How can I create an object based on an interface file definition in TypeScript?

Asked 6 years, 10 months ago Active 9 days ago Viewed 230k times

243

38

I have defined an interface like this:

```
interface IModal {
    content: string;
    form: string;
    href: string;
    $form: JQuery;
    $message: JQuery;
    $modal: JQuery;
    $submits: JQuery;
 }
```

I define a variable like this:

```
var modal: IModal;
```

However, when I try to set the property of modal it gives me a message saying that

```
"cannot set property content of undefined"
```

Is it okay to use an interface to describe my modal object and if so how should I create it?

typescript

edited Apr 25 at 15:21

Trilarion

7,302 6 43 81

asked Oct 30 '12 at 15:42

user1679941

8 Answers

329

If you are creating the "modal" variable elsewhere, and want to tell TypeScript it will all be done, you would use:

```
declare const modal: IModal;
```

If you want to create a variable that will actually be an instance of IModal in TypeScript you will need to define it fully.

```
const modal: IModal = {
    content: '',
    form: '',
    href: '',
    $form: null,
    $message: null,
    $modal: null,
    $submits: null
};
```

Or lie, with a type assertion, but you'll lost type safety as you will now get undefined in unexpected places, and possibly runtime errors, when accessing `modal.content` and so on (properties that the contract says will be there).

```
const modal = {} as IModal;
```

Example Class

```
class Modal implements IModal {
    content: string;
    form: string;
    href: string;
    $form: JQuery;
    $message: JQuery;
    $modal: JQuery;
    $submits: JQuery;
}

const modal = new Modal();
```

You may think "hey that's really a duplication of the interface" - and you are correct. If the Modal class is the only implementation of the IModal interface you may want to delete the interface altogether and use...

Rather than

```
const modal: IModal = new Modal();
```

edited Oct 2 '18 at 18:29

answered Oct 30 '12 at 15:52

Fenton
167k 47 304 332

1 Thanks. I was hoping to get away from having to define all of the modal contents initially. Would it be easier if instead of an interface I defined Modal as a class with properties and then used new and a constructor to set up all of the initial values? – user1679941 Oct 30 '12 at 16:24

1 Yes - you could define a class and then you would only have to create new instances when you needed them. Do you need an example? – Fenton Oct 30 '12 at 16:25

2 I have added an example and a note about the interface. – Fenton Oct 30 '12 at 16:37

16 `var modal = <IModal>{};` this is what I was looking for ;-) thanks! – Legends Mar 4 '17 at 21:02

1 @gen short answer, yes. Longer version - TypeScript is structurally typed, so as long as it looks like an IModal, it is as good as an IModal. – Fenton Jul 14 '17 at 14:32

168

If you want an empty object of an interface, you can do just:

```
var modal = <IModal>{};
```

The advantage of using interfaces in lieu of classes for structuring data is that if you don't have any methods on the class, it will show in compiled JS as an empty method. Example:

```
class TestClass {
    a: number;
    b: string;
    c: boolean;
}
```

compiles into

```
    }
    return TestClass;
})();
```

which carries no value. Interfaces, on the other hand, don't show up in JS at all while still providing the benefits of data structuring and type checking.

answered Jun 16 '14 at 19:37

user3180970
1,681 2 7 2

1 ` To add on to the above comment. To call a function "updateModal(IModal modalInstance) { }", you can create an inline instance of the interface 'IModal', like this: //some code here, where the updateModal function & IModal are accessible: updateModal(<IModal>{ //this line creates an instance content: '', form: '', href: '', $form: null, $message: null, $modal: null, $submits: null }); //complete off your code ` – Sunny Aug 30 '15 at 12:22

by using `var modal= <abc>{}` we just have assigned modal of type abc of empty object but where we have declared type of modal ? i am using typescript6. – Pardeep Jain Jan 7 '16 at 6:26

37

If you are using React, the parser will choke on the traditional cast syntax so an alternative was introduced for use in *.tsx* files

```
let a = {} as MyInterface;
```

https://www.typescriptlang.org/docs/handbook/jsx.html

answered Jun 14 '16 at 3:32

geg
2,318 1 20 22

17

I think you have basically **five different options** to do so. Choosing among them could be easy depending on the goal you would like to achieve.

The best way in most of the cases to **use a class and instantiate it**, because you are using TypeScript to apply type checking.

```
    //...

    //Extra
    foo: (bar: string): void;
}

class Modal implements IModal {
    content: string;
    form: string;

    foo(param: string): void {
    }
}
```

Even if other methods are offering easier ways to create an object from an interface you should consider **splitting your interface** apart, if you are using your object for different matters, and it does not cause interface over-segregation:

```
interface IBehaviour {
    //Extra
    foo(param: string): void;
}

interface IModal extends IBehaviour{
    content: string;
    form: string;
    //...
}
```

On the other hand, for example during unit testing your code (if you may not applying separation of concerns frequently), you may be able to accept the drawbacks for the sake of productivity. You may apply other methods to create mocks mostly for big third party *.d.ts interfaces. And it could be a pain to always implement full anonymous objects for every huge interface.

On this path your first option is to **create an empty object**:

```
var modal = <IModal>{};
```

Secondly to **fully realise the compulsory part of your interface**. It can be useful whether you are calling 3rd party JavaScript libraries, but I think you should create a class instead, like before:

```
var modal: IModal = {
```

```
    foo: (param: string): void => {
    }
};
```

Thirdly you can create just a **part of your interface** and create an **anonymous object**, but this way you are responsible to fulfil the contract

```
var modal: IModal = <any>{
    foo: (param: string): void => {

    }
};
```

Summarising my answer even if interfaces are optional, because they are not transpiled into JavaScript code, TypeScript is there to provide a new level of abstraction, if used wisely and consistently. I think, just because you can dismiss them in most of the cases from your own code you shouldn't.

edited Dec 14 '15 at 13:42

answered Dec 13 '15 at 17:59

Ursegor

778 7 15

16

You can do

```
var modal = {} as IModal
```

answered Dec 3 '17 at 10:26

Kangudie Muanza - Killmonger

660 5 9

Since I haven't found an equal answer in the top and my answer is different. I do:

edited Feb 1 at 0:43
Paul Rooney
13.5k 7 31 46

answered Jan 23 '18 at 9:23
Zokor
89 1 2

1

Using your interface you can do

```
class Modal() {
  constructor(public iModal: IModal) {
   //You now have access to all your interface variables using this.iModal object,
   //you don't need to define the properties at all, constructor does it for you.
  }
}
```

edited Feb 1 at 0:44
Paul Rooney
13.5k 7 31 46

answered Apr 13 '17 at 10:39
Byrd
528 1 8 16

You wrap the interface inside an object... – Ovi Trif May 2 at 14:51

0

Here is another approach:

You can simply create an ESLint friendly object like this

```
const modal: IModal = {} as IModal;
```

Or a default instance based on the interface and with sensible defaults, if any

```
const defaultModal: IModal = {
    content: "",
    form: "",
    href: "",
    $form: {} as JQuery,
    $message: {} as JQuery,
```

Then variations of the default instance simply by overriding some properties

```
const confirmationModal: IModal = {
    ...defaultModal,     // all properties/values from defaultModal
    form: "confirmForm"  // override form only
}
```

edited Aug 31 at 5:10

answered Aug 31 at 4:50

rbento

3,908 1 33 46