《ES6 快速入门》试题

备注：本套试题去面覆盖 ES6 中最常用的特性，对于一些冷门偏僻的知识不会考查，请您仔细答题。

一、 客观题（单选、多选或判断）

1. 请问以下代码将会输出什么内容？（）

```
let a = 10;
let a = 1;
console.log(a);
```

A 报错

B 输出 1

答案：A

解析：let 关键不允许在同一个作用域里面出现两个相同的变量。

2. 请问以下代码将会输出什么内容？（）

```
let [a, ...b] = [1, 2, 3, 4];
console.log(a);
console.log(b);
```

A 输出 1 和 4

B 输出 1 和[2,3,4]

答案：B

解析：解构赋值是根据模式匹配的，扩展运算符匹配成数组，故变量 b 的值是数组 [2,3,4]

3. 请问以下代码将会输出什么内容？（）

```
let [a, b] = [1];
console.log(a);
console.log(b);
```

A 抛出异常

B 输出 1 和 undefined

C 输出 1 和 1

答案：B

解析：解构赋值根据模式匹配，匹配不成功的变量会被赋值成 undefined。

4. 请问以下代码将会输出什么内容？（）

```
let [a, [b], c] = [1, [2, 3], 4];
console.log(a);
console.log(b);
console.log(c);
```

A 匹配失败，抛出异常

B 分别输出 1，2，3

C 分别输出 1，2，4

答案：C

解析：解构赋值根据模式匹配，如果能匹配上一部分也可以解构成功。

5. var {a,b,c} = { “c”:10, ” b”:9, ” a”:8 }，a、b、c 的值分别是：（）

A 10 9 8

B 8 9 10

C undefined 9 undefined

D null 9 null

答案：B

解析：对象的解构赋值不会受到属性的排列次序影响。

6. 关于关键字 const，下列说法错误的是：（）

A 用于声明常量，声明后不可修改

B 不会发生变量提升现象

C 不能重复声明同一个变量

D 可以先声明而不赋值

答案：D

解析：声明后必须赋值，否则会抛异常。

7. 请问以下代码会输出什么内容？（）

console.log(`${1+1}`);

A 输出${1+1}

B 输出 2

C 输出 1+1

答案：B

8. 关于模板字符串，下列说法不正确的是：（）

A 使用反引号标识

B 插入变量的时候使用${ }

C 所有的空格和缩进都会被保留在输出中

D ${ }中的表达式不能是函数的调用

答案：D

解析：${ }中可以放任意的 JavaScript 表达式，包括运算表达式、对象属性、函数调用等。

9. 请问以下代码会输出什么内容？（）

```
var arr = [];
for (var i = 0; i < 10; i++) {
  arr[i] = function () {
    console.log(i);
  }
}
arr[5]() ;
```

A 抛出异常

B 输出 undefined

C 输出 5

D 输出 10

答案：D

解析：由于 var 的特性，函数内部所有的 i 都会被最后一次的计数器值所覆盖，i 的最后一个值为 10。

10. 请问以下代码会输出什么内容？（）

```
var arr = [];
for (let i = 0; i < 10; i++) {
    arr[i] = function () {
        console.log(i);
    }
}
arr[5]()
```

A 抛出异常

B 输出 undefined

C 输出 5

D 输出 10

答案：C

解析：由于 let 的特性，每一次循环都会创建一个全新的计数器绑定，所以第 5 个函数对象内部的 i 会停留在当时的计数器值上面，也就是 5。

11. [1,2,3].map(x => x + 1) 的输出结果是:

A [1, 2, 3]

B [1, 3, 5]

C [2, 3, 4]

答案：C

12. [1,2,3].fill(4)的结果是：（）

A [4]

B [1,2,3,4]

C [4,1,2,3]

D [4,4,4]

答案：D

解析：fill 函数的作用是填充，所以会把原数组的每个元素填充成指定的参数。

13. 数组的扩展中，不属于用于数组遍历的函数的是：（）

A keys( )

B entries( )

C values( )

D find( )

答案：D

解析：find 函数用于找出数组中符合条件的第一个元素，并不是用于遍历数组。

14. 【多选】以下哪些类型是 ES6 中的原始（primitive）数据类型？（）

A.undefined

B.null

C.Boolean

D.String

E.Number

F.Object

G.Symbol

答案：ABCDEFG

解释：全部都是，Symbol 是 ES6 引入的一种新的原始（primitive）类型。

15. 请问以下代码会输出什么内容？（）

let s1 = Symbol('foo');

let s2 = Symbol('foo');

console.log(s1===s2);

A fale

B true

答案：A

解释：虽然两次用的构造参数相同，但并不是同一个 Symbol 实例。

16. 以下说法正确的是？（）

A 使用 Symbol 值作为属性名时，对应的属性会成为私有属性

B 使用 Symbol 值作为属性名时，对应的属性还是公开属性，只是不能用 for...of 遍历

答案：B

17. 下面运算结果，结果为 true 的是（）

A Symbol.for('name') == Symbol.for('name')

B Symbol('name') == Symbol.for('name')

C Symbol('name') == Symbol('name')

D Symbol.for('name') == Symbol('name')

答案：A

解析：Symbol.for 函数会根据参数名，去全局环境中搜索是否有以该参数为名的 symbol 值，有就返回它，没有就以该参数名来创建一个新的 symbol 值，并登记在全局环境中，而 Symbol 每次都会创建一个独一无二的值，不会登记在全局环境中。

18. 请问以下代码将会输出哪种结果？（）

```
class Animal {
    constructor() {
```

```
            this.type = "animal";
        }
        say(val) {
            setTimeout(function () {
              console.log(this); //window
              console.log(this.type + " says " + val);
            }, 1000)
        }
    }
    var animal = new Animal();
    animal.say("hi");
```

A 输出 animal says hi

B 输出 undefined says hi

答案：B

解释：因为 setTimeout 的执行上下文为 window 对象，回调函数内部的 this 指向 window，而 window 上并不存在 type 属性，故输出 undefined says hi

19. 请问以下代码将会输出哪种结果？（）

```
class Animal {
  constructor() {
        this.type = "animal";
  }
  say(val) {
    setTimeout(() => {
      console.log(this); //Animal
      console.log(this.type + ' says ' + val);
    }, 1000)
 }
```

```
}
var animal = new Animal();
animal.say("hi"); //animal says hi
```

A 输出 animal says hi

B 输出 undefined says hi

答案：A

解释：ES6 新增的箭头函数会保持词法作用域，所以这里 setTimeout 的回调函数中的 this 会指向当前实例，故输出 animal says hi

20. 请问以下代码将会输入什么结果？（）

```
const set = new Set([1,2,3,4,4])
console.log([...set] )
```

A 抛出异常

B 输出[1,2,3,4]

C 输出 Object

D 输出 Array

答案：B

解析：Set 会自动排除重复的元素，扩展运算符会在内部隐式调用 for…of 循环展开数组，故[…set]最终会得到排重之后的[1,2,3,4]。

21. 请问以下代码将会输出什么结果？（）

```
var str="abstract";
console.log([...str]);
```

A abstract

B abstrc

C ["a","b","s","t","r","c"]

D ["a", "b", "s", "t", "r", "a", "c", "t"]

答案：D

解析：扩展运算符会解开字符串，变成一个一个的字母，最终就得到了一个数组["a", "b", "s", "t", "r", "a", "c", "t"]

22. 请问以下代码将会输出什么结果？（）

```
var str="abstract";
console.log(new Set([...str]).size);
```

A 6

B 8

答案：A

解析：扩展运算符会解开字符串，变成一个一个的字母，集合会把重复的字母自动排除掉，所以最终 Set 里面有 6 个元素。

23. 请问对于以下代码，应该如何获取键 a 对应的值？（）

```
var map = new Map();
map.set('a', 'apple');
```

A、map.a

B、map.a.value

C、map.value('a')

D、map.get('a')

答案：D

24. 关于 Proxy 代理，下面说法错误的是：（）

A 可以理解成在目标对象之前，架设一层“拦截”

B Proxy 的 get 方法用于拦截某个属性的读取操作。

C Proxy 的 set 方法用于拦截对对象的写操作。

D 一旦对象设置 Proxy 代理后不可取消，所以要谨慎操作

答案：D

解析：可以用 Proxy.revocable( )来取消代理，并不是不可以取消的。

25. 关于 for...of 的简述，说法错误的是：（）

A 它可以遍历所有具有 iterator 接口的数据结构

B 不可以用 break 来终止循环

C 使用 continue 可以跳过当前循环

D 可以遍历 DOM list 对象

答案：B

解析：for...of 可以用 break 来终止循环，而传统的 forEach 则不可以用 break 终止循环，这正是 for...of 相对 forEach 的优势。

26. 下列数据结构中，不能被 for...of 遍历的是：（）

A Array 数组

B Object 对象

C String 字符串

D Set 结构

答案：B

解析：只有该数据结构实现了 Iterator 遍历器接口才可以被 for...of 遍历，而数组，字符串，Set 和 Map 结构正式这样的可遍历对象。而普通的 Object 对象并没有实现 Iterator 遍历器接口。

27. 关于 Iterator 遍历器的说法，错误的是：（）

A next( )方法是 Iterator 遍历器的核心

B 当 next( )返回对象的 done 属性为 fasle，遍历结束

C 具有 Iterator 接口的对象视为可遍历对象

D 可以自定义一个可遍历对象和其遍历行为

答案：B

解析：当 next( )返回对象的 done 属性为 fasle，表示遍历未结束，done 属性为 true 时，表示遍历结束。

28. 关于 WeakSet 结构，说法错误的是：（）

A 与 Set 结构一样，成员值都是唯一

B 成员值必须是对象

C WeakSet 中的对象都是弱引用

D 可以 forEach( )方法实现遍历

答案：D

解析：WeakSet 结构是不可遍历的，所以它不存在 forEach 方法，以及 keys()、values()、entries()方法，这是它和 Set 结构不同处之一。

29. 关于 Map 结构的介绍，下面说法错误的是：（）

A 是键值对的集合

B 创建实例需要使用 new 关键字

C Map 结构的键名必须是引用类型

D Map 结构是可遍历的

答案：C

解析：键名可以是任何数据类型，这是 Map 结构的最大特性，也是 Map 结构和传统对象 Object 最大的区别。

30. 下列 Map 结构的键名数据类型，描述错误的是：（）

A 键名可以是数组类型的值

B 键名可以是 Symbol 类型的值

C 键名值可以是 null

D 键名值不可以为 undefined

答案：D

解析：undefined 也可以做为 Map 结构的键名。

31. 关于 WeakMap 结构，下列说法错误的是：（）

A 创建实例需要使用 new 关键字

B 键名可以是任何类型的值

C WeakMap 结构不支持 clear 方法

D WeakMap 结构不可遍历

答案：B

解析：WeakMap 结构的键名必须是引用类型的值，也是它和 Map 最大不同之处。

32. 关于 Promise 对象的状态，下列说法错误的是：（）

A 三种状态分别是：pending 初始状态、fulfilled 成功、rejected 失败

B pending 初始状态可以状变成 fulfilled 成功

C rejected 失败不可以状变成 pending 初始状态

D rejected 失败可以状变成 fulfilled 成功

答案：D

解析：A、B、C 的说法都是正确的，rejected 失败和 fulfilled 成功之间不能相互转换，故 D 选项是错误的。

33. 下面关于类 class 的描述，错误的是：（）

A、 JavaScript 的类 class 本质上是基于原型 prototype 的实现方式做了进一步的封装

B、 constructor 构造方法是必须的

C、 如果类的 constructor 构造方法有多个，后者会覆盖前者

D、 类的静态方法可以通过类名调用，不需要实例化

答案：C

解析：同一个类的 constructor 构造方法只能有一个，否则程序会报错。

34. JavaScript 中类的继承使用的关键字是：（）

A extends

B inherit

C extend

D base

答案：A

解析：extends 才是 JavaScript 中类的继承关键字，其他的选项都不是。

35. 在类的继承中，关于 super 的说法错误的是：（）

A、 在子类的构造函数，必须先调用 super( )

B、 super 相当于子类的引用

C、 先调用 super( )，才可以使用 this

D、 super( )相当于父类构造函数的调用

答案：B

解析：super 是父类的引用，我们可以通过 super 来调用父类的方法和属性。

二、 主观题（简答或描述）

1. 请问以下代码将会输出什么结果，请解释原因。

```
const promise = new Promise((resolve, reject) => {
  console.log(1)
  resolve()
  console.log(2)
})
promise.then(() => {
  console.log(3)
})
console.log(4)
```

输出结果为 1 2 4 3，Promise 构造函数同步执行， then 中的函数异步执行的。

2. 请编写一个 Generator 的实例，并给出解释。

```
function* helloWorldGenerator() {
  yield 'hello';
  yield 'world';
  return 'ending';
```

```
}

var hw = helloWorldGenerator();
hw.next();//输出 { value: 'hello', done: false }
hw.next();//输出{ value: 'world', done: false }
hw.next();//输出{ value: ' ending ', done: true }
hw.next();
```

这段代码演示了 Generator 函数最基本的语法，hw.next()将会依次输出 yeild 对应的值。

3. 请编写一段代码说明 ES6 中 class/extends/super 的用法

```
class Animal {
  constructor() {
    this.type = 'animal'
  }
  says(say) {
    console.log(this.type + 'says' + say)
  }
}
let animal = new Animal()
animal.says('hello') // animal says hello

class Cat extends Animal {
  constructor() {
    super()
    this.type = 'cat'
  }
}
```

```
let cat = new Cat()
cat.says('hello') // cat says hell
```

4. 请编写一个案例演示 Decorator 修饰 Class 的基本用法

```
@ testable
class MyTestableClass {
  // ...
}

function testable(target) {
  target.isTestable = true;
}

MyTestableClass.isTestable // true
```

以上代码用装饰器的方式动态在 MyTestableClass 上增加了一个新的属性 isTestable。

5. 编写一个例子说明 exports 和 imports 的用法。

```
// a.js 中的内容
export let firstName = 'xiaofei';
export let lastName = 'Zhang';

//b.js 中的内容
import { firstName,lastName} from './a.js';
console.log(firstName);
console.log(lastName);
```

演示了最基本的模块导出和导入用法。