3.1 The Any Type

The Any type is used to represent any JavaScript value. A value of the Any type supports the same operations as a value in JavaScript and minimal static type checking is performed for operations on Any values. Specifically, properties of any name can be accessed through an Any value and Any values can be called as functions or constructors with any argument list.

The `any` keyword references the Any type. In general, in places where a type is not explicitly provided and TypeScript cannot infer one, the Any type is assumed.

The Any type is a supertype of all types.

Some examples:

```
var x: any;             // Explicitly typed
var y;                  // Same as y: any
var z: { a; b; };       // Same as z: { a: any; b: any; }

function f(x) {         // Same as f(x: any): void
    console.log(x);
}
```

3.2 Primitive Types

The primitive types are the Number, Boolean, String, Void, Null, and Undefined types and all user defined enum types.

3.2.1 The Number Type

The Number primitive type corresponds to the similarly named JavaScript primitive type and represents double-precision 64-bit format IEEE 754 floating point values.

The `number` keyword references the Number primitive type and numeric literals may be used to write values of the Number primitive type.

For purposes of determining type relationships (section 3.8) and accessing properties (section 4.10), the Number primitive type behaves as an object type with the same properties as the global interface type 'Number'.

Some examples:

```
var x: number;          // Explicitly typed
var y = 0;              // Same as y: number = 0
var z = 123.456;        // Same as z: number = 123.456
var s = z.toFixed(2);   // Property of Number interface
```