this笔记：

  首先必须要说的是，this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，实际上this的最终指向的是那个调用它的对象（这句话有些问题，后面会解释为什么会有问题，虽然网上大部分的文章都是这样说的，虽然在很多情况下那样去理解不会出什么问题，但是实际上那样理解是不准确的，所以在你理解this的时候会有种琢磨不透的感觉），那么接下来我会深入的探讨这个问题。

为什么要学习this？如果你学过函数式编程，面向对象编程，那你肯定知道干什么用的，如果你没有学过，那么暂时可以不用看这篇文章，当然如果你有兴趣也可以看看，毕竟这是js中必须要掌握的东西。

例子1：

**function** a(){

**var** user = "追梦子";

    console.log(**this**.user); *//undefined*

    console.log(**this**); *//Window*

}

a();

按照我们上面说的this最终指向的是调用它的对象，这里的函数a实际是被Window对象所点出来的，下面的代码就可以证明。

**function** a(){

**var** user = "追梦子";

    console.log(**this**.user); *//undefined*

    console.log(**this**);　　*//Window*

}

window.a();

和上面代码一样吧，其实alert也是window的一个属性，也是window点出来的。

例子2：

**var** o = {

    user:"追梦子",

    fn(){

        console.log(**this**.user);  *//追梦子*

    }

}

o.fn();

这里的this指向的是对象o，因为你调用这个fn是通过o.fn()执行的，那自然指向就是对象o，这里再次强调一点，this的指向在函数创建的时候是决定不了的，在调用的时候才能决定，谁调用的就指向谁，一定要搞清楚这个。

其实例子1和例子2说的并不够准确，下面这个例子就可以推翻上面的理论。

如果要彻底的搞懂this必须看接下来的几个例子

例子3：

**var** o = {

    user:"追梦子",

    fn:**function**(){

        console.log(**this**.user); *//追梦子*

    }

}

window.o.fn();

这段代码和上面的那段代码几乎是一样的，但是这里的this为什么不是指向window，如果按照上面的理论，最终this指向的是调用它的对象，这里先说个而外话，window是js中的全局对象，我们创建的变量实际上是给window添加属性，所以这里可以用window点o对象。

这里先不解释为什么上面的那段代码this为什么没有指向window，我们再来看一段代码。

**var** o = {

    a:10,

    b:{

        a:12,

        fn:**function**(){

            console.log(**this**.a); *//12*

        }

    }

}

o.b.fn();

这里同样也是对象o点出来的，但是同样this并没有执行它，那你肯定会说我一开始说的那些不就都是错误的吗？其实也不是，只是一开始说的不准确，接下来我将补充一句话，我相信你就可以彻底的理解this的指向的问题。

情况1：如果一个函数中有this，但是它没有被上一级的对象所调用，那么this指向的就是window，这里需要说明的是在js的严格版中this指向的不是window，但是我们这里不探讨严格版的问题，你想了解可以自行上网查找。

情况2：如果一个函数中有this，这个函数有被上一级的对象所调用，那么this指向的就是上一级的对象。

情况3：如果一个函数中有this，这个函数中包含多个对象，尽管这个函数是被最外层的对象所调用，this指向的也只是它上一级的对象，例子3可以证明，如果不相信，那么接下来我们继续看几个例子。

**var** o = {

    a:10,

    b:{

*// a:12,*

        fn:**function**(){

            console.log(**this**.a); *//undefined*

        }

    }

}

o.b.fn();

尽管对象b中没有属性a，这个this指向的也是对象b，因为this只会指向它的上一级对象，不管这个对象中有没有this要的东西。

还有一种比较特殊的情况，例子4：

**var** o = {

    a:10,

    b:{

        a:12,

        fn:**function**(){

            console.log(**this**.a); *//undefined*

            console.log(**this**); *//window*

        }

    }

}

**var** j = o.b.fn;

window.j();

这里this指向的是window，是不是有些蒙了？其实是因为你没有理解一句话，这句话同样至关重要。

this永远指向的是最后调用它的对象，也就是看它执行的时候是谁调用的，例子4中虽然函数fn是被对象b所引用，但是在将fn赋值给变量j的时候并没有执行所以最终指向的是window，这和例子3是不一样的，例子3是直接执行了fn。

this讲来讲去其实就是那么一回事，只不过在不同的情况下指向的会有些不同，上面的总结每个地方都有些小错误，也不能说是错误，而是在不同环境下情况就会有不同，所以我也没有办法一次解释清楚，只能你慢慢地的去体会。

构造函数版this：

**function** Fn(){

**this**.user = "追梦子";

}

**var** a = **new** Fn();

console.log(a.user); *//追梦子*

这里之所以对象a可以点出函数Fn里面的user是因为new关键字可以改变this的指向，将这个this指向对象a，为什么我说a是对象，因为用了new关键字就是创建一个对象实例，理解这句话可以想想我们的例子3，我们这里用变量a创建了一个Fn的实例（相当于复制了一份Fn到对象a里面），此时仅仅只是创建，并没有执行，而调用这个函数Fn的是对象a，那么this指向的自然是对象a，那么为什么对象Fn中会有user，因为你已经复制了一份Fn函数到对象a中，用了new关键字就等同于复制了一份。

除了上面的这些以外，我们还可以自行改变this的指向，关于自行改变this的指向请看JavaScript中call,apply,bind方法的总结这篇文章，详细的说明了我们如何手动更改this的指向。

更新一个小问题当this碰到return时

**function** fn()

{

**this**.user = '追梦子';

**return** {};

}

**var** a = **new** fn;

console.log(a.user); *//undefined*

再看一个

**function** fn()

{

**this**.user = '追梦子';

**return** **function**(){};

}

**var** a = **new** fn;

console.log(a.user); *//undefined*

再来

**function** fn()

{

**this**.user = '追梦子';

**return** 1;

}

**var** a = **new** fn;

console.log(a.user); *//追梦子*

**function** fn()

{

**this**.user = '追梦子';

**return** undefined;

}

**var** a = **new** fn;

console.log(a.user); *//追梦子*

什么意思呢？

如果返回值是一个对象，那么this指向的就是那个返回的对象，如果返回值不是一个对象那么this还是指向函数的实例。

**function** fn()

{

**this**.user = '追梦子';

**return** undefined;

}

**var** a = **new** fn;

console.log(a); *//fn {user: "追梦子"}*

还有一点就是虽然null也是对象，但是在这里this还是指向那个函数的实例，因为null比较特殊。

**function** fn()

{

**this**.user = '追梦子';

**return** **null**;

}

**var** a = **new** fn;

console.log(a.user); *//追梦子*

知识点补充：

1.在严格版中的默认的this不再是window，而是undefined。

2.new操作符会改变函数this的指向问题，虽然我们上面讲解过了，但是并没有深入的讨论这个问题，网上也很少说，所以在这里有必要说一下。

**function** fn(){

**this**.num = 1;

}

**var** a = **new** fn();

console.log(a.num); *//1*

为什么this会指向a？首先new关键字会创建一个空的对象，然后会自动调用一个函数apply方法，将this指向这个空对象，这样的话函数内部的this就会被这个空的对象替代。

Http:

域名：

两种访问方式：

1. <http://www.163.com:8080/welcome/index>   (restful访问方式)
2. <http://www.163.com/index.php>  （文件的方式）（可能不安全）

协议：

1. http （无状态的协议）  socket
2. https  客户端和服务器端交互  ssl加密  [https://www.alipay.com](https://www.alipay.com/)  默认端口：443
3. OSI七层结构  http（应用层 协议 公开和不公开）
4. 公开：http ftp  tftp pop3 smtp
5. RFC文档不公开：QQ、MSN、YYtcp(RPC(表示层（YY({'name':'laoshan','age':40})））)
6. 一般我们只需要关心上4层协议
7. $.post('地址'，json数据，回调函数，类型)

软刷新和硬刷新的区别：

1. 软刷新：有缓存就用缓存  F5
2. 硬刷新：有缓存也刷新     CTRL + F5

HTTP：

分为包头和包体

1. 包头：  请求头和响应头、

如何获取当前浏览器的版本？

获取响应头中的user-agent来识别。

请求行：（请求方法，请求路径，所用协议）

请求方法：

get、post、put、delete、head、options、trace

打开telnet回显功能 ctrl+]

请求头信息

请求主体信息

响应头信息  status code 200(成功)

Content-type:text/html

Content-length:5

总结HTTP构造包

GET(请求方法、请求路径、请求协议)

POST(请求方法、请求路径、请求协议)

包头内容  Content-type  Content-length

HEAD和GET基本一致 发送请求不返回请求地址的内容

GET /welcome/index HTTP/1.1

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

HEAD /welcome/index HTTP/1.1

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

PUT 想服务器传输内容  PUT

Method not allowed   405

tatus code 200 405 302(临时重定向) 301(持久重定向) 307

当有些时候只需要看看文件是否在服务器上或者服务器是否有响应：

使用head请求（只发送请求不返回请求地址的内容）

304 not modified未修改

304 确定当前浏览器缓存里面的内容有没有改变：时间、文件

last-modify:服务器上的文件最后被改动的时间

校验文件的方法：

Etag:d890-55f46f66ba463

校验文件方法：md5

php处理的页面 和 VUE处理的页面 有什么区别

后端渲染  template+data 服务器端完成

前端渲染  template  json data chrome端完成组装

1：减少HTTP连接

2：减少回传数据的大小  webpack  js/image/css

3：优化HTTP头

GET /http1/google.jpg HTTP/1.1

Host:localhost

If-Modified-Since:Fri, 01 Dec 2017 13:01:10 GMT

If-None-Match:"d890-55f46f66ba463"

301/302(永久重定向/临时重定向) robots  SEO

302 A-->B  A内容依然爬虫还是要访问

301 A-->B  A内容爬虫不爬

POST数据 在跳转的时候不丢失

referer 防盗链 apache出重写模块 mod\_rewriter .htaccess重定向文件

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

Referer:<http://localhost/http1/5.php>

header跳转 是包头跳转 先执行

script location 是包体跳转 发起了一次GET操作 200

1：客户端 与

webservice = http + xml

restful  = http + json

POST /http/test2.php HTTP/1.1

Host:localhost

Content-type:application/x-www-form-urlencoded

Content-length:23

GET /welcome/index HTTP/1.1

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

HEAD /welcome/index HTTP/1.1

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

OPTIONS / HTTP/1.1

Host:[www.hrbyueju.com](http://www.hrbyueju.com/)

GET /http/google.jpg HTTP/1.1

Host:localhost

If-Modified-Since:Fri, 01 Dec 2017 07:27:32 GMT

If-None-Match:"5d2a-55f424d4c5df6"