BOM

- 今天开始我们开始使用 js 去操作浏览器和页面中的 html 元素了

BOM

- BOM（Browser Object Model）： 浏览器对象模型
- 其实就是操作浏览器的一些能力
- 我们可以操作哪些内容
  - 获取一些浏览器的相关信息（窗口的大小）
  - 操作浏览器进行页面跳转
  - 获取当前浏览器地址栏的信息
  - 操作浏览器的滚动条
  - 浏览器的信息（浏览器的版本）
  - 让浏览器出现一个弹出框（alert/confirm/prompt）
- BOM 的核心就是 window 对象
- window 是浏览器内置的一个对象，里面包含着操作浏览器的方法

获取浏览器窗口的尺寸

- `innerHeight` 和 `innerWidth`
- 这两个方法分别是用来获取浏览器窗口的宽度和高度（包含滚动条的）

```
var windowHeight = window.innerHeight
console.log(windowHeight)

var windowWidth = window.innerWidth
console.log(windowWidth)
```

浏览器的弹出层

- `alert` 是在浏览器弹出一个提示框

```
window.alert('我是一个提示框')
```

    - 这个弹出层知识一个提示内容，只有一个确定按钮
    - 点击确定按钮以后，这个提示框就消失了

- `confirm` 是在浏览器弹出一个询问框

```
var boo = window.confirm('我是一个询问框')
console.log(boo)
```

- 这个弹出层有一个询问信息和两个按钮
- 当你点击确定的时候，就会得到 `true`
- 当你点击取消的时候，就会得到 `false`

- `prompt` 是在浏览器弹出一个输入框

```
var str = window.prompt('请输入内容')
console.log(str)
```

    - 这个弹出层有一个输入框和两个按钮
    - 当你点击取消的时候，得到的是 `null`
    - 当你点击确定的时候得到的就是你输入的内容

浏览器的地址信息

- 在 window 中有一个对象叫做 `location`
- 就是专门用来存储浏览器的地址栏内的信息的

- 就是专门用来存储浏览器的地址栏内的信息的

location.href

- `location.href` 这个属性存储的是浏览器地址栏内 url 地址的信息

```
console.log(window.location.href)
```

  - 会把中文编程 url 编码的格式

- `location.href` 这个属性也可以给他赋值

```
window.location.href = './index.html'
// 这个就会跳转页面到后面你给的那个地址
```

location.reload

- `location.reload()` 这个方法会重新加载一遍页面，就相当于刷新是一个道理

```
window.location.reload()
```

  - 注意： **不要写在全局，不然浏览器就会一直处在刷新状态**

浏览器的历史记录

- window 中有一个对象叫做 `history`
- 是专门用来存储历史记录信息的

history.back

- `history.back` 是用来会退历史记录的，就是回到前一个页面，就相当于浏览器上的 ⬅ 按钮

```
window.history.back()
```

  - 前提是你要有上一条记录，不然就是一直在这个页面，也不会回退

history.forword

- `history.forword` 是去到下一个历史记录里面，也就是去到下一个页面，就相当于浏览器上的 ➡ 按钮

```
window.history.forward()
```

  - 前提是你要之前有过回退操作，不然的话你现在就是最后一个页面，没有下一个

浏览器的版本信息（了解）

- window 中有一个对象叫做 `navigator`
- 是专门用来获取浏览器信息的

- 是专门用来获取浏览器信息的

navigator.userAgent

- `navigator.userAgent` 是获取的浏览器的整体信息

```
console.log(window.navigator.userAgent)
// Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
```

navigator.appName

- `navigator.appName` 获取的是浏览器的名称

```
console.log(window.navigator.appName)
```

navigator.appVersion

- `navigator.appVersion` 获取的是浏览器的版本号

```
console.log(window.navigator.appVersion)
```

navigator.platform

- `navigator.platform` 获取到的是当前计算机的操作系统

```
console.log(window.navigator.platform)
```

浏览器的 onload 事件

- 这个不在是对象了，而是一个事件
- 是在页面所有资源加载完毕后执行的

```
window.onload = function () {
 console.log('页面已经加载完毕')
}
```

在 html 页面中把 js 写在 head 里面

```
<html>
 <head>
 <meta charset="UTF-8" />
 <script>
 // 这个代码执行的时候，body 还没有加载
```

```
// 这个代码执行的时候，body 还没有加载
// 这个时候我们就获取不到 body 中的那个 div

// 就需要使用 window.onload 事件
window.onload = function () {
// 这个函数会在页面加载完毕以后在执行
// 那么这个时候页面的 DOM 元素都已经加载了，我们就可以获取 div 了
}
</script>
</head>
<body>
<div></div>
</body>
</html>
```

在 html 页面中把 js 写在 body 最后面

```
<html>
<head>
<meta charset="UTF-8" />
</head>
<body>
<div></div>

<script>
// 这个代码执行的时候，body 已经加载完毕了
// 在这里就可以获取到 div，写不写 window.onload 就无所谓了

window.onload = function () {
// 这个函数会在页面加载完毕以后在执行
// 那么这个时候页面的 DOM 元素都已经加载了，我们就可以获取 div 了
}
</script>
</body>
</html>
```

浏览器的 onscroll 事件

- 这个 onscroll 事件是当浏览器的滚动条滚动的时候触发
- 或者鼠标滚轮滚动的时候出发

```
window.onscroll = function () {
 console.log('浏览器滚动了')
}
```

- 注意：**前提是页面的高度要超过浏览器的可是窗口才可以**

浏览器滚动的距离

- 浏览器内的内容即然可以滚动，那么我们就可以获取到浏览器滚动的距离
- 思考一个问题？
  - 浏览器真的滚动了吗？
  - 其实我们的浏览器是没有滚动的，是一直在那里
  - 滚动的是什么？是我们的页面
  - 所以说，**其实浏览器没有动，只不过是页面向上走了**
- 所以，这个已经不能单纯的算是浏览器的内容了，而是我们页面的内容
- 所以不是在用 window 对象了，而是使用 document 对象

scrollTop

- 获取的是页面向上滚动的距离

- 一共有两个获取方式

  - `document.body.scrollTop`
  - `document.documentElement.scrollTop`

```
window.onscroll = function () {
 console.log(document.body.scrollTop)
 console.log(document.documentElement.scrollTop)
}
```

  - 两个都是获取页面向上滚动的距离
  - 区别：
    - IE 浏览器
      - 没有 `DOCTYPE` 声明的时候，用这两个都行
      - 有 `DOCTYPE` 声明的时候，只能用 `document.documentElement.scrollTop`
    - Chrome 和 FireFox
      - 没有 `DOCTYPE` 声明的时候，用 `document.body.scrollTop`
      - 有 `DOCTYPE` 声明的时候，用 `document.documentElement.scrollTop`
    - Safari
      - 两个都不用，使用一个单独的方法 `window.pageYOffset`

scrollLeft

- 获取页面向左滚动的距离

- 也是两个方法
  - `document.body.scrollLeft`
  - `document.documentElementLeft`

```
window.onscroll = function () {
 console.log(document.body.scrollLeft)
 console.log(document.documentElement.scrollLeft)
}
```

  - 两个之间的区别和之前的 `scrollTop` 一样

定时器

- 在 js 里面，有两种定时器，**倒计时定时器** 和 **间隔定时器**

倒计时定时器

- 倒计时多少时间以后执行函数
- 语法： `setTimeout(要执行的函数，多长时间以后执行)`
- 会在你设定的时间以后，执行函数

```
var timerId = setTimeout(function () {
 console.log('我执行了')
}, 1000)
console.log(timerId) // 1
```

  - 时间是按照毫秒进行计算的，1000 毫秒就是 1秒钟
  - 所以会在页面打开 1 秒钟以后执行函数
  - 只执行一次，就不在执行了
  - 返回值是，当前这个定时器是页面中的第几个定时器

间隔定时器

- 每间隔多少时间就执行一次函数
- 语法： `setInterval(要执行的函数，间隔多少时间)`

```
var timerId = setInterval(function () {
 console.log('我执行了')
}, 1000)
```

    - 时间和刚才一样，是按照毫秒进行计算的
    - 每间隔 1 秒钟执行一次函数
    - 只要不关闭，会一直执行
    - 返回值是，当前这个定时器是页面中的第几个定时器

定时器的返回值

- 设置定时器的时候，他的返回值是部分 `setTimeout` 和 `setInterval` 的
- 只要有一个定时器，那么就是一个数字

```
var timerId = setTimeout(function () {
 console.log('倒计时定时器')
}, 1000)

var timerId2 = setInterval(function () {
 console.log('间隔定时器')
}, 1000)

console.log(timerId) // 1
console.log(timerId2) // 2
```

关闭定时器

- 我们刚才提到过一个 timerId，是表示这个定时器是页面上的第几个定时器
- 这个 timerId 就是用来关闭定时器的数字
- 我们有两个方法来关闭定时器 `clearTimeout` 和 `clearInterval`

```
var timerId = setTimeout(function () {
 console.log('倒计时定时器')
}, 1000)
clearTimeout(timerId)
```

    - 关闭以后，定时器就不会在执行了

```
var timerId2 = setInterval(function () {
 console.log('间隔定时器')
}, 1000)
coearInterval(timerId2)
```

    - 关闭以后定时器就不会在执行了

- 原则上是
    - `clearTimeout` 关闭 `setTimeout`

- `clearInterval` 关闭 `setInterval`

- 但是其实是可以通用的，他们可以混着使用

```
var timerId = setTimeout(function () {
 console.log('倒计时定时器')
}, 1000)
// 关闭倒计时定时器
clearInterval(timerId)

var timerId2 = setInterval(function () {
 console.log('间隔定时器')
}, 1000)
// 关闭间隔定时器
clearTimeout(timerId2)
```

强化练习1

1. 在页面上出现倒计时
2. 回到顶部功能
3. 浏览器滚动一定高度的时候出现顶部通栏
4. 理解 onload 事件
5. 理解 onscroll 事件