1.请指出以下代码的区别：function Person(){}、var person = Person()、var person = new Person()？

1)第一个是函数声明，创建一个新函数person
2)this --global object
3）第二个是用构造函数为对象Person创建新实例person，person的原型是Person,拥有其原型的所有方法

In var person=new person()

`new` does three main things:

1. `new` creates a new object. It's just a plain old, bog standard, nothing-in-it object. It looks like `{}`. Boring, I know, but it's very important.
2. The newly created object has it's prototype set to whatever the `Person`'s prototype is right now.
3. Finally the constructor function is called (the body of `Person`) with any references to `this` replaced with the object created in step 1.

```
function Person(name) {
    this.name = name;
}
Person.prototype.introduce = function() {
    console.log("Hi, my name is " + this.name);
}

var lucy = new Person('Lucy');
lucy.introduce(); // logs out: "Hi, my name is Lucy"
```

例子：
https://amyetheredge.com/interview/js/7.html

2. This keyword

```
var a=10;
        function test(){
            a=100;
            alert(a);
            alert(this.a);
            var a;
            alert(a);
        }
        test();
```

运行结果为：100 10 100

然后我把代码改成了这个：

```
var a=10;
        function test(){
            a=100;
            alert(a);
            alert(this.a);
        }
        test();
```

输出结果变成了 100 100

为什么会这样?

Answer: 第一个程序，经过变量提升（声明前置），var a被提升到第一行，a=100定义的是局部变量a, this指向window； 第二个程序，没有局部变量a, a和this.a都为100

延伸：key(构造函数实例化对象)

这个不仅是this的作用域，还涉及到function和 new function的区别

```
var a=10;
   function test(){
      a=5;
      alert(a);
      alert(this.a);
      var a;
      alert(this.a);
      alert(a);
   }
```

问：执行test()和new test() 返回值分别为啥?

答：返回结果,alert的内容：

test(): 5,10，10，5

new test():5,undefined,undefined,5

解释下：

在第一种情况 this指拥有test的对象，这儿是windows

第二种情况this指new创建的对象，因为未定义this.a，所以undefined

错题：

问题4：this在JavaScript中如何工作的

下面的代码会输出什么结果？给出你的答案。

```
var fullname = 'John Doe';
var obj = {
   fullname: 'Colin Ihrig',
   prop: {
      fullname: 'Aurelio De Rosa',
      getFullname: function() {
         return this.fullname;
      }
   }
};

console.log(obj.prop.getFullname());

var test = obj.prop.getFullname;

console.log(test());
```

回答

答案是Aurelio De Rosa和John Doe。原因是，在一个函数中，this的行为，取决于JavaScript函数的调用方式和定义方式，而不仅仅是看它如何被定义的。

在第一个 console.log()调用中，getFullname() 被调用作为obj.prop对象的函数。所以，上下文指的是后者，函数返回该对象的fullname。与此相反，当getFullname()被分配到test变量时，上下文指的是全局对象（window）。这是因为test是被隐式设置为全局对象的属性。出于这个原因，该函数返回window的fullname，即定义在第一行的那个值。

3.立即执行函数

6. 1) Explain why the following doesn't work as an IIFE: `function foo(){ }();`. (立即执行函数)

2) What needs to be changed to properly make it an IIFE?

Answer: 1) `function foo(){ }()`中 `function foo(){}`是函数声明， 以上代码等于

```
function foo(){ /* code */ }
(); // SyntaxError: Unexpected token )
```

2） `(function foo(){ })`括号里面是函数表达式， need to wrap the anonymous function with parenthesis, so the Javascript parser treats our anonymous function as a *function expression*.

A function expression is when you assign a function to a variable or property of an object. Anything that is a Javascript expression, including function expression, returns a value. 所以想调用函数， 后加（）可运行

4.Difference between: function Person(){}, var person = Person(), and var person = new Person()?（以上三種的不同）

```
function Person(){}; // 宣告一個方法
var person = Person(); // 執行 Person 並且結果賦予給 person
var person = new Person(); // 建立一個 Person 實體
```

這題有個重點就是在於「new」這個字，有「new」的宣告將會建立一個實體來自於 Person，如果 Person 帶有 prototype 則會帶入到 person 的 __proto__ 中，而 Person 方法將變為建構式（constructor）