**HTML**

1. 如何理解html语义化

|  |
| --- |
| HTML语义化是为了解决div soup的问题，加强HTML的可视化。<header>,<nav>,<main>,<article>,<footer>来替代div的化，可以在接手别人的代码的时候，对结构一目了然。减少沟通成本。 |

**CSS**

**CSS原理·csst**

1. 如何处理CSS兼容问题
2. 盒子模型：border-box和content-box？
3. 为什么css样式初始化，目的是为了什么？

|  |
| --- |
| 浏览器css会有一些default属性，比如margin，padding不为0，box-sizing是content-box属性之类的。为了和UI设计在统一的测量标准下，将这些属性置0，在构建视图时工作效率会提高。 |

1. rem，px，em的区别
2. float和position的区别
3. block元素和inline元素的区别
4. margin坍塌
5. 清除float的方法
6. CSS中对溢出的处理
7. BCF的原理：浮动溢出的解决
8. 弹性布局
9. 流式布局
10. boostrap的底层原理？
11. CSS选择器有哪些，优先级呢？
12. 常用css伪类的应用
13. 关于iframe内部和外部变量的读取是如何的
14. transition生命周期
15. css性能？css动画优化？
16. 就动画效果，如何从js，css角度减少回流？

**CSS实战·cssd**

1. 写一下不知道宽高元素垂直水平居中方法
2. 获取元素的父节点和兄弟节点
3. 写一下节点的增删改
4. 实现一个两边宽度固定中间自适应的三列布局，圣杯布局，双飞燕
5. 实现三栏布局，中间自适应有几种方法
6. 如果有这样的场景：一个模块A作为输出，BCD等扩展模块可以在A做更改后展示A的原来内容或者驾驶CSS后的内容，想想思路
7. 如何画一个三角形
8. 如果弹出的菜单位置过于贴近边框，如何调整这个元素的位置
9. 用CSS实现一个loading效果

**Javascript**

**数据类型·jsda**

1. 基本数据类型

|  |
| --- |
| string，number，boolean，null，undefined，symbol，object基本数据类型 |

1. symbol
2. == 和 === 的区别？

|  |
| --- |
| ==在类型不相等的情况下，会做一步强制转化。然而===如果判断类型不相等的情况下，就会停下来，返回false。 |

1. typeof array null undefined NaN分别是什么

|  |
| --- |
| typeof array返回”object”； typeof null 返回”object”；typeof undefined 返回”undefined”；typeof NaN 返回”number” 。null的返回值是个例，它本该是一个基本类型。历史原因，由于ES1的代码设计想强调，空对象和重置一个对象的概念，所以返回值定为了“Object”。个人倾向这是个bug，既然设置规则null是基本类型，就应该返回null。 |

1. 把undefined和null转成Number分别是什么

|  |
| --- |
| 考察点：强制转换的个例案件， Number(null) == 0; Number(undefined) == NaN。 |

1. 你说到typeof，能不能加一个限制条件typeof只能判断是object，可以判断一下是否拥有数组的方法
2. instanceOf和constructor的区别

|  |
| --- |
| 概念要清楚，基本类型有哪些？内建对象类型有哪些？上述两种方法的适用范围是内建对象。  instantceOf判断的是一个目标是否为对标对象的实例。  判断不了非典型对象，Boolean，String，Number。  Constructor是基本类型的属性，调用指向对应的function，在prototype模型中，这部分有详细的阐述。    小知识点：判断相等的表达是应用了Boxing原理，有一步强制转化。 |

1. 如何判断是否为数组？什么情况下会原型链断裂，导致的结果是什么？

|  |
| --- |
|  |

1. 给出一个字符串(“obj.a”)，返回对象属性obj.a，类似eval的效果

**作用域·jssc**

1. 闭包

|  |
| --- |
| 闭包的概念是：闭包记住了声明scope下的变量，在别的时间空间下，调用函数时，依然可以读取到这些变量。    这是闭包的经典案例，这里面提出来的问题就是*outer*执行结束之后，函数空间消亡了，inner的n为什么还有值？  闭包在我的理解里，有着很重的时间空间属性。闭包有点像*inner*和*outer*之间的虫洞，保持着inner保持的一个通道可以访问到*outer*中的变量*n*。由于“虫洞”的存在，*outer*空间不会消亡，直到“虫洞”访问需求消失。 |

1. let和var的区别，let的产生背景？

|  |
| --- |
| 这两个关键词的最关键的区别是scope，*var*是认定的scope不包含*for*，*if*等这类函数块的，*let*把*for*，*if*等函数块认定为一个scope。如果说有一些需要在函数块中记住变量属性的表达式，ES5下写，需要加上花括号，这样子嵌套的层级会很多。我猜测是为了代码整洁的原则，ES6的时候就发明了*let*来简化代码。 |

1. var的变量提升底层原理是什么？

|  |
| --- |
| 本质上是由JS编译过程导致的。代码的执行分为两个步骤，第一步：对声明进行编译——为*var*或*function*分配地址内存。第二步执行——进行正常的赋值。  例如*var a = 2;* 编译器先不会管*a = 2*这一赋值步骤。它最先要做的是把*var a*编译掉，而运算操作留在原地。声明在编译角度上被移动到了作用域的顶端。这就是提升的底层原理。  ////////////////////////拓展话题——TDZ暂时死区///////////////////////////////    *aVar*的值符合提升的原则。*aLet*就有点意思了，它抛出来的错误实际上就是暂时死区错误。简单来说，从声明到到let的运算步骤为止的这段时空被锁死，外界在此期间寻求访问，会被反弹。 |

1. 模块化思想

|  |
| --- |
| 大致从什么是模块化？为什么要使用模块化？以及怎么写一段模块代码来梳理回答思路。  模块化思想即最小暴露原则。隐藏掉别人不需要知道的代码细节，仅提供需要使用的API。保护代码的安全性和可维护性。  书写一段模块的要点有二：1、要使用闭包原理。2、函数至少被调用执行过一次。  方法一：工厂模式    方法二：单例模式（IIFE） |

**对象系统·jsoo**

1. js的this理解， 如何改变this的指向

|  |
| --- |
| This这个参数到底代表谁，实际上是代码执行时动态形成的。要判断this的指向，需要明确的是含有this的函数到底在哪里进行了调用。This即指代的是进行调用的Object，将它作为上下文传入正在执行的函数。  ///////////////////////////////////////////////////////////////////////////////////////////////////  更改this的指向：隐性更改的话变换调用的Object即可，显性更改的话使用*call、apply、bind*函数进行再次绑定。  ///////////////////////////////////////////////////////////////////////////////////////////////////  这道题目的潜台词也是在问怎么解决this的绑定丢失的问题。混乱情况经常发生在把this作为一个参数传入回调函数的过程时，你很难预测到底谁才是真正的调用者。*{{this}}*和*this.this.*我把它们称作“this嵌套”很容易法伤丢失情况，解决方案用call或这bind    经典的this绑定丢失案例，解决方案：*obj.foo.bind(obj);* |

1. 请简述一下js原型链

|  |
| --- |
| 一般描述类，多为JAVA和C++的类概念。这种类实际上是一种拷贝行为。但是JS的类是基于JS原型链原理生成的。Js生成的实例，与父类之间形成一条链接，如果生成类中没有的属性，通过链接回溯到父类中寻找，这种行为一直可以回溯到*Object.prototype*为止。比起拷贝的类型，这种原型链形式，实际上更加节省内存。另外由于javasctipt的面向对象oo，实际上是oloo。 |

1. 在原型链上Object再往上是什么？

|  |
| --- |
| *Object.prototype*是原型链的终点了。 |

1. new生成了一个对象的过程（核心return this）

|  |
| --- |
| New关键字生成对象分为4个步骤：1、生成空白新对象，2、将新对象作为上下文传入父类。3、将新对象链接到父类默认的prototype对象上。4、父类返回值赋值给新对象。    New关键字出现的话，上面两列红框里面的代码就可以删除掉。今天看到一个理论挺有意思的：  *new Car(‘benz’);* 这句话可以看成是一个IIFE语句。 |

1. new和Object.create的区别

|  |
| --- |
| new和Object.create基本上完成的功能很像，创建一个新的空对象，链接到另一个对象。但new有构造函数而Object.create没有    Amy.constructor指向的不是Van，而是Car，因为Van.prototype没有通过new与Van创建好回溯链接，所以要向上回溯找到constructor函数。 特别要打破的思维壁垒：1、Car的constructor函数在Car里，不对，在Car.prototype里。2、Car.prototype天生就与Car有这Car.prototype.constructor = Car的链接关系，不对，Car.prototype只是默认给开辟的一块存储空间而已，是new创建了这种指向关系。Object.protoype.constructor = object除外，这是顶层设计，设计者已经写好了。 |

1. \_proto\_ 和prototype分别是什么？哪种情况下\_\_proto\_\_和prototype的指向是同一个？

|  |
| --- |
| 这道题目问的有些奇怪，大体上考察的问题：dunder和protype的指向。  直接上例子：    解析原型链的布线，需要明白的是函数的模型    子类对象通过\_\_proto\_\_指向父类的对象；函数通过prototype指向同级对象 |

1. js的继承？实现方法？

|  |
| --- |
| Js的继承说到底，只是一层语法糖，本身的机制是委托查找。经典的实现方法是原型类，变体可以使用委托设计模式，ES6中加了class语法糖，但是本质是别人已经把背后特别丑的原型类包了起来而已。  /////////////////////////////////////////////  原型类    委托类 |

1. js设计模式知道哪些，单例详细说
2. 函数式编程-柯里化
3. MVVM实现原理

**同步，异步，线程·jsth**

1. 谈谈JS的异步机制
2. 对JS单线程的理解
3. js如何实现一个栈
4. 操作系统进程和线程的区别
5. 线程的哪些资源共享，哪些资源不共享
6. promise函数
7. promise的状态有哪些
8. promise如何从then转为catch的
9. promise是如何解析异步操作的
10. 如果给a,b,c,d四个时间，执行时间分别为1，2，3，4，怎么样做才能在abc都在执行完后再执行d，除了使用promise还能使用什么方法
11. 观察者模式如何实现
12. 订阅者模式如何实现
13. settimeout异步
14. 函数节流和函数防抖知道吗，区别是什么
15. ES6 generator async/await了解吗

**语法点·jssu**

1. JS事件流
2. 事件冒泡，preventDefault和stopPropagation的区别
3. 事件捕获
4. 事件委托的原理，如何实现委托，要点是什么
5. 事件监听
6. event loop讲讲
7. 深拷贝
8. call apply bind的区别，以及bind是如何实现的
9. 假设一个object A里面的值n为1，怎么样知道n改变了，有事件绑定吗？
10. 给数字加千位分隔符
11. 正则如何将一个数千分化表示
12. jQuery原理，平时用jQuery都做过什么
13. 简单介绍一下backbone

**AJAX**

1. ajax实现过程
2. ajax是同步还是异步，怎么样实现同步
3. 表单提交和ajax的区别
4. 发送请求有哪些？get和post的区别
5. ajax返回204算是成功吗
6. localStorage存储数据格式是什么
7. 怎么样把一个数组存进localSorage
8. localstorage和cookie的区别，如何设置cookie
9. cookie有什么特征
10. 加入访问A.com存进了一个cookie，在另外一个页面用ajax向A的域名发请求会携带cookie吗
11. jsonp是如何实现的
12. 如何实现文件上传的功能
13. 页面共享数据的方法有哪些
14. 你怎么样维持用户登录状态的

**ES6**

1. typescript和es6的区别
2. es6模块新特性
3. es6的变量定义和es5的区别
4. ES6新增了哪些东西？
5. ES6的代理是什么
6. 箭头函数，箭头函数的特点
7. 箭头函数和ES5中this的指向区别
8. const和let的区别，可以改变const定义对象某个属性吗
9. 数组遍历的方法

**浏览器·browser**

1. 输入URL，浏览器是如何工作的？
2. 浏览器如何渲染页面的
3. webkit内核渲染页面过程
4. 重排（reflow）和重绘（repaint）
5. JS如何计算浏览器的渲染时间的
6. 浏览器缓存机制
7. 强缓存，协商缓存什么时候用哪个
8. 内存泄露的排除定位和解决方法
9. 垃圾回收机制
10. 前端性能优化问题
11. 项目中是是如何优化页面的

**网络·net**

1. HTTP状态码
2. http状态码301 302的区别，304是啥
3. HTTP请求头，响应头里面有哪些
4. 三次握手
5. https的请求过程
6. URL长度限制
7. 简单说一下TCP UDP
8. 浏览器如何处理未知的的tab

**框架·frame**

**Vue**

1. vue的特点
2. jQuery和vue的区别
3. Vue的生命周期
4. vue跨组件通信实现
5. vue怎么样实现数组绑定
6. vue双向绑定原理，vue-loader做了什么
7. vue的props和slot的使用区别
8. vuex的状态管理的原理是什么?
9. vue diff算法
10. vue-router不能解决情况和边界情况

**Webpack**

1. webpack工程构建工具怎么样用
2. webpack的原理
3. webpack的插件大致流程
4. webpack项目太大了怎么办
5. webpack的plugin和loader的区别
6. webpack的打包
7. webpack css兼容

**算法·algo**

1. hashmap原理
2. 数据结构有哪些?
3. 红黑树和二叉树的区别?
4. 二叉搜索树与二叉平衡树的区别？
5. 二叉树层序遍历
6. 栈和队列有什么区别，具体的应用场景
7. 排序算法有哪些，时间复杂度，选择排序怎么样搞
8. 递归，尾递归问题
9. 如何反转链表
10. 快排和冒泡原理
11. 有序的数组如何进行查找操作

**代码题·demo**

1. 如何获得用户来源（navigator.userAgent）
2. 如何实现移动端布局，适配方案？
3. 画布濡染有了解吗？
4. 详细说明解决跨域的方式
5. 兼容IE的方法
6. 如何获取当前日期？
7. 如何实现倒计时，为什么不准，校正方式
8. 假设两台电脑之间同步画板如何实现？
9. 给你一个ul列表，找到点击的li对象，把点击到的对象的字符串翻转，讲出思路并且写出代码
10. 写一个API，实现jQuery的$(selector)选择器，要求兼容IE6
11. 模拟一个双向链表
12. 预约系统如何解决高并发的问题
13. 你项目里用到第三方登录涉及的oAuth(JWT)协议的实现原理，以及你本地的实现原理，第三方登录怎么样保证安全性。
14. 现在有一大段文字，如何在页面中设置一个窗口滚动播出这段文件（轮播），轮播图如何解决卡顿问题，有手写过轮播图你，原理是什么