

前端路漫漫，这是最好的结束，也是全新的开始。京程一灯永远是您的后盾，无论何时需要帮助，我们永远都在，无论你在哪，如果需要任何帮助请随时联系我们，祝好~



## 《京程一灯》精英班第一讲试卷 姓名：

请先盖住答案，在右侧空白处作答已加深印象。

- 1.请写出弹出值，并解释为什么。(5分)

```
+function() {
  alert(a)
  a();
  var a = function() {
    console.log(1);
  }
  function a() {
    console.log(2);
  }
  alert(a)
  a();
  var c = d = a;
}();
alert(d);
alert(c);
```



- 答：

(具体代码答案大家可以自行运行下) 🍄 本题考点分为如下：

- 1.IIFE 第一行考点用了一个+号，其实就是把函数变成函数表达式，直接执行function() {}();会报错。常用的写法是(function(){} )(),此时其实创建了闭包。
- 2.变量与函数提升，但此时函数的名字也是a变量也是a,所以会造成 function a(){}, var a。此时var a因为未被定义所以被忽略了。所以顶部的输出值是2和2.接下来输出是1和1，是因为函数的提升要比变量提升的更前。局面就是var a,function a(){},a=function(){};alert(a)
- 3.作用域和连等问题 此时的var c = d = a。实际是 d = a,var c=d;所以c是is not undefined，但是要千万注意是不是严格模式下d报错。扩展的题为var a = {n:1}; var b = a; a.x = a = {n:2}; alert(a.x);// --> undefined alert(b.x);// --> {n:2}
- 4.最后大家要注意js里的分号问题哈。  
var s = { a:1 } (function(){} ) s后面没分号就报错了。上面都总结完了做做第五题吧~
- 5.function test(){console.log(1)}  
function init(){ if(false){function test(){console.log(2)}} ; test();  
init() ;

- 2.请写出如下输出值，并写出把注释掉的代码取消注释的值，并解释为什么(8分)

```

this.a = 20;
var test = {
  a: 40,
  init:()=>{
    console.log(this.a);
    function go() {
      // this.a = 60;
      console.log(this.a);
    }
    go.prototype.a = 50;
    return go;
  }
};
//var p = test.init();
//p();
new(test.init())();

```

- 答：

(去掉代码注释后的答案)：

(具体代码答案大家可以自行运行下) 🐼 本题考点分为如下：

1.this指向问题。万万需要注意this的值是在执行的时候才能确认，定义的时候不能确认(核心原理是AO) 用老袁的话呢就是this是个赖皮虫谁调用我指谁，没人管就指window(注意严格模式)。还有就是this能被修改，怎么改呢call apply bind 还有es6的箭头函数。很多面试官会考大家手写bind,注意bind有软绑定和硬绑定。还要注意bind对new无效，手写bind函数要注意完整性。

```

Function.prototype.bind = function(oThis) {
  if (typeof this !== "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() {
        return fToBind.apply(
          this instanceof fBound ? this : oThis,
          aArgs.concat(Array.prototype.slice.call(arguments))
        );
      };
  if (this.prototype) {
    fNOP.prototype = this.prototype;
  }
  fBound.prototype = new fNOP();
  return fBound;
};

```

2.原型链的基础。当一个函数做当做类使用的时候，原型链里this对属性的赋值优先级要低于构造函数内部的指向。

3.在面试的时候经常面试官会迷惑你this的情况，要分析最后this再没执行前的作用域到底在哪。

- 3.请写出如下点击li的输出值，并用三种办法正确输出li里的数字。(12分)

```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
  <li>6</li>
</ul>
<script type="text/javascript">
var list_li = document.getElementsByTagName("li");
for (var i = 0; i < list_li.length; i++) {
  list_li[i].onclick = function() {
    console.log(i);
  }
}
</script>
```

答：

(具体代码答案大家可以自行运行下) 🍄 本题考点分为如下：

1.闭包解决，闭包就是能够读取其他函数内部变量的函数。由于在javascript中，只有函数内部的子函数才能读取局部变量，所以闭包可以理解成“定义在一个函数内部的函数”。在本质上，闭包是将函数内部和函数外部连接起来的桥梁，用老袁的话是闭包是拿到了本不该是你的东西，不用盲目的害怕闭包会造成内存泄露，用完=null就完事了。还有要一阵见血的描述闭包的作用：函数作为返回值、函数作为参数传递、保护变量。

2.块级作用域和函数级作用域，ES5没有块级作用域，只有全局作用域和函数作用域，全局作用域就是最外层的作用域，ES6新增了块级作用域所以使用let即可，块级作用域要记得TDZ哈。


3.最后我们巧妙的运用了this解决(this.innerHTML),在面试的时候经常面试官会迷惑你this的情况，要分析最后this再没执行前的作用域到底在哪。

4.其实这道题可以用AO/VO/GO去解释，这样才能让整个面试的过程中面试官对你的基础知识觉得掌握的更加深入，如果你总结不好的话可以在毕业群讨论哈 🤖

- 04. 写出输出值，并解释为什么。(5分)

```
function test(m) {
  m = {v:5}
}
var m = {k: 30};
test(m);
alert(m.v);
```

- 答：


 本题考点分为如下：

1. 值类型 vs 引用类型。值类型变量包括 Boolean、String、Number、Undefined、Null，引用类型包括了 Object 类的所有，如 Date、Array、Function 等。在参数传递方式上，值类型是按值传递，引用类型是按共享传递。

2. 函数的参数是按值传递，但是该值引用的是外部 m 值得地址。所以首先函数 test(m) 和外边的 m 没有任何关系，然后值引用外部值得地所以 m.v=5 该题即是 5，但是重写了 m，内部的 m 引用了新地址，所以是 undefined。

3. 我们来深入思考一下为什么 JS 要分为按值传递和按址（引用）传递呢。JS 中这种设计的原因是：按值传递的类型，复制一份存入栈内存，这类类型一般不占用太多内存，而且按值传递保证了其访问速度。按共享传递的类型，是复制其引用，而不是整个复制其值，保证过大的对象等不会因为不停复制内容而造成内存的浪费。



4. V8 中的隐藏类其实就是按址引用。怎么确认呢？node --allow-natives-syntax test.js

翻翻我们讲的前端工程师必会的 V8 。

- 5. 请在下面写出 JavaScript 面向对象编程的混合式继承。并写出 ES6 版本的继承。

要求：汽车是父类，Cruze 是子类。父类有颜色、价格属性，有售卖的方法。Cruze 子类实现父类颜色是红色，价格是 140000，售卖方法实现输出如下语句：将红色的 Cruze 买给了小王价格是 14 万。(20分)

- 答：

 这道题的代码，必须要背着写下来！！！！什么？不会？挖个坑，跳进去，然后埋土  本题考点分为如下：

1. 原型和原型链。先来说一个 \_\_proto\_\_  图已经放到答案的最后了。

- (1) 所有的引用类型（数组、对象、函数），都具有对象特性，即可自由扩展属性（null 除外）
- (2) 所有的引用类型（数组、对象、函数）都有一个 \_\_proto\_\_ 属性，属性值是一个普通的对象

- (3) 所有的函数，都有一个prototype属性，属性值也是一个普通的对象
- (4) 所有引用类型（数组、对象、函数）\_\_proto\_\_属性值指向它的构造函数的prototype属性值
- (5) \_\_proto\_\_（隐式原型）与prototype（显式原型）

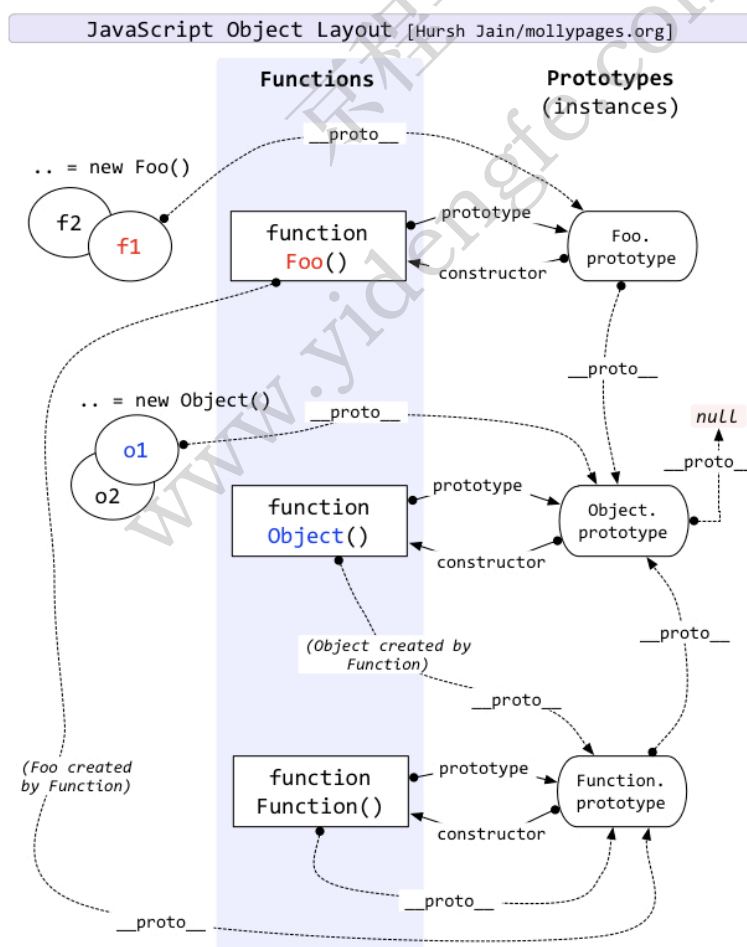
2.call和apply的区别。改变对象的执行上下文，区别就是根据你要传入的参数来做选择，不需要传参或者只有1个参数的时候，用call，当要传入多个对象时，用apply。

3.JavaScript constructor 和 Object.create()。constructor 属性返回对创建此对象的数组函数的引用。Object.create()方法会使用指定的原型对象及其属性去创建一个新的对象。（副本！！）

4.ES6的新语法Class（包括TypeScript），这里我要问你一堆面向对象的东西了。什么是构造函数和析构函数、访问控制（可见性）、对象继承、范围解析操作符（::）、Static（静态）关键字、抽象类、对象接口、重载。请问如何实现JavaScript的重载。ES6很简单直接执行super.同名方法()，然后再执行自己的方法。ES5呢？（大家翻一下预读班课程，请默写）

5.Object.prototype.a = 1; function Foo() {} var f = new Foo(); console.log(f.a); //1

在f.\_\_proto\_\_中没有找到a，那么就继续去f.\_\_proto\_\_.\_\_proto\_\_中寻找，f.\_\_proto\_\_即Foo.prototype，没有找到a，继续往上找f.\_\_proto\_\_.\_\_proto\_\_即Foo.prototype.\_\_proto\_\_。Foo.prototype就是一个普通的对象，因此Foo.prototype.\_\_proto\_\_就是Object.prototype，在这里可以找到a，这就是原型链。



🔪 小试牛刀，请将答案书写📝于下方。

- `Object.prototype.a = 'a';`  
`Function.prototype.a = 'a1';`  
`function Person(){};`  
`var yideng = new Person();`  
`console.log('p.a: ' + yideng.a);`  
`console.log(1..a);`  
`console.log(1.a);`
- `function yideng(){};`  
`yideng.__proto__.__proto__.constructor.constructor.constructor`
- 6.请写出你了解的ES6元编程 (10分)
- 答 :

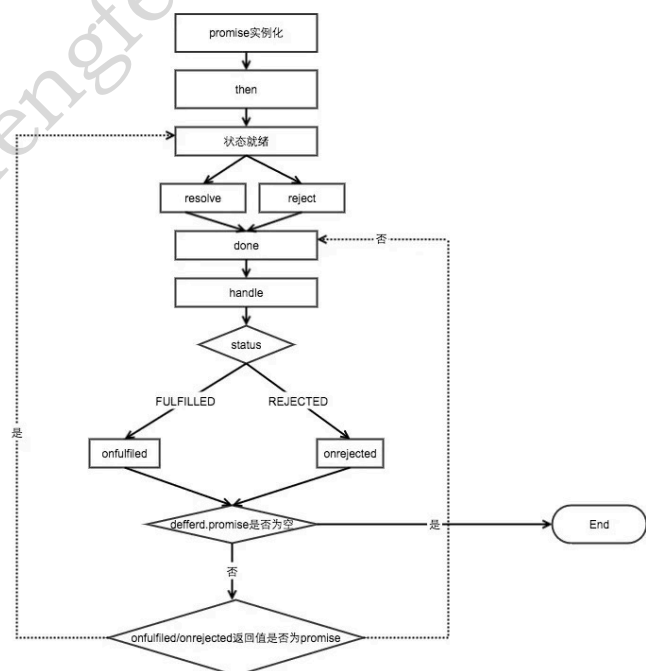
🐻元编程应用于前端的方方面面、比如Vue3新的源码、比如我们学习依赖注入的Node框架、强制开启tco.具体语法包括Proxy、Reflect、Symbol.iterator、Symbol.toPrimitive、TCO\_ENABLED = true、reflect-metadata，代码在QQ群里哈 🚲

- 7.请用ES5实现Promise A+规范代码，同时你对Promise的事物有了解么？(15分)
- 答 :

🐻1.Promise A+代码见群资料第一周代码，流程图见右侧流程，事物promise-transaction可以实现你的需求。

2.巩固一下如下这段代码？

```
let a = 0;
let yideng = async () => {
  a = a + await 10;
  console.log(a)
}
yideng();
console.log(++a);
```



- 8.请问点击<button id="test"></button>会有反应么？为什么？能解决么？ (5分)
- ```
$('#test').click(function(argument) {
  console.log(1);
});
```



```

setTimeout(function() {
  console.log(2);
}, 0);
while (true) {
  console.log(3);
}

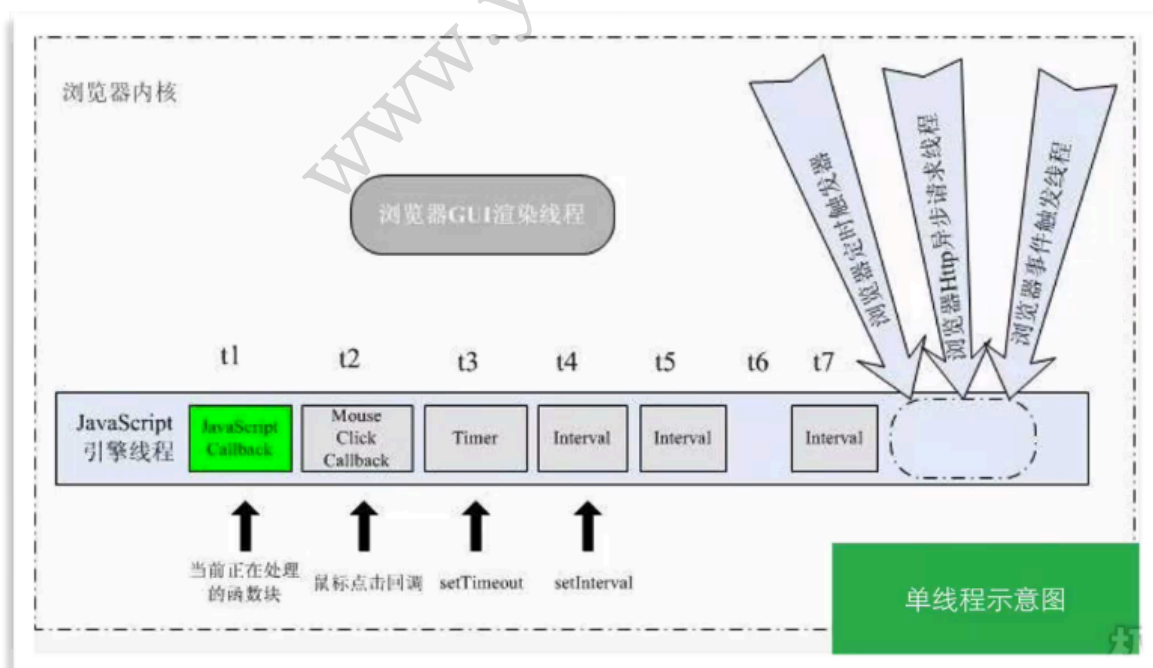
```

- 答：

🐻 这道题不要运行。。。但是答案肯定是可以的。🍄 本题考点分为如下

1.同步VS异步。JS 需要异步的根本原因是 JS 是单线程运行的，即在同一时间只能做一件事，不能“一心二用”。(下图有说明)所以Settimeout(function(){alert(1)};0);alert(2)的结果应该是 2 和 1。那么那些场景会让代码进入异步队列呢。setTimeout setInterval Ajax 事件绑定



2.线程。这个问题其实可以直接用Concurrent.Thread.js处理，这个库呢其实就是用异步来模拟了线程。那么我们也可以直接使用WebWorker来实现。说到了WebWorker就一定要想到ES2017 引入SharedArrayBuffer，允许 Worker 线程与主线程共享同一块内存。SharedArrayBuffer的 API 与 ArrayBuffer一模一样，唯一的区别是后者无法共享。多线程共享内存，最大的问题就是如何防止两个线程同时修改某个地址，或者说，当一个线程修改共享内存以后，必须有一个机制让其他线程同步。SharedArrayBuffer API 提供Atomics对象，保证所有共享内存的操作都是“原子性”的，并且可以在所有线程内同步。





- 9.请用一句话遍历变量a。(禁止用for已知var a = "abc")(10分)

- 答：

 这道题其实不仅仅有三种解法。 本题考点分为如下

1.对ES6的理解。Array.from(a), [...new Set(a)] (画个重点那么你能解释下Set的实现原理么?) 其实ES6本身就是ES5的很多语法糖,为什么这么说呢。

```
class Test {
  constructor(args) {}
  do(){
    console.log('test');
  }
}
Test.prototype.do = function() {
  console.log('test2');
};
const instance = new Test();
instance.do();
test2
```

2.对ES5的灵活运用。Array.prototype.slice.call 等等都可以。

3.直接用a.split( "" )这个其实也就是个3毛钱的技术。

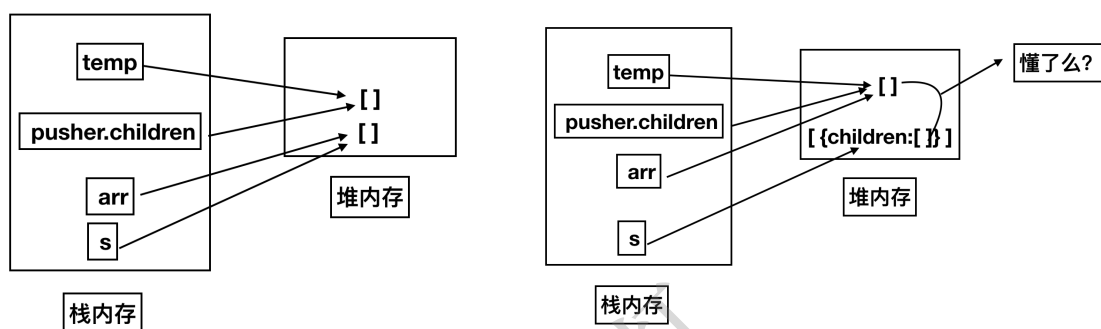
10.请写出如下输出值,并解释为什么。(12分)

```
var s = [];
var arr = s;
for (var i = 0; i < 3; i++) {
  var pusher = {
    value: "item"+i
  },tmp;
  if (i !== 2) {
    tmp = []
    pusher.children = tmp
  }
  arr.push(pusher);
  arr = tmp;
}
```

```
console.log(s[0]);
```

- 答：

🐻 这道题可能稍微有点难，考的是JS的模拟指针移动问题。🍄 本题考点分为如下



- 【附加题】.请描述你理解的函数式编程，并书写如下代码结果，最后请问如何具体的将函数式编程应用到你的项目（10分）

```
var Container = function(x) {
  this.__value = x;
}
Container.of = x => new Container(x);
Container.prototype.map = function(f){
  return Container.of(f(this.__value))
}
Container.of(3)
  .map(x => x + 1)
  .map(x => 'Result is ' + x);
```

- 答：

🐻 函数式编程是很复杂的话题但是我教给大家的东西足够了。🍄 本题考点分为如下

1.基础函子的概念。首先创建一个基础的容器Container，当一个容器具有map方法的时候她也可以叫一个函子。of是为了解决函数式编程更不像面向对象编程。接下来我们传入了新的3，x+1等，每一次都是一个变形关系。产生一个新的函子，也就是一个新的范畴。函数式编程讲的是函数要切忌纯！同时柯里化（curry）？代码组合（compose）？等等你是否还有印象。

2.项目实战。我去掉了一些模块化的判断。

```
(function() {
```

```

var _ = function(obj) {
  if (obj instanceof _) return obj;

  if (!(this instanceof _)) return new _(obj);
};

_.VERSION = '1.0.0';

_.isNaN = function(obj) {};

// 别忘了这里还有一个非常关键的mixin

return _;
}.call(this));

```

其实JS的基础知识还不止这些，比如ECMAScript 中定义了 6 种原始类型：Boolean、String、Number、Null、Undefined、Symbol（ES6 新定义）注意：原始类型不包含 Object。看了这么多希望大家不要被一面这些小把戏所难住有空多刷刷题。【终】

```

var length = 10;
function fn() {
  console.log(this.length);
}
var yideng = {
  length: 5,
  method: function(fn) {
    fn();
    arguments[0]();
  }
};
yideng.method(fn, 1);

```

```

var num = 1;
function yideng() {
  "use strict";
  console.log(this.num++);
}
function yideng2() {
  console.log(++this.num);
}
(function() {
  "use strict";
  yideng2();
})();
yideng();

```

```

function C1(name) {
  if (name) this.name = name;
}

function C2(name) {
  this.name = name;
}

function C3(name) {
  this.name = name || "fe";
}

C1.prototype.name = "yideng";
C2.prototype.name = "lao";
C3.prototype.name = "yuan";
console.log(new C1().name +
  new C2().name +
  new C3().name);

```

```

const pro = new Promise((resolve, reject) => {
  const innerpro = new
  Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(1);
    })
    console.log(2);
    resolve(3);
  })
  innerpro.then(res => console.log(res));
  resolve(4);
  console.log("yideng");
})
pro.then(res => console.log(res));
console.log("end");

```

```

var A = function() {};
A.prototype = {};
var a = new A();
A.prototype = {};
var b = new A();
console.log(a instanceof A);
console.log(b instanceof A);

```