1. 分析下面javascript代码：

   var l = 1; var n = m = 0; function myFun(x) { x = (x++) + 5; } n = myFun(l); function anotherFun(x) { x = (--x) + 5; } m = anotherFun(n); alert(n); alert(m);

   输出结果为：（D）

   A. 6 10

   B. 6 4

   C. 6 undefined

   D. undefined undefined

2. 不同的浏览器获取按下键盘码的方式不同，不可能获取按下键盘码的是(A)

```
<input type="text" id="text" >
<script>
document.getElementById("text").onkeypress = function(e) {... ...}
```

   A. this.keyCode;

   B. e.keyCode;

   C. window.event.keyCode;

   D. e.charCode;

3. 下面js语句中,a的值不是1的选项是？(B)

   A. var a = 1 && 3 && 1;

   B. var a = 1 && 2 && 3;

   C. var a = 1 || 0 || 3;

   D. var a = 0 || 1 || 2;

4. 原型模式解决全局方法只能被一个类使用的问题，但是对象实例对引用型属性的修改会导致父类原型也发生改变，所以，在实际项目中，把构造函数模式和原型模式结合来解决原型模式问题。以下代码分析，错误的是(C)

```
function stu(name, like){
  this.name = name;
  this.like = like;
}
stu.prototype.say = function(){
  document.writeIn(this.name + '原型中说话方法');
}
stu.prototype.sex = '男';
```

```
var stu1 = new stu('张三',['打球', '游泳']);
stu1.like.push('读书');
document.writeIn(stu1.like);
var stu2 = new stu('李四',['唱歌', '运动']);
document.writeIn(stu2.like);
document.writeIn(stu1.like == stu2.like);
document.writeIn(stu1.say == stu2.say);
```

A. 把不同的属性或方法放在构造方法内;

B. 把相同的属性或方法放在原型内;

C. document.writeIn(stu1.say == stu2.say);输出结果是false;

D. document.writeIn(stu1.like);输出结果是打球, 游泳, 读书;

5. setInterval("alert('hello');",2000); 这段代码的意思是(C)

A. 2000秒后弹出对话框;

B. 2秒后弹出对话框;

C. 每隔2秒钟弹出对话框;

D. 语句报错;

6. 对于

```
for (let i = 0; i < 5; i++) {
    setTimeout(function () {
        console.log(i)
    }, 1000);
}
和
for (var i = 0; i < 5; i++) {
    setTimeout(function () {
        console.log(i)
    }, 1000);
}
```

其输出结果分别为(A)

A. 55555 01234

B. 01234 01234

C. 01234 55555

D. 55555 55555

7. 下面正则表达式中,哪项不匹配()

A. var reg=/ph+..p/

var str = "phabp"

B. var reg=/ph+..p/

var str = "phhhhp"

C. var reg=/ph+..p/

var str = "phhhaap"

D. var reg=/ph+..p/

var str = "phhhaaap"

8. 下面对于cookies，sessionStorage和localStorage的描述错误的是()

A. cookie数据始终在同源的http请求中携带(即使不需要)，也会在浏览器和服务器间来回传递;

B. sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存;

C. cookie数据存储大小不能超过4k，而sessionStorage和localStiorage存储大小没有限制;

D. localStorage存储持久数据，浏览器关闭后数据不丢失除非主动删除数据; sessionStorage数据在当前浏览器窗口关闭后自动删除;

9. 下面的 JSX 代码中，哪一个无法达到预期的效果？()

A.

Hello world

B.

C.

{msg}

D. Leo

E. <div style={{height: 50}}>

F.

10. 关于typescript说法正确的是？（C）

A.类型断言可以当做类型声明或者类型转换使用；

B.当声明变量类型为对象时，可以直接用: Object声明；

C.接口可以定义函数；

D.interface IF {fn: (x: number) => void}这是声明函数的方法

11. 类式组件与函数式组件的区别说法正确的是？(B)

A、函数式组件useEffect(fn, [])完全等价于类式组件的componentDidMount；

B、函数式组件可以通过hooks达到类式组件生命周期函数的效果；

C、无论函数式组件和类式组件都可以使用hooks；

D、类式组件不能更改props但函数式组件可以；

12. 关于react hook - useState说法正确的是？(C)

A. useState返回的是一个对象，他有当前状态和设置当前状态的方法两个属性；

B. useState中解构的第二项setState和类式组件中的this.setState使用方法完全一致；

C. 从useState中解构出的第一个变量就是当前状态；

D. 多次调用useState中解构的第二项set方法会自动合并；

13. 关于自定义hook说法正确的是？(C)

A. 自定义hook无需遵从以use开头的原则，因为不会有任何影响；

B. 自定义hook在多个组件中使用state可以共享传递信息；

C. 自定义hook中可以调用其他hook，但是依旧只能在顶层调用；

D. 自定义hook必须有特殊的标识，否则无法被正常识别；

14. 关于react hook - useEffect说法错误的是？(D)

A. useEffect(() => {}, [])使用时第二个参数为空数组时相当于挂载/卸载时执行；

B. useEffect(() => {}, [a])第二个参数数组中传入值则当值更新时才会执行；

C. useEffect(() => {})在组件每次更新时都会执行；

D. useEffect本身的执行不会有任何副作用；

15. 请简单描述一下react组件的生命周期以及生命周期函数

参考答案： React的组件在第一次挂在的时候首先获取父组件传递的props，接着获取初始的state值，接着经历挂载阶段的三个生命周期函数，也就是ComponentWillMount，render，ComponentDidMount，这三个函数分别代表组件将会挂载，组件渲染，组件挂载完毕三个阶段，在组件挂载完成后，组件的props和state的任意改变都会导致组建进入更新状态，在组件更新阶段，如果是props改变，则进入ComponentWillReceiveProps函数，接着进入ComponentShouldUpdate进行判断是否需要更新，如果是state改变则直接进入ComponentShouldUpdate判定，这个默认是true，当判定不需要更新的话，组件继续运行，需要更新的话则依次进入ComponentWillMount，render，ComponentDidMount三个函数，当组件卸载时，会首先进入生命周期函数ComponentWillUnmount,之后才进行卸载，如图

16. 请简述一下reac中的虚拟DOM以及虚拟DOM的对比规则

参考答案： 当然是使用的diff算法，diff算法有三种优化形式：

tree diff：将新旧两颗DOM树按照层级遍历，只对同级的DOM节点进行比较，即同一父节点下的所有子节点，当发现节点已经不存在，则该节点及其子节点会被完全删除，不会进一步比较

component diff：不同组件之间的对比，如果组件类型相同，暂不更新，否则删除旧的组件，再创建一个新的组件，插入到删除组件的位置

element diff:在类型相同的组件内，再继续对比组件内部的元素，

17. react中获取真实DOM的方法是___或___

答案： ReactDOM.findDOMNode()或this.refs

18. 请简述一下setState的流程

参考答案： 在代码中调用setState函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。 经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个UI界面。 在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。 在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

19. 请简述一下你对react高阶组件的了解

参考答案： 高阶组件接收React组件作为参数，并且返回一个新的React组件。高阶组件本质上也是一个函数，并不是一个组件。

20. 关于redux说法错误的是？(B)

A、可维护性，可以使代码维护变得更简单；

B、redux是基于react的一个状态管理工具；

C、可以通过工具跟踪状态的变化；

D、易于测试 - Redux 的代码主要是小巧、纯粹和独立的功能；

21. 说一下redux三大原则

答案点： 单一数据源 state是只读的 使用纯函数来执行修改

22. 请简述一下react-router-dom中Link与NavLink组件之间的区别

参考答案： Link和NavLink的区别就是一个点击的时候进行跳转，另一个点击之后还会额外加一个类名，可以控制样式。 如果要更改class名的话就用activeClassName=“xxx”

23. 请简述一下ts中的类装饰器与方法装饰器的区别

参考答案： 方法装饰器表达式会在运行时当作函数被调用，传入下列3个参数：

1. 对于静态成员来说是类的构造函数，对于实例成员是类的原型对象。
2. 成员的名字。
3. 成员的属性描述符。

类装饰器表达式会在运行时当作函数被调用，类的构造函数作为其唯一的参数。

24. ts中的typeof与js中有什么区别

参考答案： ts中除了与js中相同的通过typeof判断类型外，还可以直接将类型提取，直接作为类型使用。

25. 解释一下什么是纯函数

参考答案： 简单来说，一个函数的返回结果只依赖于它的参数，并且在执行过程里面没有副作用，我们就把这个函数叫做纯函数。这么说肯定比较抽象，我们把它掰开来看：

- 函数的返回结果只依赖于它的参数。
- 函数执行过程里面没有副作用。

26. 以下 Javascript 代码的运行结果是（）

```
let name = 'outer'
function showName() {
  console.log(name)
  let name = 'inner'
}
name = 'updatedOuter'
showName()

A. outer
B. inner
C. updatedOuter
D. ReferenceError

正确答案: D
```

27. 以下关于事件监听器的说法正确的是（） 【多选】

```
A. e.target 获取的是直接绑定监听器的 DOM 元素
B. 通常 addEventListener 的第三个参数默认为true，也就是采用冒泡模式
C. 通过使用事件代理可以提高性能，也有较好的可扩展性
D. 事件捕获和事件冒泡的顺序是先捕获后冒泡

正确答案: C D
```

28. 以下 Javascript 代码的运行结果是（）

```
console.log('5' + 3, 5 + '3')

A. 8 8
B. 53 8
C. 53 53
D. 8 53

正确答案: C
```

29. 下列代码的执行结果

```
var a = 1
function fn1() {
console.log(this.a)
}
const fn2 = ()=> {
console.log(this.a)
}

const obj = {
a: 10,
fn1: fn1,
fn2: fn2
}

fn1()
fn2()
obj.fn1()
obj.fn2()

A. 1 1 10 1
B. 1 10 10 10
C. 1 10 1 1
D. 1 1 1 10
正确答案: A
```

30. Vue数据双向绑定的实现原理【问答题】

1、Vue实现数据双向绑定的原理：Object.defineProperty（）
2、vue实现数据双向绑定主要步骤：
（1)需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter。这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化。

（2)compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图。

（3)Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:
①在自身实例化时往属性订阅器(dep)里面添加自己
②自身必须有一个update()方法
③待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。
（4)MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

31. js实现简单的双向绑定【代码题】

```
<body>
    <div id="app">
    <input type="text" id="txt">
    <p id="show"></p>
</div>
</body>
<script type="text/javascript">
    var obj = {}
    Object.defineProperty(obj, 'txt', {
        get: function () {
            return obj
        },
        set: function (newValue) {
            document.getElementById('txt').value = newValue
            document.getElementById('show').innerHTML = newValue
        }
    })
    document.addEventListener('keyup', function (e) {
        obj.txt = e.target.value
    })
</script>
```

32. 如何优化SPA应用的首屏加载速度慢的问题？【问答题】

1. 将公用的JS库通过script标签外部引入，减小app.bundel的大小，让浏览器并行下载资源文件，提高下载速度；
2. 在配置 路由时，页面和组件使用懒加载的方式引入，进一步缩小 app.bundel 的体积，在调用某个组件时再加载对应的js文件；
3. 加一个首屏 loading 图，提升用户体验；

33. Vue 中的 key 到底有什么用？

key 是给每一个 vnode 的唯一 id,依靠 key,我们的 diff 操作可以更准确、更快速 (对于简单列表页渲染来说 diff 节点也更快,但会产生一些隐藏的副作用,比如可能不会产生过渡效果,或者在某些节点有绑定数据（表单）状态，会出现状态错位。)
diff 算法的过程中,先会进行新旧节点的首尾交叉对比,当无法匹配的时候会用新节点的 key 与旧节点进行比对,从而找到相应旧节点.
更准确 : 因为带 key 就不是就地复用了,在 sameNode 函数 a.key === b.key 对比中可以避免就地复用的情况。所以会更加准确,如果不加 key,会导致之前节点的状态被保留下来,会产生一系列的 bug。

34. vue中route 和 router 的区别是什么？

route是“路由信息对象”，包括path,params,hash,query,fullPath,matched,name等路由信息参数。

router是“路由实例对象”，包括了路由的跳转方法(push、replace)，钩子函数等。

35. 如何优化webpack构建的性能

一、减少代码体积
1.使用CommonsChunksPlugin 提取多个chunk之间的通用模块，减少总体代码体积
2.把部分依赖转移到CDN上，避免每次编译过程都由Webpack处理
3.对一些组件库采用按需加载，避免无用的代码

二、减少目录检索范围
1、用loader的时候，通过制定exclude和include选项，减少loader遍历的目录范围，从而加快webpack编译速度

三、减少检索路经：resolve.alias可以配置webpack模块解析的别名，对于比较深的解析路经，可以对其配置alias

36. vuex有哪几种状态和属性

state中保存着共有数据，数据是响应式的
getter可以对state进行计算操作，主要用来过滤一些数据，可以在多组件之间复用
mutations定义的方法动态修改state中的数据，通过commit提交方法，方法必须是同步的
actions将mutations里面处理数据的方法变成异步的，就是异步操作数据，通store.dispatch来分发actions，把异步的方法写在actions中，通过commit提交mutations，进行修改数据。
modules：模块化vuex

37. computed和watch有什么区别?

1. computed是计算属性,也就是计算值,它更多用于计算值的场景
2. computed具有缓存性,computed的值在getter执行后是会缓存的，只有在它依赖的属性值改变之后，下一次获取computed的值时才会重新调用对应的getter来计算
3. computed适用于计算比较消耗性能的计算场景

38. window.onload 和 document.DOMContentLoaded (注：$(document).ready()) 的区别？

一般情况下，DOMContentLoaded事件要在window.onload之前执行，当DOM树构建完成的时候就会执行DOMContentLoaded事件，而window.onload是在页面载入完成的时候，才执行，这其中包括图片等元素。大多数时候我们只是想在DOM树构建完成后，绑定事件到元素，我们并不需要图片元素，加上有时候加载外域图片的速度非常缓慢。

39. attribute和property的区别是什么？

```
attribute是dom元素在文档中作为html标签拥有的属性；
property就是dom元素在js中作为对象拥有的属性。
所以：
对于html的标准属性来说，attribute和property是同步的，是会自动更新的，
但是对于自定义的属性来说，他们是不同步的
```

40. 图片懒加载与预加载

图片懒加载的原理就是暂时不设置图片的src属性，而是将图片的url隐藏起来，比如先写在data-src里面，等某些事件触发的时候(比如滚动到底部，点击加载图片)再将图片真实的url放进src属性里面，从而实现图片的延迟加载 图片预加载是指在一些需要展示大量图片的网站，实现图片的提前加载。从而提升用户体验。常用的方式有两种，一种是隐藏在css的background的url属性里面，一种是通过javascript的Image对象设置实例对象的src属性实现图片的预加载。相关代码如下：

CSS预加载图片方式： #preload-01 { background: url(http://domain.tld/image-01.png) no-repeat -9999px -9999px; }

#preload-02 { background: url(http://domain.tld/image-02.png) no-repeat -9999px -9999px; }

#preload-03 { background: url(http://domain.tld/image-03.png) no-repeat -9999px -9999px; }

Javascript预加载图片的方式：

```
function preloadImg(url) {
   var img = new Image();
   img.src = url;
   if(img.complete) {
       //接下来可以使用图片了
       //do something here
   } else {
       img.onload = function() {
           //接下来可以使用图片了
           //do something here
       };
   }
}
```

41. ES6 箭头函数中的this和普通函数中的有什么不同

箭头函数是 ES6 中新的函数定义形式，function name(arg1, arg2) {...}可以使用(arg1, arg2) => {...}来定义。箭头函数没有this，他的this永远指向的是上层距离当前函数最近的this

```
// JS 普通函数

var arr = [1, 2, 3]
arr.map(function (item) {
```

```
  console.log(index)
  return item + 1
})

// ES6 箭头函数
const arr = [1, 2, 3]
arr.map((item, index) => {
  console.log(index)
  return item + 1
})

arr.map(item => item + 1)



function fn() {
  console.log('real', this) // {a: 100} ，该作用域下的 this 的真实的值
  var arr = [1, 2, 3]
  // 普通 JS
  arr.map(function (item) {
    console.log('js', this) // window 。普通函数，这里打印出来的是全局变量，令人费解
    return item + 1
  })

  // 箭头函数
  arr.map(item => {
    console.log('es6', this) // {a: 100} 。箭头函数，这里打印的就是父作用域的 this
    return item + 1

  })
}

fn.call({a: 100})
```

箭头函数存在的意义，第一写起来更加简洁，第二可以解决 ES6 之前函数执行中this是全局变量的问题，

42. null，undefined 的区别？

null 表示一个对象被定义了，值为“空值”； undefined 表示不存在这个值。

typeof undefined //"undefined" undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined； 例如变量被声明了，但没有赋值时，就等于undefined

typeof null //"object" null : 是一个对象(空对象, 没有任何属性和方法)； 例如作为函数的参数，表示该函数的参数不是对象；

注意： 在验证null时，一定要使用 === ，因为 == 无法分别 null 和 undefined undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。典型用法是： 1）变量被声明了，但没有赋值时，就等于undefined。 2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。 3）对象没有赋值的属性，该属性的值为undefined。 4）函数没有返回值时，默认返回undefined。

null表示"没有对象"，即该处不应该有值。典型用法是： 1） 作为函数的参数，表示该函数的参数不是对象。 2） 作为对象原型链的终点。

43. 声明变量和声明函数的提升有什么区别?

(1) 变量声明提升：变量声明在进入执行上下文就完成了。 只要变量在代码中进行了声明，无论它在哪个位置上进行声明， js引擎都会将它的声明放在范围作用域的顶部；

(2) 函数声明提升：执行代码之前会先读取函数声明，意味着可以把函数申明放在调用它的语句后面。 只要函数在代码中进行了声明，无论它在哪个位置上进行声明， js引擎都会将它的声明放在范围作用域的顶部；

(3) 变量or函数声明：函数声明会覆盖变量声明，但不会覆盖变量赋值。 同一个名称标识a，即有变量声明var a，又有函数声明function a() {}，不管二者声明的顺序，函数声明会覆盖变量声明，也就是说，此时a的值是声明的函数function a() {}。注意：如果在变量声明的同时初始化a，或是之后对a进行赋值，此时a的值变量的值。 eg: var a; var c = 1; a = 1; function a() { return true; } console.log(a);

44. new 的原理是什么？通过 new 的方式创建对象和通过字面量创建有什么区别？

在调用 new 的过程中会发生以上四件事情： 新生成了一个对象 链接到原型 绑定 this 返回新对象 根据以上几个过程，我们也可以试着来自己实现一个 new

```
function create() {
  let obj = {}
  let Con = [].shift.call(arguments)
  obj.__proto__ = Con.prototype
  let result = Con.apply(obj, arguments)
  return result instanceof Object ? result : obj
}
```

以下是对实现的分析：

- 创建一个空对象
- 获取构造函数
- 设置空对象的原型
- 绑定 this 并执行构造函数
- 确保返回值为对象
- 对于对象来说，其实都是通过 new 产生的，无论是 function Foo() 还是 let a = { b : 1 } 。
- 对于创建一个对象来说，更推荐使用字面量的方式创建对象（无论性能上还是可读性）。因为你使用 new Object() 的方式创建对象需要通过作用域链一层层找到 Object，但是你使用字面量的方式就没这个问题。 function Foo() {}

// function 就是个语法糖

// 内部等同于 new Function()

let a = { b: 1 }

// 这个字面量内部也是使用了 new Object(

首先创建一个空对象，之后让这个对象的proto指向构造函数的prototype，指向之后继承一些里面的属性，之后将对象return出去，并改变它的指向

8. focus/blur与focusin/focusout的区别与联系

focus/blur不冒泡，focusin/focusout冒泡 focus/blur兼容性好，focusin/focusout在除FireFox外的浏览器下都保持良好兼容性，如需使用事件托管，可考虑在FireFox下使用事件捕获elem.addEventListener('focus', handler, true) 可获得焦点的元素： window 链接被点击或键盘操作 表单空间被点击或键盘操作 设置tabindex属性的元素被点击或键盘操作

45. 简述javascript中this的指向

第一准则是：this永远指向函数运行时所在的对象，而不是函数被创建时所在的对象。

普通的函数调用，函数被谁调用，this就是谁。 构造函数的话，如果不用new操作符而直接调用，那即this指向window。用new操作符生成对象实例后，this就指向了新生成的对象。 匿名函数或不处于任何对象中的函数指向window 。 如果是call，apply等，指定的this是谁，就是谁。

46. 其实，GET和POST本质上两者没有任何区别。他们都是HTTP协议中的请求方法。底层实现都是基于TCP/IP协议。所谓区别，只是浏览器厂家根据约定，做得限制而已。

get是通过明文发送数据请求，而post是通过密文； get传输的数据量有限，因为url的长度有限，post则不受限； GET请求的参数只能是ASCII码，所以中文需要URL编码，而POST请求传参没有这个限制 GET产生一个TCP数据包；POST产生两个TCP数据包。对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

47. 通过new创建一个对象的时候，构造函数内部有哪些改变？

```
function Person(){}
Person.prototype.friend = [];
Person.prototype.name = '';
var a = new Person();
a.friend[0] = '王琦';
var b = new Person();
console.log(b.friend);//Array [ "王琦" ]
```

创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。 属性和方法被加入到 this 引用的对象中。 新创建的对象由 this 所引用，并且最后隐式的返回 this 。

48. 什么是原型?

原型链：简单来讲就是原型组成的链，比如函数的原型是Function，Function的原型是Object，Object的原型仍然是Object，一直追溯到最终的原型对象。 函数通过prototype来追溯原型对象，对象通过proto来追溯原型对象。

通过一个构造函数创建出来的多个实例，如果都要添加一个方法，给每个实例去添加并不是一个明智的选择。这时就该用上原型了。

在实例的原型上添加一个方法，这个原型的所有实例便都有了这个方法。

原型链继承：

```
function Show(){
this.name="run";
}

function Run(){
this.age="20"; //Run继承了Show,通过原型，形成链条
}
Run.prototype=new Show();
var show=new Run();
alert(show.name)//结果：run
```

49. typeof运算符和instanceof运算符以及isPrototypeOf()方法的区别

typeof是一个运算符，用于检测数据的类型，比如基本数据类型null、undefined、string、number、boolean，以及引用数据类型object、function，但是对于正则表达式、日期、数组这些引用数据类型，它会全部识别为object； instanceof同样也是一个运算符，它就能很好识别数据具体是哪一种引用类型。它与isPrototypeOf的区别就是它是用来检测构造函数的原型是否存在于指定对象的原型链当中；而isPrototypeOf是用来检测调用此方法的对象是否存在于指定对象的原型链中，所以本质上就是检测目标不同。

50. 全局函数eval()有什么作用？

eval()只有一个参数，如果传入的参数不是字符串，它直接返回这个参数。如果参数是字符串，它会把字符串当成javascript代码进行编译。如果编译失败则抛出一个语法错误(syntaxError)异常。如果编译成功，则开始执行这段代码，并返回字符串中的最后一个表达式或语句的值，如果最后一个表达式或语句没有值，则最终返回undefined。如果字符串抛出一个异常，这个异常将把该调用传递给eval()。

51. 序列{9,12,17,30,50,20,60,65,4,19}构造为堆后，堆所对应的的中序遍历序列可能为（B）

```
A.65,12,30,50,9,19,20,4,,17,60

B.65,12,30,9,50,19,4,20,17,60

C.65,9,30,12,19,50,4,20,17,60

D.65,12,9,30,50,4,20,9,17,60
```

52. 在不指定特殊属性的情况下，以下标签可以手动输入文本的是 (B)

A.

B.

C.

D.

53. HTML5新增的表单元素不包括 (A)

A.password；

B.color；

C.data；

D.number;

54. transition与animation的区别说法错误的是 (B)

A.transition着重属性的变化，而animation重点是在创建帧，让不同帧在不同时间点发生不同变化；

B.animation需要时间触发来达到动画的效果；

C.animation可以实现复杂的动画；

D.animation通过@keyframe控制当前帧的属性；

55. 在面向对象技术中，多态性是指（C）

A.—个类可以派生出多个类

B.—个对象在不同的运行环境中可以有不同的变体

C.针对一消息，不同对象可以以适合自身的方式加以响应

D.—个对象可以由多个其他对象组成

56. 下列说法中错误的是 (B)

A.JSON只是一种数据格式，不从属于JavaScript

B.在事件流过程中，首先发生的阶段是冒泡阶段

C.document.querySelector方法在没有找到匹配的元素时会返回null

D.ECMAScript中只有5种基本数据类型

57. 以下代码能在不同环境下（不考虑兼容性问题）正确判断变量a = [] 是数组的有：(BC)

A.a instanceof Array

B.Array.isArray(a)

C.Object.prototype.toString.call(a) === '[object Array]'

D.typeof a === 'array'

58. 执行下列语句后， 变量name的值为 （A）

function Person() {}; var person1 = new Person(); var person2 = new Person(); Person.prototype.getName = function () { return this.name; }; Person.prototype.name = 'tom'; person1.name = 'jerry'; var name = person2.getName();

A.tom

B.jerry

C.name

D.undefined

59. 下列说法中正确的是 (B)

A.JSONP不是一种跨域技术

B.调用Date构造函数而不传参数时，得到的对象自动获得当前时间

C.调用setInterval方法后得到的返回值为undefined

D.所有的RegExp实例对象都可以调用match方法

60. 根据如下代码，set.size的值为 (B)

var set = new Set([0, 2, 2, 0, 0, 5, 9, {}, {}, NaN, NaN]);

A.6

B.7

C.8

D.11

61. web storage和cookie的区别

a. Cookie的大小是受限的

b. 每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽

c. cookie还需要指定作用域，不可以跨域调用

d. Web Storage拥有setItem,getItem等方法，cookie需要前端开发者自己封装setCookie，getCookie

e. Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地"存储"数据而生

f. IE7、IE6中的UserData通过简单的代码封装可以统一到所有的浏览器都支持web storage

62. 不知宽高的盒子如何居中

```
<div class="warp"><div class="box"></div></div>

1)display:flex
.warp{display:flex;   justify-content:center;  align-item:center;}

2).warp{position:relative;}
.box{position:absolute; left:50%; right:50%;  transform:translate(-50%,-50%)}

```

```
 9 3).warp{position:relative;}
10 .box{position:absolute; left:0; right:0; top:0; bottom:0;
11  margin:auto;}
12
13 4).warp{display: table; }
14 .box{display:table-cell; text-align:center; vertical-align:middle; }
```

63. display none visibility hidden区别？

1.display:none是彻底消失，不在文档流中占位，浏览器也不会解析该元素；visibility:hidden是视觉上消失了，可以理解为透明度为0的效果，在文档流中占位，浏览器会解析该元素；

2.使用visibility:hidden比display:none性能上要好，display:none切换显示时visibility，页面产生回流（当页面中的一部分元素需要改变规模尺寸、布局、显示隐藏等，页面重新构建，此时就是回流。所有页面第一次加载时需要产生一次回流），而visibility切换是否显示时则不会引起回流。

64. 判断一个字符串中出现次数最多的字符，统计这个次数

```
var str = 'asdfssaaasasasasaa';

var json = {};

for (var i = 0; i < str.length; i++) {

    if(!json[str.charAt(i)]){

       json[str.charAt(i)] = 1;

    }else{

       json[str.charAt(i)]++;

    }

};var iMax = 0;var iIndex = '';for(var i in json){

    if(json[i]>iMax){

        iMax = json[i];

        iIndex = i;

    }

}       console.log('出现次数最多的是:'+iIndex+'出现'+iMax+'次');
```

65. http和https的区别？

Http 协议都是未加密的，http传输隐私消息非常的不安全

https就是由ssl+http协议构建进行加密传送的要比http协议安全

https需要申请证书，会需要一定的费用

http未加密，https 是加密的

http端口是80，https 是443

66. 什么是跨域？说出几种解决跨域的办法？

跨域是:指浏览器不能执行其他网站的脚本，它是由浏览器的同源策略造成的，浏览器对js实施的安全限制。

同源是指：域名、协议、端口均为相同。

解决跨域的办法

Jsonp 跨域 只能够实现get请求

PostMessage 跨域

67. Flex 布局和传统布局有什么区别？

传统布局：是基于盒模型，依赖display属性 、position属性、float属性。

Flex 布局：可以渐变、完成、响应式的实现各种页面布局，目前所有浏览器有支持。

68、下面代码的输出结果是：（A）

```
var one;
var two=null;
console.log(one==two,one===two);
```

A、true false B、true true C、false false D、false true

69、下面这段js代码执行结果是：（D）

```
var user = {
 count : 1,
 getCount: function(){
  return this.count;
 }
}
var func = user.getCount
console.log(func())
```

A、this.count B、1 C、报错 D、undefined

70、以下js操作Array的方法中不能添加元素的是：（B）

A、push B、pop C、unshift D、splice

71、以下js代码执行后，浏览器alert出来的结果分别是（B）

```
var color = 'green';
var test4399 = {
        color:'blue',
  getColor:function(){
    alert(this.color);
  }
}
var getColor = test4399.getColor;
getColor();
test4399.getColor();
```

A、undefined,red B、green,blue C、undefined,blue D、green,undefined E、blued,undefined

72、该代码在浏览器中执行，输出的日志结果是什么？（B）

```
var obj = {};
obj.log = console.log;
obj.log.call(console,this);
```

A、undefined B、window C、console D、obj

73、假设DOM结构为：

```
<div id="a"><div id="b"></div></div>
```

JS代码为：

```
document.getElementById('a').addEventListener('click', e => {console.log(1)});
document.getElementById('b').addEventListener('click', e =>
{e.preventDefault();console.log(2)});
```

当点击id为b的div时，控制台输出的内容是：(B)

A、1 2

B、2 1

C、1

D、2

74、函数的调用方式有哪些：（ABCD）【多选】

A、直接调用 B、作为对象方法调用 C、作为构造函数调用 D、通过call和apply方法调用

75、以下哪些表达式的结果为true（ACD）【多选】

A、undefined == null B、isNaN("100") C、parseInt("1a") === 1 D、[] instanceof Array

76、以下哪些操作会触发Reflow：(BC)【多选】

```
var obj = document.getElementById(“test”);
```

A、alert(obj.className) B、alert(obj.offsetHeight) C、obj.style.height = “100px” D、obj.style.color = “red”

77、用计算机解决问题的步骤一般为（D ）①编写程序 ②设计算法 ③分析问题 ④调试程序。

A、①②③④ B、③④①② C、②③①④ D、③②①④

78、JavaScript实现继承的方式，不正确的是：（D）

A、原型链继承 B、构造函数继承 C、组合继承 D、关联继承

79、下列哪项不是websocket的特性（B）

A、和http协议不同 B、客户端采用长轮询的方式向服务端发起请求 C、仍然需要至少一次客户端服务端握手 D、websocket客户端基于事件的编程模型与node类似

80、以下哪个说法是错误的：（D）

A、iframe是用来在网页中插入第三方页面，早期的页面使用iframe主要是用于导航栏这种很多页面都相同的部分，这样在切换页面的时候避免重复下载 B、iframe的创建比一般的DOM元素慢了1-2个数量级 C、iframe标签会阻塞页面的的加载 D、iframe本质是动态语言的Incude机制和利用ajax动态填充内容

81、DOM，文件对象模型，提供了树状结构的表示方法，以下描述正确的是：（D）

A、Document 不是一個document node，是一种节点格式 B、不是所有的HTML elements都是element nodes C、所有的comments都是document node D、在HTML element內的text內容也是text node

82、DOM事件流包括哪些阶段：（ABC）【多选】

A、事件捕获阶段 B、处于目标阶段 C、事件冒泡阶段 D、事件监控阶段

83、https协议的优缺点【简答】

答：优点 - 使用HTTPS协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；

HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全，可防止数据在传输过程中不被窃取、改变，确保数据的完整性。

HTTPS是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。

缺点 - https握手阶段比较费时，会使页面加载时间延长50%，增加10%~20%的耗电。

https缓存不如http高效，会增加数据开销。

SSL证书也需要钱，功能越强大的证书费用越高。

SSL证书需要绑定IP，不能再同一个ip上绑定多个域名，ipv4资源支持不了这种消耗。

84、实现一个a ja x有几个步骤【简答】

答：（1）创建XMLHttpRequest 对象。

（2）使用open方法设置请求的参数。open(method, url, 是否异步)。

（3）发送请求。

（4）注册事件。 注册onreadystatechange事件，状态改变时就会调用。

如果要在数据完整请求回来的时候才调用，我们需要手动写一些判断的逻辑。

（5）获取返回的数据。

85、 promise的状态有哪些？【简答】

答：等待(pending)、已完成(fulfilled)、已拒绝(rejected)

4、下面哪个模块可以删除目录？（C）

A、dir.close(callback)

B、fs.unlink(path, callback)

C、fs.rmdir(path[, options], callback)

D、dir.path

86、npm 安装开发依赖时的命令是（B）

A、npm i -g

B、npm i --save

C、npm i --save-dev

D、npm i -D

87、http状态码信息状态码切换协议是：（D）【单选】

A、200

B、300

C、500

D、101

88、以下哪种方式能解决跨域问题？（ACD）【多选】

A、WebSocket

B、axios

C、jsonp

D、CORS

89、下列关于promise说法正确的是（A）【单选】

A、promise是异步编程的一种解决方案；

B、promise的.then状态响应函数可以返回新的promise，或其他值，不返回值也可以我们可以认为它返回了一个null；如果返回新的promise，那么下一级.then()不会在新的promise状态改变之后执行因为它是异步的。

C、promise的原型上可以直接接收两个参数resolve和reject

D、promise不可以做链式调用；

90、关于koa说法正确的是？（D）【单选】

A、ctx.res是koa中重写的返还对象；

B、ctx.response是node中的返还对象

C、ctx.req是koa中重写的返还对象

D、ctx.state是对象命名空间，通过中间件传递信息

91、node.js 适合做什么业务？【简答】

答案：Nodejs 是单线程，非阻塞 I/O，事件驱动，它的特点决定了它适合做一些大量 I/O 的东西，比如，聊天室，表单提交等不需要大量计算的功能。做一些微信后端开发，或者做消息系统等。可以整个项目用， 也可以根据它的特点在某个模块使用，比如 socketio，打造一个消息系统等。

92、 Nodejs 中的 Stream 和 Buffer 有什么区别？【简答】

答案：Buffer：为数据缓冲对象，是一个类似数组结构的对象，可以通过指定开始写入的位置及写入的数据长度，往其中写入二进制数据。Stream：是对 buffer 对象的高级封装，其操作的底层还是 buffer 对象， stream 可以设置为可读、可写，或者即可读也可写，在 nodejs 中继承了 EventEmitter 接口，可以监听读入、写入的过程。具体实现有文件流，httpresponse 等。

93、.Express 和 koa 有什么关系，有什么区别？【简答】

答案：它既是开发平台, 也是运行环境, 也是个新的语言...它本身是基于 google 的 javascript v8 引擎开发的, 因此在编写基于它的代码的时候使用javascript 语言. 但是又不同于传统概念的javascript...它的服务端功能以及部分客户端功能必须在服务端运行, 所以它实际上是一种在服务端的开发+运行的 javascript 语言. 有一点类似于 Perl + PHP 或者 Python 的概念. 它本身可以作为 HTTP Server, 也可以当作TCP Server 用.

94、Nodejs遵循哪一套规范？（B）【单选】

A、RequireJS

B、CommonJS

C、CMD

D、AMD

95、下列关于Storage说法错误的是？（A）【单选】

A、Storage属性里有length和key

B、Storage提供了访问特定域名下的会话存储或本地存储的功能

C、操作一个域名的会话存储，可以使用 Window.sessionStorage；如果想要操作一个域名的本地存储，可以使用 Window.localStorage。

D、Storage.getItem()方法接受一个键名作为参数，返回键名对应的值。

96、以下关于 Cookie 的说法正确的是？（BCD）【多选】

A、Cookie的内容都可以通过JavaScript读取

B、Cookie过多会影响网络请求性能

C、Cookie可以用来跟踪用户的浏览行为

D、Cookie可以持久保存

97、下面关于cookie的说法错误的是？（A）

A、cookie是一小段存储在浏览器端文本信息，web应用程序可以读取cookie包含的信息

B、cookie可以存储一些敏感的用户信息，从而造成一定的安全风险

C、通过cookie提交精妙构造的移动代码，绕过身份验证的攻击叫做cookie欺骗

D、防范cookie欺骗的一个有效方法是不使用cookie验证方法，而使用session验证方法

98、下列关于localStorage的说法中不正确的是（D）【单选】

A、localStoragE.getItem（key）。该接口用于获取指定key本地存储的值

B、localStoragE.length。该接口表示对象中存储的键值对的数量

C、localStoragE.removeItem（key）。该接口用于删除指定key本地存储的值

D、localStoragE.key（index）。该接口用于将value存储到key字段。index从0开始

99、下面有关csrf的描述，说法错误的是？（D）【单选】

A、CSRF则通过伪装来自受信任用户的请求来利用受信任的网站

B、xss是实现csrf的诸多途径中的一条

C、在客户端页面增加伪随机数可以阻挡csrf

D、过滤用户输入的内容也可以阻挡csrf

19、浏览器存储中，Cookie和localStorage、sessionStorage的区别

答：

名称	cookie	localStorage	sessionStorage
相同点	都可以用来在浏览器端存储数据，都是字符串的键值对		
数据声明周期	一般由服务器生成，可设置失效时间；若在浏览器生成，默认关闭浏览器之后失效	除非被清除，否则永久有效	仅对当前对话有效，关闭当前页面或者浏览器后被清除
存储大小	4kb	一般5mb	
与服务端通信	每次都会携带在http请求头中，如果使用cookie保存过多，性能不太好	仅在客户端存储，不参与服务端通信	
用途	一般由服务器生成，来标识用户身份	勇于浏览器端缓存数据	

100、从输入url到页面加载完成发生了什么？

1、浏览器的地址栏输入URL并按下回车。
2、浏览器查找当前URL的DNS缓存记录。
3、DNS解析URL对应的IP。
4、根据IP建立TCP连接（三次握手）。
5、HTTP发起请求。
6、服务器处理请求，浏览器接收HTTP响应。
7、渲染页面，构建DOM树。
8、关闭TCP连接（四次挥手）