Consuming multiple properties via Input decorator in angular 2

Asked 2 years, 7 months ago Active 2 years, 7 months ago Viewed 4k times

7

I have this component that receives two inputs via its selector, but this can be extended to any number of inputs and any component. So, in quest of consuming multiple properties in the component itself a single `@Input()` decorator did not allow me to consume multiple properties, so as a workaround I used two decorators for two input properties but I don't think this would be the only way to tackle such a scenario.

```
export class AsyncComponent {
     @Input() waitFor: boolean;
     @Input() message: string; // TODO: Verify if multiple inputs are needed for
multiple props
 }
```

Update

```
<app-async [waitFor]="true" message="foo"><app-async>
```

So, is is possible with any other way to just use a single decorator for any number of input props? If there are more props that I am passing other than `waitFor` and `message` would I have to do the following for each prop.

```
 @Input() waitFor: boolean;
 @Input() message: string;
 @Input() someOtherProp: string;
 ....
```

Or is there any other way to just have one `Input` decorator and consume any number of properties?

angular typescript angular-components angular-decorator

1,196 2 15 34

Why do you want a single input, single input is for single value. – Roman C Feb 25 '17 at 10:23

@RomanC, So lets say I am passing 10 props to the component, does this mean I would have to wrap the decorator 10 times in the component itself? – Umair Sarfraz Feb 25 '17 at 10:28

Wrap decorator 10 times? No, it doesn't work in this way. – Roman C Feb 25 '17 at 10:30

2 Answers

11

I agree with Roman C.

I would just pass a single object (not array) containing all props:

```
@Component({
  selector: 'app-async'
})
export class AsyncComponent {
  // Single object with all props.
  @Input() props: { waitFor: boolean; message: string; };
}
```

And then the parent component:

```
@Component({
  template: '<app-async [props]="myProps"><app-async>'
})
export class ParentComponent {
  myProps = { waitFor: true, message: 'foo' };
}
```

NB. Be aware that interfaces declared on input properties are NOT ENFORCED. It is best practice to still declare them, but JavaScript can't runtime-check a TypeScript interface. This remark applies to all code samples in this thread (single input, multiple inputs...).

Thanks. I was kind of confused with array with the other answer. – Umair Sarfraz Feb 25 '17 at 11:41

You're welcome. :) – AngularChef Feb 25 '17 at 11:42

Btw I just saw your interface usage description and its not fully understandable to me. Could you please maybe explain that a bit more? – Umair Sarfraz Feb 25 '17 at 11:44

1 As you probably know your TypeScript code needs to be transpiled into JavaScript so it can be executed in the browser. TypeScript interfaces are DROPPED during transpilation, they disappear from the final JavaScript code. In other words, TypeScript interfaces are useful for **compile-time** type-checking but not for **run-time** type-checking. In your example, the value of the input property is not yet known at compile-time, only at run-time, which means the interface doesn't really help. It's still good practice to keep it (for code documentation, code completion in your IDE...). – AngularChef Feb 25 '17 at 12:25

This is very useful information, really. I really appreciate it mate. Thanks a million. – Umair Sarfraz Feb 25 '17 at 20:54

0

It's not possible if you need multiple values use an array or something

```
export class AsyncComponent {
    @Input() props: any[];
```

answered Feb 25 '17 at 10:27

Roman C
44.9k 15 47 121

I think there was a slight confusion regarding my question. I really appreciate your answer but I have made some updates to the question to make it clearer. – Umair Sarfraz Feb 25 '17 at 10:57