To create the generic utility types MyPick and MyExclude in TypeScript, follow these steps:

1. **Create MyPick Type**:
   * MyPick<T, K> should create a new type by picking a subset of properties K from the type T.
   * This behaves similarly to the built-in Pick utility type.
2. **Create MyExclude Type**:
   * MyExclude<T, U> should create a new type by excluding properties from type T that are assignable to U.
   * This behaves similarly to the built-in Exclude utility type.

Here's how you can define these utility types:

**Step 1: Create MyPick Type**

The MyPick type will create a new type by picking properties K from type T.

typescript

type MyPick<T, K extends keyof T> = {

[P in K]: T[P];

};

**Step 2: Create MyExclude Type**

The MyExclude type will exclude properties from type T that are assignable to U.

typescript

type MyExclude<T, U> = T extends U ? never : T;

**Usage Examples**

**Example for MyPick:**

typescript

interface Todo {

title: string;

description: string;

completed: boolean;

}

type TodoPreview = MyPick<Todo, 'title' | 'completed'>;

const todo: TodoPreview = {

title: "Clean room",

completed: false,

// description: "string", // Error: Property 'description' does not exist on type 'TodoPreview'

};

**Example for MyExclude:**

typescript

type T = 'a' | 'b' | 'c';

type U = 'a' | 'c';

type Result = MyExclude<T, U>; // "b"

**Full Implementation in TypeScript**

Here's the full code with usage examples included:

typescript

// Definition of MyPick

type MyPick<T, K extends keyof T> = {

[P in K]: T[P];

};

// Definition of MyExclude

type MyExclude<T, U> = T extends U ? never : T;

// Example for MyPick

interface Todo {

title: string;

description: string;

completed: boolean;

}

type TodoPreview = MyPick<Todo, 'title' | 'completed'>;

const todo: TodoPreview = {

title: "Clean room",

completed: false,

// description: "string", // Error: Property 'description' does not exist on type 'TodoPreview'

};

// Example for MyExclude

type T = 'a' | 'b' | 'c';

type U = 'a' | 'c';

type Result = MyExclude<T, U>; // "b"

In this implementation:

* MyPick<T, K> creates a type with only the properties K of type T.
* MyExclude<T, U> removes the types from T that are assignable to U.