3.2.5 The Null Type

The Null type corresponds to the similarly named JavaScript primitive type and is the type of the `null` literal.

The `null` literal references the one and only value of the Null type. It is not possible to directly reference the Null type itself.

The Null type is a subtype of all types, except the Undefined type. This means that `null` is considered a valid value for all primitive types, object types, and type parameters, including even the Number and Boolean primitive types.

Some examples:

```
var n: number = null;   // Primitives can be null
var x = null;           // Same as x: any = null
var e: Null;            // Error, can't reference Null type
```

3.2.6 The Undefined Type

The Undefined type corresponds to the similarly named JavaScript primitive type and is the type of the `undefined` literal.

The `undefined` literal denotes the value given to all uninitialized variables and is the one and only value of the Undefined type. It is not possible to directly reference the Undefined type itself.

The undefined type is a subtype of all types. This means that `undefined` is considered a valid value for all primitive types, object types, and type parameters.

Some examples:

```
var n: number;          // Same as n: number = undefined
var x = undefined;      // Same as x: any = undefined
var e: Undefined;       // Error, can't reference Undefined type
```

3.2.7 Enum Types

Enum types are distinct user defined subtypes of the Number primitive type. Enum types are declared using enum declarations (section 9.1) and referenced using type references (section 3.6.2).

Enum types are assignable to the Number primitive type, and vice versa, but different enum types are not assignable to each other.

3.2.8 String Literal Types

Specialized signatures (section 3.7.2.4) permit string literals to be used as types in parameter type annotations. String literal types are permitted only in that context and nowhere else.

All string literal types are subtypes of the String primitive type.