@ViewChild() in Angular 8+

In **Angular 8+**, the `@ViewChild()` syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

```
@ViewChild('serverContentInput') serverContentInput: ElementRef;
```

use

```
@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;
```

The same change (add `{ static: true }` as a second argument) needs to be applied to ALL usages of `@ViewChild()` (and also `@ContentChild()` which you'll learn about later) IF you plan on accessing the selected element inside of `ngOnInit()`.

If you DON'T access the selected element in `ngOnInit` (but anywhere else in your component), set `static: false` instead!

If you're using Angular 9+, you only need to add `{ static: true }` (if needed) but not `{ static: false }`.

@ViewChild() in Angular 8+

In **Angular 8+**, the `@ViewChild()` syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

```
@ViewChild('serverContentInput') serverContentInput: ElementRef;
```

use

```
@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;
```

The same change (add `{ static: true }` as a second argument) needs to be applied to ALL usages of `@ViewChild()` (and also `@ContentChild()` which you'll learn about later) IF you plan on accessing the selected element inside of `ngOnInit()`.

If you DON'T access the selected element in `ngOnInit` (but anywhere else in your component), set `static: false` instead!

If you're using Angular 9+, you only need to add `{ static: true }` (if needed) but not `{ static: false }`.

@ViewChild() in Angular 8+

In **Angular 8+**, the `@ViewChild()` syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

```
@ViewChild('serverContentInput') serverContentInput: ElementRef;
```

use

```
@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;
```

The same change (add `{ static: true }` as a second argument) needs to be applied to ALL usages of `@ViewChild()` (and also `@ContentChild()` which you'll learn about later) IF you plan on accessing the selected element inside of `ngOnInit()`.

If you DON'T access the selected element in `ngOnInit` (but anywhere else in your component), set `static: false` instead!

If you're using Angular 9+, you only need to add `{ static: true }` (if needed) but not `{ static: false }`.

@ViewChild() in Angular 8+

In **Angular 8+**, the `@ViewChild()` syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

```
@ViewChild('serverContentInput') serverContentInput: ElementRef;
```

use

```
@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;
```

The same change (add `{ static: true }` as a second argument) needs to be applied to ALL usages of `@ViewChild()` (and also `@ContentChild()` which you'll learn about later) IF you plan on accessing the selected element inside of `ngOnInit()`.

If you DON'T access the selected element in `ngOnInit` (but anywhere else in your component), set `static: false` instead!

If you're using Angular 9+, you only need to add `{ static: true }` (if needed) but not `{ static: false }`.

@ViewChild() in Angular 8+

In **Angular 8+**, the `@ViewChild()` syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

```
@ViewChild('serverContentInput') serverContentInput: ElementRef;
```

use

```
@ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;
```

The same change (add `{ static: true }` as a second argument) needs to be applied to ALL usages of `@ViewChild()` (and also `@ContentChild()` which you'll learn about later) IF you plan on accessing the selected element inside of `ngOnInit()`.

If you DON'T access the selected element in `ngOnInit` (but anywhere else in your component), set `static: false` instead!

If you're using Angular 9+, you only need to add `{ static: true }` (if needed) but not `{ static: false }`.