Navigator 对象和 Screen 对象

window.navigator 属性指向一个包含浏览器和系统信息的 Navigator 对象。脚本通过这个属性了解用户的环境信息。

1. Navigator 对象的属性

1.1. Navigator.userAgent

navigator.userAgent 属性返回浏览器的 User Agent 字符串,表示浏览器的厂商和版本信息。

下面是 Chrome 浏览器的 userAgent。

```
navigator.userAgent;
// "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/90.0.4430.212 Safari/537.36"
```

通过 userAgent 属性识别浏览器,不是一个好办法。因为必须考虑所有的情况(不同的浏览器,不同的版本),非常麻烦,而且用户可以改变这个字符串。这个字符串的格式并无统一规定,也无法保证未来的适用性,各种上网设备层出不穷,难以穷尽。所以,现在一般不再通过它识别浏览器了,而是使用"功能识别"方法,即 逐一测试当前浏览器是否支持要用到的 JavaScript 功能。

不过,通过 userAgent 可以大致准确地识别手机浏览器,方法就是测试是否包含 mobi 字符串。

```
var ua = navigator.userAgent.toLowerCase();

if (/mobi/i.test(ua)) {
    // 手机浏览器
} else {
    // 非手机浏览器
}
```

如果想要识别所有移动设备的浏览器,可以测试更多的特征字符串。

```
/mobi|android|touch|mini/i.test(ua);
```

1.2. Navigator.plugins;

Navigator.plugins 属性返回一个类似数组的对象,成员是 Plugin 实例对象,表示浏览器安装的插件,比如 Flash、ActiveX 等。

```
var pluginsLength = navigator.plugins.length;
```

```
for (var i = 0; i < pluginsLength; i++) {
    console.log(navigator.plugins[i].name);
    console.log(navigator.plugins[i].filename);
}

/*
Chrome PDF Plugin
internal-pdf-viewer

Chrome PDF Viewer
mhjfbmdgcfjbbpaeojofohoefgiehjai
*/</pre>
```

1.3. Navigator.platform

Navigator.platform 属性返回用户的操作系统信息,比如 MacIntel、Win32、Linux x86_64等。

```
navigator.platform;
// "MacIntel"
```

1.4. Navigator.onLine

navigator.onLine 属性返回一个布尔值,表示用户当前在线还是离线(浏览器断线)。

```
navigator.onLine; // true
```

有时,浏览器可以连接局域网,但是局域网不能连通外网。这时,有的浏览器的 onLine 属性会返回 true,所以不能假定只要是 true,用户就一定能访问互联网。不过,如果是 false,可以断定用户一定离线。

用户变成在线会触发 online 事件,变成离线会触发 offline 事件,可以通过 window.ononline 和 window.onoffline 指定这两个事件的回调函数。

```
window.ononline = () => {
  console.log('online');
};
window.onoffline = () => {
  console.log('offline');
};
```

当我关闭网络时,控制台输出 offline,表示触发了 window 的 onoffline 事件。当我开启网络时,控制台输出 online,表示触发了 window 的 ononline 事件。

```
window.addEventListener('offline', function (e) {
  console.log('offline');
```

```
});
window.addEventListener('online', function (e) {
  console.log('online');
});
```

1.5. Navigator.language, Navigator.languages

Navigator.language 属性返回一个字符串,表示浏览器的首选语言。该属性只读。

```
navigator.language; // "zh-CN"
```

Navigator.languages 属性返回一个数组,表示用户可以接受的语言。Navigator.language 总是这个数组的第一个成员。HTTP 请求头信息的 Accept-Language 字段,就来自这个数组。

```
navigator.languages; // ["zh-CN", "zh", "en"]
```

如果这个属性发生变化,就会在 window 对象上触发 language change 事件。

```
Navigator.geolocation;
```

Navigator.geolocation 属性返回一个 Geolocation 对象,包含用户地理位置的信息。注意,该 API 只有在 HTTPS 协议下可用,否则调用下面方法时会报错。

Geolocation 对象提供下面三个方法。

- Geolocation.getCurrentPosition():得到用户的当前位置
- Geolocation.watchPosition(): 监听用户位置变化
- Geolocation.clearWatch(): 取消 watchPosition() 方法指定的监听函数

调用这三个方法时,浏览器会跳出一个对话框,要求用户给予授权。

1.6. Navigator.cookieEnabled

navigator.cookieEnabled 属性返回一个布尔值,表示浏览器的 Cookie 功能是否打开。

```
navigator.cookieEnabled; // true
```

注意,这个属性反映的是浏览器总的特性,与是否储存某个具体的网站的 Cookie 无关。用户可以设置某个网站不得储存 Cookie, 这时 cookieEnabled 返回的还是 true。

2. Navigator 对象的方法

2.1. Navigator.javaEnabled()

navigator.javaEnabled()方法返回一个布尔值,表示浏览器是否能运行 Java Applet 小程序。

```
navigator.javaEnabled() // false
```

2.2. Navigator.sendBeacon()

Navigator.sendBeacon()方法用于向服务器异步发送数据。

这个方法主要用于满足统计和诊断代码的需要,这些代码通常尝试在卸载 (unload) 文档之前向 Web 服务器 发送数据。

```
document.addEventListener("visibilitychange", function logData() {
  if (document.visibilityState === "hidden") {
    navigator.sendBeacon("/log", analyticsData); // 在页面卸载前, 向服务器可靠的发送
  http 请求
  }
});
```

3. Navigator 的实验性属性

Navigator 对象有一些实验性属性,在部分浏览器可用。

3.1. Navigator.deviceMemory

navigator.deviceMemory 属性返回当前计算机的内存数量(单位为 GB)。该属性只读,只在 HTTPS 环境下可用。

它的返回值是一个近似值,四舍五入到最接近的 2 的幂,通常是 0.25、0.5、1、2、4、8。实际内存超过 8GB, 也返回 8。

```
if (navigator.deviceMemory > 1) {
  await import('./costly-module.js');
}
```

上面示例中,只有当前内存大于 1GB, 才加载大型的脚本。

3.2. Navigator.hardwareConcurrency

navigator.hardwareConcurrency 属性返回用户计算机上可用的逻辑处理器的数量。该属性只读。

现代计算机的 CPU 有多个物理核心,每个物理核心有时支持一次运行多个线程。因此,四核 CPU 可以提供八个逻辑处理器核心。

```
if (navigator.hardwareConcurrency > 4) {
   await import('./costly-module.js');
}
```

上面示例中,可用的逻辑处理器大于4,才会加载大型脚本。

该属性通过用于创建 Web Worker,每个可用的逻辑处理器都创建一个 Worker。

```
let workerList = [];

for (let i = 0; i < window.navigator.hardwareConcurrency; i++) {
  let newWorker = {
    worker: new Worker('cpuworker.js'),
    inUse: false,
    };
    workerList.push(newWorker);
}</pre>
```

上面示例中,有多少个可用的逻辑处理器,就创建多少个 Web Worker。

3.3. Navigator.connection

navigator.connection 属性返回一个对象,包含当前网络连接的相关信息。

- downlink: 有效带宽估计值(单位: 兆比特/秒, Mbps), 四舍五入到每秒 25KB 的最接近倍数。
- downlinkMax: 当前连接的最大下行链路速度(单位: 兆比特每秒, Mbps)。
- effectiveType:返回连接的等效类型,可能的值为 slow-2g、2g、3g、4g。
- rtt: 当前连接的估计有效往返时间,四舍五入到最接近的 25 毫秒的倍数。
- saveData: 用户是否设置了浏览器的减少数据使用量选项(比如不加载图片),返回 true 或者 false.
- type: 当前连接的介质类型,可能的值为 bluetooth、cellular、ethernet、none、wifi、wimax、other、unknown。

```
if (navigator.connection.effectiveType === '4g') {
  await import('./costly-module.js');
}
```

上面示例中,如果网络连接是46,则加载大型脚本。

2. Screen 对象

Screen 对象表示当前窗口所在的屏幕,提供显示设备的信息。window.screen 属性指向这个对象。

该对象有下面的属性。

- Screen.height:浏览器窗口所在的屏幕的高度(单位像素)。除非调整显示器的分辨率,否则这个值可以看作常量,不会发生变化。显示器的分辨率与浏览器设置无关,缩放网页并不会改变分辨率。
- Screen.width: 浏览器窗口所在的屏幕的宽度(单位像素)。
- Screen.availHeight:浏览器窗口可用的屏幕高度(单位像素)。因为部分空间可能不可用,比如系统的任务栏或者 Mac 系统屏幕底部的 Dock 区,这个属性等于 height 减去那些被系统组件的高度。
- Screen.availWidth: 浏览器窗口可用的屏幕宽度(单位像素)。
- Screen.pixelDepth: 整数,表示屏幕的色彩位数,比如 24表示屏幕提供 24位色彩。
- Screen.colorDepth: Screen.pixelDepth 的别名。严格地说,colorDepth 表示应用程序的颜色深度,pixelDepth 表示屏幕的颜色深度,绝大多数情况下,它们都是同一件事。
- Screen.orientation:返回一个对象,表示屏幕的方向。该对象的 type 属性是一个字符串,表示屏幕的具体方向,landscape-primary 表示横放,landscape-secondary 表示颠倒的横放,portrait-primary 表示竖放,portrait-secondary 表示颠倒的竖放。

下面是 Screen.orientation 的例子。

```
window.screen.orientation;
// { angle: 0, type: "landscape-primary", onchange: null }
```

下面的例子保证屏幕分辨率大于 1024 x 768。

```
if (window.screen.width >= 1024 && window.screen.height >= 768) {
    // 分辨率不低于 1024x768
}
```

下面是根据屏幕的宽度,将用户导向不同网页的代码。

```
if (screen.width <= 800 && screen.height <= 600) {
   window.location.replace('small.html');
} else {
   window.location.replace('wide.html');
}</pre>
```