很多其他的语言中（比如python），只有 null；但 JS 语言中，既有 null，又有 undefined。

很多人会弄混，由此觉得 JS 语言很麻烦。

其实不然，学习完之后，你会发现 null 和 undefined 的区别很容易理解。

Null：空对象

null 专门用来定义一个**空对象**（例如：`let a = null`）。

如果你想定义一个变量用来保存引用类型，但是还没想好放什么内容，这个时候，可以在初始化时将其设置为 null。

比如：

```
let myObj = null;
console.log(typeof myObj); // 打印结果：object
```

补充：

- Null 类型的值只有一个，就是 null。比如 `let a = null`。
- 使用 typeof 检查一个 null 值时，会返回 object。

undefined

case1：变量已声明，未赋值时

声明了一个变量，但没有**赋值**，此时它的值就是 `undefined`。举例：

```
let name;
console.log(name); // 打印结果：undefined
console.log(typeof name); // 打印结果：undefined
```

补充：

- Undefined 类型的值只有一个，就是 undefind。比如 `let a = undefined`。
- 使用 typeof 检查一个 undefined 值时，会返回 undefined。

case2：变量未声明（未定义）时

如果你从未声明一个变量，就去使用它，则会报错（这个大家都知道）；此时，如果用 `typeof` 检查这个变量时，会返回 `undefined`。举例：

```
console.log(typeof a); // undefined
console.log(a); // 打印结果：Uncaught ReferenceError: a is not defined
```

case3：函数无返回值时

如果一个函数没有返回值，那么，这个函数的返回值就是 undefined。

或者，也可以这样理解：在定义一个函数时，如果末尾没有 return 语句，那么，其实就是 `return undefined`。

举例：

```
function foo() {}

console.log(foo()); // 打印结果：undefined
```

case4：调用函数时，未传参

调用函数时，如果没有传参，那么，这个参数的值就是 undefined。

举例：

```
function foo(name) {
    console.log(name);
}

foo(); // 调用函数时，未传参。执行函数后的打印结果：undefined
```

实际开发中，如果调用函数时没有传参，我们可以给形参设置一个默认值：

```
function foo(name) {
    name = name || 'jike';
}

foo();
```

等学习了 ES6 之后，上方代码也可以这样写：

```
function foo(name = 'jike') {}

foo();
```

其他区别

null 和 undefined 有很大的相似性。看看 `null == undefined` 的结果为 `true` 也更加能说明这点。

但是 `null === undefined` 的结果是 false。

```
var a = null == undefined;
console.log(a);
```

它们虽然相似，但还是有区别的，其中一个区别是，和数字运算时：

- 10 + null 结果为 10。
- 10 + undefined 结果为 NaN。

规律总结：

- 任何数据类型和 undefined 运算都是 NaN;
- 任何值和 null 运算，null 可看做 0 运算。