

XML 原理与应用

夏天

中国人民大学

```
<script>
```

```
<!--
```

```
function ga(o,e){  
  if (document.getElementById
```

```
    a=o.id.substring(1);  
    p = "";
```

```
    r = "";  
    g = e.target;  
    (g) {  
      g.id
```

本章学习目标

- 了解 JavaScript 的起源
- 掌握 JavaScript 的测试方法
- 熟悉 JavaScript 的基本语法，能借助于文档进行基本的脚本编写
- 了解浏览器对象模型 BOM
- 理解 JavaScript 的定时器操作

目录

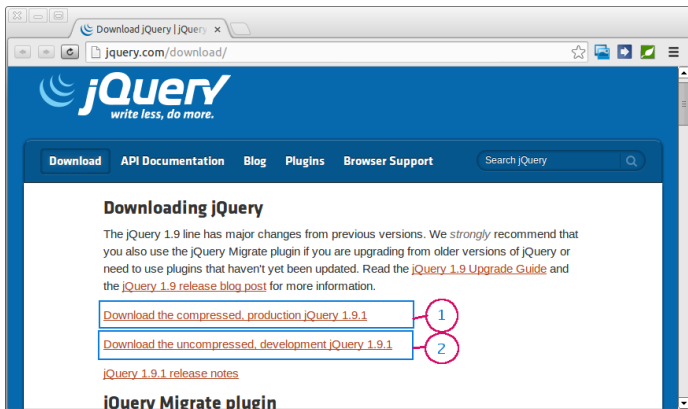
- JavaScript 概述
- JavaScript 的测试方法
- JavaScript 的变量和常量
- JavaScript 的基本语句
- 函数和数组
- 对象
- 浏览器对象模型 BOM
- 定时器

8.1 JavaScript 概述

- 1995 年 → Brendan Eich → 网景公司 → JavaScript
- JavaScript vs Applet
- JavaScript vs JScript
- 当前 JavaScript 的三大组成部分
 - ECMAScript 语言核心
 - DOM(Document Object Model) 文档对象模型
 - BOM(Browser Object Model) 浏览器对象模型

- JavaScript 的跨浏览器问题
- jQuery 解决方案
 - John Resig 创建的一个轻量级 JavaScript 库
 - 充分利用了 CSS 的优势，支持扩展，能够抽象浏览器的不一致性，并支持隐式迭代和连缀操作

- 下载使用



- 嵌入方法

```
<script type="text/javascript" src="jquery.js"></script>
```

- 使用托管版本

```
<script type="text/javascript"  
src=" http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js  
"></script>
```


8.2 JavaScript 的测试方法

- 与网页的关联测试方法
- 在页面加载之后运行 JavaScript
- 浏览器内置的 JavaScript 控制台

8.2.1 与网页的关联测试方法

- 编写 JavaScript 脚本文件，并保存到以.js 结尾的文件中，然后通过 HTML 语言中 script 标记的 src 属性进行关联
- 在 <script> 标记的开始和结束之间，直接嵌入 JavaScript 脚本
- 考虑二者的优缺点

示例

```
1 <html>
2   <head>
3     <title>JavaScript DEMO</title>
4     <script type="text/javascript" src="jquery.js"></script>
5     <script type="text/javascript">
6       $(document).ready(function() {
7         alert('hello world');
8       });
9     </script>
10  </head>
11  <body>
12    <h1>JavaScript Test</h1>
13  </body>
14 </html>
```

8.2.2 在页面加载之后运行 JavaScript

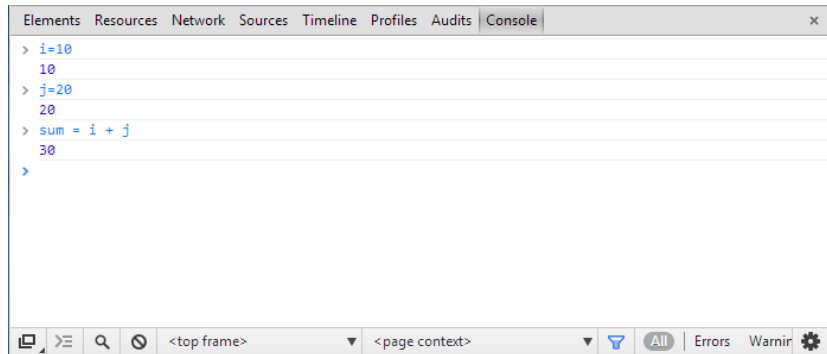
- JavaScript 默认在网页中的添加位置立即加载运行
 - 问题
 - 难度
- jQuery 的实现方式

```
$(document).ready(function() {  
    //将想在网页加载完后再运行的代码放在此处...  
});
```

8.2.3 利用浏览器内置的 JavaScript 控制台

- Firefox
 - “Tools” → “Web Developer” → “Web Console”
 - CTRL+SHIFT+K
- CHROME
 - “Tools” → “JavaScript Console”
 - CTRL+SHIFT+J
- OPERA
 - “Developer Tools” → “Web Inspector”
 - CTRL+SHIFT+I
- IE
 - “Tools” → “F12 Developer Tools”
 - F12

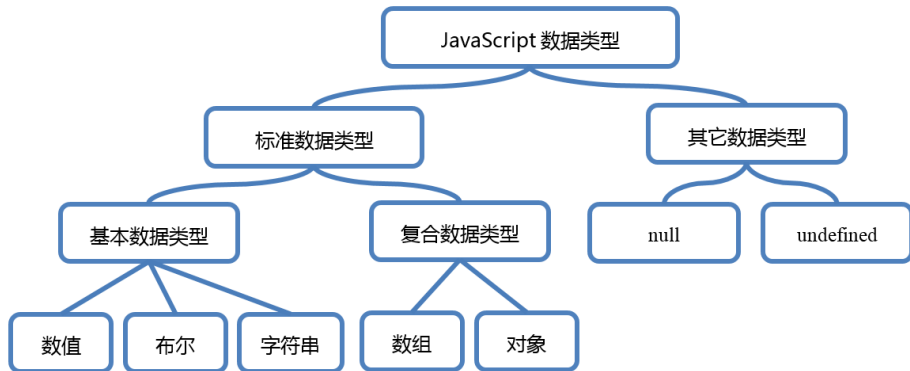
Google Chrome 的 JavaScript 控制台



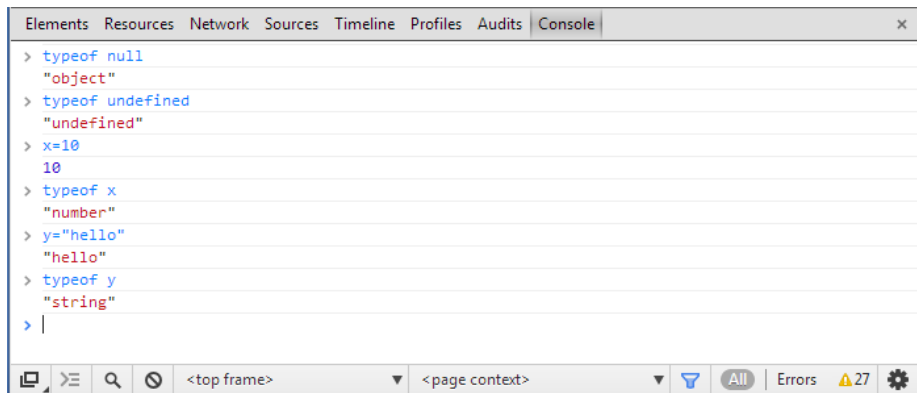
8.3 JavaScript 的变量和常量

- 数据类型
- 变量的声明和赋值
- 变量的作用域
- 常量

8.3.1 数据类型



利用控制台查看数据类型



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the results of several JavaScript commands entered in the prompt. The commands and their outputs are as follows:

- `> typeof null` returns `"object"`
- `> typeof undefined` returns `"undefined"`
- `> x=10` returns `10`
- `> typeof x` returns `"number"`
- `> y="hello"` returns `"hello"`
- `> typeof y` returns `"string"`
- `> |` (The prompt is currently empty, showing a vertical bar)

The console interface includes a toolbar at the bottom with icons for clearing, expanding, searching, and toggling console visibility. It also shows the current frame as `<top frame>` and the page context as `<page context>`. The filter is set to `All`, and there are 27 errors listed.

8.3.2 变量的声明和赋值

- 变量声明

```
var userName;
```

- 变量赋值

```
username="李白";
```

- 声明和赋值同时进行

```
var userName="李白";
```

8.3.3 变量的作用域

- 全局变量
 - 在函数之外声明的变量
- 局部变量
 - 在函数之中声明的变量
- 局部变量对全局变量具有覆盖作用
- 考虑以下代码的输出结果：

```
1 var a = 6;  
2 function myfunction(){  
3     var a = 7;  
4     var b = 8;  
5     return a+b;  
6 }  
7 console.info(myfunction());
```

8.3.4 常量

- 不变的量，用关键字 `const` 来声明

```
const A = 10;  
const B = .314;
```

- 常量类型

- 数字常量
- 非数字 NaN(Not A Number)
- 布尔常量：true、false
- 字符串
- 转义字符
 - `\b` (退格)、`\f` (换页)、`\n` (换行)、`\r` (回车)、`\t` (跳格)、`\'` (单引号)、`\"` (双引号)

8.4 JavaScript 的基本语句

- 注释语句
- 条件语句
- 循环语句

8.4.1 注释语句

```
1 //这一行是注释内容
2 /*这里
3  *也是注释内容
4  */
```

8.4.2 条件语句

- 条件运算符
 - 比较运算符
 - $>$, $>=$, $<$, $<=$, $=$, $!=$
 - 逻辑运算符
 - $\&\&$, $\|\|$, $!$
- if 语句

if 语句

```
1 var hour = 20;  
2 if (hour == 20) {  
3     alert("Now is 20 o'clock.");  
4 }
```


if-else-else if 语句 I

```
1 <html>
2   <head>
3     <title>IF DEMO</title>
4     <meta http-equiv="Content-Type"
5       content="text/html; charset=utf-8"/>
6   </head>
7   <body>
8     <h1>条件语句测试</h1>
9     <script type="text/javascript">
10       var d = new Date(); //get current date and time
11       var hour = d.getHours();
12
13       if((hour>=7) && (hour<12)) {
14         msg = '上午';
15       } else if((hour>=12) && (hour<18)){
```

if-else-else if 语句 II

```
16      msg = '下午';  
17      } else if((hour>=18) && (hour<22)) {  
18          msg = '晚上';  
19      } else {  
20          msg = '夜间';  
21      }  
22  
23      document.write('现在是 : ' + msg);  
24  </script>  
25  </body>  
26  </html>
```

8.4.3 循环语句

- for

```
for (初始变量; 循环条件; 变量更新){  
    循环主体语句;  
}
```

- while

```
while (循环条件){  
    循环主体语句;  
}
```

- break & continue

循环语句示例 I

```
1 <html>
2   <head>
3     <title>LOOP DEMO</title>
4     <meta http-equiv="Content-Type"
5       content="text/html; charset=utf-8"/>
6   </head>
7   <body>
8     <h1>条件语句测试</h1>
9     <script type="text/javascript">
10       var i=0, n=30, sum=0;
11       var lastNumber = 0;
12       while(i < n){
13         if(i%2 == 0){
14           i++;
15           continue;
16         } else {
```

循环语句示例 II

```
16         sum = sum + i;  
17         lastNumber = i;  
18         i++;  
19     }  
20     if(i >= 50) break;  
21 }  
22 document.write('1+3+5+...' + lastNumber + '=' + sum);  
23 </script>  
24 </body>  
25 </html>
```

8.5 函数和数组

- 函数
- 数组

8.5.1 函数 I

- 创建函数

```
function 函数名称(参数列表) {  
    函数内部的语句段  
}
```

- 例子

```
1 //函数示例1：函数中包括了alert这个内置的弹出警告框函数  
2 function welcome () {  
3     alert ( "hello,this is a function!");  
4 }  
5  
6 //函数示例2：对a、b两个参数进行求和，并用return语句返回结果  
7 function sum ( a,b) {  
8     return a+b;
```

8.5.1 函数 II

9 }

- 字面量方式创建函数

var 变量名称 = **function** 可选函数名称(函数参数){语句段};

- 例子

```
1 var test = function() {  
2   alert("hello world");  
3 }  
4 test();
```


函数参数

- 形式参数（形参）vs 实际参数（实参）
 - 形式参数只是一个指代，而真正在调用中起作用的是实参
- 例子

```
1 function plus(a,b) { //这里a,b是形式参数
2   return a+b;
3 }
4 plus(3, 5); //将两个数字作为实参传入
5 plus(plus(1,2),plus(2,3)); //将两个函数作为实参传入
```

- Arguments

函数调用与返回

```
1 function add (x,y){  
2   return x+y;  
3 }  
4 alert (add(3,9));
```

8.5.2 数组

• 创建数组

```
1 var base = [1,2,3,4,5,6,7,8];  
2 var table = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]; //此处构造了一个二维数组
```

• 访问和修改

```
1 var base = [1,2,3,4,5,6,7,8,9];  
2 base[0] = 10;  
3 base[1] = 20;  
4 for (var i=0;i<base.length;i++)  
5     base[i]++;  
6 console.info(base);
```

8.6 对象

- 创建对象
- 属性和方法
- 基本类型和引用类型
- 原型与继承
- 类方法

8.6.1 创建对象

- 对象类型

- Object
- Function (函数对象)
- Array (数组对象)
- Date
- Math
- RegExp (正则表达式)
- Error (错误)
- 字符串、布尔、数值的包装对象 : String、Boolean、Number

通过 new 构造对象

```
1 var num=new Object();//自定义一个Object对象的实例并命名为num
2 num.x=1;
3 num.y=2; //为num对象设置两个属性x和y，并分别赋值为1和2
4 var z=new Array(3); //定义一个长度为3的数组对象z
5 var a=new Array (1,2,3,4);
   //定义一个数组对象a并设定前4个元素为1、2、3、4
6 var b=new function(){
7     alert("hello world");
8 } //定义一个函数对象b
```

通过字面量构造对象

```
1 obj_1 = {  
2   x: "h1",  
3   y: function () {  
4     alert("Hello world!");  
5   }  
6 };  
7 obj_2 = {  
8   z: [1,2,3,4],  
9   add: function(a,b) {  
10    return a+b;  
11  }  
12 };
```

8.6.2 属性和方法

```
1 var obj = {};  
2 obj.color = "red";  
3 obj.text = "hello";  
4 obj.work = function(worker){  
5     console.info(worker + " is working!");  
6 };
```


8.6.3 基本类型和引用类型 I

- 基本类型
 - 指数值、字符串、布尔值等这类原始值
- 引用类型
 - 指对象类型

8.6.3 基本类型和引用类型 II

• 例子

```
1 var a = 6;
2 var b = a;
3 var b = 7; //此时b的值为7, a的值仍为6
4 console.info("a=" + a + ", b=" + b);
5 var oneday = new Date(2050, 1, 1);
6 var c = oneday;
7 c.setDate(11);
8 oneday.getDate(); //此时变量c和oneday的日期的天数均变为11
9 console.info("oneday.getDate()=" + oneday.getDate() + ", c.getDate()="
    + c.getDate());
```

– 思考：输出结果为？

8.6.4 原型与继承 I

- JavaScript 借助于原型 (prototype) 实现继承
 - 把 prototype 看作是一个模版，新创建的自定义对象则是这个模版 (prototype) 的一个拷贝
- 例子

```
1 function Animal(name, age) {  
2     this.name = name;  
3     this.age = age;  
4 }  
5 Animal.prototype = {  
6     getName: function(){  
7         return this.name;  
8     },  
9     getAge: function(){  
10        return this.age;  
}
```

8.6.4 原型与继承 II

```
11  }  
12  }  
13  var tom = new Animal("Tom Cat", 2);  
14  console.info(tom.getName() + " is " + tom.getAge() + " years old.");
```

- 每一个对象都有相应的原型
- prototype 属性默认初始化为一个对象

8.6.5 类方法 I

- 实例方法、原型方法和类方法
- 例子

```
1 function Animal(name) {  
2     this.name=name;  
3     //以下定义的introduce 为实例方法  
4     this.introduce=function(){  
5         console.info("I am " + this.name);  
6     }  
7 }  
8  
9 //类方法  
10 Animal.run = function(){  
11     console.info("I can run");  
12 }  
13
```

8.6.5 类方法 II

```
14 //原型方法
15 Animal.prototype.introduceInChinese=function(){
16     console.info("我是"+this.name);
17 }
18
19 //测试
20 var tom=new Animal("Tom Cat");
21 tom.introduce();
22 Animal.run();
    //注意：类方法只能通过类名称调用，不能通过实例调用，如tom.run()
23 tom.introduceInChinese();
```

8.7 浏览器对象模型 BOM

- Window
- Document
- Navigator
- Location
- Screen
- History

- window 是全局对象，每个窗口（包括浏览器窗口和框架窗口）都对应于一个 Window 对象
- 引用方式
 - window. 属性名称或方法名
 - 当访问当前窗口的 window 对象，可以省略 window
 - window 对象是代码在浏览器中解释时的全局对象，当运行环境不是浏览器时，window 对象不存在

常见的 Window 对象的属性和方法

属性/方法名称	描述
document	返回浏览器内置的 Document 对象
navigator	返回浏览器内置的 Navigator 对象
location	返回浏览器内置的 Location 对象
history	返回浏览器内置的 History 对象
screen	返回浏览器内置的 Screen 对象
name	设置或返回窗口的名称
open()	创建一个新的浏览器窗并载入指定的 url
close()	关闭一个浏览器窗口
alert()	弹出一个消息框
confirm()	弹出一个确认框
prompt()	弹出一个提示框

常见的 Document 对象的属性和方法

属性/方法名称	描述
body	访问文档的 body 元素
domain	返回当前文档的域名
title	返回当前文档的标题
URL	返回当前文档的 URL
referrer	返回载入当前文档的原文档的 URL
getElementById()	返回对拥有指定 id 的第一个对象的引用
getElementsByTagName()	返回带有指定标签名称的对象集合
write()	向当前文档写入 HTML 表达式或 JS 代码
writeln()	等同于 write()，但在输出后会添加换行符

练习

- 通过 JavaScript 控制台获取网页 <http://www.ruc.edu.cn/> 中的所有链接

8.8 定时器

- 一次性定时器
 - 设置 : `setTimeout(f, milliseconds);`
 - 取消 : `clearTimeout(timer);`
- 重复定时器
 - 设置 : `setInterval(f, milliseconds);`
 - 取消 : `clearInterval(timer);`

定时器示例

```
1 <html>
2   <head>
3     <title>setTimeout DEMO</title>
4     <script type="text/javascript" src="jquery.js"></script>
5     <script type="text/javascript">
6       $(document).ready(function() {
7         function onTimer(){
8           alert("DING DING");
9         };
10        var timer = setTimeout(onTimer, 2000);
11      });
12    </script>
13  </head>
14  <body>
15    <h1>setTimeout DEMO</h1>
16  </body>
17 </html>
```

END

