1. 列举html中至少三个实体

> `div,img,span` ;其中也分别对应块元素，行内块元素，行内元素

2. 我们经常看到火车都是一串的，它们由一节节车厢连成一列火车组。每节车厢连接是有规则的，规则如下：每种车厢都有它唯一标识的一个单词，现在给定一个单词开头的字母，求出以这个字母开头的最长的火车组（每种车厢最多在火车组中出现两次），在两个单词相连时，其重合部分算作一部分，例如money和neymar，如果接成一列火车组则变为moneymar，另外相邻的两部分不能存在包含关系，例如break和breakfast 不能相连。请写一个函数，将给定的数据录入并实现该功能。（必答题）

> **题目意思：**数组里面有n个元素（每个元素是字符串类型），每个元素至多可以重复取两次，求取出元素拼接成最大的长度；其中拼接的规则有：相邻元素存在包含关系的元素不能拼接；相邻元素的头尾重合部分的长度计算公式为相邻元素的长度之和减去字符串重合部分
>
> **算法思路：** 假设字符数组有n个元素，那儿使用递归算法算出2*n个元素的排列组合出来的拼接的字符串数组；然后判断每次取出的最长的字符串去除两个条件后计算出的最大长度，是否符合最大，不符合，继续查找

具体实现：

```
        var arr = ['winney', 'winneyfung', 'break', 'zhuzu', 'moon'];

        const getRepeat = function (preWord, nextWord) {
            let len = preWord.length;
            let repeat = 0;
            let lastChar = preWord[len - 1];
            let nlIdx = nextWord.indexOf(lastChar);
            if (nlIdx > -1 && nlIdx != 0) {
                repeat++;
                let j = nlIdx - 1;
                for (let i = len - 2; i >= 0 && j >= 0; i-- , j--) {
                    if (nextWord[j] !== preWord[i]) {
                        return repeat;
                    } else {
                        repeat++;
                    }
                }
            } else if (nlIdx == 0) {
                return 1;
            }
            return repeat;
        }

        const isContains = function (preWord, nextWord) {
            return (preWord.lastIndexOf(nextWord) != -1 || nextWord.lastIndexOf(preWord) !=
-1);
        }
        const permuteWithRule = function (words) {
            let result = "";
            const process = function (words, prevReuslt) {

                if (words.length > 1) {
```

```
                    words.forEach((word, i) => {
                        let currentResult;
                        let remianWords = words.slice();
                        if (!prevReuslt) {
                            currentResult = [].concat(word);
                            remianWords.splice(i, 1);
                            process(remianWords, currentResult);
                        } else {
                            currentResult = prevReuslt;
                            const curLen = currentResult.length;
                            let lastWord = currentResult[curLen - 1];
                            if (!isContains(lastWord, word)) {
                                currentResult =
currentResult.concat(word.substr(getRepeat(lastWord, word)));
                                remianWords.splice(i, 1);
                                process(remianWords, currentResult);
                            }
                        }
                    });
                } else {
                    let row = prevReuslt ? prevReuslt.concat(words) : [].concat(words);
                    row = row.join("");
                    if (row.length > result.length) {
                        result = row;
                    }
                }
            }
            process(words);
            return result;
        }

        const getMaxTrain = function (words) {
            return permuteWithRule([...words, ...words]);
        }
        console.log(getMaxTrain(arr));
```

3. 怎么在网页中实现绝对定位?

> 给元素设置样式`position:absolute;`，该绝对定位元素的设置相对位移的话，一般是相对离它最近的非staitc的祖先元素;

4. HTTP状态码知道哪些?

> 200 请求成功
>
> 301 没有权限
>
> 304 自上次请求，请求的内容没有修改过
>
> 404 找不到对应请求的服务
>
> 500 服务器异常
>
> 503 服务不可用

5. get和post的区别?

> ge的请求数据会直接暴露在请求的url中，post的请求数据，放在请求体中，并且get发送的请求的数据受浏览器url的长度限制而有有大小限制，post没有；本质区别是get请求一般具有幂等性（主要看后端配置http协议时是怎么配置的，一般都是配置幂等），所谓幂等性指的是请求某个资源的时候，应该具有同等的副作用；正因为get具有幂等性；所以在网络不好的情况下，浏览器会尝试重连，所以如果用get请求去做增操作，后果可想而知；所以restful风格的接口也有规范，获取数据用get请求，提交表单等增操作使用post请求；

6. display与visibility 有何异同?

> display是控制一个元素的显示方式，属性值可以是`block、inline、none`等等；visibility则控制一个元素的可见性，属性值可以是`visible、hidden`;而且一个元素只要设置了`display: none`,那么它的后代元素都会隐藏,并且它的空间不会继续保留，这也导致了浏览器会触发重排重绘；而一个元素就算设置了`visibility:hidden`，它会隐藏，但是只要它的后代设置了`visibility: visible`，那么它的后代是可见的；并且设置了`visibility:hidden`的元素的空间是继续保留的，被空白占位；并且只会触发重绘；

7. 你有哪些性能优化的方法.

> 1. 在操作dom时，注意缓存查询dom的结果
> 2. 批量更新dom的时候，尽量使用模板语法，更新样式，尽量通过类名修改
> 3. 小图标之类的可以使用精灵图，或者用base64编码代替http请求；也可以采用`webp`等新图片格式，同一个页面多图片，可以实现图片懒加载；
> 4. 关键的样式文件在head中引入；
> 5. 脚本文件放在`<body>`后引入；可以使用异步加载脚本；
> 6. 脚本文件，样式文件压缩；
> 7. 使用cdn加速；
> 8. 计算复杂的结果可以缓存；
> 9. 服务器配置gzip压缩；