Undefined vs Not Defined vs TDZ

Implicit Global Variables

Case 1

```
a = 10;
console.log(a); // 10
```

Case 2

```
console.log(a); // error: a is not defined
a = 10;
```

Case 3

```
console.log(a); // undefined
var a = 10;
```

Temporal Dead Zone

1. In JavaScript, the Temporal Dead Zone (TDZ) is a behavior that occurs during the variable creation phase of the execution context.

2. It refers to the period between the creation of a variable and its initialization, during which the variable cannot be accessed.

3. Attempting to access a variable during the TDZ will result in a ReferenceError.

Case 4

```
console.log(a); // error: can not access a before initialization
let a = 10;
```

Yes, let and const variables can be hoisted. If let and const varibles could not be hoisted, we would have get the error a is not defined.

```
let a = 10;
{
  console.log(a); // error: cannot access a before initialization
  let a = 20;
}
```

In this example, we try to access the variable a before it is declared. This results in a ReferenceError because the variable is in the TDZ and cannot be accessed until it has been declared.

To avoid TDZ errors, it's best practice to declare variables at the beginning of their scope, before they are accessed. This will ensure that they are not in the TDZ and can be accessed without throwing a ReferenceError.