Global 作用域

1. 通过 var 声明的所有**全局变量和函数**都会变成 window 对象的属性和方法。

窗口关系

window.top：始终指向**最上层（最外层）窗口**，即浏览器窗口本身。
window.parent：始终指向**当前窗口的父窗口**。如果当前窗口是最上层窗口，则 parent = top = window。
window.self：始终**指向 window**。

1. 既然有了 window 为什么还有 self？只是为了与 top 和 parent 保持一致。
2. 这几个属性实际使用场景？
3. iframe 嵌入窗口？
4. window.open(url,name)打开的窗口？这里的 name 参数就是 window.name 的值。

窗口位置

1. screenLeft、screenTop
2. moveTo(要移动到的新位置的绝对坐标 x 和 y)、moveBy(相对当前位置在两个方向上移动的像素数)

像素比

物理分辨率转换为逻辑分辨率；DPI：每英寸像素数（dots per inch）；**window.devicePixelTatito**

窗口大小

innerWidth、innerHeight ：浏览器**窗口自身**的大小（不管是在最外层 window 上，还是在窗格<frame>中）
outerWidth、outerHeight ：浏览器窗口中**页面视口**的大小
document.documentElement.clientWidth、document.documentElement.clientHeight

1. 上述 6 个属性都会改变，缩放、拖动；
2. 浏览器窗口自身的精确尺寸不好确定；那如何确定页面视口大小？

```
function getsTheViewportSize() {
    let pageWidth = window.innerWidth,
        pageHeight = window.innerHeight;
    if (typeof pageWidth !="number") {
        if (document.compatMode === "CSS1Compat") {
        pageWidth = document.documentElement.clientWidth;
        pageHeight = document.documentElement.clientHeight;
        } else {
        pageWidth = document.body.clientWidth;
        pageHeight = document.body.clientHeight;
        }
    }
}
```

3. document.compatMode？
4. resizeTo()、resizeBy() 缩放窗口的方法只能应用到最上层的 window 对象

视口位置

scrollX/pageXOffset、scrollY/pageYOffset（度量文档相对于视口滚动距离）
滚动页面：scroll(滚动到)、scrollTo(滚动到)、scrollBy(滚动了)；参数：(x，y)或者 ScrollToOptions 字典（{top:100,left:100,behavior:'smooth'}）

导航与打开新窗口

- let windowObjectReference = window.open(strUrl, strWindowName, [strWindowFeatures])

1. windowObjectReference 打开的新窗口对象的引用。如果**调用失败，返回值会是 null** 。如果父子窗口满足“同源策略”，你可以通过这个引用访问新窗口的属性或方法。windowObjectReference.close()和 windowObjectReference.closed。
2. strWindowName 字符串还可以用来**作为超链接 <a> 或表单 <form> 元素的目标属性值。**可以用来避免打开同一个窗口（列表详情按钮，打开新窗口，name 可以设置为 uuid）。
3. newWin.opener、当调用 newWin.opener.close()后 newWin.opener 为 null。newWin.opener=null 则表示新打开的标签页不需要与打开它的标签页通信，因此新打开的标签页可以在独立的进程中运行。注意：newWin.opener=null 是不可逆的操作
4. 准确判断调用 window.open()的弹窗是否被浏览器屏蔽了？

```
function openWindow(url) {
    let blocked = false;
    try {
        let win = window.open(url, "_blank");
        if (win == null) {
        blocked = true;
        }
    } catch (error) {
        blocked = true;
    }
    return blocked;
}
```

定时器

setTimeout

常用写法：

```
var timeoutID = scope.setTimeout(function[, delay]);
```

附加参数：

```
var timeoutID = scope.setTimeout(function[, delay, arg1, arg2, ...]); // 一旦定时器到期，它们会作为参数传递给 function
```

使用字符串：

```
var timeoutID = scope.setTimeout(code[, delay]);  // 不推荐的，原因和使用 eval()一样，有安全风险
```

1. 第二个参数只是告诉 JavaScript 引擎在指定的毫秒数过后把任务添加到任务队列中。
2. 只要是在指定的时间到达之前调用 clearTimeout()，就可以取消超时任务。在任务执行后再调用 clearTimeout()没有效果。
3. 所有超时执行的代码都会在全局作用域中的一个匿名函数中运行。因此函数中的 this 值在非严格模式下始终指向 window，而在严格模式下是 undefined。如果提供的是一个箭头函数，那么 this 会保留为定义它时所在的词汇作用域。【this 问题】

setInterval

1. 每隔一段时间会向队列中添加一个任务，并不关心什么时候会被执行。存在的问题？
2. 使用 setTimeout 实现一个循环任务？
3. 最好不要使用 setInterval()

系统对话框

alert(message)、var isOk = confirm(message)、var value = prompt(text, value) ：

1. 无法使用 CSS 设置样式；同步执行（阻塞 JavaScript）。
2. JavaScript 可显示的两种对话框：find()和 print()。
3. 重点关注 print()

FrontEnd-DuKang