Daily_Learning(2019-11-7)

Vue 中的 computed 和 watch 的区别在哪里

```
computed：计算属性

计算属性是由data中的已知值，得到的一个新值。
这个新值只会根据已知值的变化而变化，其他不相关的数据的变化不会影响该新值。
计算属性不在data中，计算属性新值的相关已知值在data中。
别人变化影响我自己。(有缓存)

watch：监听数据的变化

监听data中数据的变化
监听的数据就是data中的已知值
我的变化影响别人

1.watch擅长处理的场景：一个数据影响多个数据

2.computed擅长处理的场景：一个数据受多个数据影响
```

改造下面的代码，使之输出0 - 9，写出你能想到的所有解法。

```
for (var i = 0; i< 10; i++){
    setTimeout(() => {
        console.log(i);
    }, 1000)
}
```

方法一

原理：

- 利用 setTimeout 函数的第三个参数，会作为回调函数的第一个参数传入
- 利用 bind 函数部分执行的特性

代码 1：

```
for (var i = 0; i < 10; i++) {
  setTimeout(i => {
    console.log(i);
  }, 1000, i)
}
```

代码 2：

```
for (var i = 0; i < 10; i++) {
  setTimeout(console.log, 1000, i)
}
```

代码 3：

```
for (var i = 0; i < 10; i++) {
  setTimeout(console.log.bind(Object.create(null), i), 1000)
}
```

方法二

原理：

- 利用 let 变量的特性 — 在每一次 for 循环的过程中，let 声明的变量会在当前的块级作用域里面（for 循环的 body 体，也即两个花括号之间的内容区域）创建一个文法环境（Lexical Environment），该环境里面包括了当前 for 循环过程中的 i

代码 1：

```
for (let i = 0; i < 10; i++) {
  setTimeout(() => {
    console.log(i);
  }, 1000)
}
```

等价于

```
for (let i = 0; i < 10; i++) {
  let _i = i;// const _i = i;
  setTimeout(() => {
    console.log(_i);
  }, 1000)
}
```

方法三

原理：

- 利用函数自执行的方式，把当前 for 循环过程中的 i 传递进去，构建出块级作用域。IIFE 其实并不属于闭包的范畴。
- 利用其它方式构建出块级作用域

代码 1：

```
for (var i = 0; i < 10; i++) {
  (i => {
    setTimeout(() => {
      console.log(i);
    }, 1000)
  })(i)
}
```

代码 2：

```
for (var i = 0; i < 10; i++) {
```

```
  try {
    throw new Error(i);
  } catch ({
    message: i
  }) {
    setTimeout(() => {
      console.log(i);
    }, 1000)
  }
}
```