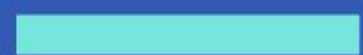


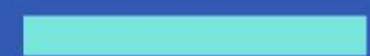
如何在面试中更好的发挥自己

F前端-董杨

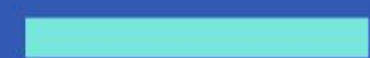
自我成长及面试分享



对于计算机和前端的思考



面试成败的因素

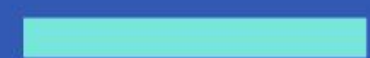




面试成败的因素

- 技术、学历、经验
- 沟通表达、抗压
- 基础与潜力
- 态度

来点技术——一道最普通的面试题



Bind的实现

```
Function.prototype.bind = function(oThis) {  
  if (typeof this !== 'function') {  
    // closest thing possible to the ECMAScript 5  
    // internal IsCallable function  
    throw new TypeError('Function.prototype.bind - what is trying to be bound is not callable');  
  }  
  
  var aArgs  = Array.prototype.slice.call(arguments, 1),  
      fToBind = this,  
      fNOP   = function() {},  
      fBound = function() {  
        // this instanceof fBound === true时,说明返回的fBound被当做new的构造函数调用  
        return fToBind.apply(this instanceof fBound  
                              ? this  
                              : oThis,  
                               // 获取调用时(fBound)的传参.bind 返回的函数入参往往是这么传递的  
                               aArgs.concat(Array.prototype.slice.call(arguments)));  
      };  
  
  // 维护原型关系  
  if (this.prototype) {  
    // 当执行Function.prototype.bind()时, this为Function.prototype  
    // this.prototype(即Function.prototype.prototype)为undefined  
    fNOP.prototype = this.prototype;  
  }  
  // 下行的代码使fBound.prototype是fNOP的实例,因此  
  // 返回的fBound若作为new的构造函数,new生成的新对象作为this传入fBound,新对象的__proto__就是fNOP的实例  
  fBound.prototype = new fNOP();  
  
  return fBound;  
};
```




Bind的实现

- This
- 闭包
- Instanceof原理
- 原型链
- 函数柯里化
- New的过程
- 继承



Bind的实现

```
> function Rectangle(length,width){  
    this.l = length  
    this.w = width  
}  
Rectangle.prototype.getArea = function(){  
    return this.l*this.w  
}  
function Square(length){  
    Rectangle.call(this,length,length)  
}  
Square.prototype = Object.create(Rectangle.prototype,{  
    constructor:{  
        value:Square,  
    }  
})  
})
```

```
< ▶ Rectangle {constructor: f}
```

```
> let square = new Square(3)
```

```
< undefined
```

```
> console.log(square.getArea())
```

```
9
```

```
< undefined
```

```
> square instanceof Square
```

```
< true
```

```
> square instanceof Rectangle
```

```
< true
```

19.2.3.2 Function.prototype.bind (*thisArg*, ...*args*)

When the **bind** method is called with argument *thisArg* and zero or more *args*, it performs the following steps:

1. Let *Target* be the **this** value.
2. If **IsCallable**(*Target*) is **false**, throw a **TypeError** exception.
3. Let *args* be a new (possibly empty) **List** consisting of all of the argument values provided after *thisArg* in order.
4. Let *F* be ? **BoundFunctionCreate**(*Target*, *thisArg*, *args*).
5. Let *targetHasLength* be ? **HasOwnProperty**(*Target*, "**length**").
6. If *targetHasLength* is **true**, then
 - a. Let *targetLen* be ? **Get**(*Target*, "**length**").
 - b. If **Type**(*targetLen*) is not **Number**, let *L* be 0.
 - c. Else,
 - i. Let *targetLen* be **ToInteger**(*targetLen*).
 - ii. Let *L* be the larger of 0 and the result of *targetLen* minus the number of elements of *args*.
7. Else let *L* be 0.
8. Perform ! **DefinePropertyOrThrow**(*F*, "**length**", **PropertyDescriptor** {[[**Value**]]: *L*, [[**Writable**]]: **false**, [[**Enumerable**]]: **false**, [[**Configurable**]]: **true**}).
9. Let *targetName* be ? **Get**(*Target*, "**name**").
10. If **Type**(*targetName*) is not **String**, let *targetName* be the empty string.
11. Perform **SetFunctionName**(*F*, *targetName*, "**bound**").
12. Return *F*.

NOTE 1 Function objects created using **Function.prototype.bind** are exotic objects. They also do not have a **prototype** property.

NOTE 2 If *Target* is an arrow function or a **bound function** then the *thisArg* passed to this method will not be used by subsequent calls to *F*.

THANKS.

真诚欢迎大家
加入我们团队

 ByteDance 字节跳动