Navigator 对象，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。

Navigator 对象的属性

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`。

```javascript
navigator.userAgent
// "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
29.0.1547.57 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。

```javascript
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。

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

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。

```javascript
```

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

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

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 。

```javascript
navigator.platform
// "Linux x86_64"
```

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）。

```javascript
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。

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

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。

```javascript
navigator.language // "en"
```

`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。

```javascript
navigator.languages  // ["en-US", "en", "zh-CN", "zh", "zh-TW"]
```

如果这个属性发生变化，就会在`window`对象上触发`languagechange`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

注意，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。

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。

```javascript
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`。

Navigator 对象的方法

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。

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

Navigator.sendBeacon()

`Navigator.sendBeacon()`方法用于向服务器异步发送数据，详见《XMLHttpRequest 对象》一章。

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`。

下面是`Screen.orientation`的例子。

```javascript
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。

```javascript
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。

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