TypeScript Interview Questions - Coding Interview

Section 1: Introduction

- Setting up Typescript (educational)
- Javascript vs Typescript
  - Q1: What is the difference between Javascript and Typescript?
- Does Typescript improve our code out of the box?
  - Q1: Does Typescript improve our code if we just change the extension of the file?

Section 2: Core Typescript

- How to define basic types inside Typescript?
  - Q1: How can we define basic types in Typescript?
- What is the difference between explicit vs implicit types?
  - Q1: What is the difference between explicit vs implicit types?
- Type a function getFullName correctly
  - Q1: Write a function getFullName which gets name and surname and returns a full name
- What is interface in Typescript?
  - Q1: What is an interface in Typescript?
- What is type in Typescript?
  - Q1: What is a type in Typescript?
- What is the difference between an interface and a type?
  - Q1: What is the difference between type and interface?
- What is union in Typescript?
  - Q1: What is union in Typescript?
- How to narrow the union in Typescript?
  - Q1: What do you know about type narrowing?
- What is void in Typescript?
  - Q1: What is void in Typescript?
- What is never in Typescript?
  - Q1: What is never in Typescript?
- What is any in Typescript?
  - Q1: What is any in Typescript?
- What is unknown in Typescript?
  - Q1: What is unknown in Typescript?
- How to work with DOM in Typescript?
  - Q1: How to work with DOM in Typescript?
- How to work with classes in Typescript?
  - Q1: How to work with classes in Typescript?
- What is an enum in Typescript?
  - Q1: What is enum in Typescript?

- What are generics in Typescript?
  - Q1: How to work with generics in Typescript?
- What is a tuple in Typescript?
  - Q1: What is the difference between an array and a tuple?
- What is optional property in Typescript?
  - Q1: How to use optional properties in object?
  - Q2: How to use an elvis operator
- How to cover dynamic keys in the object?
  - Q1: How to type keys in object?
- What is index signature in Typescript?
  - Q1: What is index signature in Typescript?
- What is a record type in Typescript?
  - Q1: How to use Record helper in Typescript?
- What is omit and pick in Typescript?
  - Q1: How to use Omit and Pick in Typescript?
- What is readonly in Typescript?
  - Q1: How to use Readonly helper in Typescript?
- What is partial in Typescript?
  - Q1: How to use Partial helper in Typescript?
- What is required in Typescript?
  - Q1: How to use Required helper in Typescript?
- How to use Typescript together with React?
  - Q1: How to use Typescript together with React framework?
- What is type inference in Typescript?
  - Q1: What is type inference in Typescript?
- What is literal type in Typescript?
  - Q1: What is literal type in Typescript?
- What is tsconfig.json file?
  - Q1: What is tsconfig.json file?
- What are the core components of Typescript?
  - Q1: What does Typescript as a language consists of?
- How to transpile Typescript to Javascript?
  - Q1: How to transpile file to Javascript?
- What is d.ts file in Typescript?
  - Q1: What is .d.ts file?
  - Q2: Why do we need it?
- What is map file in Typescript?
  - Q1: What is .map file and why do we need it?

Section 2: Core Typescript

- Introduction for advanced section (educational)
- What is function overloading in Typescript?
  - Q1: What is function overloading in Typescript?
- What is extends in Typescript?
  - Q1: What does extends keyword do in Typescript?

- What is infer in Typescript?
    - Q1: What does infer keyword do in Typescript?
- Do it yourself - readonly
    - Q1: Write a Readonly helper for Typescript

    ```
    interface Todo {
      title: string
      description: string
    }
    const todo: MyReadonly<Todo> = {
      title: 'Hey',
      description: 'foobar'
    }
    todo.title = 'Hello' // Error: cannot reassign a readonly property
    todo.description = 'bar' // Error: cannot reassign a readonly property
    ```

- Do it yourself - first
    - Q1: Implement a generic First that takes an Array T and returns it's first element's type

    ```
    type arr1 = ['a', 'b', 'c']
    type arr2 = [3,2,1]
    type head1 = First<arr1> // expected to be 'a'
    type head2 = First<arr2> // expected to be 3
    ```

- Do it yourself - tuple length
    - Q1: For given tuple, you need to create a genetic Length, pick the length of the tuple

    ```
    type tesla = ['tesla', 'model 3', 'model x', 'model y']
    type spaceX = ['falcon 9', 'falcon heavy', 'dragon', 'starship',
    'human spaceflight']
    type teslaLength = Length<tesla> // expected 4
    type spaceXLength = Length<spaceX> // expected 5
    ```

- Do it yourself - if
    - Q1: Implement the util type If<C, T, F> which accepts condition C, a truthy value T, and a falsy value F. C is expected to be either true or false while T and F can be any type.

    ```
    type A = If<true, 'a', 'b'> // expected to be 'a'
    type B = If<false, 'a', 'b'> // expected to be 'b'
    ```

- Do it yourself - concat
    - Q1: Implement the Javascript Array.concat function in the type system. A type takes the two arguments. The output should be a new array that includes inputs in ltr order

    ```
    type Result = Concat<[1], [2]> // expected to be [1,2]
    ```