ANGULAR 2 EXAMPLES

1. Components :

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

@Component({

selector: 'my-component',

template: `

<div>Hello my name is {{name}}.

<button (click)="sayMyName()">Say my name</button>

</div>

`

})

export class MyComponent {

name: string;

constructor() {

this.name = 'Max'

}

sayMyName() {

console.log('My name is', this.name)

}

}

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

@Component ({

selector: 'my-app',

template: ` <div>

<h1>{{appTitle}}</h1>

<div>Angular2 Componenets</div>

</div> `,

})

export class AppComponent {

appTitle: string = 'Welcome';

}

template: '

<div>

<h1>{{appTitle}}</h1>

<div>Angualr 2 Template</div>

</div>

'

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

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

appTitle: string = 'Welcome';

}

# Angular 2 - Transforming Data USING PIPE

Angular 2 has a lot of filters and pipes that can be used to transform data.

### Result

The property value will be converted to lowercase.

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

TutorialName: string = 'Angular JS2';

appList: string[] = ["Binding", "Display", "Services"];

}

Next, ensure the following code is present in the app/app.component.html file.

<div>

The name of this Tutorial is {{TutorialName}}<br>

The first Topic is {{appList[0] | lowercase}}<br>

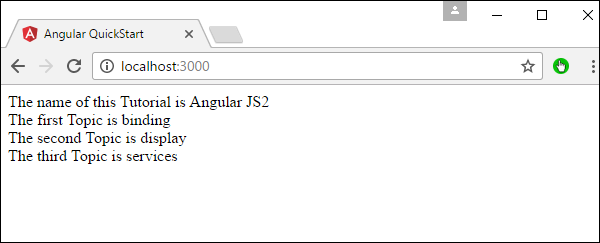
The second Topic is {{appList[1] | lowercase}}<br>

The third Topic is {{appList[2]| lowercase}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



## uppercase

This is used to convert the input to all uppercase.

### Syntax

Propertyvalue | uppercase

### Parameters

None.

### Result

The property value will be converted to uppercase.

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

TutorialName: string = 'Angular JS2';

appList: string[] = ["Binding", "Display", "Services"];

}

Next, ensure the following code is present in the app/app.component.html file.

<div>

The name of this Tutorial is {{TutorialName}}<br>

The first Topic is {{appList[0] | uppercase }}<br>

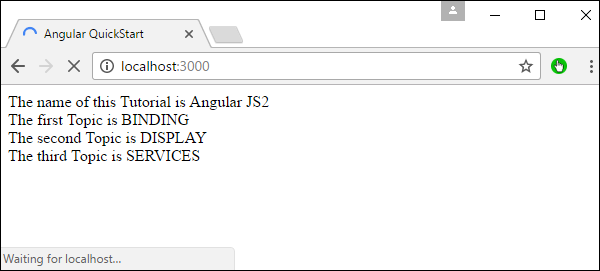
The second Topic is {{appList[1] | uppercase }}<br>

The third Topic is {{appList[2]| uppercase }}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



## slice

This is used to slice a piece of data from the input string.

### Syntax

Propertyvalue | slice:start:end

### Parameters

* **start** − This is the starting position from where the slice should start.
* **end** − This is the starting position from where the slice should end.

### Result

The property value will be sliced based on the start and end positions.

### Example

First ensure the following code is present in the app.component.ts file

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

TutorialName: string = 'Angular JS2';

appList: string[] = ["Binding", "Display", "Services"];

}

Next, ensure the following code is present in the app/app.component.html file.

<div>

The name of this Tutorial is {{TutorialName}}<br>

The first Topic is {{appList[0] | slice:1:2}}<br>

The second Topic is {{appList[1] | slice:1:3}}<br>

The third Topic is {{appList[2]| slice:2:3}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



## date

This is used to convert the input string to date format.

### Syntax

Propertyvalue | date:”dateformat”

### Parameters

**dateformat** − This is the date format the input string should be converted to.

### Result

The property value will be converted to date format.

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

newdate = new Date(2016, 3, 15);

}

Next, ensure the following code is present in the app/app.component.html file.

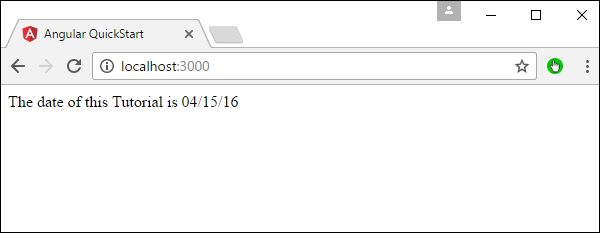
<div>

The date of this Tutorial is {{newdate | date:"MM/dd/yy"}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



## currency

This is used to convert the input string to currency format.

### Syntax

Propertyvalue | currency

### Parameters

None.

### Result

The property value will be converted to currency format.

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

newValue: number = 123;

}

Next, ensure the following code is present in the app/app.component.html file.

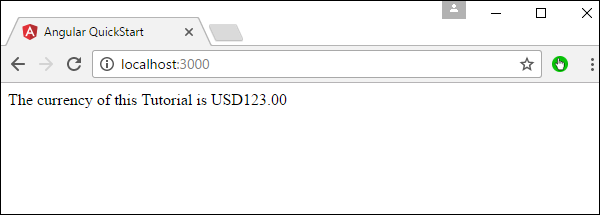
<div>

The currency of this Tutorial is {{newValue | currency}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



## percentage

This is used to convert the input string to percentage format.

### Syntax

Propertyvalue | percent

### Parameters

None

### Result

The property value will be converted to percentage format.

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

newValue: number = 30;

}

Next, ensure the following code is present in the app/app.component.html file.

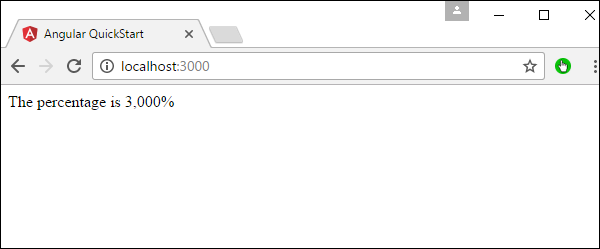
<div>

The percentage is {{newValue | percent}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.



There is another variation of the percent pipe as follows.

### Syntax

Propertyvalue | percent: ‘{minIntegerDigits}.{minFractionDigits}{maxFractionDigits}’

### Parameters

* **minIntegerDigits** − This is the minimum number of Integer digits.
* **minFractionDigits** − This is the minimum number of fraction digits.
* **maxFractionDigits** − This is the maximum number of fraction digits.

### Result

The property value will be converted to percentage format

### Example

First ensure the following code is present in the app.component.ts file.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

newValue: number = 0.3;

}

Next, ensure the following code is present in the app/app.component.html file.

<div>

The percentage is {{newValue | percent:'2.2-5'}}<br>

</div>

### Output

Once you save all the code changes and refresh the browser, you will get the following output.

## Percent Pipe

# Angular 2 - User Input

In Angular 2, you can make the use of DOM element structure of HTML to change the values of the elements at run time. Let’s look at some in detail.

The Input Tag

In the app.component.ts file place the following code.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

template: '

<div>

<input [value] = "name" (input) = "name = $event.target.value">

{{name}}

</div>

'

})

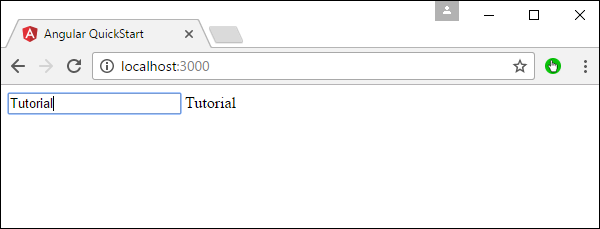
export class AppComponent { }

Following things need to be noted about the above code.

* **[value] = ”username”** − This is used to bind the expression username to the input element’s value property.
* **(input) = ”expression”** − This a declarative way of binding an expression to the input element’s input event.
* **username = $event.target.value** − The expression that gets executed when the input event is fired.
* **$event** − Is an expression exposed in event bindings by Angular, which has the value of the event’s payload.

Once you save all the code changes and refresh the browser, you will get the following output.

You can now type anything and the same input will reflect in the text next to the Input control.



Click Input

In the app.component.ts file place the following code.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

template: '<button (Click) = "onClickMe()"> Click Me </button> {{clickMessage}}'

})

export class AppComponent {

clickMessage = 'Hello';

onClickMe() {

this.clickMessage = 'This tutorial!';

}

}

Once you save all the code changes and refresh the browser, you will get the following output.



When you hit the Click Me button, you will get the following output.

## Click me Button

**DIRECTIVES** :

A **directive** is a custom HTML element that is used to extend the power of HTML. Angular 2 has the following directives that get called as part of the BrowserModule module.

* ngif
* ngFor

If you view the app.module.ts file, you will see the following code and the BrowserModule module defined. By defining this module, you will have access to the 2 directives.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule ({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

**Step 1**

**APP.COMPONENT.TS** :

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

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

appTitle: string = 'Welcome';

appStatus: boolean = true;

}

**Step 2** − Now in the app.component.html file, add the following code.

<div \*ngIf = 'appStatus'>{{appTitle}} ANGULAR 2 DIRECTIVE </div>

Let’s now take a look at an example of how we can use the \*ngFor directive.

**Step 1** − First add a property to the class named appList. This will be of the type which can be used to define any type of arrays.

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

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

appTitle: string = 'Welcome';

appList: any[] = [ {

"ID": "1",

"Name" : "One"

},

{

"ID": "2",

"Name" : "Two"

} ];

}

Hence, we are defining the appList as an array which has 2 elements. Each element has 2 sub properties as ID and Name.

**Step 2** − In the app.component.html, define the following code.

<div \*ngFor = 'let lst of appList'>

<ul>

<li>{{lst.ID}}</li>

<li>{{lst.Name}}</li>

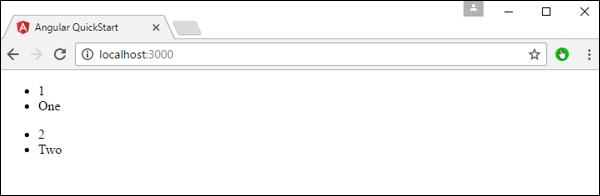
</ul>

</div>

In the above code, we are now using the ngFor directive to iterate through the appList array. We then define a list where each list item is the ID and name parameter of the array.

Once we add the above code, we will get the following output in the browser.

### Output



1. INPUT

import { Component, Input } from '@angular/core';

@Component({

selector: 'user-profile',

template: '<div>{{user.name}}</div>'

})

export class UserProfile {

@Input() user;

constructor() {}

}

1. SIMPLE FORM

##### **APP.MODULE.TS**

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

// We need to import the ReactiveFormsModule and HttpModule

import { ReactiveFormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

ReactiveFormsModule,

HttpModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

##### **LOGIN-PAGE.HTML**

<form [formGroup]="loginForm" (ngSubmit)="doLogin($event)">

<input formControlName="email" type="email" placeholder="Your email">

<input formControlName="password" type="password" placeholder="Your password">

<button type="submit">Log in</button>

</form

##### **APP.COMPONENT.TS**

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

import { FormBuilder, Validators } from '@angular/forms';

@Component({

selector: 'login-page',

templateUrl: 'login-page.html'

})

export class LoginPage {

public loginForm = this.fb.group({

email: ["", Validators.required],

password: ["", Validators.required]

});

constructor(public fb: FormBuilder) {}

doLogin(event) {

console.log(event);

console.log(this.loginForm.value);

}

}

**Angular 2 - Data Binding**

**Step 1** − Download any 2 images. For this example, we will download some simple images shown below.



**Step 2** − Store these images in a folder called **Images** in the app directory. If the images folder is not present, please create it.

**Step 3** − Add the following content in app.component.ts as shown below.

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

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

appTitle: string = 'Welcome';

appList: any[] = [ {

"ID": "1",

"url": 'app/Images/One.jpg'

},

{

"ID": "2",

"url": 'app/Images/Two.jpg'

} ];

}

**Step 4** − Add the following content in app.component.html as shown below.

<div \*ngFor = 'let lst of appList'>

<ul>

<li>{{lst.ID}}</li>

<img [src] = 'lst.url'>

</ul>

</div>

In the above app.component.html file, we are accessing the images from the properties in our class.

### Output

The output of the above program should be like this −

