

BOM

从入门到工作2020版

饥人谷版权所有

版权声明

- 本内容版权归属杭州饥人谷教育科技有限公司所有。
- 任何媒体、网站、个人未经本公司授权不得转载、链接、转帖，或以其他方式复制、发布、发表。
- 已获得饥人谷授权的媒体、网站、个人在使用时必须注明“资料来源：饥人谷”
- 如咨询课程，微信联系xiedaimala02 或 xiedaimala03 ，或者点击链接直接咨询 <https://dwz.cn/pPsjntxY>

BOM

- BOM(Browser Object Model) 浏览器对象模型，提供了独立于内容的、可以与浏览器窗口进行互动的对象结构

window

- ✓ var声明的全局变量和function声明的函数是window的属性
- ✓ let 、const声明的变量不是

```
var age = 2
let sex = 'male'

function printName(){
  console.log(age)
}

console.log(window.age)
window.printName()
console.log(window.sex) // undefined
```

navigator

- ✓ 浏览器相关信息
- ✓ navigator.userAgent

```
console.log(navigator.userAgent)
/*
  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36"
  "Mozilla/5.0 (iPhone; CPU iPhone OS 11_0 like Mac OS X)
  AppleWebKit/604.1.38 (KHTML, like Gecko) Version/11.0 Mobile/15A372
  Safari/604.1"
  "Mozilla/5.0 (iPad; CPU OS 11_0 like Mac OS X) AppleWebKit/604.1.34
  (KHTML, like Gecko) Version/11.0 Mobile/15A5341f Safari/604.1"
  "Mozilla/5.0 (Linux; Android 5.0; SM-G900P Build/LRX21T)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Mobile
  Safari/537.36"
*/

const isIPhone = () => /iphone/i.test(navigator.userAgent)
const isIOS = () => /iphone|ipad/i.test(navigator.userAgent)
const isAndroid = () => /android/i.test(navigator.userAgent)
```

screen

✓ 屏幕相关信息

```
console.log(screen.width)
console.log(screen.height)
console.log(screen.orientation)

screen.orientation.onchange = function() {
  console.log(screen.orientation.type)
  // "landscape-primary" 水平宽度大于垂直高度
  // "portrait-primary" 垂直高度大于水平宽度
}
```

尺寸

```
document.documentElement.offsetHeight //页面文档高度

screen.height //屏幕高度

//浏览器视窗高度
document.documentElement.clientHeight
window.innerHeight

//页面滚动的垂直距离
window.scrollY
document.documentElement.scrollTop

//距离最近的是定位元素的祖先元素
element.offsetParent

//到距离最近的是定位元素的祖先元素到距离
element.offsetTop

//获取元素在视窗中的位置
element.getBoundingClientRect()

//元素距离页面顶部的距离
element.getBoundingClientRect().top + document.body.scrollTop

//判断元素是否出现在窗口视野中
const isShow = node => node.getBoundingClientRect().top <
```

滚动

```
//平滑滚动到哪里
window.scrollTo({
  top: 100,
  left: 0,
  behavior: 'smooth'
})

//平滑滚动多少
window.scrollBy({
  top: window.innerHeight,
  left: 0,
  behavior: 'smooth'
})
```

- 回到顶部
 - ✓ <http://js.jirengu.com/jobij/>
- FullPage滚动
 - ✓ <http://js.jirengu.com/zaboc>

alert(), prompt(), confirm()

- ✓ 弹框提示
- ✓ 接收输入
- ✓ 确定取消

```
alert('Hello World')  
  
let result = prompt('您的年龄? ')  
  
let result = confirm('你最近好吗? ')
```

URL 编码解码

- ✓ decodeURI()
- ✓ decodeURIComponent()
- ✓ encodeURI()
 - 不会对 : / ? & = # 编码
- ✓ encodeURIComponent()
 - 会对 : / ? & = # 编码

Cookie、Session

Cookie

- ✓ cookie是存储在浏览器上的一小段数据，用来记录某些当页面关闭或者刷新后仍然需要记录的信息。在控制台用「document.cookie」查看你当前正在浏览的网站的cookie。
- ✓ cookie可以使用 js 在浏览器直接设置（用于记录不敏感信息，如用户名），也可以在服务端通使用 HTTP 协议规定的 set-cookie 来让浏览器种下cookie，这是最常见的做法。（打开一个网站，清除全部cookie，然后刷新页面，在network的Response headers试试找一找set-cookie吧）
- ✓ 每次网络请求 Request headers 中都会带上cookie。所以如果 cookie 太多太大对传输效率会有影响。
- ✓ 一般浏览器存储cookie 最大容量为4k，所以大量数据不要存到cookie。
- ✓ 设置cookie时的参数：
 - path: 表示 cookie 影响到的路径，匹配该路径才发送这个 cookie。
 - expires 和 maxAge: 告诉浏览器 cookie 时候过期，maxAge 是 cookie 多久后过期的相对时间。不设置这两个选项时会产生 session cookie，当用户关闭浏览器时，就被清除。一般用来保存 session 的 session_id。
 - secure: 当 secure 值为 true 时，cookie 在 HTTP 中是无效，在 HTTPS 中才有效
 - httpOnly: 浏览器不允许脚本操作 document.cookie 去更改 cookie。一般情况下都应该设置这个为 true，这样可以避免被 xss 攻击拿到cookie。

Session

✓ 当一个用户打开淘宝登录后，刷新浏览器仍然展示登录状态。服务器如何分辨这次发起请求的用户是刚才登录过的用户呢？这里就使用了session保存状态。用户在输入用户名密码提交给服务端，服务端验证通过后会创建一个session用于记录用户的相关信息，这个 session 可保存在服务器内存中，也可保存在数据库中。

- 创建session后，会把关联的session_id 通过setCookie 添加到http响应头部中。
- 浏览器在加载页面时发现响应头部有 set-cookie字段，就把这个cookie 种到浏览器指定域名下。
- 当下次刷新页面时，发送的请求会带上这条cookie， 服务端在接收到后根据这个session_id来识别用户。

✓ cookie 是存储在浏览器里的一小段「数据」，而session是服务器用来识别身份的一段数据，有时候也特指身份识别机制。

Session鉴权流程

session鉴权注册流程

浏览器

服务器

(1) 第一次访问饥人谷网站，我要注册

(2) 我的用户名jirengu, 密码: 123456

(4) 注册好了

(3) 数据库没有jirengu这个用户，允许注册，我给你创建一个数据库条目，username: jirengu, salt: 3a09, secret: SHA256(密码+salt)

数据库直接存储密码被“脱库”就完蛋了，还是存储密码经过散列处理后的结果好

session鉴权登录流程

(1) 我要登录

(2) 我的用户名jirengu, 密码: 123456

原作者:若愚@饥人谷

(5) 登录成功了。响应报文有Set-Cookie字段，那我设置这个cookie

10分钟后

(6) 我要访问我的个人页面，浏览器输入http://jirengu.com/user

(7) 我要访问/user, Cookie是浏览器自动带的

(9) 拿到了

200 OK
Set-Cookie: s_id=1b2ra9

登录成功

GET /user HTTP/1.1
Cookie: s_id=1b2ra9

200 OK

jirengu的更多数据

(3) 数据库有jirengu这个用户，计算一下你的secret。SHA256(你的密码+之前存储的salt) 结果和之前存储的secret一样，你的密码是对的。

(4) 创建一个session对象，我可以放到内存里也可以放到数据库里。session对象长这样 {id:1, username:'jirengu', avatar: '头像地址', s_id: '1b2ra9' ...}, 把session_id用setCookie给你

(8) 收到请求。请求报文带的有Cookie，我先拿出来比较一下。我的session集合里有一个sid相同的session，对应的用户名是jirengu。原来你是jirengu，我从数据库查找jirengu更多信息，发给你

知乎 @若愚

localStorage

- 特点

- ✓ localStorage HTML5本地存储web storage特性的API之一，用于将大量数据（最大5M）保存在浏览器中
- ✓ 保存后数据永远存在不会失效过期，除非用 js手动清除。
- ✓ 不参与网络传输

```
let information = {name: 'hunger', friends: ['Bob', 'Alice']}
localStorage.detail= JSON.stringify(information)
localStorage['age'] = 20
localStorage.setItem('sex', 'male')

localStorage.removeItem('sex')
console.log(localStorage.getItem('age'))

let oldInfo = JSON.parse(localStorage.detail)
oldInfo.friends.push('David')
localStorage.detail= JSON.stringify(oldInfo)

localStorage.clear()
```