Advanced Concepts and Features

- Everything is subtype of any
- Never is subtype of everything
- Super type - inverse of subtype
- Object is super type of an array
- Any is super type of everything
- we can save sub type on super type
- **Type widening**
  - For variable that might change TS assigns it primitive type
  - For non-changing variable TS assigns literal value
  - For values where we use **let** and assign **null**, TS will infer it as **any**
  - For values where we use **const** and assign **null**, TS will infer it as **null**
  - For enum with let TS will infer enum Name
  - For enum with const TS will infer enum value
- **Typecasting**
  - **as** or **<type>** - keyword, where we want TS to infer particular type
  - as - preferred
  - **!** mark at the end of variable declaration, we let TS know that value will definitely exists
- **Totality**
  - TS checks for corner cases where a function might return undefined
- **Type Checking**
  - **typeof** - primitive type check
  - **discriminated unions** - helps in identifying object type in union of objects
- **keying-in or Index Accessed Types**
  - new type inherit from object type already defined
- **Keyof**
  - Union of all keys that are present in a type
  - if key is defined as string it will always be union of string and numbers
    - because js strings are coerced as numbers behind the scenes
- **typeof**
  - similar to JS typeof
  - can be used to assign and infer types from (variables)
- **Mapped Types**
  - will loop through list to create keys for map
- C**onditional Types**
  - advanced types
  - add condition to TS type system
  - we can check whether one type extends another type and perform action based on boolean val
  - no JS variables are involved in conditions
- **Infer keyword**
  - can only be used inside a conditional type
  - can be used infer types of function as well

- can be used to know return type of function that are not known or where fn returns large objects