一面

基础知识考察，所有试题由浅入深，可以挑3个左右的方向面试一下

dom

- svg和canvas的区别？优缺点?
  - 前者是矢量图，不失真，后者为位图（标量图）；
  - 前者为好多标签组成，后者只有一个标签；
  - 前者适合大规模数据展示，后者适合小规模，东西太多耗内存；
- 定义<!DOCTYPE>。以此引出浏览器渲染机制，domtree，cssdomTree，renderTree，什么是回流和重绘，如何减少回流重绘。
- Dom事件级别，举例（dom0，dom2，dom3），如何自定义事件，event对象的常见应用

css

- 什么是BFC
  - Block formatting contexts（块级格式化上下文）：首先BFC是一个独立的布局环境，BFC中元素的布局是不受外界影响的。
  - 如何创建一个BFC
    - float的值不为none
    - position的值不为static或者relative
    - display的值为table-cell、table-caption、inline-block、flex、inline-flex中的一个
    - overflow的值不为visible
  - BFC的使用场景
    - 使用BFC来防止外边距折叠
    - 使用BFC来包含浮动，解决容器高度塌陷的问题
    - 使用BFC来防止文字环绕
    - 在多列布局中使用BFC，解决最后一列被挤到下一行的问题
- css的优先级是如何计算的?
  - 优先级就近原则，同权重情况下样式定义最近者为准；
  - 载入样式以最后载入的定位为准；
  - 优先级为: !important > id > class > tag；
  - !important比标签内Style权重高
- flex布局用过吗，子容器有个复合属性flex，他的三个属性值分别代表什么?
  - 是flex-grow，flex-shrink，felx-basis三个属性的简写；
  - flex-grow代表：弹性盒子项的拉伸因子，即子项分配父项剩余空间的比，默认值为0；
  - flex-shrink代表：弹性盒子项的缩小因子，即当所有子项相加的宽度大于父项的宽度，每个子项的缩小占比 ，默认值为1；
  - felx-basis代表：元素在主轴方向上的初始大小，即子项的宽度，默认值auto；如果这个值不为auto时元素的width属性将失效，即他的权重比width高；
- css 里的定位，浮动，有哪些，区别是什么；

- 附加一个场景页面存在内容区和导航条，导航条位于页面底部高50px，内容区可以滚动；要求用尽可能多的方法去实现；
  - 答： fixed; absolute; flex; calc(100vh - 50px)
- 导航条内有三个图片 30px * 30px，均匀分布，把导航四等分；用尽可能多的方法实现
  - 答： flex; absolute; text-align-last;
- 导航条内图片点击时出现背景从中心扩散到整个按钮的效果
  - 答：加一个 div 作为背景层，点击时有动画
  - 答2：当前标签加动画， 在 background 里画圆
- 如何实现一个轮播图； 三张图片，停留3秒，切换动画0.3s；
  - js+css; css animation

跨域

- 由来
  - 什么是同源策略及限制，什么组成了源，跨域会导致哪些不能无法操作。
  - 前后端通信的几种方式，跨域通信的几种方式
- 解决方法
  - jsonp：实现原理，有什么局限； 只能发get请求
  - cors：实现方式，配置跨域头，这些跨域头的含义，跨域携带cookie的问题
  - 转为同域： 做代理

javascript

- 引起内存泄漏的操作有哪写
  - 全局变量引起的内存泄漏
  - 闭包引起的内存泄漏
  - 被遗忘的定时器
  - 未清理的dom元素引用
  - dom清空或删除时，事件未清除导致的内存泄漏
- 基础数据类型和引用数据类型的区别
  - 基本数据类型是存放在栈中的简单数据段，数据大小确定，内存空间大小可以分配，它们是直接按值存放的，所以可以直接按值访问
  - 引用数据类型是存放在堆内存中的对象，变量保存的其实是栈内存中的一个指针，这个指针指向堆内存。
- 数据类型相关
  - instanceof原理是什么？右边变量的 prototype 在左边变量的原型链上即可
  - function与object如何判断？Object.prototype.toString
    - Object.prototype.toString.call('leon'); //"[object String]"
    - Object.prototype.toString.call(1234); //"[object Number]"
  - 类型比较（==），比较原则：string与boolean比较需要先转成number，string、boolean与number比较需要转成number，object与其他类型比较需要先转为Primitive(具体通过valueOf和toString方法)
- 继承方式
  - 原型链、call&apply、原型+构造函数、共享原型+构造函数、es6
  - 各自实现方式的优缺点

- promise 与 async-await的使用场景
  - promise： 解决异步回调嵌套问题、三个状态修改后不可改变、更多用于需要自定义异步请求成功失败状态
  - async-await：await后面的异步过程需要包装成promise（如果后面是同步代码则自动转为promise），async返回的是promise，上下顺序执行更直观。
- 数组有哪些方法，说尽可能多的
  - 有哪些是会改变数组原始值的
  - 哪些用作遍历的，有什么区别； map, foreach, reduce, sort, filter, every, some
  - slice/splice/split 有什么区别
- js的运行机制，什么是运行栈，什么是任务队列，任务队列分为哪几种，异步任务举例。同步任务和异步任务同时存在的执行流程。
- 前端错误分类：即时运行错误，资源加载错误和他们的捕获方式
- 创建对象的几种方法，区别
- instanceof的原理【可引出prototype和__proto的含义区别】
- new运算符都发生了什么

```
new F()
obj.__proto__ = F.prototype;
F.call(obj);
```

  - 建立一个空对象 var obj = {}
  - 原型链，将obj的__proto__成员指向了构造函数的prototype
  - 将步骤1新创建的对象作为this的上下文
  - 如果构造函数return了一个新的“对象”，那么这个对象就会取代整个new出来的结果。如果构造函数没有return对象，那么就会返回步骤1所创建的对象，即隐式返回this
- 箭头函数的特殊之处，函数调用的几种方式

```
// 如下函数打印的结果
const animal = {
  name: 'cat',
  say: (...args) => console.log(this.name, 'say', ...args),
  // 将此方法转义成 es5
  oSay(...args) { console.log(this.name, 'say', ...args) }
}
animal.say(1, 2, { x: 3 }); // 打印的结果

// 转义 es5
var animal = {
  name: 'cat',
  oSay: function() {
      var args = Array.prototype.slice.call(arguments, 0)
      args.splice(0, 0, this.name, 'say')
      console.log.apply(this, args)
  }
}
```

- 作用域相关
  - this指向问题：this指当前执行环境的所有者，优先级：bind > call,apply > new > 隐式

```
function f(){
return this.a;
}
var g = f.bind({a:"azerty"});
console.log(g()); //azertyvar
o ={a:37,f:f,g:g};
console.log(o.f(),o.g()); // 37,azerty
```

接口设计

- restful api 参考todo list wiki
- 实现在一个新闻页的展示，一级标题， 二级标题， 作者， 头像， 文章内容包括文字，图片，加粗，标红
  - 设计个接口
  - 实现这个需求