网易严选前端实习一面

估计一轮游了，这次被问得一脸懵逼，半个小时就结束了。网易忙着严选五周年活动，笔试和面试隔了2周多

自我介绍

怎么学习前端的

不用自带函数，手写原地翻转数组的一部分

用了双指针，第三变量交换法

问我有什么别的方法。

解构赋值，

利用和或者位运算

```
arr[i] += arr[j];
arr[j] = arr[i] – arr[j];
arr[i] = arr[i] – arr[j];
```

或者

```
arr[i] ^= arr[j];
arr[j] ^= arr[i];
arr[i] ^= arr[j];
```

知道ES的class吧，将下面用ES5表示

```
class person {
    constructor (name) {
        this.name = name;
    }
    call () {
        return ”I\’m” + this.name;
    }
    static isPerson() {
        return true;
    }
}
```

我只是知道有static这个静态方法，不知道怎么用，瞎写的

```
function person(name) {
    this.name = name;
    this.isPerson() {
        return true;
    }
}
person.prototype.call (name) {
    return "I\'m" + this.name;
}
```

这样两种方法被没有区分度，面试官说直接静态方法挂载在对象上

明白了，我太菜了

```
function person(name) {
    this.name = name;
}
person.prototype.call (name) {
    return "I\'m" + this.name;
}
person.isPerson() {
    return true;
}
```

JSON.stringify有哪些应用场景

转成JSON格式，结合JSON.parse实现简单对象的深拷贝，不知道有什么用处了

查MDN文档

JSON.stringify用作 JavaScript

注意 JSON 不是 JavaScript 严格意义上的子集，在 JSON 中不需要省略两条终线（Line separator 和 Paragraph separator），但在 JavaScript 中需要被省略。因此，如果 JSON 被用作 JSONP 时，下面方法可以使用：

```
function jsFriendlyJSONStringify (s) {
    return JSON.stringify(s).
        replace(/\u2028/g, '\\u2028').
        replace(/\u2029/g, '\\u2029');
}

var s = {
    a: String.fromCharCode(0x2028),
    b: String.fromCharCode(0x2029)
};
try {
    eval('(' + JSON.stringify(s) + ')');
} catch (e) {
    console.log(e); // "SyntaxError: unterminated string literal"
}

// No need for a catch
eval('(' + jsFriendlyJSONStringify(s) + ')');

// console.log in Firefox unescapes the Unicode if
//   logged to console, so we use alert
alert(jsFriendlyJSONStringify(s)); // {"a":"\u2028","b":"\u2029"}
```

y 结合 localStorage 的例子

一些时候，你想存储用户创建的一个对象，并且，即使在浏览器被关闭后仍能恢复该对象。下面的例子是 JSON.stringify 适用于这种情形的一个样板：

```
// 创建一个示例数据
var session = {
    'screens' : [],
    'state' : true
};
session.screens.push({"name":"screenA", "width":450, "height":250});
session.screens.push({"name":"screenB", "width":650, "height":350});
session.screens.push({"name":"screenC", "width":750, "height":120});
session.screens.push({"name":"screenD", "width":250, "height":60});
session.screens.push({"name":"screenE", "width":390, "height":120});
session.screens.push({"name":"screenF", "width":1240, "height":650});

// 使用 JSON.stringify 转换为 JSON 字符串
// 然后使用 localStorage 保存在 session 名称里
localStorage.setItem('session', JSON.stringify(session));
```

```
// 然后是如何转换通过 JSON.stringify 生成的字符串，该字符串以 JSON 格式保存在
localStorage 里
var restoredSession = JSON.parse(localStorage.getItem('session'));

// 现在 restoredSession 包含了保存在 localStorage 里的对象
console.log(restoredSession);
```

那么什么情况下JSON.stringify会有bug呢

开始瞎回答了

说是循环引用，比如

```
var obj = {
    obj: {
        obj: 1
    }
}
```

现场测试可以，我那个尴尬。

然后写了symbol类型，这次报了错（蒙对了一个）

具体还是再次参考MDN文档

JSON.stringify()将值转换为相应的JSON格式：

- 转换值如果有 toJSON() 方法，该方法定义什么值将被序列化。
- 非数组对象的属性不能保证以特定的顺序出现在序列化后的字符串中。
- 布尔值、数字、字符串的包装对象在序列化过程中会自动转换成对应的原始值。
- undefined、任意的函数以及 symbol 值，在序列化过程中会被忽略（出现在非数组对象的属性值中时）或者被转换成 null（出现在数组中时）。函数、undefined 被单独转换时，会返回 undefined，如JSON.stringify(function(){}) or JSON.stringify(undefined).
- 对包含循环引用的对象（对象之间相互引用，形成无限循环）执行此方***抛出错误。
- 所有以 symbol 为属性键的属性都会被完全忽略掉，即便 replacer 参数中强制指定包含了它们。
- Date 日期调用了 toJSON() 将其转换为了 string 字符串（同Date.toISOString()），因此会被当做字符串处理。
- NaN 和 Infinity 格式的数值及 null 都会被当做 null。
- 其他类型的对象，包括 Map/Set/WeakMap/WeakSet，仅会序列化可枚举的属性。

```
JSON.stringify({});                  // '{}'
JSON.stringify(true);                // 'true'
JSON.stringify("foo");               // ""foo""
JSON.stringify([1, "false", false]); // '[1,"false",false]'
JSON.stringify({ x: 5 });            // '{"x":5}'
```

```
JSON.stringify({x: 5, y: 6});
// "{"x":5,"y":6}"

JSON.stringify([new Number(1), new String("false"), new Boolean(false)]);
// '[1,"false",false]'

JSON.stringify({x: undefined, y: Object, z: Symbol("")});
// '{}'

JSON.stringify([undefined, Object, Symbol("")]);
// '[null,null,null]'

JSON.stringify({[Symbol("foo")]: "foo"});
// '{}'

JSON.stringify({[Symbol.for("foo")]: "foo"}, [Symbol.for("foo")]);
// '{}'

JSON.stringify(
    {[Symbol.for("foo")]: "foo"},
    function (k, v) {
        if (typeof k === "symbol"){
            return "a symbol";
        }
    }
);


// undefined

// 不可枚举的属性默认会被忽略：
JSON.stringify(
    Object.create(
        null,
        {
            x: { value: 'x', enumerable: false },
            y: { value: 'y', enumerable: true }
        }
    )
);

// "{"y":"y"}"
```

请你手写form有哪些元素，

完了，HTML全忘了，憋了半天想了想说select、checkbox，让我手写怎么用的我居然不会了，问我会不会写HTML，我说平时这些都是查了之后复制粘贴的，抽自己一下，HTML基础都快忘光了，确实也很久没碰了

文本域（Text Fields）文本域通过 <input type=”text”> 标签来设定，当用户要在表单中键入字母、数字等内容时，就会用到文本域。

```
<form>
    First name: <input type=”text” name=”firstname”><br>
    Last name: <input type=”text” name=”lastname”>
</form>
```

浏览器显示如下：

First name: Last name:

注意:表单本身并不可见。同时，在大多数浏览器中，文本域的默认宽度是 20 个字符。

密码字段

密码字段通过标签 <input type=”password”> 来定义:

```
<form>
    Password: <input type=”password” name=”pwd”>
</form>
```

注意:密码字段字符不会明文显示，而是以星号或圆点替代。

单选按钮（Radio Buttons）

<input type=”radio”> 标签定义了表单单选框选项

```
<form>
    <input type=”radio” name=”sex” value=”male”>Male<br>
    <input type=”radio” name=”sex” value=”female”>Female
</form>
```

<input type=”checkbox”> 定义了复选框. 用户需要从若干给定的选择中选取一个或若干选项。

```
<form>
```

```
    <input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
    <input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

提交按钮(Submit Button)

<input type="submit"> 定义了提交按钮.

当用户单击确认按钮时，表单的内容会被传送到另一个文件。表单的动作属性定义了目的文件的文件名。由动作属性定义的这个文件通常会对接收到的输入数据进行相关的处理。:

```
<form name="input" action="html_form_action.php" method="get">
    Username: <input type="text" name="user">
    <input type="submit" value="Submit">
</form>
```

实现未知长高子元素父元素居中对齐，我思路有两个，结果又写错了

使用transform大法

```
.child{
    left :50%,
    top: 50%;
    transform: translate(-50%, -50%);
}
```

使用flex布局

```
.parent {
    display: flex;
    justify-content: center;
    align-items: center;
}
```