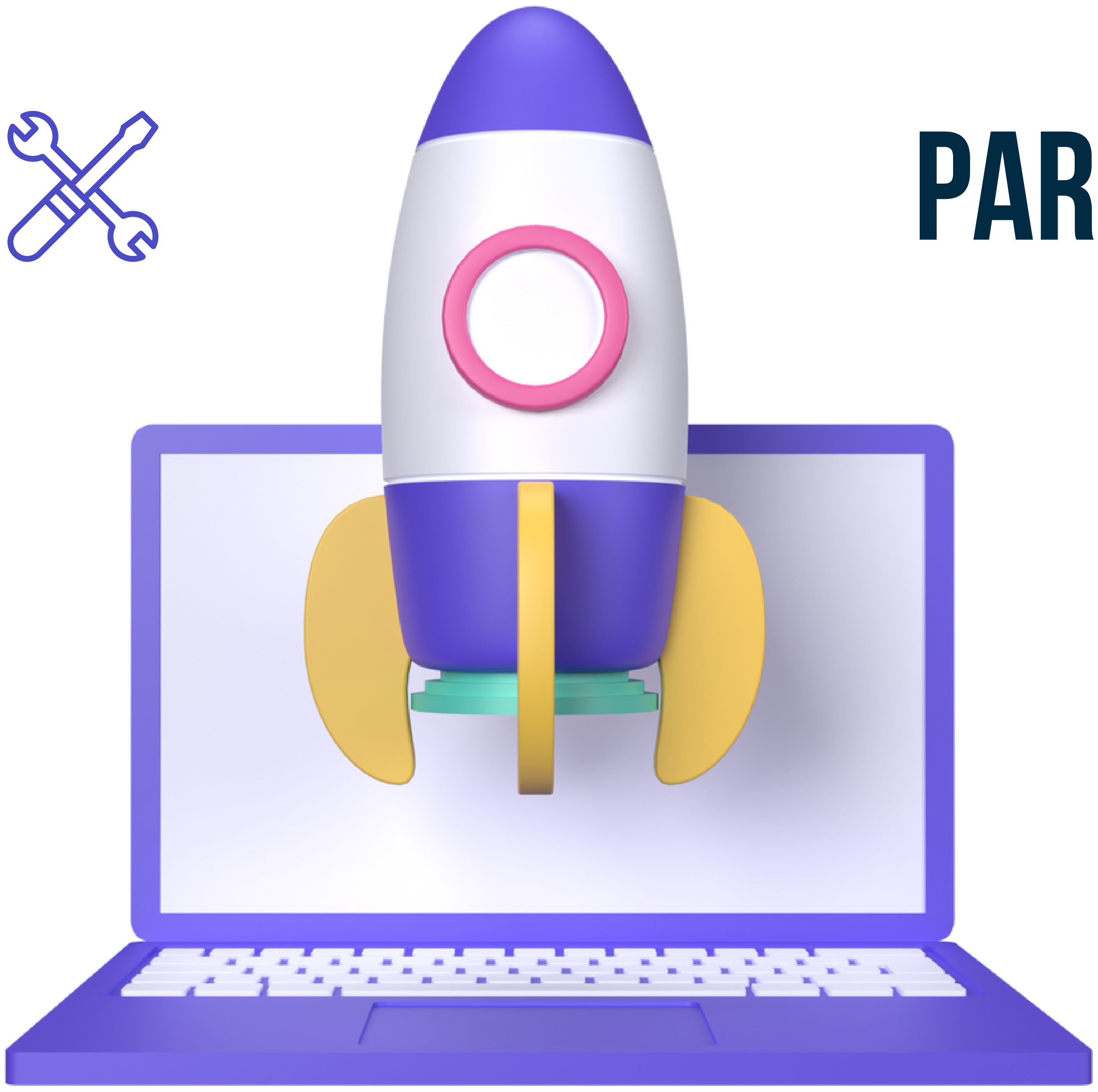
TS

PART 2



TYPESCRIPT UTILITIES

YOU SHOULD KNOW

SWIPE

Required

The opposite to Partial is Required utility type, which makes everything required

```
interface Person {
 name?: string | undefined;
 age?: number | undefined;
 email?: string | undefined;
}

type RequiredPerson = Required<Person>;

//RequiredPerson will be same as:
interface Person {
 name: string;
 age: number;
 email: string;
 }
```

Exclude

Exclude utility type allows you to create a new type by removing members of an union

```
type NumberOrString = number | string;
type OnlyNumber =
Exclude<NumberOrString, string>;

// same as:
// type OnlyNumber = number;

const num: OnlyNumber = 5;
const str: OnlyNumber = 'hello';
```

Error: Type '"hello"' is not assignable to type 'number'.

You can even exlude mulitple members of an union:

```
type NumberStringOrBoolean = number |
string | boolean;
type OnlyBoolean =
Exclude<NumberStringOrBoolean, string |
number>;

// same as:
type OnlyBoolean = boolean;
```

Extract

The opposite to Exclude is Extract utitlity type that allows you to pick a or multiple members from an union

```
type NumberOrString = number | string |
boolean;
type OnlyNumber = Extract<NumberOrString,
number>;

// same as:
type OnlyNumber = number;
```

Parameters

The Parameters utility type lets you extract the type of parameters from a function.

```
function add(a: number, b: string, c:boolean): string {
  return a + b;
}

type AddReturnType = Parameters<typeof add>;

// type AddReturnType = [a: number, b: string,
c:boolean];
```

Awaited

Awaited utility type allows you to extract the resolved type of a promise or other type that uses await.

```
type promiseNumber = Promise<number>


type justNumber = Awaited<Promise<number>>
// type justNumber = number
```

Awaited and ReturnType combined

Here’s an example of using ReturnType and Awaited together:

```
async function fetchData(): Promise<string> {
  // fetch data from API and return a string
}

type ResolvedResult = Awaited<ReturnType<typeof
fetchData>>;
// type ResolvedResult = string
```

In this example, we define an async function that returns a Promise of a string (Promise<string>). We then use ReturnType to extract the return type of fetchData and pass it as an argument to Awaited to unwrap the promised's resolved type.

WANT MORE CONTENT LIKE THIS

LIKE, REPOST, COMMENT

AND FOLLOW ME