JavaScript高级程序设计

什么是JavaScript

1. JavaScript是一门用来与网页交互的脚本语言，包含以下三个组成部分：
    a. ECMAScript：由ECMA-262定义并提供其核心功能
    b. DOM（文档对象模型）：提供与网页内容交互的方法的接口
    c. BOM（浏览器对象模型）：提供与浏览器交互的方法和接口

HTML中的JavaScript

<script>元素

属性

1. async：可选，表示应立即开始下载脚本，但不能阻止其他页面动作（异步执行）。只对外部脚本文件有效。
2. charset：可选，字符集。
3. crossorign：可选，配置相关请求的CORS（跨源资源共享）设置。默认不使用CORS
4. defer：可选，脚本立即下载，但推迟执行。仅对外部脚本文件有效。
5. integrity：可选，允许比对接收到的资源和指定的加密签名以验证子资源的完整性。
6. src：可选。
7. type：可选，代表代码块中脚本语言的内容类型（也称MIME类型）

执行顺序

1. 在不使用defer或者async的情况下，包含在<script>中的代码由上至下执行

注意点

1. <script>中的代码尽量不要出现</script>，浏览器会将其视为结束标志；如果一定要使用，使用转义字符，例：

```
1
```

2. 使用src属性的<script>标签元素不应该在<script></script>中再包含其他代码（也就是一个<script>标签，行内式和外部文件式只能选一个）

跨域

1. <script>元素可以包含来自外部域的JavaScript文件
2. 若src属性是一个指向不同于的url，则浏览器会向此指定路径发送一个GET请求，此初始请求不受浏览器同源策略的限制，但返回的JavaScript仍受限制
3. 好处：通过不同的域分发JavaScript（就是我们引入外部包的过程）
4. 可使用integrity属性进行防范

位置

1. 通常将所有的JavaScript引用放在<body>元素中的页面内容后面

动态加载脚本

```
1
```

<noscript>元素

1. <noscript>可以是一种出错提示手段
2. 在以下任一条件被满足时，<noscript>中的内容就会被渲染
    a. 浏览器不支持脚本
    b. 浏览器对脚本的支持被关闭

```
1
```

语言基础

语法

标识符

1. 标识符是变量、函数、属性或函数参数的名称
2. 标识符的组成如下：
    a. 第一个字符必须是一个字母、下划线、或美元符号
    b. 剩下的其他字符可以是字母、下划线、美元符号或数字
3. 推荐使用驼峰大小写形式
4. 关键字、保留字、true、false和null不能作为标识符

关键字

break	do	in	typeof
case	else	instanceof	var
catch	export	new	void
class	extends	return	while
const	finally	super	with
continue	for	switch	yield
default	if	throw	this
function	debugger	delete	import
try			

保留字

始终保留	严格模式下保留		模块代码中保留
enum	implements	package	await
	public	interface	
	protected	private	
	static	let	

变量

1. 可以保存任意类型的数据，每个变量都是一个保存任意值的占位符
2. 变量有三个：var、let和const
3. const优先，let次之，不使用var

var

1. 不初始化的情况下，变量会保存一个特殊值undefined
2. 使用var操作符定义的变量会成为包含它的函数的局部变量
3. 在函数内部定义变量时省略var，可以创建一个全局变量（严格模式下会报错，且不推荐这么做）
4. var声明提升：
    a. 使用var声明变量时，变量会发生变量提升
    b. 所谓提升，是把所有变量声明提升到函数作用域的顶部
5. 可以使用var声明同一个变量
6. 用var在全局作用域中声明的变量会成为window对象的属性
7. 具体用例：

```Git
// var的作用域

// var声明提升
```

let

1. let声明的范围是块作用域

2. let不允许在同一个块作用域中出现冗余声明
3. let声明的变量不会在作用域中提升
4. 用let在全局作用域中声明的变量不会成为window对象的属性
5. 具体用例：

```Git
// 混用var与let

// for循环中的let声明
```

const

1. const的行为与let基本相同
2. 用const声明变量的同时必须初始化变量，且该变量不允许进行修改
3. 如果const变量引用的是一个对象，修改该对象内部的属性方法是允许的
4. 具体用例：

```Git
// for-of 和 for-in
```

数据类型

1. 6种简单数据类型：Undefined、Null、Boolean、Number、String、Symbol
2. 1种复杂数据类型：Obiect

typeof操作符

1. 使用typeof返回的字符串及其意义

字符串	意义
"undefined"	值未定义
"boolean"	值为布尔值
"string"	值为字符串
"number"	值为数值
"object"	值为对象或null
"function"	值为函数
"symbol"	值为符号

2. 具体用例：

```Git
// typeof 操作符
```

undefined类型

1. undefine类型只有一个值，就是特殊值undefined
2. 变量声明了但是没有赋值是，变量的值为undefined（明确是空对象指针null和为初始变量的区别）
3. 对未声明的变量，只能执行一个有用的操作，就是对它调用typeof，返回值为undefined
4. 具体事例：

```Git
// 包含undefined值的变量与未定义变量的区别

// 明确检测undefined这个字面值
```

Null类型

1. Null类型只有一个值，即特殊值null
2. 在定义将来要保存对象值的变量时，建议用null初始化
3. 具体事例：

```
// null与undefined表面相等

// 明确检测null这个字面值
```

Boolean类型

1. Boolean类型有两个字面值，true和false
2. 所有其他ECMAScript类型的值都有相应的布尔值的等价形式
3. 可使用Boolean（[任意类型的数据]）转型函数将其他值转换为布尔值
4. 不同类型与布尔值的转换规则

数据类型	转换为true的值	转换为false的值
Boolean	true	false
String	非空字符串	“”（空字符串）
Number	非零数值（包括无穷值）	0、NaN
Object	任意对象	null
Undefined	N/A（不存在）	undefined