

# Training TypeScript

## Module: Conditional Types



Peter Kassenaar

[info@kassenaar.com](mailto:info@kassenaar.com)

# What are conditional types?

*"In programs, we have to make decisions **based on input**. Given the fact that values can be easily introspected, decisions are also based on the **types** of the inputs. **Conditional types help describe the relation between the types of inputs and outputs.***

# Official docs on Conditional Types

The screenshot shows the TypeScript Handbook page for Conditional Types. The left sidebar contains a navigation menu with items like 'Get Started', 'Handbook', 'The TypeScript Handbook', 'The Basics', 'Everyday Types', 'Narrowing', 'More on Functions', 'Object Types', 'Type Manipulation' (which is expanded to show 'Creating Types from Types', 'Generics', 'Keyof Type Operator', 'Typeof Type Operator', 'Indexed Access Types', 'Conditional Types' (highlighted), 'Mapped Types', 'Template Literal Types', 'Classes', and 'Modules'), 'Reference', 'Tutorials', 'What's New', and 'Declaration Files'. The main content area is titled 'Conditional Types' and includes an introductory paragraph, two code examples, and a definition of conditional types. The first example shows an interface 'Animal' with a 'live()' method, and an interface 'Dog' that extends 'Animal' with a 'woof()' method. It then shows a conditional type 'Example1' that is 'Dog' if 'number' is 'string', and 'number' otherwise. The second example shows a conditional type 'Example2' that is 'RegExp' if 'number' is 'string', and 'string' otherwise. The text explains that conditional types take the form '(condition ? trueExpression : falseExpression)' in JavaScript. A final example shows 'SomeType extends OtherType ? TrueType : FalseType;'. The bottom of the page explains that when the type on the left of the 'extends' is assignable to the one on the right, the type in the first branch is used, otherwise the type in the latter branch is used. A footer note states that conditional types might not immediately seem useful but can be useful to determine if 'Dog' is a subtype of 'Animal'. On the right side, there is a red-bordered box containing the text 'On this page' followed by a list of links: 'Conditional Type Constraints', 'Inferring Within Conditional T...', and 'Distributive Conditional Types'. Below this is a section titled 'Is this page helpful?' with 'Yes' and 'No' buttons.

TypeScript Download Docs Handbook Community Playground Tools Search Docs

## Conditional Types

At the heart of most useful programs, we have to make decisions based on input. JavaScript programs are no different, but given the fact that values can be easily introspected, those decisions are also based on the types of the inputs. *Conditional types* help describe the relation between the types of inputs and outputs.

```
interface Animal {
  live(): void;
}
interface Dog extends Animal {
  woof(): void;
}

type Example1 = Dog extends Animal ? number : string;
// type Example1 = number

type Example2 = RegExp extends Animal ? number : string;
// type Example2 = string
```

Conditional types take a form that looks a little like conditional expressions ( `condition ? trueExpression : falseExpression` ) in JavaScript:

```
SomeType extends OtherType ? TrueType : FalseType;
```

When the type on the left of the `extends` is assignable to the one on the right, then you'll get the type in the first branch (the "true" branch); otherwise you'll get the type in the latter branch (the "false" branch).

From the examples above, conditional types might not immediately seem useful - we can tell ourselves whether or not `Dog`

**On this page**

- Conditional Type Constraints
- Inferring Within Conditional T...
- Distributive Conditional Types

**Is this page helpful?**

Yes No

<https://www.typescriptlang.org/docs/handbook/2/conditional-types.html>



# Conditional Types

Setting a type based on a true|false condition

# What is a conditional type?

- Basically you define the type of a certain `type` on the outcome of an if-else statement.
- You mostly use the **ternary operator** `? ... :` for that.
- This is useful if you have types that extend from each other and you want to create a derived type based on a generic condition.

The basic layout of a conditional type is

```
type SomeType = T extends U ? TypeX : TypeY
```

## Conditional types in plain English...

*“If some type  $T$  is assignable to some type  $U$ , then choose/return type  $X$ . Otherwise choose/return type  $Y$ ”*

# Example Conditional Type

- See comments in the code.

```
// basetypes
type Diesel = {
  type: "petroleum" | "bio" | "synthetic"; // fuel types available to Diesel engines
};
type Gasoline = {
  type: "hybrid" | "conventional" | "unleaded"; // fuel types available to Gas engines
};
// derived types
type Bus = {
  engine: Diesel; // so we KNOW a Bus variable has "petroleum" | "bio" | "synthetic"
};
type Car = {
  engine: Gasoline; // likewise we KNOW a Car uses "hybrid" | "conventional" petrol
};

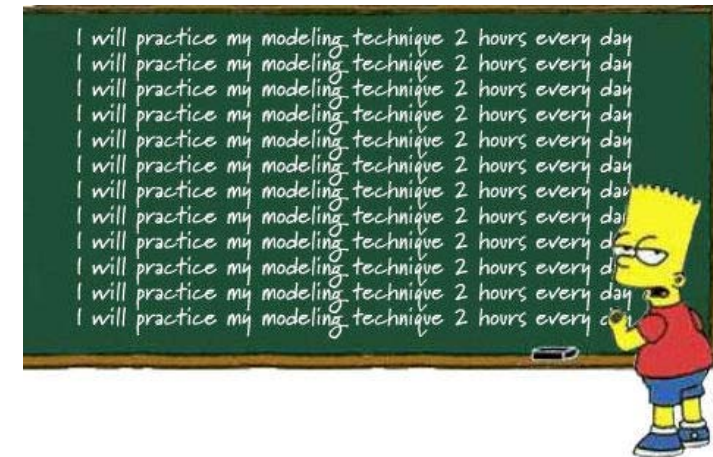
// OUR CONDITIONAL TYPE:
type Engine<T> = T extends { engine: unknown } ? T["engine"] : never;
type BusEngine = Engine<Bus>;

// The actual Variable busEngine, of type BusEngine
const busEngine: BusEngine = {
  type: "unleaded" // INVALID!
};
```

# Workshop

- Create a type or an interface and create some variables, based on that type
- Create a new variable, but make it conditional, like in the example and slides
- Example code: `../35-conditional-types.ts`.
- Docs:

<https://www.typescriptlang.org/docs/handbook/2/conditional-types.html>






# More info on Conditional Types

typescript conditional types

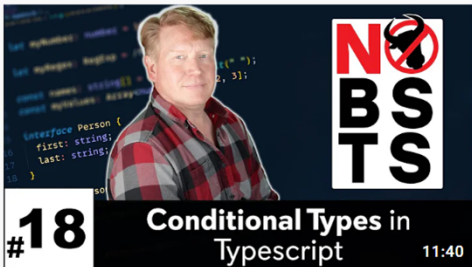
FILTERS About these results



**Conditional Types - Advanced TypeScript**  
22K views • 2 years ago  
Dmytro Danylov

A conditional type ( $T \text{ extends } U ? X : Y$ ) in TypeScript selects one of two possible types based on some condition. Take a ...

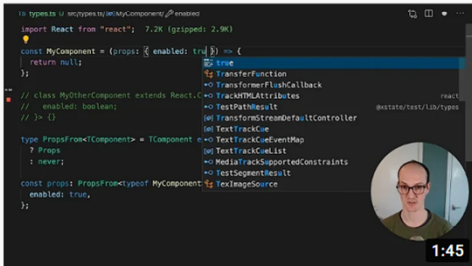
Conditional type definition | A basic example | Deferred evaluation of a conditional type | Nested... 8 moments



**No BS TS #18 - Conditional Types in Typescript**  
7.5K views • 1 year ago  
Jack Herrington


Turns out you can define types conditionally based on the type check against a different type. Let's try it out on a fetch function that ...

Introduction | What are conditional types? | Importing libraries | fetchPokemon with conditionals |... 6 chapters



**Extracting React Props using CONDITIONAL TYPES - Advanced TypeScript**  
4.1K views • 3 months ago  
Matt Pocock

Become a TypeScript Wizard with Matt's upcoming TypeScript Course: <https://www.mattpocock.com/> Follow Matt on Twitter ...



**TypeScript Berlin Meetup #2: Generics, Conditional types and Mapped types**  
11K views • 2 years ago  
Prisma

[https://www.youtube.com/results?search\\_query=typescript+conditional+types](https://www.youtube.com/results?search_query=typescript+conditional+types)

