实习题目

不使用**sizeof**来输出当前**int**的长度

“

```
int main(){
    num = 10;
    $i=1;
    while(num!=0){
        num<<1;
        i++;
    }
    printf("%d",i);
}
main();
// 可以根据左移右移来判断有几位
```

*js*跨域的方法

- 基于*iframe-location.hash(document.domain)*的实现/修改*document.domain*来跨子域

```
A页面：
  <iframe src="b.com/b.html" id="a"></iframe>
  <script type="text/javascript">
      var a = document.getElementById("a").onload = function(){
          a.b();
      }
  </script>
B页面：
  <script type="javascript">
      document.domain = "a.com";
      function b(){}
  </script>
```

- *window.name*
- 使用*HTML5*的*window.postMessage*方法跨域// 例子见该文件统一目录的*a.html && b.html*
- *jsonp*,就是把跨域的*js*当成一个节点使用，并返回结果*(*数据*)*。

```
// 服务器端：
  <?php
  function() {
      $str = "callback({
          some json || some xml || array
      })";
      echo $str;// 目前只用过json
  }
  ?>
// 请求端：
  <script type="text/javascript">
  $.ajax({
```

```
        url:"this url",
        dataType:'jsonp',
        jsonp:"jsoncallback",
        success: function(data){
            console.log(data);
        },
        error: function(err) {
            console.log(err);
        }
    })
    </script>
```

attachEvent和addEventListen的区别

“

- attachEvent是ie支持的属性，addEventListen是非ie支持的属性(联系到事件冒泡和事件捕获)
- 两种方法都是监听处理函数的叠加而不是覆盖
- 可用它监听dom节点事件
- node.attachEvent(eventType,fn);node.addEventListen(eventType,fn,false);

处理表单输入的函数

“

- php的tram()//好像还有clearHtml()
- js可以写个正则匹配

事件冒泡和事件捕获

```
<html>
<body>
    <div>
        <p></p>
    </div>
</body>
</html>
```

“

- 冒泡：子级元素先触发，父级元素后触发 p->div->body
- 捕获：父级元素先触发，子级元素后触发 body->div->p
- 阻止事件冒泡：在ie下是使用event.cancelBubble = true,ff下是event stopPropagation()
- IE只支持事件冒泡，不支持事件捕获

ajax返回的类型

“

- 原生的*ajax*返回*responseText*，*responseXML*，即*text*和*xml*
- 然后*json*是由之前返回的*JSON.parse(text)*
- 其他的*(json*、*html*、*jsonp*、*script)*都是字符串传输过来在客户端进行转换和处理的

jssdk接口有哪些

“

- 基础接口*(*判断浏览器是否支持*js*接口*)*
- 分享接口*(*分享到朋友圈那些*)*
- 图像接口*(*图片上传，拍照那些*)*
- 音频接口*(*当然是声音的播放那些*)* + 智能接口*(*识别音频并返回*)*
- 地理接口*(*类似*html5*的*geolocation)*
- 微信支付*(*扫一扫那些*)*

js的数据对象有哪些

“

- 一开始我有点懵了，我以为是*object*对象那些
- *String(),Number(),Boolen(),Object(),Array(),function(),Math(),RegExp(),Date()*
- 事件监听，捕获/冒泡，冒泡阻止，事件委托
- 常去的技术博客*(*我竟然想到的是--真阿当*)*
- 一分钟自我介绍*(*感觉都会来这套，然而我并不知道应该说什么*)*

```
function test(){
  alert("aaaaaa");
}
function test(){
  alert("bbbbbb");
}
test();//会返回什么？
// 因为加载的后面的js程序会覆盖前面的函数，而不是依次执行
// 有点类似onclick的那个事情
```

深拷贝、浅拷贝

“

- 自我理解是浅拷贝只是值(对象)拷贝，深拷贝是地址、原型链那些都拷贝，所以深拷贝要加上 *hasOwnProperty*
- 浅拷贝是指源对象与拷贝对象共用一份实体，仅仅是引用的变量不同(名称不同)。
- 深拷贝是指源对象与拷贝对象互相独立，其中任何一个对象的改动都不会对另外一个对象造成影响。
  ```*js Object.prototype.clone = function() { var Constructor = this.constructor,*

```
    obj = new Construct();
```

```
for(var attr in this) {
    if (this.hasOwnProperty(attr)) {
        if (typeof(this[attr]) !== "function") {
            if (this[attr] === null) {
                obj[attr] = null;
            }else {
                obj[attr] = this[attr].clone();
            }
        }
    }
}

return obj;
```

}```

预防跨站攻击

“

- 我也是这么说的啊。。。不过好像新浪的学长有些怀疑
- 通过*script*的方式进行获取，通过*script*的方式先传入一个*callback*，那么数据就会被传入的函数执行。
- 可以通过发起*JavaScript*的跨域请求来绕过同源策略,会导致不同源或域之间的数据泄露
- 在*jsonp*请求中加入一个随机数
- 在*jsonp*中不要加入个人数据

css盒模型

“

- *http://www.osmnoo.com/translation/213.html*
- *https://zh.wikipedia.org/wiki/IE%E7%9B%92%E6%A8%A1%E5%9E%8B%E7%BC%BA%E9%99%B7*
- *w3c*标准：*margin->border->padding->width/height->content* (从外到内)
- *ie*标准：*margin->width/height->content* (从外到内) ---*ie*的*padding*和*border*都包含在指定的宽度和高度
- *box-sizing*：*content-box*(默认,*w3c*标准) || *border-box*(*ie*标准) || *inherit*
```

git怎么返回上一个提交的版本

“

- *git reset --hard HEAD^* //返回上个版本
- *git reset --hard 3228164* // 返回这个版本
- *git reset HEAD readme.txt* //撤销暂存区*readme.txt* 将其退回工作区
- *git reflog* // 退回返回的上一个版本。。类似撤销返回上个版本
- *git statu* // 查看缓存区的内容
- *git checkout -- readme.txt* //丢弃工作区中*readme.txt*的修改
- *git rm --readme.txt* //和*linux*的命令很像啊
- *git log* //查看提交记录

Tcp三次握手，四次握手

“

*

fileReader返回的是二进制

“

*

块级作用域和函数作用域

“

*

(AMD)Require和**(CMD)sea.js**的区别

“

- *https://github.com/seajs/seajs/issues/277*
- 都是模块加载
- *AMD*是提前执行，*CMD*是延迟执行
- *CMD*依赖就近，*AMD*依赖前置

数据结构

“

- 链表、栈、队列

java的继承和多态

“

-

js缓存问题

“

- *html5*缓存*appcache*
- 在*js*那加上时间戳，判断是否时间一致然后加载

网站优化