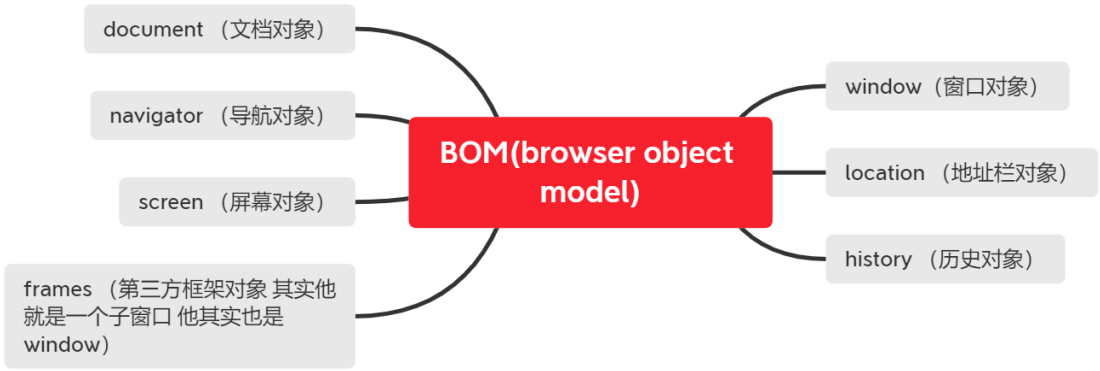
day08 BOM

BOM概述

BOM全称 (browser object model) 浏览器对象模型，**主要用操作浏览器相关的内容**（调用浏览器相关的功能及获取浏览器携带的内容），BOM是JavaScript的重要组成，但是他缺少规范，所以他是通过共有对象来解决没有规范的问题。（前面BOM相关的内容没有融入w3c 导致他有兼容问题）。ie的bom对象里面的相关参数及方法是最多的，由于是共有对象所以这些参数都被其他浏览器采用了。



BOM的共有对象

- window 窗口对象（浏览器的global对象）
- location 地址栏对象
- history 历史对象
- navigator 导航对象（浏览器信息对象）
- screen 屏幕对象 （适配）
- frames 框架对象 （其实也是个window）
- document 文档对象

window

概述:

window窗口对象，他是浏览器的global对象，他包含所有的全局变量。

属性

- caches 返回一个缓存对象 CacheStorage
- closed 是否关闭（默认值为false）
- cookieStore 存储cookie的空间
- crossOriginIsolated cors设置（跨域设置）
- **innerHeight 窗口可操作区域的高度**
- **innerWidth 窗口可操作区域的宽度**
- indexedDB 浏览器内置数据库对象
- localStorage 本地存储
- sessionStorage 他是session生命周期本地存储

- chrome 谷歌内核的属性
- console 控制台
- **scrollX 滚动栏的坐标 scrollY 滚动栏的坐标**
- ....

其他的所有对象都是window对象的属性 window可以被省略

方法

打印方法（控制台打印方法）

```
// 控制台打印方法
console.log('日志打印') //日志打印
console.error('错误打印') //错误打印
console.warn('警告打印') //警告打印
console.info('信息打印') //信息打印
console.debug('调试打印') //调试打印
console.group('分组打印') //分组打印
console.table('表格打印') //表格打印
```

弹框方法

- alert 弹信息框 没有返回值
- prompt 弹输入框 有返回值返回值为输入的内容 （string）
- confirm 弹选择框 有返回值 返回为boolean类型

```
//弹窗
console.log(alert('hello'))
console.log(prompt('请输入内容'))
console.log(confirm('请问你是否是单身贵族'))
```

打开窗口的方法 open

```
document.getElementById('openBtn').onclick = function(){
    //打开的url路径地址  打开的方式(a标签的target一致)  打开时的配置(使用=号赋值 使用,号隔开)

 window.open('http://www.baidu.com','_blank','width=100,height=200,left=200,top=200')
}
```

关闭窗口的方法 close （关闭当前窗口）

```
document.getElementById('closeBtn').onclick = function(){
    //打开的url路径地址  打开的方式(a标签的target一致)  打开时的配置(使用=号赋值 使用,号隔开)
    window.close()
}
```

改变窗口位置的方法

- moveTo
- moveBy

```
//moveTo 给定实际坐标 moveBy 给定变化的距离
window.moveTo(30,30) //到达30,30的坐标
window.moveBy(20,20) //10 10 变成 30,30的坐标
```

改变窗口大小的方法

- resizeTo
- resizeBy

```
//resizeTo 给定实际大小 resizeBy 在当前下发生变化
window.resizeTo(30,30) //大小改为width 30 height 30大小
window.resizeBy(30,30) //原本的width增加30 原本的height也增加30
```

改变的滚动栏位置的方法 (*)

- scrollTo
- scrollBy

```
document.getElementById('scrollToBtn').onclick = function(){
    window.scrollTo(300,300) //滚动栏x轴到达30 y轴到达30
    console.log(window.scrollX,window.scrollY)
}
document.getElementById('scrollByBtn').onclick = function(){
    window.scrollBy(300,300) //滚动栏x轴到达30 y轴到达30
    console.log(window.scrollX,window.scrollY)
}
```

打印机功能调用 print打印方法 (*)

- print (打印的基础实现)

```
window.print()
```

find 查找

```
window.find()
```

窗口 foucs 获取焦点 blur 失去焦点

```
window.foucs()
window.blur()
```

setInterval 和 setTimeout

clearInterval 和 clearTimeout

fetch 发送一个异步请求 (*)

```
//fetch 异步的请求 Axios的底层实现 （内核为xhr）
var obj = window.fetch('http://www.baidu.com')
console.log(obj)
```

Location (*)

概述

location是地址栏对象，他可以获取地址栏上的所有信息。

```
https://mbd.baidu.com/newspage/data/landingsuper?
context=%7B%22nid%22%3A%22news_9664507230212976443%22%7D&n_type=-1&p_from=-1
```

- https:// 协议
- mbd.baidu.com 域名（解析ip+端口号）
- 浏览器访问根据协议的不同指定不同的端口号 80端口http 443端口https
- /newspage/data/landingsuper 路径地址
- ? context=%7B%22nid%22%3A%22news_9664507230212976443%22%7D&n_type=-1&p_from=-1 传递参数（get请求）

location的相关属性（都支持赋值）

- hash 获取#后面携带的内容（#通常在最后面）
- host 主机 ip地址+端口号
- hostname 主机名 ip地址
- href url路径
- port 端口号
- pathname 路径名
- search 获取?后面传递的参数
- protocol 协议
- origin 跨域地址
- ancestorOrigins 获取倒序排列的文档对象及来源的浏览器上下文 （插件开发）

```
console.log(location)
//获取hash前面带#
console.log(location.hash)
//默认自己添加#
location.hash = 'abc'
console.log(location.host)
//localhost == 127.0.0.1
//location.host = "123.123.123.123:8080" 一般不建议设置
console.log(location.hostname)//主机名其实就是个ip地址
console.log(location.href)//路径
// 跳转页面
// location.href = 'http://www.baidu.com' //不会产生历史页面的
console.log(location.origin)
console.log(location.port)
//获取?后的内容 里面的写法为key=value&key=value
console.log(location.search)
location.search = 'name=jack&age=19'
//http（不安全 明文 80） 及 https （安全 加密 443（openssl（对称加密 非对称加密 hash加密等等） ））  超文本传输协议
console.log(location.protocol)//协议
//只读属性 返回的是一个domstringlist对象
console.log(location.ancestorOrigins)
```

location的方法

- assgin 跳转页面 （会产生历史页面）
- replace 替换url跳转页面

- reload 重新加载页面

```
//获取元素
document.getElementById('btn1').onclick=function(){
    location.assign('http://www.baidu.com')
}
//等同herf赋值
document.getElementById('btn2').onclick=function(){
    location.replace('http://www.baidu.com')
}
document.getElementById('btn3').onclick=function(){
    location.reload()
}
```

练习

将https://www.baidu.com/s?ie=UTF-8&wd=mdn&tn=15007414_dg里面的key value提取组成一个对象

```
// 将https://www.baidu.com/s?ie=UTF-8&wd=mdn&tn=15007414_dg里面的key value提取组成一个对象
function getUrlParams(url){
    //使用?进行分割
    url = url.split('?')[1]
    var param = {}
    // 使用&符号进行分割
    var params = url.split('&')//数组
    //遍历数组
    for(var paramStr of params){
        var arr = paramStr.split('=')
        //arr这个数组第一个是key 第二个是value
        param[arr[0]] = arr[1]
    }
    return param
}
console.log(getUrlParams('https://www.baidu.com/s?ie=UTF-8&wd=mdn&tn=15007414_dg'))
```

history （*）

概述

history对象是历史对象，他记录所有的历史页面。

history的属性

- length 历史页面个数（包含当前页面）
- state 值 （默认值为null）
- scrollRestoration 滚动恢复属性 （auto || manual）

history的方法

- forward 前进
- back 后退
- go 去任意历史页面

```
//常用的方法 前进 后退 任意跳转 事件会自动传递参数 参数叫event
document.getElementById('forward').onclick = function(){
    history.forward()
}
document.getElementById('back').onclick = function(){
    history.back()
}
document.getElementById('go').onclick = function(){
    //0为区间 0是自己 1为前进 -1为后退
    history.go(-1)
}
```

- pushstate 添加一个state（新增历史页面）
- replacestate 替换state （不会新增历史页面）

```
document.getElementById('pushstate').onclick = function(){
    //第一个参数为数据 state的数据值 第二个 unsend 常用值为"" 第三为url 改变的url
    history.pushState('hello',"",'./location对象讲解.html')
    console.log(history.state)
    console.log(history.length)
}
document.getElementById('replacestate').onclick = function(){
    //第一个参数为数据 state的数据值 第二个 unsend 常用值为"" 第三为url 改变的url
    history.replaceState('你好',"",'./location对象讲解.html')
    console.log(history.state)
    console.log(history.length)
}
```

pushstate及replacestate都会进行的操作

会影响state值 会改变url地址 但是不会跳转页面（页面不会刷新）

navigator

概述

navigator他是属于浏览器的导航对象，里面包含浏览器的相关信息以及你的系统信息。

属性

- **userAgent 用户信息**
- appName
- appVersion
- language (国际化)
- ...

```
//携带 浏览器版本信息以及系统版本信息兼容
console.log(navigator.userAgent) //用户相关信息
console.log(navigator.appName) //应用名
console.log(navigator.appVersion)//应用版本
console.log(navigator.language)//语言
```

screen 屏幕

概述:

用于屏幕相关信息 （适配 大屏可视化）

属性

- width 屏幕宽
- height 屏幕高
- availWidth 屏幕可视区宽度
- availHeight 屏幕可视区高度
- availLeft 屏幕可视区离左边的距离
- availTop 屏幕可视区离上边的距离

```
console.log(screen.width)//宽
console.log(screen.height)//高
console.log(screen.availHeight) //可视区高度
console.log(screen.availWidth) //可视区宽度
console.log(screen.availLeft) //可视区离左边的距离
console.log(screen.availTop) //可视区离上边的距离
```

document

概述

document是文档对象，他指代的是html整个文档。包含用于操作对应的html文档的相关内容。他是整个DOM里面最大的对象，他是属于BOM的。

相关属性

- body 获取body
- forms 获取所有的表单
- head 获取head

```
console.log(document.body)
console.log(document.forms)
console.log(document.head)
```

BOM是路由的底层实现，所有的前端JS框架的路由底层都是BOM。

路由

概述

根据不同的url地址来给你渲染不同的内容，路由主要分为俩种。前端路由，后端路由。

后端路由

后端介绍

后端 主要提供数据的，以及对应的数据进行相关的业务处理。后端用到的语言主要java，php，node.js...

后端路由

接口路由 （json格式 restful接口）

根据不同的接口返回不同的数据

渲染路由 （ssr 服务器渲染 前后端不分离）

根据不同的url地址来渲染不同的页面 (后端服务器的压力大)

优点

利于seo （搜索引擎优化）

首页渲染速度快 （主页都是做ssr）

缺点

服务器压力大

维护不便

前端路由

根据不同url地址来渲染不同的页面（浏览器解析）

前端路由划分

页面路由 （根据不同的url跳转不同的页面 （刷新操作））

```
location.href = 地址
location.assign('地址')
location.replace('地址')
```

hash路由 （根据不同hash值渲染不同的内容 不刷新）

```
<body>
    <button onclick="location.hash = 'hello'">hello</button>
    <button onclick="location.hash = 'byby'">byby</button>
    <div id="context"></div>
</body>
</html>
<script>
    //监听事件 onhashchange
    window.onhashchange = function(){
        var hash = location.hash
        if(hash=='#hello'){
            document.getElementById('context').innerHTML = 'hello'
        }else{
            document.getElementById('context').innerHTML = 'byby'
        }
    }
</script>
```

history路由 （根据不同的url来渲染不同的内容 不刷新）

```
<body>
  <button onclick="history.pushState('','','./document.html')">1</button>
  <button onclick="history.replaceState('','','./histroy.html')">2</button>
  <div id="context"></div>
</body>
</html>
<script>
```

```
    //监听事件 onpopstate
    //等待back 或者 go 或 forwad才触发
    window.onpopstate = function(){
      if(location.href.search('document')){
        document.getElementById('context').innerHTML = 'hello'
      }else{
        document.getElementById('context').innerHTML = 'byby'
      }
    }
</script>
```

hash路由和history路由常用于SPA（单页应用程序）程序，不刷新页面也能实现视图的切换。所以对应的在vue或者react的路由底层设计中只有俩种模式一种为hash模式一种为history模式。这种程序一般运用在前后端分离的基础上。

单页应用程序的优缺点

优点

无刷新 减少了页面的回流（重新渲染）

业务更加清晰 代码结构更加明了

缺点

不利于seo

所以为了解决单页应用不利于seo的问题出现了一个新的技术叫预渲染