# 选择题

**第一题：**A

(function(x, f = () => x) {

var x;

var y = x;

x = 2;

return [x, y, f()];

})(1)

A、 [2, 1, 1]

B、 [2, undefined, 1]

C、 [2, 1, 2]

D、 [2, undefined, 2]

解析：本题主要考察的知识点是1、参数值与函数体内定义的重名变量的优先级；2、ES6的默认参数；

3、箭头函数。 在本题中，先执行x的定义，然后函数参数x=1，接着是y = x = 1,接着再x = 2，

第三个是执行f函数，箭头函数如果只是表达式，那么等价于return 表达式，由于箭头函数的作用域

等于定义时的作用域，那么函数定义时x=1，所以最后的return x 等价于 return 1

**第二题：**B

(function() {

return [

(() => this.x).bind({ x: 'inner' })(),

(() => this.x)()

]

}).call({ x: 'outer' });

A、 ['inner', 'outer']

B、 ['outer', 'outer']

C、 [undefined, undefined]

D、 Error

解析：本题主要考察的是箭头函数的作用域问题，箭头函数的作用域等于定义时的作用域，所以通过

bind设置的this是无效的。那么结果就显而易见了。

**第三题：**B

let x, { x: y = 1 } = { x }; y;

A、 undefined

B、 1

C、 { x: 1 }

D、 Error

解析：本题主要考察的是对象赋值，先定义x，然后在赋值的时候会执行一次y=1，最后返回y的值。

**第四题:**A

(function() {

let f = this ? class g { } : class h { };

return [

typeof f,

typeof h

];

})();

A、 ["function", "undefined"]

B、 ["function", "function"]

C、 ["undefined", "undefined"]

D、 Error

解析：本题主要考察定义函数变量时，命名函数的名称作用域问题。在定义函数变量时，函数名称只能

在函数体中生效。

**第五题：**B

(typeof (new (class { class () {} })))

A、 "function"

B、 "object"

C、 "undefined"

D、 Error

解析：本题主要考察对象的类型，和原型方法。该提可以分解如下：

// 定义包含class原型方法的类。

var Test = class{

class(){}

};

var test = new Test(); //定义类的实例

typeof test; //出结果

**第六题：**C

typeof (new (class F extends (String, Array) { })).substring

A、 "function"

B、 "object"

C、 "undefined"

D、 Error

解析：本题主要考察ES6中class的继承，以及表达式的返回值和undefined的类型。题目其实可以按

照如下方式分解：

//由于JS的class没有多继承的概念，所以括号被当做表达式来看

(String, Array) //Array,返回最后一个值

(class F extends Array); //class F继承成Array

(new (class F extends Array)); //创建一个F的实例

(new (class F extends (String, Array) { })).substring; //取实例的

substring方法，由于没有继承String，Array没有substring方法，那么返回值为undefined

typeof (new (class F extends (String, Array) { })).substring; //对

undefined取typeof

**第七题：**B

[...[...'...']].length

A、 1

B、 3

C、 6

D、 Error

解析：本题主要考察的是扩展运算符...的作用。扩展运算符是将后面的对象转换为数组，具体用法

是：

[...<数据>] 比如 [...'abc']等价于["a", "b", "c"]

**第八题：**D

typeof (function\* f() { yield f })().next().next()

A、 "function"

B、 "generator"

C、 "object"

D、 Error

解析：本题主要考察ES6的生成器。题目可以如下分解：

function\* f() { yield f }; //定义一个生成器

var g = f(); //执行生成器

var temp = g.next(); //返回第一次yield的值

console.log(temp); //测试，查看temp，其实是一个object

temp.next()；//对对象调用next方法，无效

**第九题：**D

typeof (new class f() { [f]() { }, f: { } })[`${f}`]

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

解析：本题主要考察ES6的class，以及动态属性和模板字符串等。 实际上这个题动态属性和模板字符

串都是烟雾弹，在执行new class f()的时候，就已经有语法错误了。

**第十题：**B

typeof `${{Object}}`.prototype

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

解析：本题考察的知识点相对单一，就是模板字符串。分解如下：

var o = {Object},

str = `${o}`;

typeof str.prototype;

**第十一题**D

((...x, xs)=>x)(1,2,3)

A、 1

B、 3

C、 [1,2,3]

D、 Error

解析：本题主要考察的是Rest参数的用法，在ES6中，Rest参数只能放在末尾，所以该用法的错误

的。

**第十二题：**D

let arr = [ ];

for (let { x = 2, y } of [{ x: 1 }, 2, { y }]) {

arr.push(x, y);

}

arr;

A、 [2, { x: 1 }, 2, 2, 2, { y }]

B、 [{ x: 1 }, 2, { y }]

C、 [1, undefined, 2, undefined, 2, undefined]

D、 Error

解析：本题看起来是考察let的作用域和of迭代的用法。实则是考察let的语法，let之后是一个参数

名称。所以，语法错误

**第十三题：**D

(function() {

if (false) {

let f = { g() => 1 };

}

return typeof f;

})()

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

解析：本题非常有迷惑性，看似考察的let的作用域问题，实则考察了箭头函数的语法问题。

**第十四题：**

(function(x, f = () => x) {

var x;

var y = x;

x = 2;

return [x, y, f()];

})(1)

A、 [2, 1, 1]

B、 [2, undefined, 1]

C、 [2, 1, 2]

D、 [2, undefined, 2]

解析：本题主要考察的知识点是1、参数值与函数体内定义的重名变量的优先级；2、ES6的默认参数；3、箭头函数。 在本题中，先执行x的定义，然后函数参数x=1，接着是y = x = 1,接着再x = 2，第三个是执行f函数，箭头函数如果只是表达式，那么等价于return 表达式，由于箭头函数的作用域等于定义时的作用域，那么函数定义时x=1，所以最后的return x 等价于 return 1

**第十五题：**

(function() {

return [

(() => this.x).bind({ x: 'inner' })(),

(() => this.x)()

]

}).call({ x: 'outer' });

A、 ['inner', 'outer']

B、 ['outer', 'outer']

C、 [undefined, undefined]

D、 Error

解析：本题主要考察的是箭头函数的作用域问题，箭头函数的作用域等于定义时的作用域，所以通过bind设置的this是无效的。那么结果就显而易见了。

**第十六题：**

let x, { x: y = 1 } = { x }; y;

A、 undefined

B、 1

C、 { x: 1 }

D、 Error

解析：本题主要考察的是对象赋值，先定义x，然后在赋值的时候会执行一次y=1，最后返回y的值。

**第十七题**

(function() {

let f = this ? class g { } : class h { };

return [

typeof f,

typeof h

];

})();

A、 ["function", "undefined"]

B、 ["function", "function"]

C、 ["undefined", "undefined"]

D、 Error

解析：本题主要考察定义函数变量时，命名函数的名称作用域问题。在定义函数变量时，函数名称只能在函数体中生效。

**第十八题**

(typeof (new (class { class () {} })))

A、 "function"

B、 "object"

C、 "undefined"

D、 Error

解析：本题主要考察对象的类型，和原型方法。该提可以分解如下：

// 定义包含class原型方法的类。

var Test = class{

class(){}

};

var test = new Test(); //定义类的实例

typeof test; //出结果

**第十九题：**

typeof (new (class F extends (String, Array) { })).substring

A、 "function"

B、 "object"

C、 "undefined"

D、 Error

解析：本题主要考察ES6中class的继承，以及表达式的返回值和undefined的类型。题目其实可以按照如下方式分解：

//由于JS的class没有多继承的概念，所以括号被当做表达式来看

(String, Array) //Array,返回最后一个值

(class F extends Array); //class F继承成Array

(new (class F extends Array)); //创建一个F的实例

(new (class F extends (String, Array) { })).substring; //取实例的substring方法，由于没有继承String，Array没有substring方法，那么返回值为undefined

typeof (new (class F extends (String, Array) { })).substring; //对undefined取typeof

**第二十题：**

typeof (new class f() { [f]() { }, f: { } })[`${f}`]

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

解析：本题主要考察ES6的class，以及动态属性和模板字符串等。 实际上这个题动态属性和模板字符串都是烟雾弹，在执行new class f()的时候，就已经有语法错误了。

**第二十一题**

typeof `${{Object}}`.prototype

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

解析：本题考察的知识点相对单一，就是模板字符串。分解如下：

var o = {Object},

str = `${o}`;

typeof str.prototype;

**第二十二题**

((...x, xs)=>x)(1,2,3)

A、 1

B、 3

C、 [1,2,3]

D、 Error

解析：本题主要考察的是Rest参数的用法，在ES6中，Rest参数只能放在末尾，所以该用法的错误的。

**第二十三题**

let arr = [ ];

for (let { x = 2, y } of [{ x: 1 }, 2, { y }]) {

arr.push(x, y);

}

arr;

A、 [2, { x: 1 }, 2, 2, 2, { y }]

B、 [{ x: 1 }, 2, { y }]

C、 [1, undefined, 2, undefined, 2, undefined]

D、 Error

解析：本题看起来是考察let的作用域和of迭代的用法。实则是考察let的语法，let之后是一个参数名称。所以，语法错误

**第二十四题**

(function() {

if (false) {

let f = { g() => 1 };

}

return typeof f;

})()

A、 "function"

B、 "undefined"

C、 "object"

D、 Error

25.

constructor() {

super();

this.state = {

val: 0

};

}

componentDidMount() {

this.setState({val: this.state.val + 1});

console.log(this.state.val); // 第 1 次 log

this.setState({val: this.state.val + 1});

console.log(this.state.val); // 第 2 次 log

setTimeout(() => {

this.setState({val: this.state.val + 1});

console.log(this.state.val); // 第 3 次 log

this.setState({val: this.state.val + 1});

console.log(this.state.val); // 第 4 次 log

}, 0);

}

render() {

return null;

}

};

问上述代码中 4 次 console.log 打印出来的 val 分别是多少？

25. 组件中的状态维护，是通过（）来维护

a) ref b) state c) props d)dom

26. 哪个是react中 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

a) router b)dom c)keys d)render

keys 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

render () {

return (

<ul>

{this.state.todoItems.map(({task, uid}) => {

return <li key={uid}>{task}</li>

})}

</ul>

)

}

27. 在react生命周期中，哪些生命周期函数只能执行一次（）(多选)

a)getInitialState b)getDefaultState c)componentWillMount d)render

28. 在react中定义组件的方式有()(多选)

a) class b) function c) Object d)Component

29. 在react中获取数据的方式（）（多选）?

a）ajax b)$.ajax() c)fetch d)axios

30.在react中，跨组件传递参数,是通过()

a) context b) state c) task d)param

31.在组件中，如何修改state属性的值？

a) state = 值 b)通过getState的方法 c)通过setState的方法 d)set方法

32. 在redux中，存储了整个应用的state的是(）

a)localstorge b)sessionStorage c)store d)libary

33. 在redux中,监听state发生变化的是（）？

a) store.getState() b)store.dispatch c)store.subscribe() d)listener()

34. 在redux中，改变state ，必须触发action,reducer就是根据传入的各种action不同,相对应对state进行处理，最后返回一个新state的函数,reducer这个函数必须要有两个参数（）？

a)state b)action c)redux d)dom

35. 合并reducer的方法是（）

a)union b)combineReducers c)all D)and

36. 在babel中，插件集合的是()

a)core b)plugs c)presets d)babelrc

37. nodejs主要特点（）(多选)

a)单线程 b)非io阻塞 c)多线程 d)异步编程

38. nodejs的开发框架（）（多选）

a)express b)koa c)nodal d)ejs

39. nodejs的全局对象（）

a) window b)global c)\_\_dirname d)\_\_dirname

40. 在nodejs中导入模块，使用（）

a)import b)export c)require d)exports

41.在nodejs中创建web服务器，需要导入（）模块

a)fs b)url c)querystring d)http

42. express的核心特性：(多选)

a)可以快速地搭建一个完整网站的前端框架

b)可以设置中间件来响应 HTTP 请求。

c)定义了路由表用于执行不同的 HTTP 请求动作。

d)可以通过向模板传递参数来动态渲染 HTML 页面

43. nodejs连接mysql需要（）配置信息（多选）

a)服务器地址

b)用户名

c)密码

d)数据库名

44. nodejs需要读取文件内容，需要引入()模块

a) http b)fs c)url d)file

45. 在nodejs中获取url的查询参数串()：

a)req.app b)req.body c)req.query d)req.route

46. 在nodejs中渲染ejs模板：通过以下()命令

a)res.render b)res.send c)res.write d)res.end

1.下面不属于ECMAScript规范的范围的是：（ ）

A、数据类型

B、语法

C、DOM事件

D、内置对象和函数的标准库

答案：C

解析：DOM事件不属于ECMAScript的部分；ECMAScript定义的内容：语法、类型、原型和继承、内置对象和函数的标准库。

考点： ECMAScript定义规范的理解，

2.下面不属于关键字let的特点的是：（ ）

A、只在 let 命令所在的代码块内有效

B、会产生变量提升现象

C、同一个作用域，不能重复声明同一个变量

D、不能在函数内部重新声明参数

答案：B

解析：使用var关键字才会产生变量提升的现象。关键字let不会产生变量提升现象，所以必须先声明定义后使用，否则程序抛出异常。

考点：关键字let的使用注意事项，

3.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，不赋值。

答案：D

解析：声明后必须赋值，负责程序会抛出异常。

考点：关键字const的使用注意事项，来自 的内容。

4.在数组的解构赋值中，var [ a,b,c ] = [ 1,2 ]结果中，a、b、c的值分别是：（）

A、1  2  null

B、1  2  undefined

C、1  2  2

D、抛出异常

答案：B

解析：在解构赋值过程中，赋值不成功，变量的值会变成undefined，不会抛出异常。

考点：数组的结构赋值使用注意事项。

5.在对象的解构赋值中，var {a,b,c} = { “c”:10, ”b”:9, ”a”:8 } 结果中，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。

考点：对象解构赋值和数组解构赋值的区别。

6.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。

考点：模板字符串的特点。

7.关于字符串扩展的新函数，下面描述错误的是：（）

A、includes函数用于判断字符串中是否含有指定的子字符串

B、repeat函数将目标字符串重复N次，目标字符串被修改

C、startsWidth函数判断指定的子字符串是否出现在目标字符串头部位置

D、endWidth函数判断指定的子字符串是否出现在目标字符串尾部位置

答案：B

解析：repeat函数将目标字符串重复N次，会返回一个新的字符串，不影响目标字符串。

考点：String字符串的新特性repeat函数的使用。

8.关于数值的扩展，window.isNaN(“abc”) 和Number.isNaN(“abc”)的结果分别是：（）

A、true   false

B、false   true

C、true   true

D、false   false

答案：A

解析：window.isNaN函数会把非数值的参数转化成数值再进行判断，而Number. isNaN只对数值类型有效，非数值类型的参数一律返回false。

考点：isNaN函数从全局函数移植到Number对象后的区别。

9.关于数值扩展的Number.isInteger函数，下面说法错误的是：（）

A、用来判断是否是整数，返回布尔值。

B、Number.isInteger(3);结果为：true

C、Number.isInteger(3.0);结果为：false

D、Number.isInteger(3.01);结果为：false

答案：C

解析：在JavaScript内部对整数和浮点数采用一样的存储方式，因此小数点后如果都是0的浮点数，都会被认为是整数。

考点：Number.isInteger函数使用的注意事项。

10.数组扩展的fill( )函数，[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函数的参数会把原数组的每个元素填充成指定的参数。

考点：对fill( )函数的了解和使用

11.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。

考点：利用数组的新特性来实现数组的遍历。

12.关于对象的扩展，错误的是：（）

A、ES6中对象的表示法更加简洁

B、对象的属性名可以是表达式

C、对象的方法名不可以是表达式

D、当属性名为表达式的时候，需要使用[ ]标识

答案：C

解析：对象的方法名同样可以是表达式，使用[ ]标识。

考点：对象的新特性的使用注意事项

13.关于对象的Object.is函数，错误的是：（）

A、用于判断两个值是否严格相等

B、用于判断两个值是否抽象相等

C、Object.is(3,'3') 结果是：false

D、Object.is(3,3.0) 结果是：true

答案：B

解析：Object.is函数判断两个值是否严格相等，或者说全等，作用等同于全等符号：===

考点：Object.is函数的了解和使用注意事项。

14.函数的扩展中，关于rest参数的描述，正确的是：（）

A、获取函数剩下部分的参数

B、rest参数不可以是函数的最后一个参数

C、获取函数的第一个参数

D、一个名叫rest的参数

答案：A

解析：rest参数用于表示且必须是函数的最后一个函数，后面不可以再跟其他参数，故B选项是错的。它的表示法表示法使用...（三个点）+ 自定义的参数名，故D选项是错的

考点：rest参数的理解和使用

15.关于箭头函数的描述，错误的是：（）

A、使用箭头符号=>定义

B、参数超过1个的话，需要用小括号（）括起来

C、函数体语句超过1条的时候，需要用大括号{ }括起来，用return语句返回

D、函数体内的 this 对象，绑定使用时所在的对象

答案：D

解析：函数体内的 this 对象，绑定定义时所在的对象，而不是使用时所在的对象。这点很重要，连载第十节的时候还特意加了一节讲这个内容。

考点：箭头函数的使用注意事项。

16.关于Symbol，错误的说法是：（）

A、是ES6新增的一种数据类型

B、Symbol() === Symbol() 结果为false

C、Symbol('same') === Symbol('same') 结果为true

D、当symbol值作为对象的属性名的时候，不能用点运算符获取对应的值。

答案：C

解析：symbol是独一无二的值，虽然它们的描述都是“'same'”，但是对应的值还是不一样的，所以结果为：false。

解析：关于Symbol的使用注意事项，

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。

解析：Symbol和Symbol.for创建变量的区别，

18.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。

考点：Proxy代理的设置和取消代理

19.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

考点：for...of遍历相对于传统便利方式的优势，

20.关于set结构，下面说法错误的是：（）

A、创建一个实例需要用new关键字

B、结构成员都是唯一的，不允许重复

C、使用add方法添加已经存在的成员会报错

D、初始化的时候接受数组作为参数

答案：C

解析：使用add方法添加已经存在的成员，会自动忽略相同的值，相同的值保留一个。

考点：set结构的特点，

21.关于Set结构的实例方法，下面说法错误的是：（）

A、set方法用于添加成员

B、clear方法用于清除所有成员。

C、entries方法返回成员的位置索引和值的遍历器

D、values方法返回成员值的便利器

答案：C

解析：返回的是键名和键值的遍历器；特别注意的是：set结构的键名和键值是同一个值。

考点：set结构键名和键值的特点，

22.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。

考点：Set和WeakSet的不同点，

23.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。

考点：Map结构的键名特点

24.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。

考点：Map结构键名的值类型

25.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。

考点：Map和WeakMap的不同点，

26.ES6的新特性Promise对象的设计初衷是：（）

A、 更好地实现遍历具有iterator接口的数据结构

B、 为对象的操作增加了一层“拦截”

C、 独一无二的值，用于对象属性，避免属性名冲突

D、 让开发者更合理、更规范地用于处理异步操作

答案：D

解析：遍历具有iterator接口的数据结构是for...of的作用；为对象的操作增加了一层“拦截”是Proxy代理的设计初衷；独一无二的值，避免属性名冲突是Symbol的设计初衷。而D选项，是Promise对象的设计初衷。

考点：Promise对象的作用

27.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。

考点：Promise对象三种状态的转换关系，

28.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。

考点：类class的声明，

29.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。

考点：类的继承知识，

30.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。

考点：类的继承，

31.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。

考点：具有Iterator接口可遍历对象，

32.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。

考点：Iterator遍历器的原理，

33.关于新特性Generator函数的描述，错误的是：（）

A、Generator函数，又称生成器函数

B、声明Generator函数的关键字是：function\*

C、Generator函数执行后得到的一个生成器

D、使用return语句使Generator函数暂停执行，直到next方法的调用

答案：D

解析：使函数暂停执行的关键字是yield，不是return；return语句是使函数停止执行并退出。

考点：Generator函数的特点的内容。

34.Generator函数的yield关键字的作用是：（）

A、停止执行

B、退出函数

C、暂停执行，等待next( )方法调用

D、停止执行，可自行恢复执行

答案：C

解析：Generator函数可以有很多个yield。而return代表的是终止执行，yield代表的是暂停执行，后续通过调用生成器的next( )方法，可以恢复执行。

考点：yield关键字的关键字，

35.在Generator函数中，yield\*语句的作用是：（）

A、调用另外的Generator函数

B、暂停执行，待next( )方法调用后恢复

C、停止执行，退出函数

D、调用任意的其他函数

答案：A

解析：关键字yield\*来实现调用另外的Generator函数。如果一个Generator函数A执行过程中，进入（调用）了另一个Generator函数B，那么会一直等到Generator函数B全部执行完毕后，才会返回Generator函数A继续执行。

考点：关键字yield\*的作用，

36.关于ES6的module模块，下列说法错误的是：（）

A、可以有效解决大型系统文件复杂的依赖问题

B、使用export语句可以选择性地向外部暴露自己的属性或者方法

C、使用import语句导入其他模块的属性或者方法

D、目前大部分主流浏览器都支持module模块

答案：D

解析：目前(2016年上半年)还没有浏览器支持module模块的开发，需要借助转码工具才可以使用。A选项是ES6的设计初衷之一，B选项是export语句的正确描述，C选项是import语句的正确描述。

考点：module模块的使用和实现，

37.module模块中，对下列语句的描述，错误的是：（）

A、export 导出

B、import 导入

C、export default 默认导出

D、import \* as  重命名

答案：D

解析：import \* as：星号符\*实现的是整体导入。而重命名的实现方式是：import { name as myname }。

考点：module模块的关键字语句含义，

38.module模块中的注意事项，下列说法错误的：（）

A、除了对象类型以外，导出的属性对外都是只读的

B、导入不存在的变量，值为undefined

C、导入不存在的变量，程序会抛出异常

D、可以为导入的属性和方法取任意名字

答案：C

解析：导入不存在的变量，程序不会抛出异常，只是值为undefined。

考点：module模块的注意事项，

39.想要获取Map实例对象的成员数，利用的属性是：（）

A、size

B、length

C、sum

D、members

答案：A

解析：属性就是size：获取实例的成员数，其他选项都不是，很多初学者误以为是length。

考点：Map结构的知识点，

40.关于定义常量的关键字const，定义一个Object对象{“name”:”Jack”}，再对属性name 的值进行修改，如：obj.name = “John”。下列说法正确的：（）

A、修改常量，程序跑出异常

B、程序不抛出异常，修改无效

C、修改成功，name的值为John

D、程序不抛出异常，name的值为undefined

答案：C

解析：用const来声明一个对象类型的常量，就是传址赋值。而不可修改的是对象在内存中的地址，而不是对象本身。所以修改name并不是修改对象的内存地址，所以可以成功修改。

考点：const定义一个对象时候的注意事项，还记得装修师傅的故事吗？

# 简答题

1. Promise对象只有三种状态：
   1. 异步操作“未完成”（pending）
   2. 异步操作“已完成”（resolved，⼜称fulfilled）
   3. 异步操 作“失败”（rejected）
   4. 这三种的状态的变化途径只有两种： 异步操作从“未完成”到“已完成”——异步操作从“未完成”到“失败”。 这种变化只能发⽣⼀次，⼀旦当前状态变为“已完成”或“失败”，就意味着不会再有新的状态变化了。
   5. 因 此，Promise对象的最终结果只有两种： 异步操作成功，Promise对象传回⼀个值，状态变为resolved。——异步操作失败，Promise对象抛出 ⼀个错误，状态变为rejected。
2. nodejs的特点

单线程 非io阻塞 事件环

Node.js 异步编程的直接体现就是回调。

Node.js 是单进程单线程应用程序，但是通过事件和回调支持并发，所以性能非常高。

Node.js 基本上所有的事件机制都是用设计模式中观察者模式实现。

Node.js 单线程类似进入一个while(true)的事件循环，直到没有事件观察者退出，每个异步事件都生成一个事件观察者，如果有事件发生就调用该回调函数.

1. nodejs常用的系统模块有哪些，它们的功能是什么？
2. 如何在nodejs中获取get和post的请求参数？
3. express中间件的功能是什么？
4. express中的常用模板有哪些？
5. 箭头有哪些新特点？
6. let 和const:

let是更完美的var，不是全局变量，具有块级函数作用域,大多数情况不会发生变量提升。const定义常量值，不能够重新赋值，如果值是一个对象，可以改变对象里边的属性值

let

1、let声明的变量具有块级作用域

2、let声明的变量不能通过window.变量名进行访问

3、形如for(let x..)的循环是每次迭代都为x创建新的绑定

1. 如何避免回调地狱？

模块化：将回调函数转换为独立的函数

使用流程控制库，例如[aync]

使用Promise

使用aync/await

1. 什么是事件循环？
2. 使用NPM有哪些好处？

通过NPM，你可以安装和管理项目的依赖，并且能够指明依赖项的具体版本号。

对于Node应用开发而言，你可以通过package.json文件来管理项目信息，配置脚本， 以及指明项目依赖的具体版本。

1. react的生命周期有哪些，分为哪些执行阶段和执行顺序，并说明组件props发生改变时，执行的生命周期函数顺序
2. 请说明webpack 的作用
3. React中Element 和 Component 有何区别？

简单地说，一个 React element 描述了你想在屏幕上看到什么。

换个说法就是，一个 React element 是一些 UI 的对象表示。

一个 React Component 是一个函数或一个类，

它可以接受输入并返回一个 React element

(通常是通过 JSX ，它被转化成一个 createElement 调用）。

1. 描述事件在React中的处理方式

为了解决跨浏览器兼容性问题，

React 中的事件处理程序将传递 SyntheticEvent 的实例，

React 的浏览器本机事件的跨浏览器包装器。

React 将使用单个事件监听器监听顶层的所有事件。

这对于性能是有好处的，这也意味着在更新DOM时，React 不需要担心跟踪事件监听器

1. 什么是Nodejs？以及优缺点  
   Nodejs是一个JavaScript的运行环境，是一个服务器端的“JavaScript解释器”，用于方便高效地搭建一些响应速度快、易于扩展的网络应用。它采用事件驱动、异步编程，为网络服务而设计。  
   Nodejs的优缺点  
   优点：  
   （1）nodejs是基于事件驱动和无阻塞的，非常适合处理并发请求，因此构建在Nodejs上的代理服务器相比其他技术实现的服务器要好一点。  
   （2）与nodejs代理服务器交互的客户端代码由js语言编写，客户端与服务端都采用一种语言编写。  
   缺点：  
   （1）nodejs是一个相对新的开源项目，不太稳定，变化速度快  
   （2）不适合CPU密集应用。如果有长时间运行的计算（比如大循环），将会导致CPU时间片不能释放，使得后续I/O无法发起。
2. 关于 javascript 模块化，AMD和CMD的含义和区别

[AMD](https://github.com/amdjs/amdjs-api/wiki/AMD)是"Asynchronous Module Definition"的缩写，意思就是"异步模块定义"。它采用异步方式加载模块，模块的加载不影响它后面语句的运行。所有依赖这个模块的语句，都定义在一个回调函数中，等到加载完成之后，这个回调函数才会运行。  
AMD也采用require()语句加载模块，但是不同于CommonJS。  
主要有两个Javascript库实现了AMD规范：[require.js](http://requirejs.org/)和[curl.js](https://github.com/cujojs/curl)。

AMD 是 RequireJS 在推广过程中对模块定义的规范化产出。  
CMD 是 SeaJS 在推广过程中对模块定义的规范化产出。  
区别：  
1. 对于依赖的模块，AMD 是**提前执行**，CMD 是**延迟执行**。  
2. CMD 推崇**依赖就近**，AMD 推崇**依赖前置**。

1. .rem的理解:(

rem 的62% 是个 什么意思 16px=1rem 1px=62.5%rem;

迭代器和生成器的理解

1.array,set,map

都属于迭代器 iterable(可迭代),类型 ,可以使用for in 来遍历

2.一直对for in 的误解

for in 只可以遍历对象,而数组就是一个对象

3.迭代器的遍历:

iterable内置的forEach方法

4.如何初始化 ,创建实例

var s = new Set(['A', 'B', 'C']);

var m = new Map([[1, 'x'], [2, 'y'], [3, 'z']]);

1. Es6中常用的map操作

# 编程题

1）利用rest参数完成一个任意数的求和功能？

function add(...values) {

let sum = 0

for(var val of values) {

sum += val

}

return sum

}

2）利用nodejs完成用户的登录，将登录的用户及密码显示在页面上

3)完成sum(a,b)以及sum(a)(b)的功能

4）nodejs连接mysql数据库，实现个人信息注册功能，注册成功后打印注册信息

表（user[id,name,password,gender,status,remark]）

5) NodeJS实现简单的HTTP服务器，当请求成功后把Student[id,name,gender,age,address]对象 ，以json的方式返回

6）nodejs实现文件上传

7）利用解构赋值完成两数的交换