Deaily_Learning(2019-11-15)

箭头函数与普通函数（function）的区别是什么？构造函数（function）可以使用 new 生成实例，那么箭头函数可以吗？为什么？

箭头函数是普通函数的简写，可以更优雅的定义一个函数，和普通函数相比，有以下几点差异：

1、函数体内的 this 对象，就是定义时所在的对象，而不是使用时所在的对象。

2、不可以使用 arguments 对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替。

3、不可以使用 yield 命令，因此箭头函数不能用作 Generator 函数。

4、不可以使用 new 命令，因为：

- 没有自己的 this，无法调用 call，apply。
- 没有 prototype 属性 ，而 new 命令在执行时需要将构造函数的 prototype 赋值给新的对象的 __proto__

new 过程大致是这样的：

```
function newFunc(father, ...rest) {
  var result = {};
  result.__proto__ = father.prototype;
  var result2 = father.apply(result, rest);
  if (
    (typeof result2 === 'object' || typeof result2 === 'function') &&
    result2 !== null
  ) {
    return result2;
  }
  return result;
}
```

打印什么?

```
const person = {
  name: "Lydia",
  age: 21
}

const changeAge = (x = { ...person }) => x.age += 1
const changeAgeAndName = (x = { ...person }) => {
  x.age += 1
  x.name = "Sarah"
}

changeAge(person)
changeAgeAndName()

console.log(person)
```

- A: {name: ”Sarah”, age: 22}
- B: {name: ”Sarah”, age: 23}
- C: {name: ”Lydia”, age: 22}
- D: {name: ”Lydia”, age: 23}

无论是changeAge和changeAgeAndName功能有一个默认的参数，即一个*新*创建的对象{ ...person }。该对象具有该对象中所有键/值的副本person。

```
> person = {
    name: "Lydia",
    age: 21
  }
◂ ▸{name: "Lydia", age: 21}
> x = { ...person }
◂ ▸{name: "Lydia", age: 21}
> x == person
◂ false
```

首先，我们调用该changeAge函数并将person对象作为参数传递。此函数将age属性的值增加1. personis now { name: ”Lydia”, age: 22 }。

然后，我们调用该changeAgeAndName函数，但是不传递参数。相反，的值x等于一个*新的*对象：{ ...person }。由于它是一个新对象，因此不会影响该person对象的属性值。person仍然等于{ name: ”Lydia”, age: 22 }。

C