NgClass Example Conditionally Apply Class

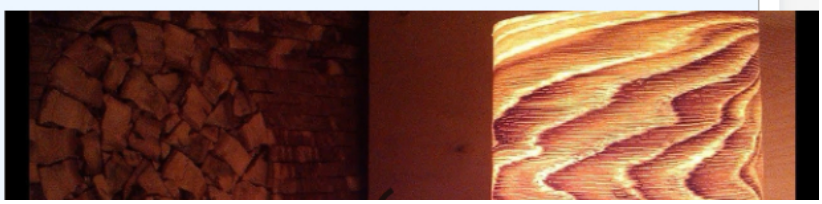
Leave a Comment / March 15, 2023 / 5 minutes of reading

⟵ NgIf Angular Tutorial NgStyle ⟶

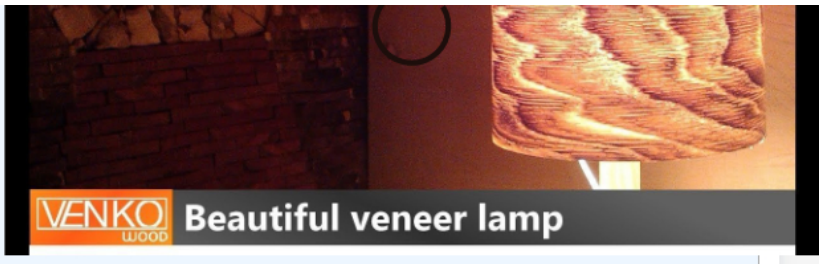
The **Angular** **ngClass** Directive is an Angular Attribute Directive, which allows us to add or remove CSS classes to an HTML element. `ngClass` makes adding a conditional class to an element easier, hence dynamically changing the styles at runtime. We can also add multiple CSS Classes.

Table of Contents

NgClass
ngClass Examples
NgClass with a String
NgClass with Array



NgClass with Object
Applying class from component
Conditionally applying class
Vs. Class
Summary



NgClass

The `ngClass` is a directive that Angular uses to dynamically add or remove CSS classes to an HTML element based on certain conditions.

```

<element [ngClass]="expression">...</element>

```

Where

The **element** is the DOM element to which we want to apply the CSS class.

The **expression,** when evaluated, must return a String, Array, or object. The `ngClass` Directive extracts the class name from the result and applies it to the element.

Let us explore all of them with examples.

We can add/remove classes to and from the element in three ways. One using the DOM ClassName Property. The second option is to use the Class shorthand (class binding). The third option is to use the `NgClass` Directive, which we cover in this tutorial.

ngClass Examples

Create a new angular project using ng new. Empty the **app.component.html**. Open the

create a new angular project using ng new. Empty the app.component.html. Open the **app.component.css** with the following CSS styles.

```
.primary {
  color: red;
}
.secondary {
  color: green;
}

.big {
  font-size: 20px;
}

.small {
  font-size: 12px;
}

.italics {
  font-style: italic;
}

.is-active {
  font-weight: bolder;
}
.section {
  margin: 10px;
}

```

You can view the source code from StackBlitz.

NgClass with a String

You can use the string as an expression and bind it directly to the `ngClass` attribute. If you want to assign multiple classes, separate each with space, as shown below.

```
<element [ngClass]="'cssClass1 cssClass2'">...</element>

```

Add the following to the **app.template.html**

Add the following to the **app.template.html**:

```
//Example 1
<div [ngClass]="'primary big'">Sample Text</div>
```

The above example code adds the multiple CSS Classes, `primary` & `big`, to the `div` element.

You can also use the `ngClass` without a square bracket. In that case, the expression is not evaluated but assigned directly to the `class` attribute. As shown below, we must remove the double quote around the expression.

```
//Example 2
<div ngClass="primary big">Sample Text</div>
```

NgClass with Array

If the `ngClass` expression returns an array, Angular treats each array element as a CSS class.

The syntax is as follows.

```
<element [ngClass]="['cssClass1', 'cssClass2']">...</element>
```

This example below, passes the array with two elements. `ngClass` will treat each element as a class. The following code applies `primary` and `big` classes to the `div` element.

```
<div [ngClass]="['primary', 'big']">Sample Text</div>
```

If there is a space between the string of an element, then both will be treated as a separate class. For example, in the code below, the array element at index 0 has the value `primary big`. It will treat this as two different classes.

```

<div [ngClass]="['primary big','italics']">Sample Text</div>

```

NgClass with Object

You can also bind the `ngClass` to an object. The **properties** (keys) of the object will act as a class name. The properties must return a boolean value. If the return value is **true**, the class is applied. Else not.

The syntax is as follows.

```

<element [ngClass]="{'cssClass1': true, 'cssClass2': true}">...</element>

```

If the property name contains spaces, it will treat it as two classes. But you need to use quotes around the property name. Quotes are also required if the class name has `-` etc.

In the example below, an object is bound to the `ngClass`. The object has two properties, `primary` and `big`. Since both return true, `ngClass` will apply both classes to the element.

```

<div [ngClass]="{ primary: true, big: true }">Sample Text</div>

```

The first property in the code below (`primary italics`) has a space. `ngClass` will treat it as

two separate classes. Hence it will apply three classes, `primary`, `italics` & `big`, to the element.

```
<div [ngClass]="{ 'primary italics': true, big: true }">Sample Text</div>
```

The property is-active has a –. Hence use the quote; else, it will result in an error.

```
<div [ngClass]="{ 'is-active': true, big: true }">Sample Text</div>
```

Applying class from component

We can dynamically change the CSS Classes from the component.

Create the following variable in the component class.

```
cssVar: string = 'primary big';
cssArray = ['primary', 'big'];
cssClass = {
  primary: true,
  big: true,
};
```

We can use them in the Template.

```
<b>Example 5: Value comming from the component method</b> <br />
<b>Example 5A (string)</b> <br />
<div [ngClass]="cssVar">Sample Text</div>

<b>Example 5B (array)</b> <br />
<div [ngClass]="cssArray">Sample Text</div>
```

```
<b>Example 5C (Object)</b> <br />
<div [ngClass]="cssClass">Sample Text</div>

```

You can modify the variables anytime; the view will reflect these changes.

The `ngClass` Directive switches the CSS class based on the `flag`'s value.

```

<div [ngClass]="flag ? 'primary' : 'secondary'">
 Sample Text
 <button (click)="flag = !flag">Change Color</button>
</div>

```

Conditionally applying class

One of the most important use cases of `ngClass` is we can conditionally apply the CSS classes.

For example, add the following code in the component class. The `getClass` return **primary** if the value of `num` is less than **50** and `secondary` otherwise.

```

numbers = [30, 40, 50, 60, 70, 80];
 getClass(num) {
  if (num <= 50) return 'primary';
  else return 'secondary';
 }

```

In the HTML template, we use `ngFor` to loop over the numbers array and display the list. We invoke the `getClass` method and assign the return value to `ngClass` Directive.

```

<ul>
 <li *ngFor="let num of numbers">
```

```
    <div [ngClass]="getClass(num)">{{ num }}</div>
  </li>
</ul>

```

You can write the expression in the Template itself, as shown below.

```

<ul>
  <li *ngFor="let num of numbers">
    <div [ngClass]="{ primary: num <= 50, secondary: num > 50 }">
      {{ num }}
    </div>
  </li>
</ul>

```

Vs. Class

You can also set CSS classes using the ClassName property or class binding. But ngClass is more flexible and is the preferred method.

You can view the source code from StackBlitz.

Summary

- `ngClass` is used the add conditional class to a HTML element.
- The syntax is `<element [ngClass]="expression">...</element>`
- The expression must return a string, array, or object.

Read More

1. Angular Tutorial
2. Angular Directives
3. ngFor
4. ngSwitch

4. ngSwitch
5. ngIf
6. ngClass
7. ngStyle
8. ngFor Trackby
9. Custom Directive
10. Angular ngStyle
11. Style Binding in Angular
12. Angular Global Styles
13. Angular Component Styles
14. View Encapsulation in Angular
15. Angular Directives

⟵ NgIf Angular Tutorial NgStyle ⟶

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name* Email* Website

Post Comment »

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Copyright © 20