前端面试讲义

一面（概览）

- 页面布局类
- CSS盒模型
- DOM事件类
- HTTP协议类
- ES6
- 面向对象类
- 通信类
- 安全类
- 算法类

二面（概览）

- 渲染机制
- 异步线程
- 页面性能
- 错误监控
- mvvm
- 工作原理
- 工作流

三面（概览）

- 业务能力
- 团队协作
- 前端架构
- 性能优化
- 职业规划

常见面试题

1. 浏览器打开kaikeba.com 在console里写段代码，统计开课吧首页用了多少种html标签

LEVEL.0

```
let tags = document.getElementsByTagName("*")

let tagnames = []
for (let i = 0; i < tags.length; i++) {
  const element = tags[i];
  tagnames.push(element.tagName)
}

let arr = []
let count = 0
for (let j = 0; j < tagnames.length; j++) {
  let tagname = tagnames[j]
  if(arr.indexOf(tagname)==-1){
    count += 1
    arr.push(tagname)
  }

}
console.log(count)
```

优化点
* 数组查询复杂度高 改为对象

Level1

```
// 获取所有节点
let tags = document.getElementsByTagName("*")

// 遍历 获取所有节点的类型 是DIV 还是A
let tagnames = []
for (let i = 0; i < tags.length; i++) {
  const element = tags[i];
  tagnames.push(element.tagName)
}
// 去重的逻辑(还有level0 就是用数组记录是否第一次出现)
let obj = {}
let count = 0
for (let j = 0; j < tagnames.length; j++) {
```

```
  let tagname = tagnames[j]
  if(!obj[tagname]){
    // tagname第一次出现 累加
    count += 1
    obj[tagname] = true
  }

}
console.log(count)
```

优化点

- 节点到节点类型 可以用map做映射
- 新的数据结构Set

Level2

```
 let tags = [...document.getElementsByTagName("*")].map(v=>v.tagName)
 let count = new Set(tags).size
 console.log(count)

// 甚至可以骚包成一行
 let count = new Set([...document.getElementsByTagName("*")].map(v=>v.tagName)).siz
e
```

1. 冒泡排序 传统算法 挨个对比交换

```
let arr = [5,6,1,2,3,10,8]

for (let j = 0; j < arr.length-1; j++) {
  for (let i = 0; i < arr.length-1-j; i++) {
    if(arr[i]>arr[i+1]){
      [ arr[i], arr[i+1] ] = [ arr[i+1], arr[i] ]
    }
  }
}
```

1. react VS vue

JSX 虚拟DOM 单项数据流 VS 响应式 双向绑定 template模板

1. 输入URL后，发生了什么事

- 域名DNS解析
- TCP三次握手 简历链接
- 后端接受响应 拼接html 和header的Content-Length属性
- 浏览器获取html内容开始渲染
    - 解析html内容 产生dom树
    - 解析css 产生CSS Rule Tree
    - DOM和CSSOM合并后的render Tree
    - 浏览器计算layout 开始渲染