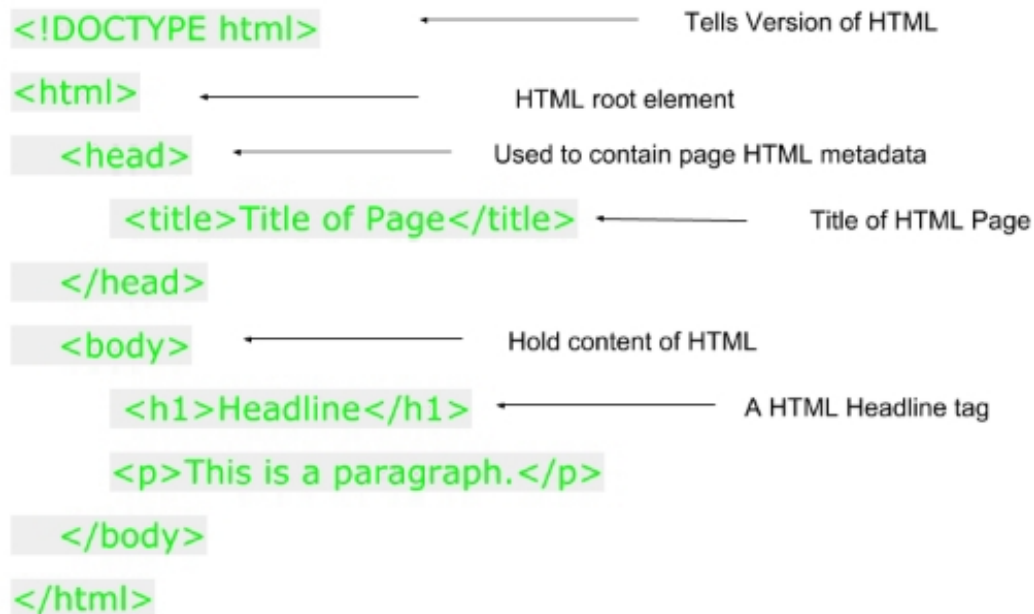
Elements and Tag

HTML uses predefined tags and elements that tell the browser about the display properties of content. If a tag is not closed then that effect is applied by the browser until the end of the page.

HTML page structure

Below is the Basic structure of the HTML page. It contains certain elements such as head, title, body, ... etc. Those elements are used to construct web pages blocks.

HTML PAGE STRUCTURE



<DOCTYPE! html> :Uses this tag to tell the version of HTML. This currently indicates the version is HTML 5.

<html>: This is called the root element of HTML, and is used to wrap all code.

<head>: Head tag contains metadata, caption, CSS page etc. All the HTML elements which can be used within the element `< head >` are:

- `<style>`
- `<title>`
- `<base>`
- `<noscript>`
- `<script>`
- `<meta>`
- `<title>`

<body>: Body tag encloses all the data a web page has from texts to links. This element contains all of the content that you see rendered in the

browser.

Example: Any text editor (notepad) may be used to create HTML page. Then save the file using the extension to.htm or.html and open the file in the browser. It will get a response from the HTML page.

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo web page</title>
        <style>
            h1 {
                color:#009900;
                font-size:46px;
            }
            p {
                font-size:17px;
                margin-top:-25px;
                margin-left:15px;
            }
        </style>
    </head>
    <body>
        <h1>HtmlforGeeks</h1>
        <p>A computer science portal for geeks</p>
    </body>
</html>
```

Output

HtmlforGeeks

A computer science portal for geeks

Features of HTML:

Learning is easy, and easy to use.

It is independent of platform.

A web page can include images, video and audio.

Could add hypertext to the text.

It is the language of markup.

Advantages:

It is a language of simple markup. Its easy to implement.

Helps develop Web programming fundamentals.

Professional career boost.

HTML is used for web-building.

It is compatible with all browsers.

It can be built into other languages such as CSS, JavaScript etc.

Disadvantages:

HTML can only create static web pages, so that other languages have to be used for dynamic web pages.

You have to write a large amount of code to create a simple web page.

Safety feature isn't good.

HTML - Basics

There are different tags that we need to consider and include in HTML when starting code.

These tags help to organize and format elements in our script or web pages in a basic way.

These step-by - step procedures will guide you through the HTML write process.

Basic HTML Document

Each HTML document starts with an HTML tag. While this is not mandatory but it is a good convention to start the document with the tag mentioned below:

`<!DOCTYPE html>`

<html> : Each HTML code must be contained between the basic HTML tags. Beginning with **<html>** tag and ending with **<html>** tag.

<head> : Next comes the head tag which contains all the web page or document header information such as the page title and other miscellaneous information. This information is contained in the head tag that opens and ends with **<head>** . The contents of this tag will of course be explained in the subsequent sections.

<title>: We may use the `< title >` tag to mention the title of a web page. This is header information, and is therefore mentioned in the header tags. The tag starts with `<titl >` and closes with `< /title >`

<body>: The next step of all the tags we've learned so far is the most important. The Body tag contains the page's actual body that will be visible to all users. With `<body>` this opens, and ends with `</body>`. Any content contained within this tag will be displayed on the web page whether it is writings or images or audios or videos or even links. Later in the section we will see how we can insert mentioned content into our web pages using different tags.

The entire code pattern will appear something like this:

```
<html>
<head>
    <!-- Information about the page -->
    <!--This is the comment tag-->
    <title>HtmlforGeeks</title>
</head>
<body>
    <!--Contents of the webpage-->
</body>
</html>
```

HTML Headings

These tags help us give headings about a webpage 's content. These tags are written mainly inside the body tag.

HTML gives us six tag headings from < h1 > to < h6 >.

Each tag displays the heading in a different font size and style.

```
<html>
<head>
    <title>HtmlforGeeks</title>
</head>
<body>
    <h1>Hello HtmlforGeeks</h1>
    <h2>Hello HtmlforGeeks</h2>
    <h3>Hello HtmlforGeeks</h3>
    <h4>Hello HtmlforGeeks</h4>
    <h5>Hello HtmlforGeeks</h5>
    <h6>Hello HtmlforGeeks</h6>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

HTML Paragraph

These tags help us write statements about paragraphs in a web page. They begin with tag <p> and end with tag </p>. Here the tag
 is used to break line and act as a return to the carriage. The
 tag is empty.

```
<html>
<head>
    <title>HtmlforGeeks</title>
</head>
<body>
    <h1>Hello HtmlforGeeks</h1>
    <p>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
    </p>
</body>
</html>
```

Output

Hello HtmlforGeeks

A Computer Science portal for geeks
A Computer Science portal for geeks
A Computer Science portal for geeks

HTML Horizontal Lines

Using the < hr > tag, the page is broken into different parts, creating horizontal margins using a horizontal line running from left to right side of the page.

This is also an empty tag, and takes no further statements.

```
<html>
<head>
    <title>HtmlforGeeks</title>
</head>
<body>
    <h1>Hello HtmlforGeeks</h1>
    <p>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
    </p>
    <hr>
    <p>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
    </p>
    <hr>
    <p>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
        A Computer Science portal for geeks<br>
    </p>
    <hr>
</body>
</html>
```

Output

Hello HtmlforGeeks

A Computer Science portal for geeks
A Computer Science portal for geeks
A Computer Science portal for geeks

A Computer Science portal for geeks
A Computer Science portal for geeks
A Computer Science portal for geeks

A Computer Science portal for geeks
A Computer Science portal for geeks
A Computer Science portal for geeks

HTML Images

The image tag is used for inserting a picture into our web page. The image source to be inserted is placed inside the tag ``.

```
<html>
```

```
<head>
```

```
    <title>HtmlforGeeks</title>
```

```
</head>
```

```
<body>
```

```
    
```

```
</body>
```

```
</html>
```

HTML - Paragraph

<p> tag:

In HTML the <p> tag defines a paragraph. These have tags which open and close.

Thus anything named in p> and /p> is treated as a paragraph.

Most browsers read a line as a paragraph even if we don't use the closing tag, i.e., /p>, but this can lead to unforeseen results.

So, both are a good convention and we need to use the closing tag.

Syntax:

```
<p> Content </p>
```

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Paragraph</title>
</head>
<body>
    <p>A Computer Science portal for geeks.</p>
    <p>It contains well written, well thought articles.</p>
</body>
</html>
```

Output

A Computer Science portal for geeks.

It contains well written, well thought articles.

Looking at the webpage, we see that just a few spaces are added before and after a paragraph. By default HTML does so. Let's look at just a few paragraph tag properties:

- As already mentioned, before and after any paragraph, the <p> tag automatically adds space, which is basically margins that the

browser adds.

- If a user adds multiple spaces, this is reduced to a single space by the browser.
- When a user adds several lines, the browser reduces them to a single line.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Display_Paragraph</title>
</head>
<body>
    <p>
        This paragraph has multiple
        lines. But HTML reduces them
        to a single line, omitting
        the carriage return we have used.
    </p>
    <p>
        This paragraph    has multiple
        spaces.        But HTML reduces them
        all to a        single space, omitting
        the extra spaces and line we have used.
    </p>
</body>
</html>
```

Output

This paragraph has multiple lines. But HTML reduces them to a single line, omitting the carriage return we have used.

This paragraph has multiple spaces. But HTML reduces them all to a single space, omitting the extra spaces and line we have used.

 tag:

There's a way to let the HTML know where the browser needs to change the lines by using tag
. These tags are without a closing tag. So, just one opening tag will change that line.

Syntax:

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Display_Paragraph</title>
</head>
<body>
    <p>
        This paragraph has multiple<br>lines.
        But HTML reduces them<br>to a single
        line, omitting<br>the carriage return
        we have used.
    </p>
</body>
</html>
```

Output

This paragraph has multiple lines. But HTML reduces them to a single line, omitting the carriage return we have used.

<align> attribute:

The < p > tag specifically supports the alignment attribute and enables us to align our paragraphs in alignment to the left, right or centre.

Syntax:

<p align="value">

Example

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Paragraph</title>
</head>
<body>
    <p align="center">Welcome Geeks</p>
    <p align="left">A Computer Science
portal for geeks.</p>
    <p align="right">It contains well
written, well thought articles.</p>
</body>
</html>
```

Output

Welcome Geeks

A Computer Science portal for geeks.

It contains well written, well thought articles

The <pre> element:

We have seen how the paragraph tag ignores all changes in lines and extra spaces within a paragraph, but there is a way to preserve that through the use of < pre > tags.

It includes an opening and a closing tag, too.

It displays a text in a fixed height and width and retains the extra lines and spaces that we use.

Syntax:

```
<pre> Content </pre>
```

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Display_Paragraph</title>
</head>
<body>
    <pre>
```

This paragraph has multiple
lines. But it is displayed
as it is unlike the paragraph
tag.

</pre>

<pre>

This paragraph has multiple
spaces. But it is displayed
as it is unlike the paragraph
tag.

</pre>

</body>

</html>

Output

This paragraph has multiple
lines. But it is displayed
as it is unlike the paragraph
tag.

This paragraph has multiple
spaces. But it is displayed
as it is unlike the paragraph
tag.

HTML - Text Formatting

HTML gives us the ability to format text just like we do in MS Word or any software for editing text. We 'd go through just a few such options in this article.

Making text Bold or Strong:

We can use the `< b >` tag to make the text bold. The tag uses the tag both to open and to close. The text which must be bold must be in the tag `< b >` and `< /b >`.

Also, we can use the `< strong >` tag to make the text strong, with added semantic value. It opens with `< strong >`, and ends with tag `< /strong >`.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Bold</title>
</head>
<body>
    <!--Normal text-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in Bold-->
    <p><b>Hello HtmlforGeeks</b></p>
    <!--Text in Strong-->
    <p><strong>Hello HtmlforGeeks</strong></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Making text Italic or emphasizing:

Using the <i> tag to italicize the text. It opens with <i> and terminates with tag </i>.

The tag is used to emphasize the text, with added semantic significance. It opens with and concludes with tag .

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Italic</title>
</head>
<body>
    <!--Normal text-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in Italics-->
    <p><i>Hello HtmlforGeeks</i></p>
    <!--Text in Emphasize-->
    <p><em>Hello HtmlforGeeks</em></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Highlighting a text:

Highlighting a text in HTML can also be done using the tag `< mark >`. It has a `<mark>` opening tag, and a `</mark>` closing tag.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Highlight</title>
</head>
<body>
    <!--Text in Normal-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in Highlight-->
    <p><mark>Hello HtmlforGeeks</mark></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Making a text Subscript or Superscript:

The element `<sup>` is used to superscript a text, and the element `<sub>` is used to subscript a text. Both have opening tag and closing tag.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Superscript and Subscript</title>
</head>
<body>
    <!--Text in Normal-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in Superscript-->
    <p>Hello <sup>HtmlforGeeks</sup></p>
    <!--Text in Subscript-->
```

```
<p>Hello <sub>HtmlforGeeks</sub></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Hello HtmlforGeeks

Making text smaller:

The element `<small>` serves to make the text smaller.

The text to be shown smaller should be written inside the tag `<small>` and `</small>`.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Small</title>
</head>
<body>
    <!--Text in Normal-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in small-->
    <p><small>Hello HtmlforGeeks</small></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

Striking through the text:

The element `` is used to mark the part as deleted by striking through the text.

It also has an opening tag and an end tag.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Delete</title>
</head>
<body>
    <!--Text in Normal-->
    <p>Hello HtmlforGeeks</p>
    <!--Text in Delete-->
    <p><del>Hello HtmlforGeeks</del></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

~~Hello HtmlforGeeks~~

Adding a text:

The element `<ins>` is used to underline a text which marks the part as inserted or added.

It also has an opening tag and an end tag.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Inserting</title>
</head>
<body>
    <!--Text in Normal-->
    <p>Hello HtmlforGeeks</p>
```

```
<!--Text in Insert-->
<p><ins>Hello HtmlforGeeks</ins></p>
</body>
</html>
```

Output

Hello HtmlforGeeks

Hello HtmlforGeeks

HTML - Quotations

The HTML Quotation elements are used to insert quoted texts into a web page, which is a portion of texts that are different from the normal texts on the web page.

1.<q> element:

The element < q > is used for setting a set of text within quotation marks. It has tags which open and close.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Quotations</title>
</head>
<body>
    <h3>HtmlforGeeks</h3>
    <p>The quick brown fox jumps over the lazy dog<br></p>
    <!--Inside quotes-->
    <p><q>The quick brown fox jumps over the lazy dog</q></p>
</body>
</html>
```

Output

HtmlforGeeks

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

2.<blockquote> element:

Also, the < blockquote > element is used differently for quotations.

Instead of putting the text into quotes, the alignment changes to make it unique to others.

It has tags which open and close.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Blockquote</title>
</head>
<body>
    <h3>HtmlforGeeks</h3>
    <p>The quick brown fox jumps over the lazy dog<br></p>
    <!--Inside blockquotes-->
    <p><blockquote>The quick brown fox jumps
over the lazy dog</blockquote></p>
</body>
</html>
```

Output

HtmlforGeeks

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

3.<address> element:

We can define an address in a webpage using the < address > element, and emphasize the text inside the address tag.

It has tags which open and close.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Address</title>
</head>
<body>
    <h3>HtmlforGeeks</h3>
```

```
<address>
<p>Address:<br>
530-B, Central Steet,<br>
Sector-123, Noida,Uttarpradesh – 201305</p>
</address>
</body>
</html>
```

Output

HtmlforGeeks

Address:

530-B, Central Steet,

Sector-123, Noida,Uttarpradesh – 201305

4.<abbr> element:

The < abbr > element is used as an acronym or abbreviation for defining a text.

When you mouse over the < abbr > element, the title attribute can be used to show the full version of the abbreviation / acronym.

It has tags which open and close. This is advantageous for browsers and search engines.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Abbreviations</title>
</head>
<body>
<h3>HtmlforGeeks</h3>
<!--Here the marked text is the acronym-->
<p>Welcome to <abbr title="HtmlforGeeks">HfG</abbr></p>
</body>
</html>
```

Output

HtmlforGeeks

Welcome to HfG

HTML - Tables

A table is an array of data in rows and columns, or perhaps in a more complex structure. Tables are widely used in data analysis , research, and communication.

Tables are useful for various tasks, such as presenting text and numerical information.

Tables may be used in tabular form layout to compare two or more items.

Databases are created through tables.

Defining Tables -HTML

Defines an HTML table with the tag , "table." The tag "tr" is used to define each table row.

A table header tag is "th" . Table headings are set to bold and centered by default.

A table data / cell tag is "td" .

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
    <table style="width:100%">
```

```
    <tr>
```

```
    <th>Firstname</th>
```

```
    <th>Lastname</th>
```

```
    <th>Age</th>
```

```
    </tr>
```

```
    <tr>
```

```
    <td>Priya</td>
```

```
    <td>Sharma</td>
```

```
    <td>24</td>
```

```
    </tr>
```

```
    <tr>
```

```
    <td>Arun</td>
```

```
<td>Singh</td>
<td>32</td>
</tr>
<tr>
<td>Sam</td>
<td>Watson</td>
<td>41</td>
</tr>
</table>
```

```
</body>
```

```
</html>
```

Output

Firstname	Lastname	Age
Priya	Sharma	24
Arun	Singh	32
Sam	Watson	41

Adding Border Spacing - Html Table

Border spacing specifies the intercell space. We must use the CSS border-spacing property to set the border spacing for a table.

```
table {
    border-spacing: 5px;
}
```

Example

```
<html>
```

```
<head>
```

```
<style>
table, th, td {
```



```

        border: 1px solid black;
    }
    table {
        border-spacing: 5px;
    }
</style>
</head>

<body>

    <table style="width:100%">
    <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
    </tr>
    <tr>
    <td>Priya</td>
    <td>Sharma</td>
    <td>24</td>
    </tr>
    <tr>
    <td>Arun</td>
    <td>Singh</td>
    <td>32</td>
    </tr>
    <tr>
    <td>Sam</td>
    <td>Watson</td>
    <td>41</td>
    </tr>
    </table>

</body>

</html>

```

Output

Firstname	Lastname	Age
Priya	Sharma	24

Arun	Singh	32
Sam	Watson	41

Creating Nested Tables - HTML

Simply nesting tables means creating a table within another table.

Nesting tables can lead to complex layouts of tables which are visually interesting .

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
  <table border=5 bordercolor=black>
```

```
    <tr>
```

```
      <td>
```

```
        Fisrt Column of Outer Table
```

```
      </td>
```

```
      <td>
```

```
        <table border=5 bordercolor=grey>
```

```
          <tr>
```

```
            <td>
```

```
              First row of Inner Table
```

```
            </td>
```

```
          </tr>
```

```
          <tr>
```

```
            <td>
```

```
              Second row of Inner Table
```

```
            </td>
```

```
          </tr>
```

```
        </table>
```

```
      </td>
```

```
    </tr>
```

```
  </table>
```

```
</body>
```

</html>

Output

Fisrt Column of Outer Table	First row of Inner Table
	Second row of Inner Table

HTML - Lists

A list is a record of short pieces of information, such as names of people, typically written or printed with a single thing on each line, and ordered in a manner that makes it easy to find a particular thing.

Example

- A shopping list
- To-do list

Lists in HTML

HTML offers three ways to specify data lists. All lists must have one or more listings

ELEMENTS

List types that can be used in HTML include:

ul: An unordered list. Using plain bullets this will list items.

ol: An ordered list. This will use various numeral schemes to list your items.

dl: A list of definitions. This arranges your items the same way a dictionary arranges them.

The Unordered HTML List

An unordered list begins with the tag "ul." Each list item begins with the tag "li." By default the list items are marked with bullets i.e. small black circles.

Example

```
<!DOCTYPE html>
<html>
<body>

<h2>Grocery list</h2>

<ul>
<li>Bread</li>
```

```
<li>Eggs</li>
<li>Milk</li>
<li>Coffee</li>
</ul>
```

```
</body>
</html>
```

Output

Grocery list

- Bread
- Eggs
- Milk
- Coffee

The HTML Ordered List

An ordered list begins with the tag "ol." Each item to the list starts with the tag "li." By default the list items are marked with numbers.

EXAMPLE

```
<!DOCTYPE html>
<html>
<body>

<h2>Grocery List</h2>

<ol>
<li>Bread</li>
<li>Eggs</li>
<li>Milk</li>
<li>Coffee</li>
</ol>

</body>
```

</html>

Output

Grocery List

1. Bread
2. Eggs
3. Milk
4. Coffee

The HTML Description List

A list of terms is a description list, with a description of each term.

The tag < dl > defines the list of descriptions, the tag < dt > defines the term name and each term is described by the tag < dd >.

Example

<!DOCTYPE html>

<html>

<body>

<h2>A Description List</h2>

<dl>

<dt>Coffee</dt>

<dd>- 500 gms</dd>

<dt>Milk</dt>

<dd>- 1 ltr Tetra Pack</dd>

</dl>

</body>

</html>

Output

A Description List

Coffee

- 500 gms

Milk

- 1 ltr Tetra Pack

Nested List in HTML:

A nest list is a list containing a list inside a list.

```
<!DOCTYPE html>
<html>
<body>

<h2>A Nested List</h2>

<ul>
<li>Coffee</li>
<li>Tea
    <ul>
    <li>Black tea</li>
    <li>Green tea</li>
    </ul>
</li>
<li>Milk</li>
</ul>

</body>
</html>
```

Output

A Nested List

- Coffee
- Tea
 - Black tea
 - Green tea
- Milk

HTML - Attributes

An attribute is used to convey additional element information.

- All elements of HTML may possess attributes. Further information about an element is provided by attributes.
- It will require two parameters: a name and a value.
- These define the element's properties, and are placed within the element 's opening tag.
- The name parameter takes the name of the property we want to assign to the element and the value takes the value of the properties or the extent of the property names which can be aligned over the element.
- Each name has a certain value which must be written in quotes

Syntax:

<element attribute_name="attribute_value">

Most commonly used Attributes in HTML:

src Attribute :

If we want to insert an image into a webpage, then the < img > tag and the src attribute must be used. We will need to specify the image address inside the double quote as the value of the attribute.

```
<html>
```

```
<head>
```

```
    <title>src Attribute</title>
```

```
</head>
```

```
<body>
```

```
    
```

```
</body>
```


</html>

alt Attribute :

As the name goes, this is an alternative tag used to display or show something if the primary attribute, i.e. the < img > tag, does not display the value assigned to it.

This may also be used to describe the image to a developer .

```
<html>
<head>
    <title>alt Attribute</title>
</head>
<body>
    <!--If the image is not found or the img field
    is left blank the alt value gets displayed-->
    <br>
    <img src="" alt="Since the src value is blank,the alt value is
displayed">
</body>
</html>
```

The id Attribute :

This attribute is used to give an element a unique identifier.

Situations may arise when we need to access a specific element which might have a name similar to the others.

In that case, we are providing different ids for different elements so they can be accessed in a unique way.

In general, the properties that extend I d usage are used in CSS.

```
<html>
<head>
    <title>id Attribute</title>
</head>
<body>
    <p id = "GfG">Hello geeks<br>
```

```
<p id = "ui">This is unique to this paragraph<br>
<p id = "head">This is also unique to this paragraph
</body>
</html>
```

The title Attribute :

The title attribute is used to explain an element when mouse is hovering over it.

The behavior differs with different elements but the value is generally displayed while the mouse pointer is loaded or hovering over it.

```
<html>
<head>
    <title>title Attribute</title>
</head>
<body>
    <h3 title="Hello HtmlforGeeks">Hover to see the effect</h3>
</body>
</html>
```

The href Attribute :

Uses this attribute to specify a link to any address. This attribute is used in conjunction with tag <a>. The link inside the href attribute is linked to the text inside the tag that is displayed.

Clicking on the text redirects us to the link.

The link is opened in the same tag by default, but by using the **target** attribute and setting its value to "**_blank,**" we are redirected to another tab or window based on the browser configuration.

```
<html>
<head>
    <title>link Attribute</title>
</head>
<body>
```

```
<a href="https://www.learnhtml.org/">
```

Click to open in the same tab

```
</a><br>
```

```
<a href="https://www.learnhtml.org/" target="_blank">
```

Click to open in a different tab

```
</a>
```

```
</body>
```

```
</html>
```

HTML - Spell Check

In HTML, the spell check feature is used to detect grammatical or spelling errors in text fields.

The function SpellCheck can be applied to HTML forms using the attribute SpellCheck.

The spell check attribute is an enumerated attribute .

That defines whether the HTML element will be checked for errors or not.

It can be used in HTML with the fields "input" and "textarea."

Syntax :

spellcheck attribute in an input field in html

```
<input type="text" spellcheck="value">
```

Syntax for spellcheck in a **textarea** field

```
<textarea type="text" spellcheck="value"></textarea>
```

The value assigned to spell check in the above syntax will define whether spell check is enabled on the element or not. There are two valid values in the spell check attribute which are:

True: It defines error checking for the HTML element.

False: This define that errors should not be checked for the HTML element.

If the attribute is not set it takes the default value that is generally defined by the type of element and the browser. Of the ancestral element, the value can also be inherited.

Enabling spell check in an HTML form :

The spell check attribute is set to 'true' to allow spell check in an HTML form. The sample HTML program with spell check enabled is below.

```
<!DOCTYPE html>
<html>
<body>
<h3>Example of Enabling SpellCheck</h3>
```

```
<form>
<p>
<input type="text" spellcheck="true">
</p>
<p>
<textarea spellcheck="true"></textarea>
</p>
<button type="reset">Reset</button>
</form>
</body>
</html>
```

Disabling Spell Check in a HTML Form :

The spell check attribute is set to "false" to deactivate spell check in an HTML The sample HTML program with spell check disabled is shown below.

```
<!DOCTYPE html>
<html>
<body>
<h3>Example of Disabling SpellCheck</h3>
<form>
<p>
<input type="text" spellcheck="false">
</p>
<p>
<textarea spellcheck="false"></textarea>
</p>
<button type="reset">Reset</button>
</form>
</body>
</html>
```

HTML - Color Styles and HSL

In Html colors to make the page more attractive. Here are the various styles which can be used by combining different colors to create new colours.

1.Hexadecimal Style : In this style we define the color in hexadecimal number, in 6 digits (from 0 to F). '#' denotes it. The first two digits show red, next two green and the last two blue.

Examples: If we want all 'h1' tags, the purple color.

```
h1{  
color:#00FF00;  
}
```

2.RGB Style [Red Green Blue]: In this we must give 3 numbers indicating the amount of red , green and blue colors required in the mixed color, respectively. Each color has a range from 0 to 255.

Example: If we want all 'h1' tags green color

```
h1{  
color:rgb(0, 255, 0);  
}
```

Note : rgb(255, 255, 255) is White color and rgb(0, 0, 0) is Black color

3.RGBA Style [Red Green Blue Alpha] : This style enables us to make the color transparent at will. The degree of transparency indicates Alpha. The green , blue, and red range from 0 to 255 and the alpha range from 0 to 1.

Example: If we want green color for all 'h1' tags.

```
h1{  
color:rgba(11, 99, 150, 1);  
}
```

4 . HSL colors : Here 'H' means hue, 'S' means saturation and 'L' means lightness. The HSL color values shall be as follows:

Syntax:

```
hsl(hue, saturation, lightness)
```

Hue is the image's own colour. Its range is between 0 and 360. Its 0 is red, 120 is green and 240 is blue.

The intensity / purity of the hue is **saturation**. 0 percent for a gray shade and 100 percent for a full colour.

When full saturation of color occurs, the color is considered in the purest version.

Lightness is the brightness of the space in colour. 0 per cent is black, and 100 per cent white

```
h1{
  color:#00FF00;
  background-color: hsl(200, 20%, 40%);
  color: hsl(300, 30%, 90%);
}
```

We have 4096 different combinations of colors in total as we have the range of red , green and blue from 00 to FF each so we have $16 * 16 * 16$ different combinations of colours.

Then, we can achieve even more creative and large number of colors with hue , saturation and lightness.

```
<!-- Write HTML code here -->
<head>
  <title>HtmlforGeeks</title>
  <style type="text/css">
    h1{
      color:#0FFFFF0;
      background-color: hsl(200, 50%, 20%);
      color: hsl(200, 20%, 90%);

    }
    h4{
      color:rgb(0, 255, 0);
      background-color: hsl(150, 20%, 40%);
      color: hsl(360, 30%, 90%);
    }
    li{
```

```
        color:rgba(11, 99, 150, 1);
        background-color: hsl(250, 45%, 60%);
        color: hsl(175, 35%, 87%);
    }
</style>
</head>
<body>
    <h1>HtmlforGeeks</h1>
    <h4>Programming Languages</h4>
    <ul>
        <li>Java</li>
        <li>C++</li>
        <li>C</li>
    </ul>
</body>
</html>
```

HtmlforGeeks

Programming Languages

- Java
- C++
- C

HTML - Layout

Page layout is the part of graphic design dealing with visual element arrangements on a page.

The page layout is used to improve the look of the web pages.

To achieve a smooth flow of information and eye movement for maximum efficacy or impact, it establishes the overall appearance, relative importance, and relationships between the graphic elements.

Header Section	
Navigation Bar	
Index	Content section
Footer Section	

Page Layout Information:

Header: The portion of a front end used at the top of the page. Use < header > tag to add header section to web pages.

Navigation bar: Navigation bar is identical with menu list. Using hyperlink it is used to display information about content.

Index / Sidebar: it contains additional information or advertisements and does not need to be added to the page always.

Content Section : The section of content is the main part where content is displayed.

Footer: The section of the footer contains the contact information and other web-page queries. Always put the footer section on the bottom of web pages. The < footer > tag is used in Web pages to set the footer.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Layout</title>
    <style>
    .head1 {
    font-size:40px;
    color:#009900;
    font-weight:bold;
    }
    .head2 {
    font-size:17px;
    margin-left:10px;
    margin-bottom:15px;
    }
    body {
    margin: 0 auto;
    background-position:center;
    background-size: contain;
    }
    .menu {
    position: sticky;
    top: 0;
    background-color: #009900;
    padding:10px 0px 10px 0px;
    color:white;
    margin: 0 auto;
    overflow: hidden;
    }
    .menu a {
    float: left;
    color: white;
```

```
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 20px;
}
.menu-log {
right: auto;
float: right;
}
footer {
width: 100%;
bottom: 0px;
background-color: #000;
color: #fff;
position: absolute;
padding-top:20px;
padding-bottom:50px;
text-align:center;
font-size:30px;
font-weight:bold;
}
.body_sec {
margin-left:20px;
}
</style>
</head>

<body>

<!-- Header Section -->
<header>
<div class="head1">HtmlforGeeks</div>
<div class="head2">A computer science portal for geeks</div>
</header>
<!-- Menu Navigation Bar -->
<div class="menu">
<a href="#home">HOME</a>
<a href="#news">NEWS</a>
```

```
<a href="#notification">NOTIFICATIONS</a>
<div class="menu-log">
<a href="#login">LOGIN</a>
</div>
</div>
<!-- Body section -->
<div class = "body_sec">
<section id="Content">
<h3>Content section</h3>
</section>
</div>
<!-- Footer Section -->
<footer>Footer Section</footer>
</body>
</html>
```

Output



INTRODUCTION TO CSS

Cascading Style Sheets, called CSS, is a language simply designed to simplify the process of making web pages presentable. CSS lets you apply web pages with styles. More importantly, CSS allows you to do this regardless of the HTML which makes up each web page.

There are three CSS types that are given below:

Inline CSS

Internal or embedded CSS

External CSS

Inline CSS:

Inline CSS contains the CSS property which is known as inline CSS in the body section attached with element. This type of style is specified by the style attribute within an HTML tag

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline CSS</title>
  </head>

  <body>
    <p style = "color:#009900; font-size:50px;
      font-style:italic; text-align:center;">
      HtmlForGeeks
    </p>
  </body>
</html>
```

Output

HtmlForGeeks

Internal or Embedded CSS:

This may be used when unique styling of a single HTML document is required. The set of CSS rules should be in the Head section of the HTML file i.e. the CSS is embedded in the HTML file.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Internal CSS</title>
    <style>
      .main {
        text-align:center;
      }
      .GFG {
        color:#009900;
        font-size:50px;
        font-weight:bold;
      }
      .geeks {
        font-style:bold;
        font-size:20px;
      }
    </style>
  </head>
  <body>
    <div class = "main">
      <div class ="GFG">HtmlForGeeks</div>

      <div class ="geeks">
        A computer science portal for geeks
      </div>
    </div>
  </body>
</html>
```

Output

HtmlForGeeks

A computer science portal for geeks

External CSS:

External CSS contains a separate CSS file that only contains style property using tag attributes (e.g. class, I d, heading, ... etc.). CSS property written with.css extension in a separate file and should be linked to the HTML document using link tag.

This means that style can only be set once for each element, and will be applied across web pages.

Example:

Syntax

```
body {  
    background-color:powderblue;  
}  
.main {  
    text-align:center;  
}  
.GFG {  
    color:#009900;  
    font-size:50px;  
    font-weight:bold;  
}  
#geeks {  
    font-style:bold ;  
    font-size:20px;  
}
```

Below is the HTML file, using the external style sheet created

Link tag is used to link the html web page to the external style sheet.

The href attribute is used to specify where the external style sheet file is located.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="geeks.css"/>
  </head>

  <body>
    <div class = "main">
      <div class ="GFG">GeeksForGeeks</div>
      <div id ="geeks">
        A computer science portal for geeks
      </div>
    </div>
  </body>
</html>
```

Properties of CSS:

Inline CSS has the highest priority, then comes Internal / Embedded followed by the least prioritized External CSS. You can define multiple style sheets on a single page. If the styles are defined in multiple style sheets for an HTML tag then the order below is followed.

Since Inline has the highest priority, Inline styles override any styles which are defined in the internal and external style sheets.

Internal or Embedded is second in the priority list and overrides the styles in the style sheet on the external.

External style sheet has the lowest priority. If there are no styles defined in the inline or internal style sheet then the rules for the external style sheet are applied.

ANGULAR BASICS

PROGRAMMING FOR BEGINNERS
J KING

AngularJS

Angular JS is Google's open source JavaScript platform for creating Web apps. Anybody can use it freely, modify it and share it.

The AngularJS tutorial covers all AngularJS topics including mvc, expressions, directives, controllers, modules, scopes, filters, dom, forms, ajax, validation, utilities, animation, dependency injection, views, w3.css, etc.

Angular Js is Google made.

This is an excellent platform for the development of single step applications and business line applications.

Advantages

Dependency Injection: Dependency Injection describes a design pattern where components are given their dependencies rather than hard-coding them within the object.

Two way Data binding: AngularJS generates data binding two ways between the selected entity and the orderProp model. OrderProp is then used as commandBy filter data.

Testing: Angular JS is built in a way we can evaluate from the very beginning. So, evaluating all of its components by unit testing and end-to - end testing is very easy.

Model View Controller: Application in Angular JS is very simple to build in a clean MVC manner. Simply break the application code into MVC components i.e. Controller, View and model.

AngularJS First Example

AngularJS Applications are a combination of HTML and JavaScript. What you need first is an HTML page.

```
<!DOCTYPE html >
<html>
<head>
.
.
</head>
<body>
.
.
</body>
</html>
```

Second, the AngularJS JavaScript file must be included in the HTML page so we can use AngularJS:

```
<!DOCTYPE html >
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">
</script>
</head>
<body>
.
.
</body>
</html>
```

AngularJS First Example

Below is a simple example of "Hello Word," created with AngularJS. It specifies the AngularJS app of the Model, View, Controller.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">
</script>
</head>
<body ng-app="myapp">
<div ng-controller="HelloController" >
<h2>Hello {{helloTo.title}} !</h2>
</div>

<script>
angular.module("myapp", [])
  .controller("HelloController", function($scope) {
    $scope.helloTo = {};
    $scope.helloTo.title = "World, AngularJS";
  });
</script>
</body>
</html>
```

View Part

```
<div ng-controller="HelloController" >
<h2>Hello {{helloTo.title}} !</h2>
</div>
```

Controller Part

```
<script>
angular.module("myapp", [])
  .controller("HelloController", function($scope) {
    $scope.helloTo = {};
    $scope.helloTo.title = "World, AngularJS";
  });
</script>
```

AngularJS Data Binding

Data Binding is a very useful and efficient tool used in the development of software technologies. This serves as a bridge between the application's vision and the business logic.

AngularJS follows the pattern of two-way data binding.

One-way Data Binding

The one-way data binding is an approach in which the data model takes a value and incorporates it into an HTML feature. There's no way the model can be modified from view. This is used on traditional modeling systems. Such systems merely connect data in one direction.

Two-Way Data Binding

In Angular apps, data-binding is the automatic synchronization of data between the view components and model and.

Binding data helps you to view the model as the one-source-of-truth in your query. The vision is often a projection of the image. The view represents the transition if the paradigm is changed, and vice versa.

```
<!DOCTYPE html >
<html>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js" >
</script>
<body>
<div ng-app = "" ng-init = "firstName='Ajeet'" >
<p> Input something in the input box: </p>
<p> Name: <input type = "text" ng-model = "firstName" ></p>
<p> You wrote: {{ firstName }} </p>
</div>
</body>
</html>
```

Let's take another example where two text fields and two ng-model directives are linked together:

```
<!DOCTYPE html >
<html>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js" >
</script>
<body>
<div data-ng-app = "" data-ng-init = "quantity=1;price=20" >
<h2> Cost Calculator </h2>
Quantity: <input type = "number" ng-model = "quantity" >
Price: <input type = "number" ng-model = "price" >
<p><b> Total in rupees: </b> {{quantity * price}} </p>
</div>
</body>
</html>
```

AngularJS Expressions

Expressions are used in AngularJS to link the application data to HTML. AngularJS addresses the expression, and returns the result where the expression is written exactly.

Double braces {{expression}} used to write Expressions. They can also be written within a directive:

ng-bind = "expression" .

AngularJS expressions are very similar to the expressions in JavaScript. These can include the literals, operators, and variables. For instance:

{{ 5 + 5 }} or {{ firstName + " " + lastName }}

AngularJS Expressions Example

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app>
<p>A simple expression example: {{ 5 + 5 }}</p>
</div>
</body>
</html>
```

Note: When you delete the "ng-app" command, HTML shows the phrase without solving it.

See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
```

<p>If you remove the directive "ng-app", HTML will display the expression without solving it.</p>

<div>

<p>A simple expression example: {{ 5 + 5 }}</p>

</div>

</body>

</html>

Also, anywhere you want to write expressions, AngularJS will resolve the expression and return the output.

Let's take one example by changing its value to change the color of the input box.

See this example:

<!DOCTYPE html>

<html>

<script

src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">

</script>

<body>

<p>Change the value of the input field:</p>

<div ng-app="" ng-init="myCol='pink'">

<input style="background-color:{{myCol}}" ng-model="myCol" value="{{myCol}}">

</div>

<p>AngularJS resolves the expression and returns the result.</p>

<p>The background color of the input box will be whatever you write in the input field.</p>

</body>

</html>

AngularJS Numbers

AngularJS numbers are similar to the numbers on JavaScript.

<!DOCTYPE html>

<html>

<script

src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">


```
</script>
<body>
<div ng-app="" ng-init="quantity=5;cost=5">
<p>Total in dollar: {{ quantity * cost }}</p>
</div>
</body>
</html>
```

The same example can be used with ng-bind:

See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="quantity=5;cost=5">
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
</body>
</html>
```

AngularJS Strings

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="firstName='Sonoo';lastName='Jaiswal'">
<p>My full name is: {{ firstName + " " + lastName }}</p>
</div>
</body>
</html>
```

Same with ng-bind:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="firstName='Sonoo';lastName='Jaiswal'">
<p>My full name is: <span ng-bind="firstName + ' ' + lastName"></span>
</p>
</div>
</body>
</html>
```

AngularJS Objects

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="person=
{ firstName:'Sonoo',lastName:'Jaiswal'}">
<p>My name is {{ person.firstName }}</p>
</div>
</body>
</html>
```

Same with ng-bind:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="person=
{ firstName:'Sonoo',lastName:'Jaiswal'}">
<p>The name is <span ng-bind="person.firstName"></span></p>
```

```
</div>
</body>
</html>
```

AngularJS Arrays

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The first result is {{ points[0] }}</p>
</div>
</body>
</html>
```

Same with ng-bind:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The first result is <span ng-bind="points[0]"></span></p>
</div>
</body>
</html>
```

AngularJS Expressions Vs JavaScript expressions:

AngularJS expressions can be written in HTML while the expressions in JavaScript can not.

AngularJS expressions allow filters but JavaScript expressions do not allow filters.

AngularJS expressions do not accept conditionals, loops and exceptions, though JavaScript expressions do.

AngularJS Directives

AngularJS helps you to expand HTML by adding new attributes. Such qualities are called directives.

In AngularJS there is a collection of built-in directive that provides features for your applications. Often, you can create your own directives.

Directives are special attributes starting with ng- prefix. The most common directives follow suit:

O ng-app: The AngularJS Code begins this directive.

O ng-init: This directive initializes data concerning the query.

O ng-model: This directive specifies the model to use in AngularJS as the variable.

O ng-repeat: For each object in a list, this directive repeats html items.

ng-app directive

The Root Element determines by ng-app guideline.

It starts an AngularJS application and initializes or bootstraps the application automatically when loading web page containing AngularJS Application.

It is also used in AngularJS modules to load different modules of AngularJS.

Example

In the following example, we have defined an AngularJS application by default using a div element attribute ng-app.

```
<div ng-app = "">  
  ...  
</div>
```

ng-init directive

ng-init directive initializes data concerning the AngularJS program. It specifies the initial values for an application under AngularJS.

We'll initialize an array of countries in the following example. To describe array of countries, we use JSON syntax.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'},  
{locale:'en-PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">  
...  
</div>
```

ng-model directive:

The ng-model directive describes the model / variable that is to be used in AngularJS.

In the following example, a model named "name" was specified

```
<div ng-app = "">  
...  
  <p>Enter your Name: <input type = "text" ng-model = "name"></p>  
</div>
```

ng-repeat directive

Directive ng-repeat repeats html elements for each object in a collection; We have iterated over an array of countries in the following example.

```
<div ng-app = "">  
...  
  <p>List of Countries with locale:</p>  
  
  <ol>  
    <li ng-repeat = "country in countries">  
      {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}  
    </li>  
  </ol>
```

AngularJS directives Example

Let us take an example in using all the directives mentioned above:

```
<!DOCTYPE html>
```

```

<html>
<head>
  <title>AngularJS Directives</title>
</head>
<body>
  <h1>Sample Application</h1>

  <div ng-app = "" ng-init = "countries = [{locale:'en-
IND',name:'India'}, {locale:'en-PAK',name:'Pakistan'}, {locale:'en-
AUS',name:'Australia'}]">
    <p>Enter your Name: <input type = "text" ng-model = "name">
  </p>
    <p>Hello <span ng-bind = "name"></span>!</p>
    <p>List of Countries with locale:</p>

    <ol>
      <li ng-repeat = "country in countries">
        {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
      </li>
    </ol>
  </div>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>
</body>
</html>

```

To add directives

See this example:

```

<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>

<div ng-app="myApp" w3-test-directive></div>

```

```
<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "This is a directive constructor. "
    };
});
</script>
</body>
</html>
```


AngularJS Controllers

AngularJS controllers are used for managing the AngularJS application data flow.

Using ng-controller directive to describe a controller.

A controller is an object of JavaScript which contains attributes / properties and functions. Every controller accepts \$scope as a parameter that refers to the application / module to be managed by the controller.

AngularJS Controller Example

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{ firstName + " " + lastName }}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = "Aryan";
    $scope.lastName = "Khanna";
});
</script>

</body>
</html>
```

Note

- Here the application AngularJS runs inside the < div > specified by ng-app="myApp".
- The AngularJS directive is an attribute called ng-controller="myCtrl".
- The feature MyCtrl is JavaScript.
- AngularJS invokes a controller with the object \$scope.
- In AngularJS, \$scope is the object of operation (the owner of the variables and functions).
- The controller generates two (variables) properties within the domain (firstName and lastName).
- The ng-model directives bind controller properties (firstName and lastName) to the input fields.

AngularJS controller example

With methods

(variables as functions)

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>
</script>
```

```
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "Aryan";
    $scope.lastName = "Khanna";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    };
});
</script>

</body>
</html>
```

AngularJS Controller in external files

For broader implementations, the controllers are usually housed for external archives.

To store controller build an external file called "personController.js."

"PersonController.js" means

```
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Robert",
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    }
});
```

See this example:

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
First Name: <input type="text" ng-model="firstName"><br>
```

```
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</div>
<script src="personController.js"></script>
</body>
</html>
```

Angular 8 Tutorial

Angular Group released their new update, known as Angular 8.

If you're familiar with Angular's previous edition, it won't be hard for you. You can update your Angular CLI to version 8 with ease.

What's Angular 8?

Angular 8 is a TypeScript-based client-side architecture that is used to build dynamic Web applications.

Except for having some comprehensive features it is very similar to its previous models.

Note: Dynamic web applications are essentially dynamic websites, i.e. www.gmail.com, www.yahoo.com, etc. with a propensity to alter data / information by 3 parameters:

Time to time (e.g. updating web applications for news updates)

Location-to-localisation (e.g. web apps for weather reports)

User-to - user (e.g., applications of the Gmail, Facebook)

New Features - Angular 8

Angular 8 supports TypeScript 3.4

It supports Web Workers

Ivy Rendering Engine is the latest compiler to Angular 8

Angular 8 provides the lazy-loaded modules with complex imports.

Improve ngUpgrade

TypeScript 3.4

Angular 8 supports TypeScript 3.4, and your Angular 8 project is required to run. So update your version of TypeScript to 3.4.

Web workers class

JavaScript is a single thread so asynchronous occurrence is normal for more sensitive tasks such as data calls. Web Workers facilitates the execution of intensive CPU computations in the background thread, freeing up the main thread to update the user interface.

If your application is unresponsive while processing data, web workers can also be helpful.

If you want to outsource such a calculation to a context, we will first use the Angular CLI to build the Web worker.

Ivy and Bazel

The new rendering engine is Ivy, and the latest development system is Bazel.

In Angular 8 both are set for proper use.

The preview of these two is going to be available early. Ivy is the latest Angular compiler / runtime and Angular 8 is the first update to formally deliver an opt-in move to Ivy.

In Angular version 9 Ivy is expected to be a default rendering engine.

Bazel offers one of Angular 8's newest features as a possibility to develop your CLI application faster.

Prerequisite for Angular 8

You need to have Node.js installed version > 10.

NPM will also be updated, since it is used by default.

You have to have MongoDB built on your machine.

Workflow of Angular 8

We'll be building two different projects here:

Another for the front end (in Angular) and another for the backend (in Node.js Express MongoDB). We'll also build a backend API that's used by frontend.

Following technologies used generally

Node: 12.4.0

Angular CLI: 8.0.2

NPM: v12.4.0

MongoDB shell version v4.0.10

MongoDB version v4.0.10

Windows 10

To check “**Node and Angular CLI version**” , use **ng --version** command.

To check “**npm version**” , use **node -v** command.

To check “**MongoDB version**” , use **mongod --version** command.

To check “**MongoDB shell version**” , use **mongo --version** command.

Create an Angular 8 project

Angular 8 project by using the below command:

ng new angular8project

“**angular8project**” is the name of the project here.

Node.js command prompt

```
C:\Users\J...>ng new angular8project
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE angular8project/angular.json (3497 bytes)
CREATE angular8project/package.json (1288 bytes)
CREATE angular8project/README.md (1032 bytes)
CREATE angular8project/tsconfig.json (438 bytes)
CREATE angular8project/tslint.json (1985 bytes)
CREATE angular8project/.editorconfig (246 bytes)
CREATE angular8project/.gitignore (629 bytes)
CREATE angular8project/browserslist (429 bytes)
CREATE angular8project/karma.conf.js (1027 bytes)
CREATE angular8project/tsconfig.app.json (210 bytes)
CREATE angular8project/tsconfig.spec.json (270 bytes)
CREATE angular8project/src/favicon.ico (5430 bytes)
CREATE angular8project/src/index.html (302 bytes)
CREATE angular8project/src/main.ts (372 bytes)
CREATE angular8project/src/polyfills.ts (2838 bytes)
CREATE angular8project/src/styles.css (80 bytes)
CREATE angular8project/src/test.ts (642 bytes)
CREATE angular8project/src/assets/.gitkeep (0 bytes)
CREATE angular8project/src/environments/environment.prod.ts (51 bytes)
CREATE angular8project/src/environments/environment.ts (662 bytes)
CREATE angular8project/src/app/app-routing.module.ts (245 bytes)
CREATE angular8project/src/app/app.module.ts (393 bytes)
CREATE angular8project/src/app/app.component.html (1152 bytes)
CREATE angular8project/src/app/app.component.spec.ts (1122 bytes)
CREATE angular8project/src/app/app.component.ts (219 bytes)
CREATE angular8project/src/app/app.component.css (0 bytes)
```

Node.js command prompt

```
CREATE angular8project/e2e/protractor.conf.js (810 bytes)
CREATE angular8project/e2e/tsconfig.json (214 bytes)
CREATE angular8project/e2e/src/app.e2e-spec.ts (644 bytes)
CREATE angular8project/e2e/src/app.po.ts (251 bytes)

> core-js@2.6.9 postinstall C:\Users\JavaTpoint\angular8project\node_modules\babel-runtime\node_modules\core-js
> node scripts/postinstall || echo "ignore"

> core-js@2.6.9 postinstall C:\Users\JavaTpoint\angular8project\node_modules\karma\node_modules\core-js
> node scripts/postinstall || echo "ignore"

> @angular/cli@8.0.3 postinstall C:\Users\JavaTpoint\angular8project\node_modules\@angular\cli
> node ./bin/postinstall/script.js

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 1011 packages from 1041 contributors and audited 19005 packages in 210.005s
found 0 vulnerabilities

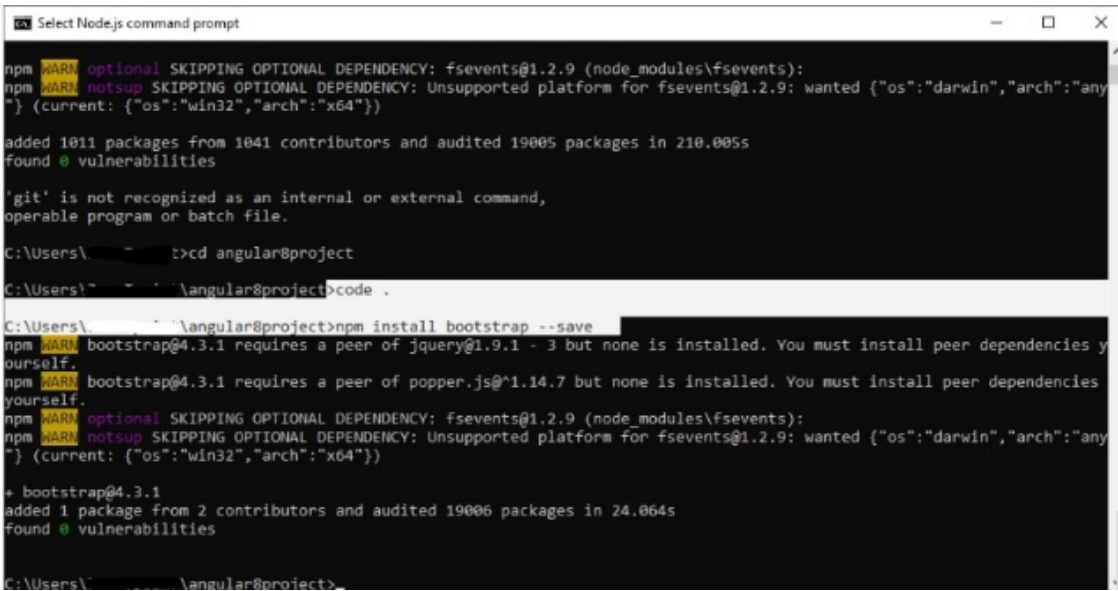
'git' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\JavaTpoint>
```

Install Bootstrap 4 CSS framework

Following command to install “bootstrap” in ur project.

`npm install bootstrap --save`



```
Select Node.js command prompt

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 1011 packages from 1041 contributors and audited 19005 packages in 210.005s
found 0 vulnerabilities

'git' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\...>cd angular8project
C:\Users\... \angular8project>code .

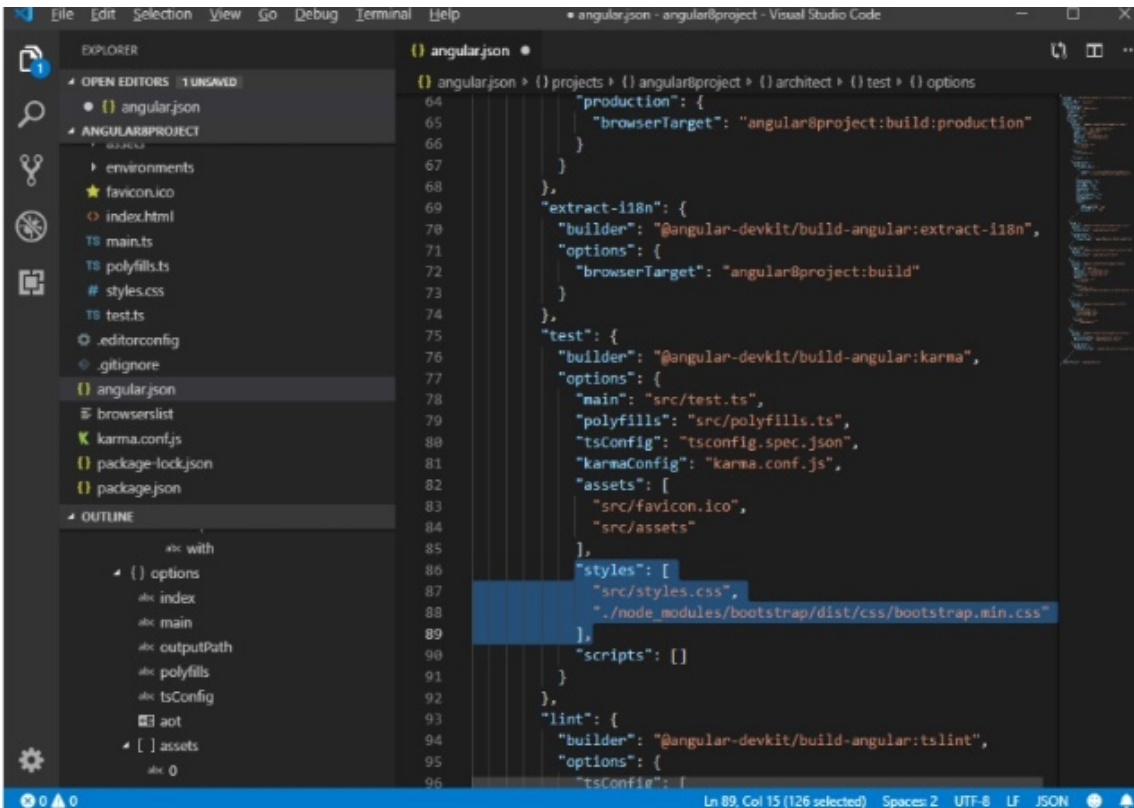
C:\Users\... \angular8project>npm install bootstrap --save
npm WARN bootstrap@4.3.1 requires a peer of jquery@1.9.1 - 3 but none is installed. You must install peer dependencies yourself.
npm WARN bootstrap@4.3.1 requires a peer of popper.js@^1.14.7 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ bootstrap@4.3.1
added 1 package from 2 contributors and audited 19006 packages in 24.064s
found 0 vulnerabilities

C:\Users\... \angular8project>
```

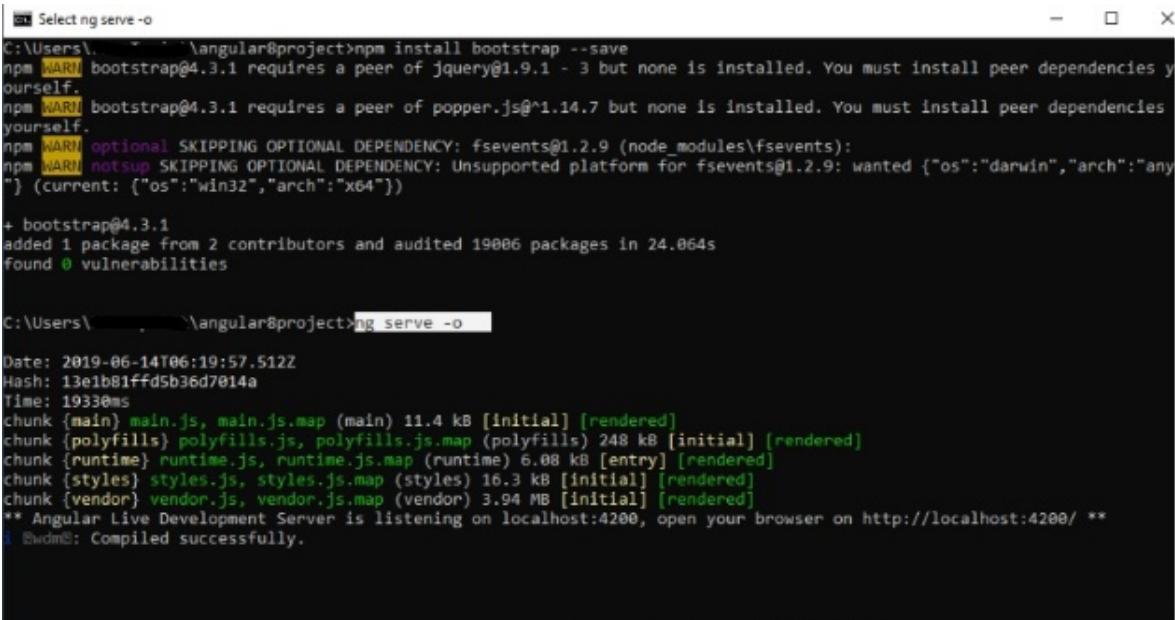
Now, apply the following code inside the angular.json file.

1. `"styles" : [`
2. `"src/styles.css" ,`
3. `"./node_modules/bootstrap/dist/css/bootstrap.min.css"`
4. `],`



Start the Angular development server by apply the following command.

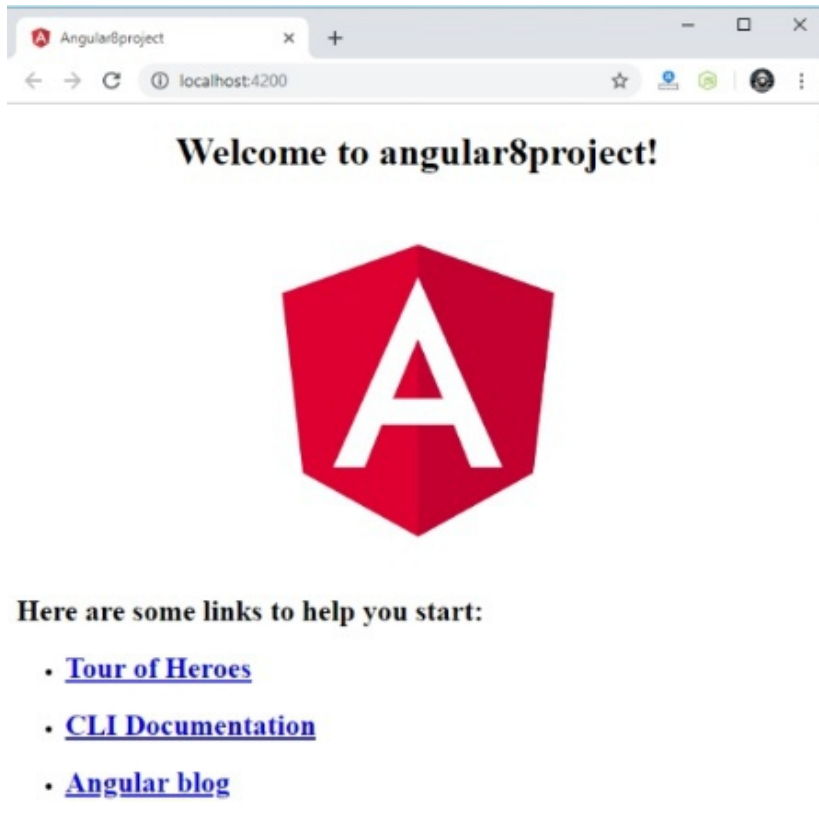
`ng serve -o`



The server starts in the following host

<http://localhost:4200/>

U will see the output. That is the initial Angular home screen.



Generate Angular Components

Using the following command to create 3 Angular Components:

```
ng g c product-add --skipTests= true  
ng g c product-get --skipTests= true  
ng g c product-edit --skipTests= true
```

```
Node.js command prompt

C:\Users\... \angular8project>ng g c product-add --skipTests=true
CREATE src/app/product-add/product-add.component.html (30 bytes)
CREATE src/app/product-add/product-add.component.ts (288 bytes)
CREATE src/app/product-add/product-add.component.css (0 bytes)
UPDATE src/app/app.module.ts (493 bytes)

C:\Users\... \angular8project>ng g c product-get --skipTests=true
CREATE src/app/product-get/product-get.component.html (30 bytes)
CREATE src/app/product-get/product-get.component.ts (288 bytes)
CREATE src/app/product-get/product-get.component.css (0 bytes)
UPDATE src/app/app.module.ts (593 bytes)

C:\Users\... \angular8project>ng g c product-edit --skipTests=true
CREATE src/app/product-edit/product-edit.component.html (31 bytes)
CREATE src/app/product-edit/product-edit.component.ts (292 bytes)
CREATE src/app/product-edit/product-edit.component.css (0 bytes)
UPDATE src/app/app.module.ts (697 bytes)

C:\Users\... \angular8project>
```

All the above mentioned three components are automatically added to **app.module.ts** file.

Now, we are going to configure the routing of angular components within an **app-routing.module.ts** file.

Now, write the below code inside an **app-routing.module.ts** file:

```
// app-routing.module.ts
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { ProductAddComponent } from './product-add/product-add.component';
import { ProductEditComponent } from './product-edit/product-edit.component';
import { ProductGetComponent } from './product-get/product-get.component';
const routes: Routes = [
  {
    path: 'product/create',
    component: ProductAddComponent
  },
  {
```

```

    path: 'edit/:id' ,
    component: ProductEditComponent
  },
  {
    path: 'products' ,
    component: ProductGetComponent
  }
];
@NgModule ({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Create Angular Navigation

Write the below code within the **app.component.html** file.

```

<nav class = "navbar navbar-expand-sm bg-light " >
  <div class = "container-fluid " >
    <ul class = "navbar-nav " >
      <li class = "nav-item " >
        <a routerLink= "product/create " class = "nav-link " routerLinkActive=
"active " >
          Create Product
        </a>
      </li>
      <li class = "nav-item " >
        <a routerLink= "products " class = "nav-link " routerLinkActive= "active
" >
          Products
        </a>
      </li>
    </ul>
  </div>
</nav>
<div class = "container " >
  <router-outlet></router-outlet>

```

</div>

Angular 8 Introduction

Angular is the most popular JavaScript framework and platform for developing mobile and desktop Web apps or single-page (SPAs) client-side (front-end) applications.

Angular community released their latest version, called Angular 8. If you're familiar with Angular's previous edition, it won't be hard for you. You can update your older version of Angular to the new Angular 8 edition with ease.

Angular 8 is written in TypeScript, and JavaScript is complied with. They use Angular 8 to build interactive web applications. Except for having some comprehensive features it is very similar to its previous models.

Features and Advantages of Angular 8

The Angular group released its new update, Angular 8, with an impressive list of updates and improvements including the much anticipated Ivy compiler as an opt-in option.

Most famous Angular 8 features

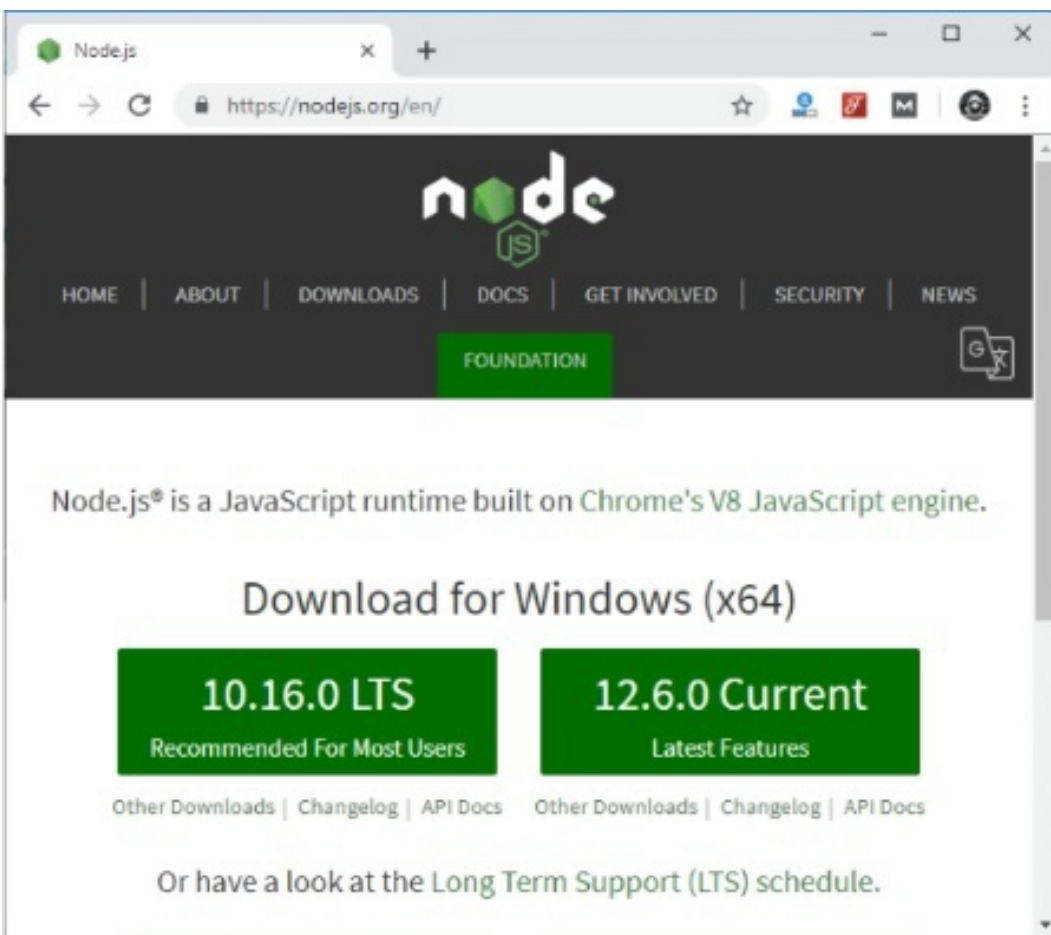
- Support TypeScript 3.4
- Supports Web Workers
- Preview of Ivy available
- Lazy loading
- Improvement of ngUpgrade

Angular 8 Installation

You must have Node.js installed on your machine and set up a development framework and npm package manager before setting up configuration for Angular development using the Angular CLI tool.

Install Node.js

Angular requires a version 10.9.0 or later of Node.js. It is available for download from <https://nodejs.org/en/>



You'll need to install it on your device after downloading.

Once you have Node.js installed on your system, open command prompt node.js.


```
Node.js command prompt
Your environment has been set up for using Node.js 10.16.0 (x64) and npm.
C:\Users\...>
```

In a terminal / console window, run **node -v** to check your version.

```
Node.js command prompt
Your environment has been set up for using Node.js 10.16.0 (x64) and npm.
C:\Users\JavaTpoint>node -v
v10.16.0
C:\Users\JavaTpoint>
```

Use npm to Install Angular CLI

Enter the below command to install Angular CLI

`npm install -g @angular/cli`

Or

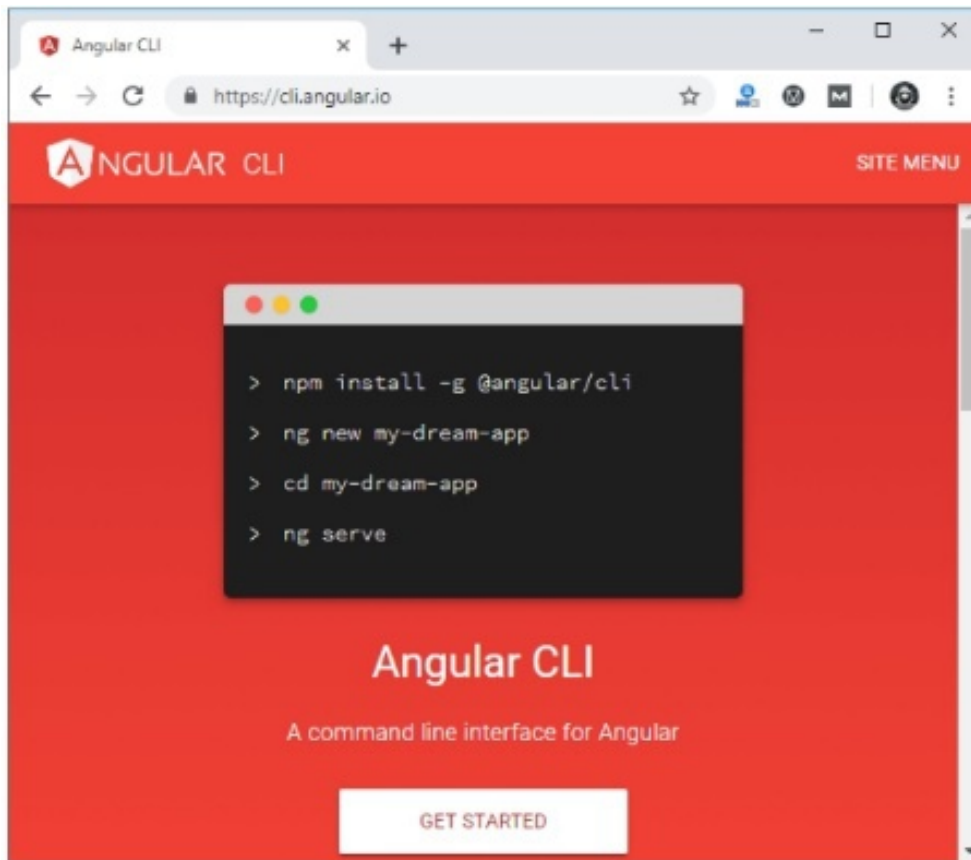
`npm install -g @angular/cli@latest`

```
Node.js command prompt
Your environment has been set up for using Node.js 10.16.0 (x64) and npm.
C:\Users\JavaTpoint>node -v
v10.16.0
C:\Users\JavaTpoint>npm install -g @angular/cli@latest
C:\Users\JavaTpoint\AppData\Roaming\npm\ng -> C:\Users\JavaTpoint\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng
> @angular/cli@8.1.0 postinstall C:\Users\JavaTpoint\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js
+ @angular/cli@8.1.0
added 16 packages from 14 contributors, removed 5 packages and updated 26 packages in 40.223s
C:\Users\JavaTpoint>
```

Or

From Angular CLI official website <https://cli.angular.io/>

To create an Angular App, you'll see the entire cli guide. The first command to install Angular CLI



Node and Angular CLI version check,

use **ng --version** command.

```
Node.js command prompt
C:\Users\J...nt>ng --version

Angular CLI

Angular CLI: 8.1.0
Node: 10.16.0
OS: win32 x64
Angular:
...
Package      Version
-----
@angular-devkit/architect 0.801.0
@angular-devkit/core      8.1.0
@angular-devkit/schematics 8.1.0
@schematics/angular       8.1.0
@schematics/update        0.801.0
rxjs                    6.4.0

C:\Users\J...nt>
```

Now your machine has Angular 8 .

Angular 8 First App

We will going to see how to create an Angular 8 application.

To create an app

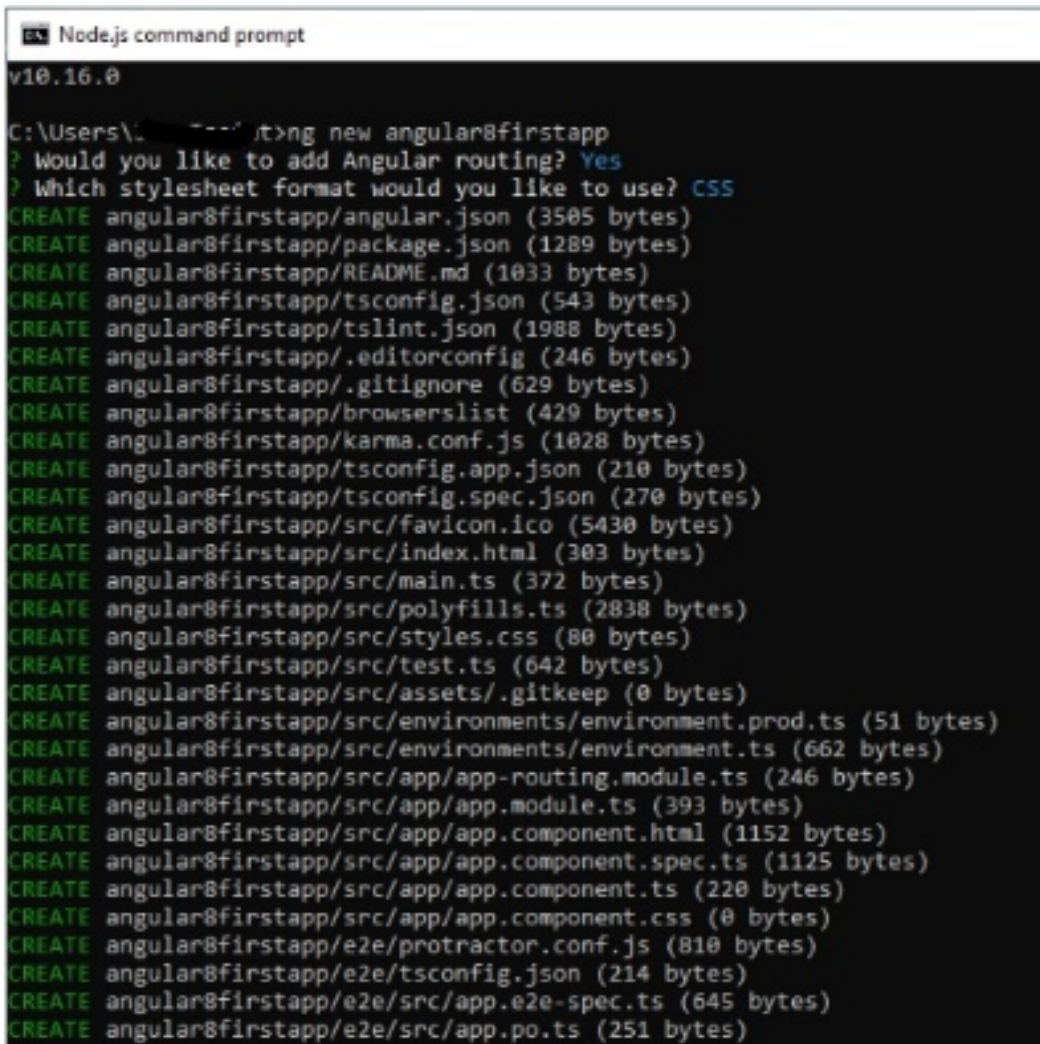
Syntax:

`ng new app_name`

For example:

We r going to create an app named "angular8firstapp"

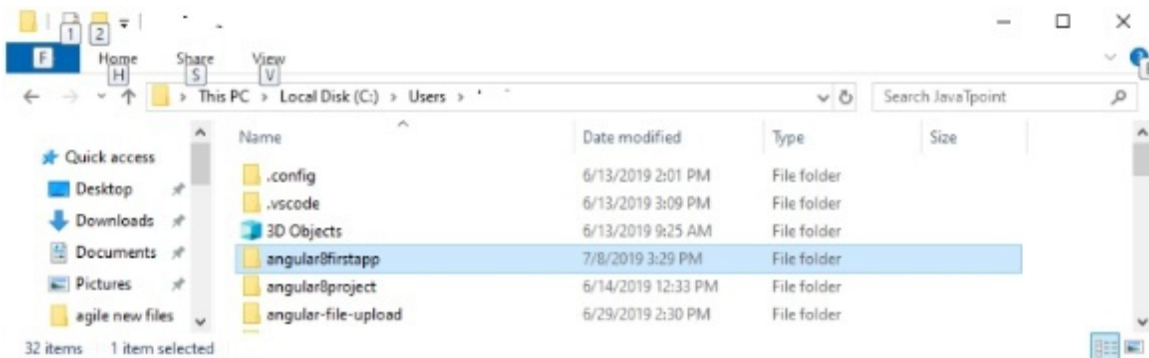
`ng new angular8firstapp`

A screenshot of a Node.js command prompt window titled "Node.js command prompt". The window shows the command "ng new angular8firstapp" being executed. The output lists the files created for the new Angular application, including configuration files, source files, and test files, along with their sizes in bytes. The files listed are: angular.json (3505 bytes), package.json (1289 bytes), README.md (1033 bytes), tsconfig.json (543 bytes), tslint.json (1988 bytes), .editorconfig (246 bytes), .gitignore (629 bytes), browserslist (429 bytes), karma.conf.js (1028 bytes), tsconfig.app.json (210 bytes), tsconfig.spec.json (270 bytes), src/favicon.ico (5430 bytes), src/index.html (303 bytes), src/main.ts (372 bytes), src/polyfills.ts (2838 bytes), src/styles.css (80 bytes), src/test.ts (642 bytes), src/assets/.gitkeep (0 bytes), src/environments/environment.prod.ts (51 bytes), src/environments/environment.ts (662 bytes), src/app/app-routing.module.ts (246 bytes), src/app/app.module.ts (393 bytes), src/app/app.component.html (1152 bytes), src/app/app.component.spec.ts (1125 bytes), src/app/app.component.ts (220 bytes), src/app/app.component.css (0 bytes), e2e/protractor.conf.js (810 bytes), e2e/tsconfig.json (214 bytes), e2e/src/app.e2e-spec.ts (645 bytes), and e2e/src/app.po.ts (251 bytes).

```
Node.js command prompt
v10.16.0

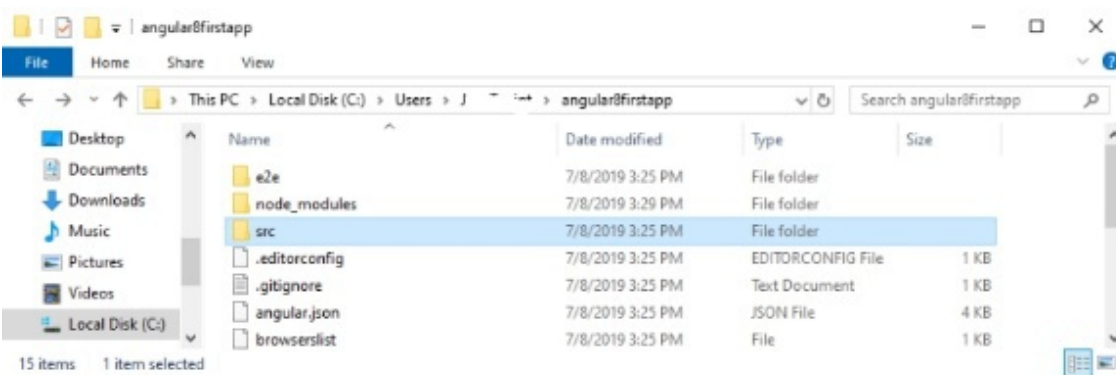
C:\Users\...>ng new angular8firstapp
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE angular8firstapp/angular.json (3505 bytes)
CREATE angular8firstapp/package.json (1289 bytes)
CREATE angular8firstapp/README.md (1033 bytes)
CREATE angular8firstapp/tsconfig.json (543 bytes)
CREATE angular8firstapp/tslint.json (1988 bytes)
CREATE angular8firstapp/.editorconfig (246 bytes)
CREATE angular8firstapp/.gitignore (629 bytes)
CREATE angular8firstapp/browserslist (429 bytes)
CREATE angular8firstapp/karma.conf.js (1028 bytes)
CREATE angular8firstapp/tsconfig.app.json (210 bytes)
CREATE angular8firstapp/tsconfig.spec.json (270 bytes)
CREATE angular8firstapp/src/favicon.ico (5430 bytes)
CREATE angular8firstapp/src/index.html (303 bytes)
CREATE angular8firstapp/src/main.ts (372 bytes)
CREATE angular8firstapp/src/polyfills.ts (2838 bytes)
CREATE angular8firstapp/src/styles.css (80 bytes)
CREATE angular8firstapp/src/test.ts (642 bytes)
CREATE angular8firstapp/src/assets/.gitkeep (0 bytes)
CREATE angular8firstapp/src/environments/environment.prod.ts (51 bytes)
CREATE angular8firstapp/src/environments/environment.ts (662 bytes)
CREATE angular8firstapp/src/app/app-routing.module.ts (246 bytes)
CREATE angular8firstapp/src/app/app.module.ts (393 bytes)
CREATE angular8firstapp/src/app/app.component.html (1152 bytes)
CREATE angular8firstapp/src/app/app.component.spec.ts (1125 bytes)
CREATE angular8firstapp/src/app/app.component.ts (220 bytes)
CREATE angular8firstapp/src/app/app.component.css (0 bytes)
CREATE angular8firstapp/e2e/protractor.conf.js (810 bytes)
CREATE angular8firstapp/e2e/tsconfig.json (214 bytes)
CREATE angular8firstapp/e2e/src/app.e2e-spec.ts (645 bytes)
CREATE angular8firstapp/e2e/src/app.po.ts (251 bytes)
```

That a folder is created. This is ur first created app of Angular 8.

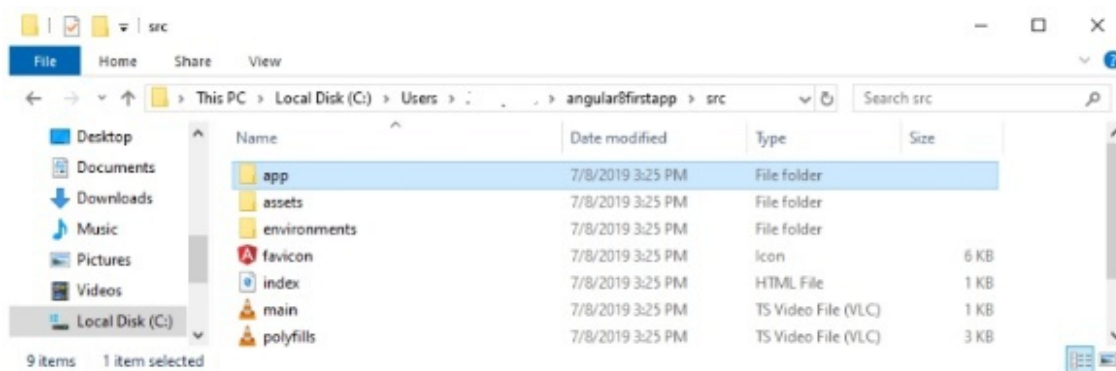


Open this folder.

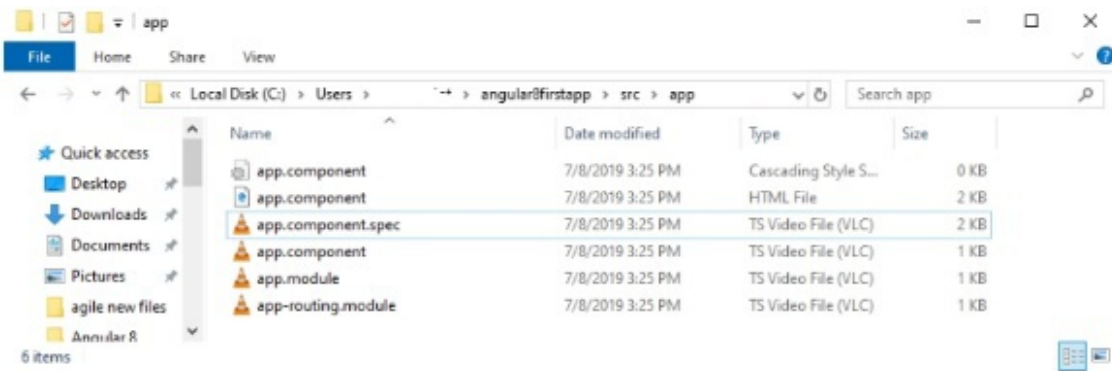
U will see the some subfolders.



Here, the project 's principal folder is src. Open the src , and see other subfolders.



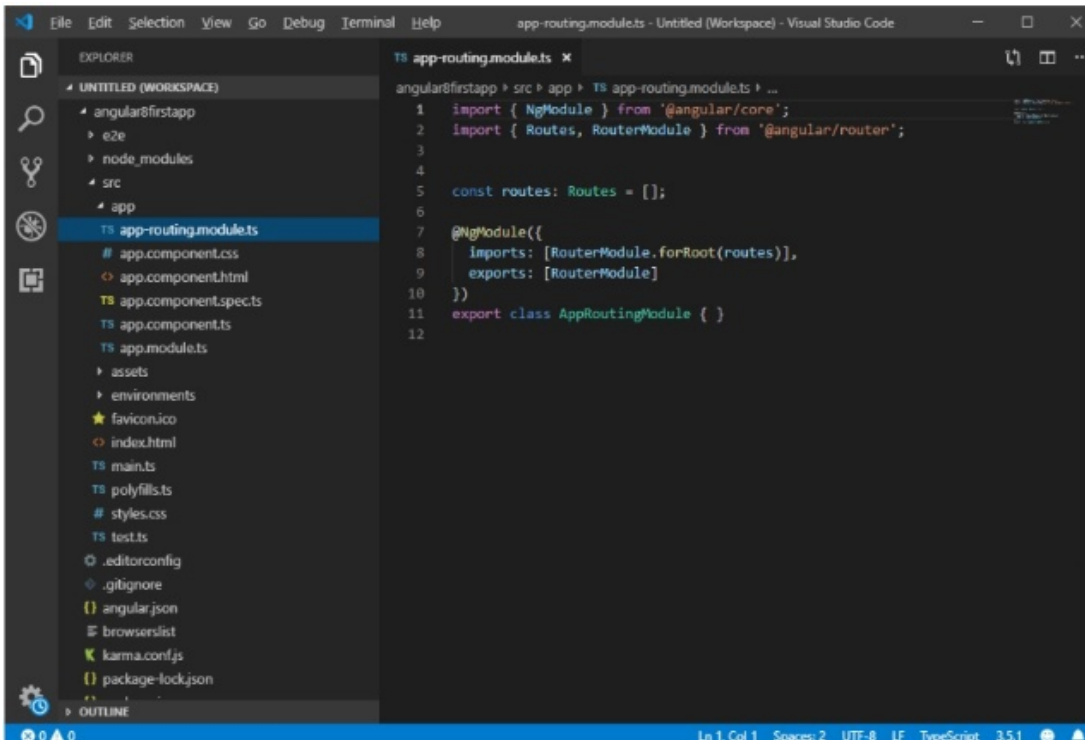
The root of your Angular 8 app is this App folder. Open this folder and you'll see some files like .ts, html , and css.



Install Visual Studio Code IDE or JetBrains WebStorm

VS Code is lightweight and simple to set up, with a wide variety of built-in application editing, formatting, and refactoring capabilities. Utilization is safe. This also offers a large range of extensions which will increase the productivity considerably.

JetBrains WebStorm is also a great IDE for developing Angular applications. It's simple, attractive and software very easy to use, but it's not free to use.



Run your app

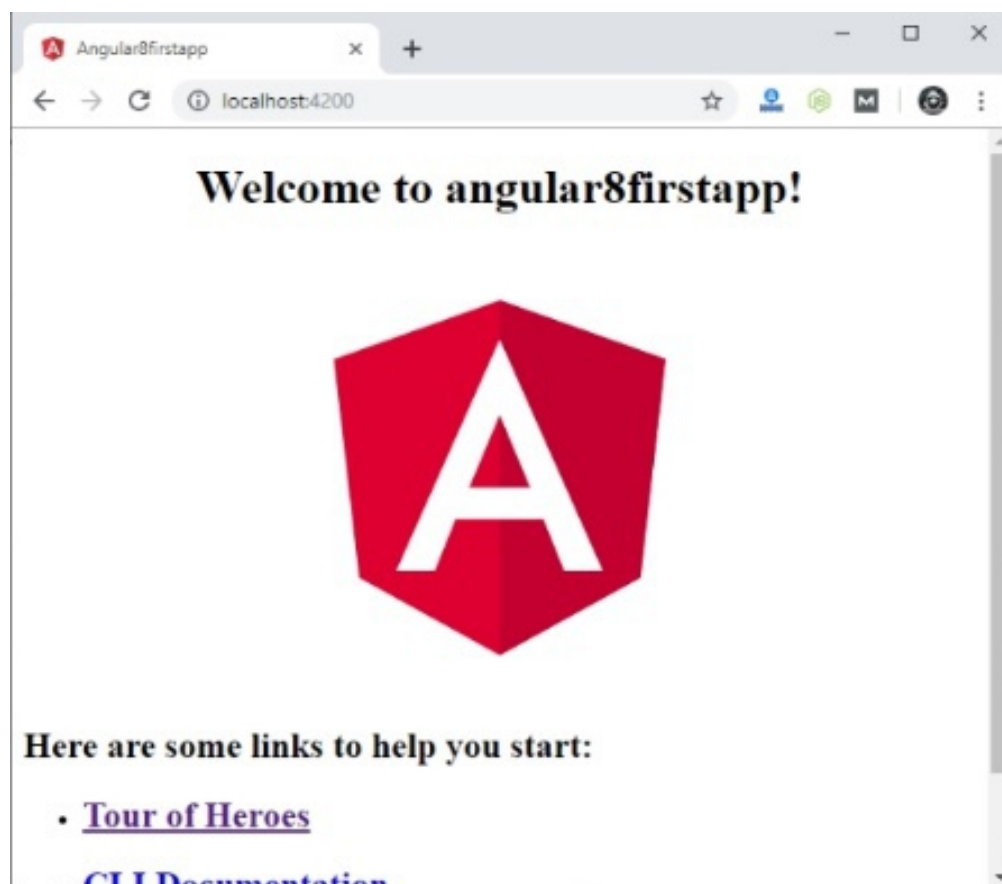
Open your node.js command prompt and use the cd command to go into your project and then run the ng serve command to compile and run your program.

ng serve


```
Select ng serve
C:\Users\...>cd angular8firstapp
C:\Users\...>ng serve
10% building 3/3 modules 0 active [wds]: Project is running at http://localhost:4200/w
ebpack-dev-server/
[wds]: webpack output is served from /
[wds]: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 11.4 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 251 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.09 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.3 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.07 MB [initial] [rendered]
Date: 2019-07-08T11:07:18.954Z - Hash: 0cccfd6c4ab828db762c - Time: 14038ms
** Angular Live Development Server is listening on localhost:4200, open your browser on
http://localhost:4200/ **
[wds]: Compiled successfully.
```

In your browser go to: <http://localhost:4200/>



Now, U can see your app is now running.

How an Angular's app get loaded and started

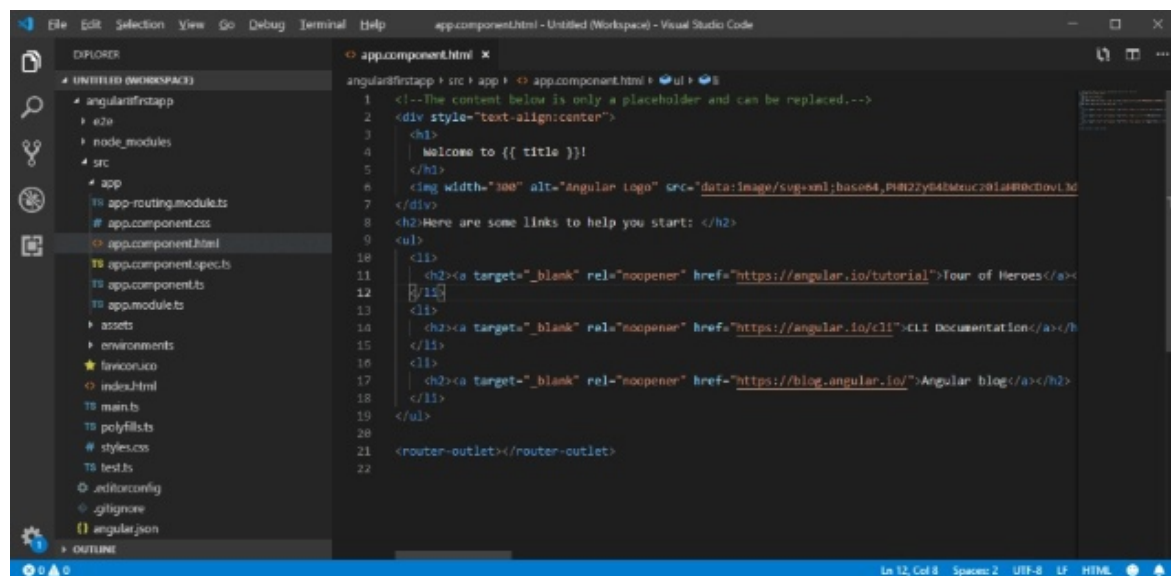
It is a simple Angular app generated through the use of ng new app name instruction, and nothing in the app is edited. The App is called angular8firstapp.

Now, we'll learn how to load and launch the Angular 's app.

Let's delete all the code from the file app.component.html, and write some basic HTML code. For instance:

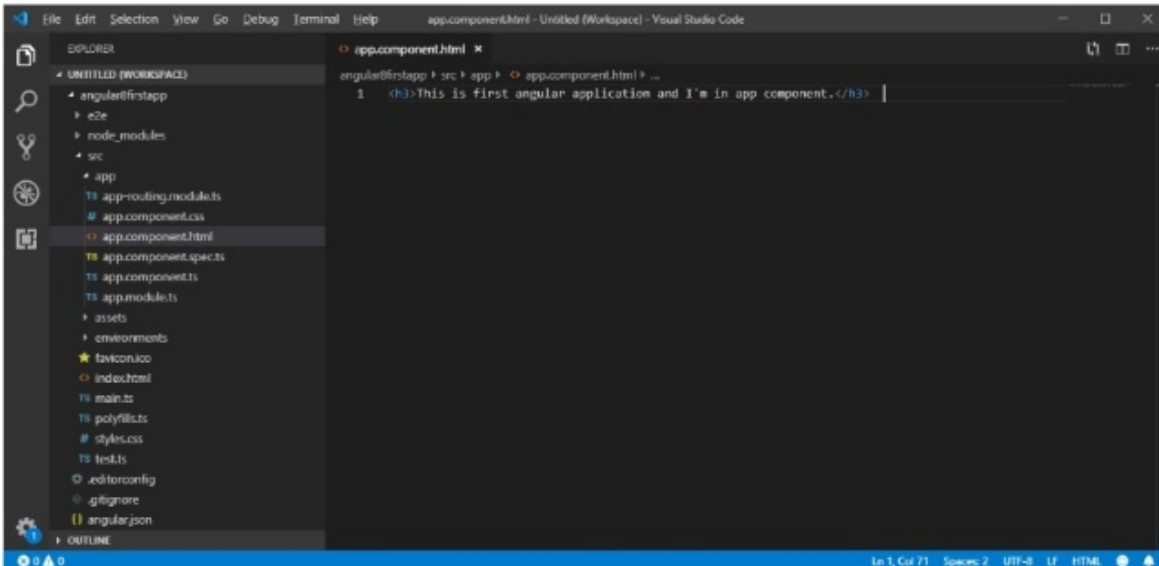
<h3> This is first angular application and I'm in app component. </h3>

This is the original code in the app.component.html file

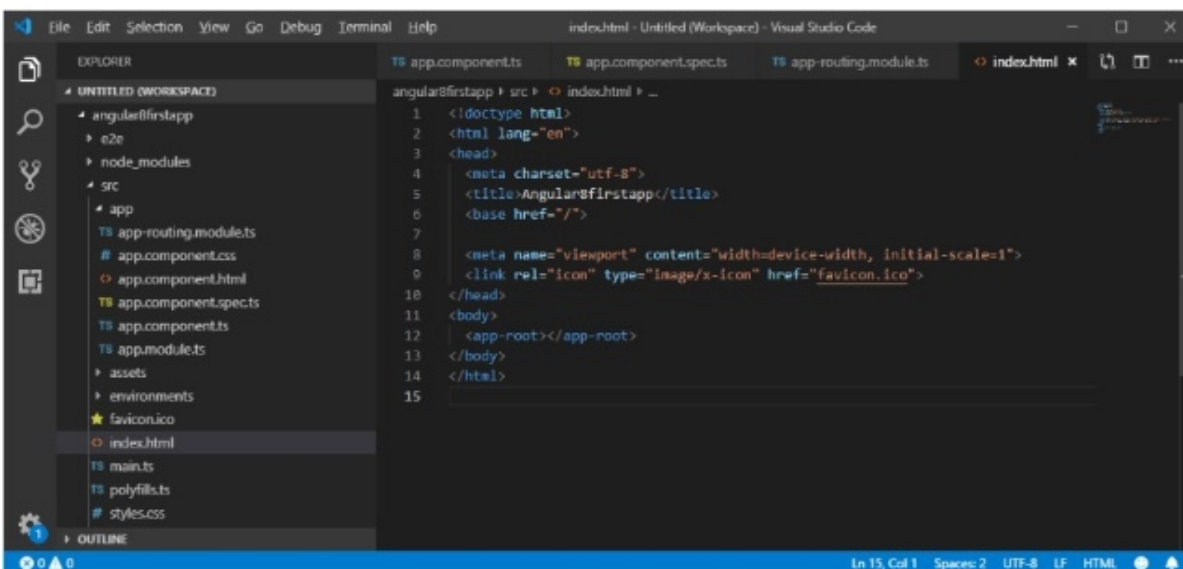
A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure for 'angular8firstapp' with folders like 'src' and 'app'. The 'app.component.html' file is selected and open in the main editor. The code in the editor is as follows:

```
1 <!--The content below is only a placeholder and can be replaced.-->
2 <div style="text-align:center">
3   <h1>
4     Welcome to {{ title }}!
5   </h1>
6   Tour of Heroes</a>
12   </li>
13   <li>
14     <a target="_blank" rel="noopener" href="https://angular.io/cli">CLI documentation</a>
15   </li>
16   <li>
17     <a target="_blank" rel="noopener" href="https://blog.angular.io/>Angular blog</a>
18   </li>
19 </ul>
20
21 <router-outlet></router-outlet>
22
```

Now, it's replaced and looked this way:



Here the server doesn't support the above script. The server had supported a file with index.html.



Angular is a platform that allows us to create "Single Page Applications," and here index.html is the one page the server produced.

Index.html:

```
<!doctype html>
<html lang= "en" >
<head>
  <meta charset= "utf-8" >
  <title>Angular8firstapp</title>
  <base href= "/" >
```

```
<meta name= "viewport" content= "width=device-width, initial-scale=1"
>
<link rel= "icon" type= "image/x-icon" href= "favicon.ico" >
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

The code above looks like a standard HTML file, except here the < title > tag indicates the same title as the title of the app in the browser. But the code for < body > varies from standard HTML code. Here, you can see the tag "< app-root >" given by the CLI. We may assume that, by default, one component is generated whenever we create a project from CLI, that is, "app component."

Now, see a file named "app.component.ts." It is a file called TypeScript. You can see the selector property here.

```
import { Component } from '@angular/core' ;
@Component ({
  selector: 'app-root' ,
  templateUrl: './app.component.html' ,
  styleUrls: [ './app.component.css' ]
})
export class AppComponent {
  title = 'angular8firstapp' ;
}
```

The selector property can be seen to contain the string as index.html script.

This information allows the Angular to put this part with the component template into an index.html file.

The component template is "./app.component.html," so Angular uses this section of the index.html file body.

Now, you see how index.html file contains a "app-root." Now let's see "How does Angular trigger?"

Once ng-serve builds the program, it generates "bundles" and adds these to index.html file automatically at runtime.

Thus, the first code must be executed from the "main.ts" file from such packages, i.e. the "main.ts" file is the first file from which the program is started.

Main.ts file:

```
import { enableProdMode } from '@angular/core' ;
import { platformBrowserDynamic } from '@angular/platform-browser-
dynamic' ;
import { AppModule } from './app/app.module' ;
import { environment } from './environments/environment' ;
if (environment.production) {
    enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule)
    . catch (err => console.error(err));
```

Here the Angular application begins by bootstrap process.

It refers to AppModule which looks into the folders of the application.

In the file "app.module" you can see that the index.html file is evaluated by a bootstrap array which is simply a list of all the components.

See file with app.module.ts:

```
import { BrowserModule } from '@angular/platform-browser' ;
import { NgModule } from '@angular/core' ;
import { AppRoutingModule } from './app-routing.module' ;
import { AppComponent } from './app.component' ;
@NgModule ({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule,
        AppRoutingModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
```

```
export class AppModule { }
```

Now, you can see that it loads the Angular program as:

```
main.ts    > >  app.Module.ts    > >  app.component.ts    > >  index.html  
          > >  app.component.html
```

Angular 8 Architecture

Angular 8 is a platform and framework which is used in HTML and TypeScript to construct client applications.

Angular 8 is written in TypeScript. It provides both core and optional functionality as a collection of TypeScript libraries you can import into your applications.

An Angular application's basic building blocks are NgModules which provide a compiling background for components.

NgModules compile the related code into functional sets; a collection of NgModules determines an Angular app.

An application often has at least one root module that enables bootstrapping, and usually has a lot more feature modules.

- Components describe views, which are collections of screen elements that Angular can choose from and modify according to the logic and data of your program.
- Components use services which provide features that are not directly related to views. Service providers can be embedded as dependencies into the components, making the code modular, reusable and efficient.

Angular 8 Components

Within Angular 8, all components and services are merely classes with decorators labeling their types and providing metadata that direct Angular to do things.

Every Angular application always has at least one root component known to link a page hierarchy to page DOM.

Each component defines a class that contains data and logic for the application, and is associated with an HTML template that defines a view to be displayed in a target environment.

Modules

Angular 8 NgModules vary from those of other JavaScript modules. Growing Angular 8 app comes with a root module known as the AppModule. It provides the mechanism of bootstrap which launches the application.

In general, there are several functional modules available in any Angular 8 app.

Some essential features of Modules Angular 8:

Like other JavaScript modules, Angular 8 NgModules import the functionalities of other NgModules.

NgModules allow the exportation and use of their own features via other NgModules. For example, you can import the Router NgModule if you want to use the Router service in your app.

Template, Directives and Data Binding

A template is used in Angular 8 to combine HTML with Angular Markup, and to modify HTML elements before display. Template directives provide logic for the program, and binding markup links data from the application and DOM.

There are two forms of binding to data:

1. **Event Binding:** Event binding is used to bind events to your app and by modifying the application data to respond to user feedback in the target environment.
2. **Property Binding:** Property binding is used to transfer data into the HTML from component class and allows you to interpolate values that are derived from the data of your application.

Services and Dependency Injection

In Angular 8, developers build a data or logic service class that isn't associated with a particular view and want to share across components.

Dependency Injection (DI) is used to make the classes of your components lean and effective. DI does not fetch server data, validate user input, or log directly into the console; it simply makes services to these tasks.

Routing3

Within Angular 8, Router is a NgModule that provides a service that enables developers to establish a navigation path between the various application states and to display hierarchies within their application.

This operates in the same way as navigation works in a browser. E.g.:

- Enter a URL in the address bar, and navigate the browser to the correct tab.
- On a page, click the link and the user will navigate to a new tab.
- Press back or forward buttons on the screen, and the screen can move back or forward according to the history pages you have seen.

Angular 8 Directives

The DOM is manipulated with the Angular 8 Directives.

You may change the appearance, behavior or layout of a DOM element by using the Angular directives. It helps to extend HTML, too.

Angular 8 directives can be classified in 3 categories based on how they behave:

- Component Directives
- Structural Directives
- Attribute Directives

Component Directives : Directives on components are implemented in the main class. They contain the specifics of how to process, instantiate and use the component at runtime.

Structural Directives : Structural directives begin with a sign of *. These directives are used to **modify and change the DOM entity structure**. For example, *ngIf directive, *ngSwitch directive, and *ngFor directive.

Attribute Directives : Directives on attributes are used to **modify the DOM elements look and actions**. For example: ngClass directive, and ngStyle directive etc.

Angular 8 ngIf Directive

According to the expression, the ngIf directive is used to add or delete HTML elements.

The expression has to return a value in Boolean.

If the expression is false then the element will be removed, the element will be inserted otherwise.

It is akin to AngularJS 'ng-if' directive.

ngIf Syntax

```
<p *ngIf = "condition " >  
  condition is true and ngIf is true .  
</p>  
<p *ngIf = "!condition " >  
  condition is false and ngIf is false .  
</p>
```

The *ngIf directive form with an "else" block

```
<div *ngIf= "condition; else elseBlock " >  
Content to render when condition is true .  
</div>  
<ng-template #elseBlock>  
Content to render when condition is false .  
</ng-template>
```

The DOM element isn't hidden by the ngIf directive. It removes the whole element from the DOM, along with its subtree.

It also removes the corresponding state freeing up the allocated resources to the item.

Most commonly the * ngIf directive is used to conditionally display an inline template. See the example below:

```
@Component ({
```

```

selector: 'ng-if-simple' ,
template: `
  <button (click)= "show = !show " >{{show ? 'hide ' : 'show ' }}</button>
  show = {{show}}
  <br>
  <div *ngIf= "show " >Text to show</div>
  `
,
})
export class NgIfSimple {
  show: boolean = true ;
}

```

Same template example with else block

```

@Component ({
  selector: 'ng-if-else' ,
  template: `
    <button (click)= "show = !show " >{{show ? 'hide ' : 'show ' }}</button>
    show = {{show}}
    <br>
    <div *ngIf= "show; else elseBlock " >Text to show</div>
    <ng-template #elseBlock>Alternate text while primary text is
    hidden</ng-template>
    `
  ,
})
export class NgIfElse {
  show: boolean = true ;
}

```

Angular 8 *ngFor Directive

The * ngFor directive is used to repeat a portion of the HTML template from an iterable list (Collection) once for every object.

The ngFor is a formal angular directive in AngularJS and is identical to ngRepeat. Some local variables, such as Index, First, Last, odd and even * ngFor directive are exported.

Syntax of ngFor

Simplified syntax for the ngFor directive:

```
<li *ngFor = "let item of items; " > .... </li>
```

ngFor Directive

You need to build a block of HTML elements to Use ngFor directive, which can show a single item in the list of objects.

You can use the ngFor directive afterwards to tell angular to repeat the block of HTML elements for each item in the list.

Example for *ngFor Directive

First, you must build an Application with angular. Open the app.component.ts after that, and add the code below.

The following code provides a list of Top 3 movies in a film series. Let's build a template that shows these films in a tabular format.

```
import { Component } from '@angular/core' ;
@Component ({
  selector: 'movie-app' ,
  templateUrl: './app/app.component.html' ,
  styleUrls:[ './app/app.component.css' ]
})
export class AppComponent
{
  title: string = "Top 10 Movies" ;
  movies: Movie[] =[
```

```

        {title: 'Zootopia' ,director: 'Byron Howard, Rich Moore' ,cast: 'Idris
Elba, Ginnifer Goodwin, Jason Bateman' ,releaseDate: 'March 4, 2016' },
        {title: 'Batman v Superman: Dawn of Justice' ,director: 'Zack Snyder'
,cast: 'Ben Affleck, Henry Cavill, Amy Adams' ,releaseDate: 'March 25,
2016' },
        {title: 'Captain America: Civil War' ,director: 'Anthony Russo, Joe
Russo' ,cast: 'Scarlett Johansson, Elizabeth Olsen, Chris Evans'
,releaseDate: 'May 6, 2016' },
        {title: 'X-Men: Apocalypse' ,director: 'Bryan Singer' ,cast: 'Jennifer
Lawrence, Olivia Munn, Oscar Isaac' ,releaseDate: 'May 27, 2016' },
    ]
}
class Movie {
    title : string;
    director : string;
    cast : string;
    releaseDate : string;
}

```

Open the app now. Component.html, and add the code below:

```

<div class ='panel panel-primary' >
    <div class ='panel-heading' >
        {{title}}
    </div>
    <div class ='panel-body' >
        <div class ='table-responsive' >
            <table class ='table' >
                <thead>
                    <tr>
                        <th>Title</th>
                        <th>Director</th>
                        <th>Cast</th>
                        <th>Release Date</th>
                    </tr>
                </thead>
                <tbody>
                    <tr *ngFor="let movie of movies;" >

```

```
        <td>{{movie.title}}</td>
        <td>{{movie.director}}</td>
        <td>{{movie.cast}}</td>
        <td>{{movie.releaseDate}}</td>
    </tr>
</tbody>
</table>
</div>
</div>
</div>
```

It will display the movies in tabular form when you run the program.

Angular 8 ngSwitch Directive

NgSwitch is a structural directive in Angular 8 that is used to add / remove DOM Element.

It is identical to the C # switch statement.

The ngSwitch directive has a switch expression applied to the container element.

Syntax of ngSwitch

```
<container_element [ngSwitch]="switch_expression" >
  <inner_element * ngSwitchCase = "match_expression_1" > .. </inner_element >
  <inner_element * ngSwitchCase = "match_expression_2" > .. </inner_element >
  <inner_element * ngSwitchCase = "match_expression_3" > .. </inner_element >
  <inner_element *ngSwitchDefault > .. </element >
</container_element>
```

ngSwitch Directive Example

Use the following code inside your application's app.component.ts file:

```
class Item {
  name: string;
  val: number;
}
export class AppComponent {
  items: Item[] = [{name: 'One', val: 1}, {name: 'Two', val: 2}, {name: 'Three', val: 3}];
  selectedValue: string = 'One';
}
```

Use the following code inside your application's app.component.html file:

```
<select [(ngModel)]="selectedValue" >
```

```
    <option * ngFor = "let item of items; " [value]="item.name " >
    {{item.name}} </option>
</select>
<div class = 'row' [ngSwitch]="selectedValue" >
    <div * ngSwitchCase = "One" > One is Pressed </div>
    <div * ngSwitchCase = "Two" > Two is Selected </div>
    <div *ngSwitchDefault > Default Option </div>
</div>
```


Data Binding in Angular 8

Data binding is Angular 8's core concept and is used to describe communication between a component and the DOM.

This is a technique for connecting your knowledge to your layer of view.

Simply put, you might say that data binding is a connection between your component type code and the user's template.

It makes interactive applications easy to define without having to push and pull data.

Data binding can either be one-way or two-way data binding.

One-way databinding

One way to communicate to the data is simply to interact in one way that changes the HTML template while making changes in TypeScript code.

Or

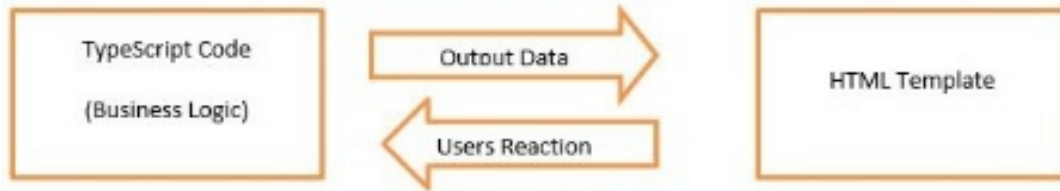
In single-way data binding the Model value is used on the View (HTML page), but from the View Model can not be modified. An example of one-way data link is angular interpolation/string interpolation, property binding and event binding.

Two-way databinding

Automatic data synchronization occurs between the model and the view during two-way data binding. Change in both components is expressed here.

If you make adjustments to the code, it is reflected in the View and mirrored in the code when you make modifications to the Model.

It is done instantly and automatically, ensuring that the HTML template and TypeScript code are updated at all times.



Property Binding in Angular 8

Property Binding is also a one-way technique for binding the data. In property binding we bind a property of a DOM element to a field in our component TypeScript code which is a given property.

In fact Angular internally transforms string interpolation into property binding.

For example:

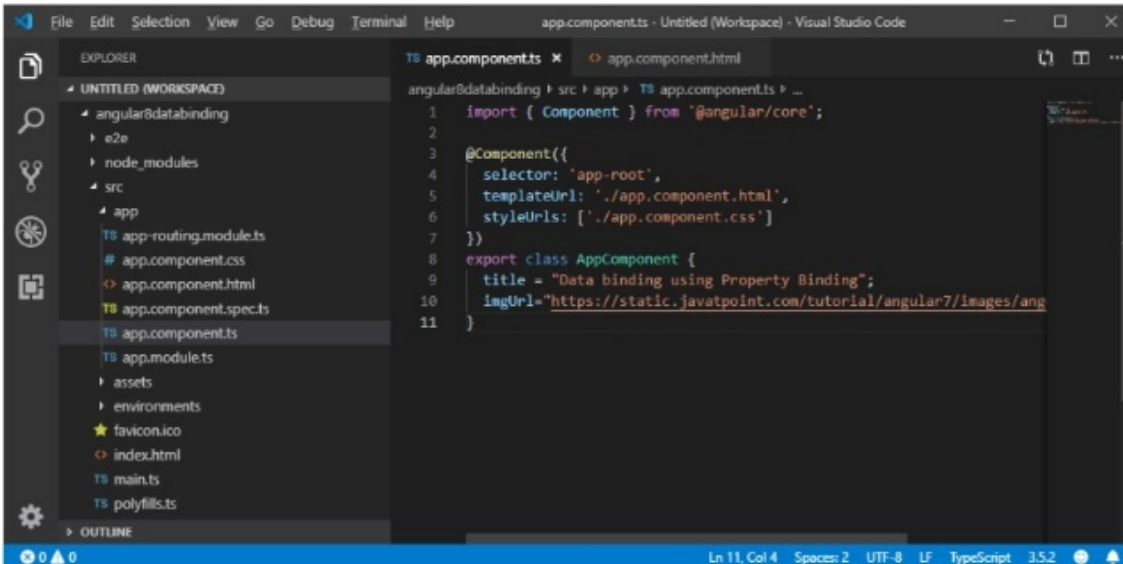
```
<img [src]="imgUrl" />
```

Property binding is favored over string interpolation because it has a shorter and simpler code. String interpolation should be used when you simply want to show any dynamic data from a component viewing between headings such as h1, h2, p etc.

Property Binding Example

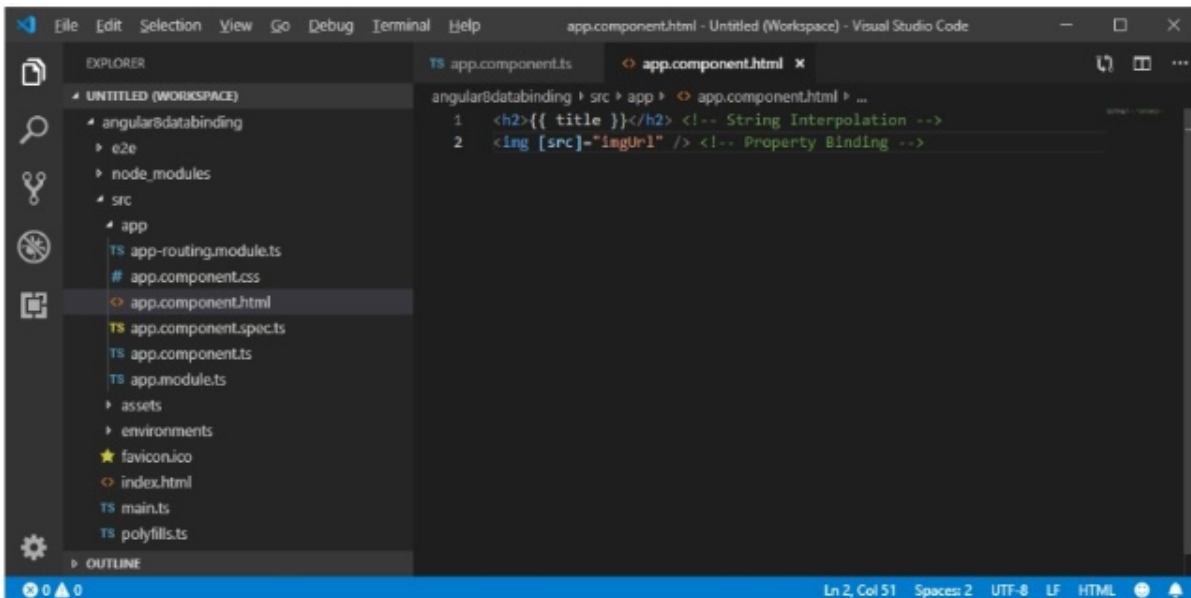
Open file app.component.ts and add the code below:

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = "Data binding using Property Binding " ;
  imgUrl = "https://static.javatpoint.com/tutorial/angular7/images/angular-7-logo.png " ;
}
```



Open app.component.html now, and use the following code for binding properties:

```
<h2> {{ title }} </h2> <!-- String Interpolation -->
<img [src]="imgUrl" /> <!-- Property Binding -->
```



Run the command `ng serve`, and open the local host to see the result.