JavaScript三大技术框架分析

React

React于2013年5月开源，由Facebook开发维护。

- 优点：
    i. 高效，快捷灵活
    ii. 组件化，性能强
    iii. 强调SEO，优化搜索引擎
    iv. 界面高度响应，负载时间少

- 缺点：
    i. 项目体积大
    ii. 如果要写大型应用要加上ReactRouter和Flux框架使用

- 使用场景：
  react的使用基本上是大型项目的首选，组件化和灵活性是大型项目的条件，其次，react native可以让react运行在移动设备上。

Vue

Vue是2014年2月开源的，由在AngularJS工作过的尤雨溪团队维护。

- 优点
    i. 简单易于上手，普及度高
    ii. 依赖性小，性能好
    iii. 组件化，复用性强

- 缺点：
    i. 对于大应用，模板不太易于调试，不易于重构和解构
    ii. SEO 难度较高
    iii. 初次加载耗时多

- 使用场景：
  vue的全家桶Vue-router,Vuex,服务端渲染，以及vue的虚拟dom，组件化，性能，不差于react，对于开发中型前端项目来说，vue是一个很好的选择。

Angular

Angular是2012年开源的，由Google开发维护。

- 优点：

  i. 是一个比较完善的前端框架，包含服务，模板，数据双向绑定，模块化，路由，过滤器，依赖注入等所有功能；
  ii. ng模块化比较大胆的引入了Java的一些东西（依赖注入），能够很容易的写出可复用的代码，对于敏捷开发的团队来说非常有帮助。
  iii. 模板功能强大丰富，自带了极其丰富的angular指令。

- 缺点：

  i. angular入门很容易，但深入后概念很多, 学习中较难理解代码量大
  ii. 对于特别复杂的应用场景,性能差

- 使用场景：
  当项目对性能要求不高的时候，可以使用angular

React VS Angular

React.JS是基于构建一些可重用的代码块，它通过客户端和服务器端渲染减少了页面加载时间，强调SEO，在处理搜索引擎优化时，这可能是很大的优势。与Angular.JS相比，React.JS还更加专注于用户界面，使界面高度响应，负载时间减少，中断次数减少。

React VS Vue

如果拿react跟vue比较的话，使用起来会相对复杂，比如，不能使用指令，遍历不方便，而vue相对react而言，没有react灵活，搭配自如，但是它开发起来很高效，各有优势不相上下。

总结：

框架的选型不仅要看项目本身，还要综合公司团队，团队的技术栈可能直接导致项目框架的选型。

建议

如果对项目的框架没有太大的要求，建议选择vue,因为vue的插件，组件，生态系统对于我们一般的项目已经足够了，虽然vue的是个人主导的，react是Facebook团队维护的，社区比较繁荣，但vue适合很多项目，而且Vue是Github上最受欢迎的开源项目之一。同时， 在JavaScript框架/函数库中，根据2018年度在GitHub上新增star数量，Vue所获得的星标数已超过React，并高于Backbone.js、Angular 2、jQuery等项目。Vue爆发力最强,使用率高。

前端面试题

js

1. 看下列代码，将会输出什么?

```
var a = 1;
function fun(){
      console.log(a);
      var a = 2;
      console.log(a);
}
fun();
```

答案：输出undefined 和 2。（考点：1、变量作用域 2、变量声明提升）

2. 写出下面函数的执行结果。

> 解：在 JavaScript 中，研究 this 一般都是 this 的指向问题，核心就是 this 永远指向最终调用它的那个对象，除非改变 this 指向或者箭头函数那种特殊情况,函数调用的环境不同，所得到的结果也是不一样的

```
function test() {
    console.log(this);
}
var obj = {
    foo: function (){console.log(this.bar)},
      bar: 1
};
var foo = obj.foo;
var bar = 2;

test()
obj.foo()
foo()
```

答案： window，1，2

3. 写出下面函数的执行结果。

```
foo.prototype.a = function(){console.log(5)};
foo.a = function(){console.log(8)};
function foo(){
```

```
  this.a = function(){console.log(1)};
  foo.a = function(){console.log(2)};
  a = function(){console.log(3)};
  var a = function(){console.log(8)};
};
foo.a();
var obj = new foo();
obj.a();
foo.a();
```

答案：8，1，2

4. 已知有字符串foo=”get-element-by-id”,写一个function将其转化成驼峰表示法”getElementById”。(用代码写出来)

```
function combo(msg){
    var arr=msg.split("-");
    for(var i=1;i<arr.length;i++){
        arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substring(1);
    }
    msg=arr.join("");
    return msg;
}
```

5. 有这样一个URL：`https://www.google.com/search?a=1&b=2&c=&d=xxx&e`，请写一段JS程序提取URL中的各个GET参数(参数名和参数个数不确定)，将其按key-value形式返回到一个json结构中，如{a:'1', b:'2', c:'', d:'xxx', e:undefined}(用代码写出来)。

```
function serlize(url) {
   var result = {};
   url = url.substr(url.indexOf("?") + 1);
   var args = url.split("&");
    for (var i = 0, len = args.length; i < len; i++) {
        var arg = args[i];
        var item = arg.split('=');
        result[item[0]] = item[1];
     }
     return result;
}
serlize('https://www.google.com/search?a=1&b=2&c=&d=xxx&e');
```

6. 如何消除一个数组里面重复的元素？(用代码写出来)

```
var arr = [1, 2, 3, 3, 4, 4, 5, 5, 6, 1, 9, 3, 25, 4];
```

```
function deRepeat() {
        var newArr = [];
        var obj = {};
        var index = 0;
        var l = arr.length;
        for (var i = 0; i < l; i++) {
          if (obj[arr[i]] == undefined){
            obj[arr[i]] = 1;
            newArr[index++] = arr[i];
          } else if (obj[arr[i]] == 1)
            continue;
        }
        return newArr;
  }
  var newArr2 = deRepeat(arr);
  console.log(newArr2);
```

7. 说说javascript中常见的内存泄露陷阱

   - 内存泄露会导致一系列问题，比如：运行缓慢，崩溃，高延迟
   - 内存泄露是指你用不到（访问不到）的变量，依然占居着内存空间，不能被再次利用起来
   - 意外的全局变量，这些都是不会被回收的变量（除非设置 null 或者被重新赋值），特别是那些用来临时存储大量信息的变量
   - 周期函数一直在运行，处理函数并不会被回收，jq 在移除节点前都会，将事件监听移除
   - js 代码中有对 DOM 节点的引用，dom 节点被移除的时候，引用还维持

8. 跨域问题，谁限制的跨域，怎么解决

   浏览器的同源策略导致了跨域，用于隔离潜在恶意文件的重要安全机制
   解决的方法：

   - jsonp ，允许 script 加载第三方资源
   - nginx 反向代理（nginx 服务内部配置 Access-Control-Allow-Origin）
   - cors 前后端协作设置请求头部，Access-Control-Allow-Origin 等头部信息
   - iframe 嵌套通讯，postmessage

9. 说说你记得的所有的排序，他们的原理是什么?

   - 冒泡排序：双层遍历，对比前后两个节点，如果满足条件，位置互换，直到遍历结束。
   - 快速排序：取数组中间的那一个数，然后遍历所有数，小于该数的push到一个数组，大

于该数的push到另外一个数组，然后递归去排序这两个数组，最后将所有结果连接起来。

- 选择排序：声明一个数组，每次取数组里面找数组中的最大值或者最小值，取出来后push到声明的数组中，直到输入数组为空。

CSS

1. 两种以上方式实现已知或者未知宽度的垂直水平居中。

```
//方法1
.wraper {
    position: relative;
    .box {
      position: absolute;
      top:50%;
      left:50%;
      width:100px;
      height:100px;
      margin: -50px 50px;
    }
}

// 方法2
.wraper {
    display: table;
    .box {
      display: table-cell;
      vertical-amiddle;
    }
}

// 方法3
.wraper {
  .box {
    display: flex;
    jucontent:center;
    align-items: center;
    height:100px;
  }
}

// 方法4
.wraper {
  position: relative;
  .box {
    position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
  }
```

```
}
```

2. 说说svg和canvas各自的优缺点?
   svg:

   - 优点:
     矢量图，不依赖于像素，无限放大后不会失真。以dom的形式表示，事件绑定由浏览器直接分发到节点上。
   - 缺点:
     dom形式，涉及到动画时候需要更新dom，性能较低。

   canvas

   - 优点:
     定制型更强，可以绘制绘制自己想要的东西。非dom结构形式，用JavaScript进行绘制，涉及到动画性能较高。
   - 缺点:
     事件分发由canvas处理，绘制的内容的事件需要自己做处理。依赖于像素，无法高效保真，画布较大时候性能较低。

3. CSS优化、提高性能的方法有哪些

   - 移除空的css规则（Remove empty rules）
   - 正确使用display的属性
   - 不滥用浮动、web字体
   - 不声明过多的font-size
   - 不在选择符中使用ID标识符
   - 遵守盒模型规则
   - 尽量减少页面重排、重绘
   - 抽象提取公共样式，减少代码量

Vue

1. 组件之间通信，父组件如何向子组件通信，子组件如何向父组件通信，非父子组件、兄弟组件之间的如何进行数据传递?

- 父组件向子组件通信
  子组件通过 props 属性，绑定父组件的数据，实现通信。

- 子组件向父组件通信

  将父组件的事件在子组件中通过 $emit 触发。

- 非父子组件、兄弟组件之间的数据传递

```
let event =new Vue();
event.$on('eventName',(value){

});
event.$emit('eventName','this is a message.')
```

2. 分别写一个Vue中的全局和局部自定义指令
   全局：

```
Vue.directive('focus', {
    inserted: function(el) {
      el.focus();
    }
})
```

   局部：

```
directives: {
  focus: {
    inserted: function (el){
        el.focus();
    }
  }
}
```

3. Vue中如何监控某个属性值的变化？比如现在需要监控data中， obj.a 的变化。

   方法1：

```
watch: {
      obj: {
        hand(newValue,oldValue) {
```

```
            console.log('obj changed')
        },
      deep:true
    }
}
```

方法2：

```
watch: {
  obj.a': {
      handler (newName, oldName) {
        console.log('obj.a changed')
      }
    }
}
```

方法3:
通过computed来实现,当依赖改变时，便会重新计算一个新值。

```
computed: {
    a1 () {
      return  this.obj.a
    }
}
```

4. js实现简单的vue的双向绑定 (用代码写出来)

```
<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>
```

5. vue如何自定义一个过滤器？(用代码写出来)

html代码：

```
<div id="app">
     <input type="text" v-model="msg" />
     {{msg| capitalize }}
</div>
```

JS代码：

```
var vm=new Vue({
    el:"#app",
    data:{
        msg:''
    },
    filters: {
      capitalize: function (value) {
        if (!value) return ''
        value = value.toString()
        return value.charAt(0).toUpperCase() + value.slice(1)
      }
    }
})
```

全局定义过滤器:

```
Vue.filter('capitalize', function (value) {
  if (!value) return ''
  value = value.toString()
  return value.charAt(0).toUpperCase() + value.slice(1)
})
```

React

1. 说说react diff 原理

    - 把树形结构按照层级分解，只比较同级元素。
    - 给列表结构的每个单元添加唯一的 key 属性，方便比较。

- React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
- 合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.
- 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

2. 为什么虚拟 dom 会提高性能?
   虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

   用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

3. 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，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

   这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。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。

20道题

- js : 9
- css : 3
- Vue : 5
- React : 3
  这些都是从网上收集整理的一些不是很难的题，但是包括重要的知识点的考查，都是一些面试常考的类型。由于对Angular不是很了解，就没有收集，React了解一点，整理了3道经典题。