WEB前端

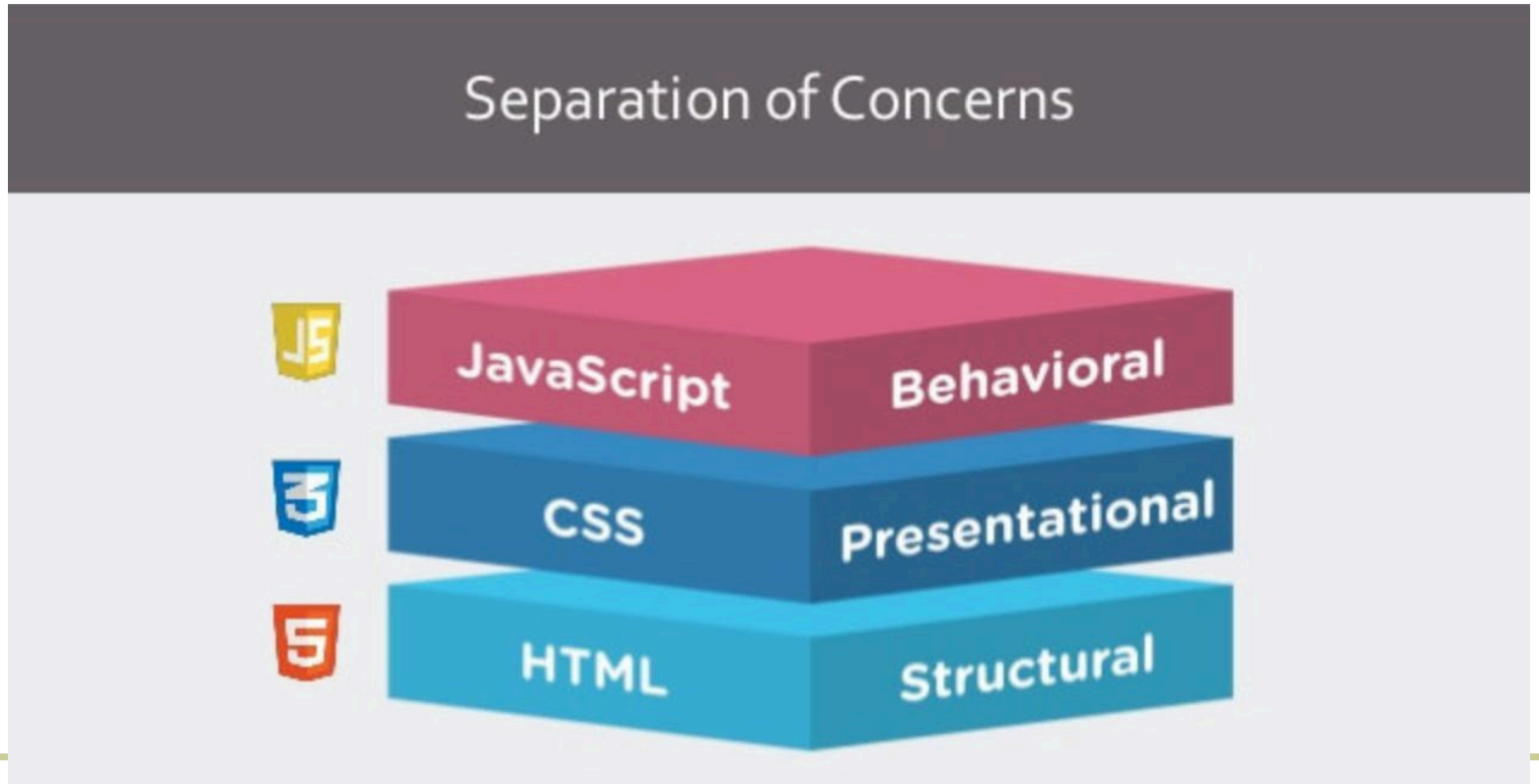
南京大学软件学院

互联网计算

网页开发的原则

关注点分离：各种技术只负责自己的领域，不要混合在一起，形成耦合。

对于网页开发来说，主要是三种技术分离：



网页开发的原则

- **HTML 语言**

  负责网页的结构，又称语义层。

- **CSS 语言**

  负责网页的样式，又称视觉层。

- **JavaScript 语言**

  负责网页的逻辑和交互，又称逻辑层或交互层。

HTML

- HTML简介
- HTML标签
- 剖析一个HTML元素
- 元素的属性
- 分析HTML文档
- 学习网址
- 特殊字符
- HTML语法：注释、块级元素和内联元素等
- HTML的CSS样式

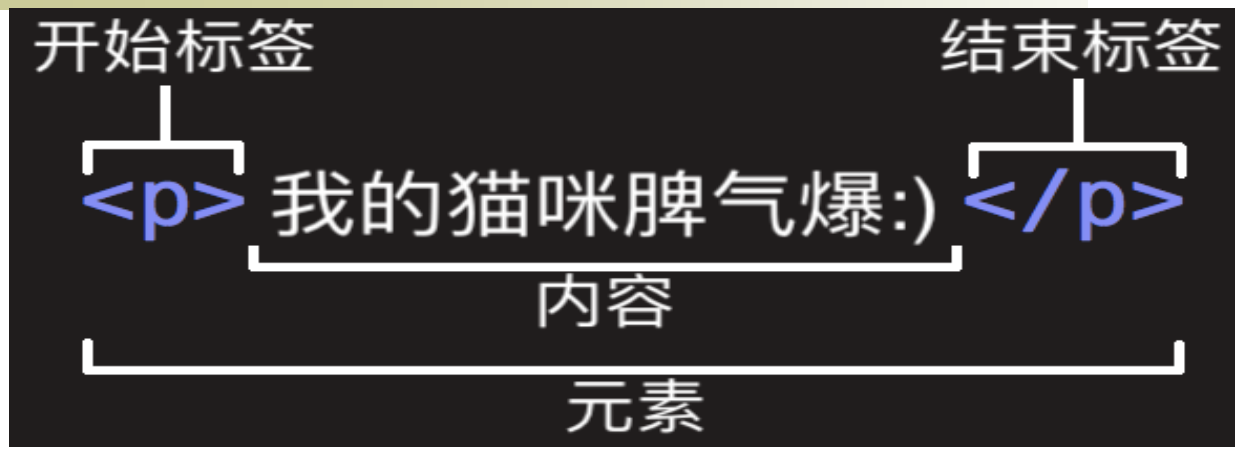
HTML简介

- HTML (Hyper Text Markup Language),是一种超文本标记语言，用来描述网页。
- HTML由一系列的元素（elements）组成，这些元素可以用来包围不同部分的内容，使其以某种方式呈现。
- Web 浏览器的作用是读取 HTML 文档，并以网页的形式显示出它们。浏览器不会显示 HTML 标签，而是使用标签来解释页面的内容。
- 一个HTML文件的后缀名是.htm 或者是.html。
- 用文本编辑器就可以编写HTML文件。

HTML标签

- HTML使用标记标签来描述网页。
  - HTML标签是由尖括号包围的关键词，比如 <html>。
  - HTML 标签通常是成对出现的，比如 和 。标签对中的第一个标签是开始标签，第二个标签是结束标签。
  - HTML 标签不区分大小写。也就是说，输入标签时既可以使用大写字母也可以使用小写字母。例如，标签 <title> 写作<title>、<TITLE>、<Title>、<TiTlE>，等等都可以正常工作。不过，从一致性、可读性等各方面来说，最好仅使用小写字母。

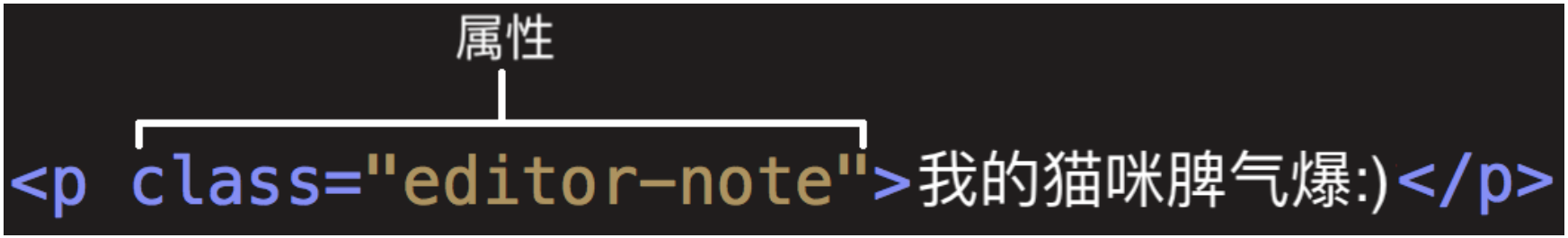
剖析一个HTML元素



这个元素的主要部分有：

- **开始标签**：包含元素的名称（上图中为 p），被<、>所包围。
  表示元素从这里开始或者开始起。
- **结束标签**：与开始标签相似，只是其在元素名之前包含了一个斜杠。
  表示着元素的结尾 。
- **内容**：元素的内容，本例中就是所输入的文本本身。
- **元素**：开始标签、结束标签与内容相结合，便是一个完整的元素。

元素的属性



属性包含元素的额外信息。在上述例子中，这个class属性给元素赋了一个识别的名字，这个名字此后可以被用来识别此元素的样式信息和其他信息。

一个属性必须包含如下内容：

1.在元素和属性之间有个空格space (如果有一个或多个已存在的属性，就与前一个属性之间有一个空格.)

2.属性后面紧跟着一个“=”符号.

3.有一个属性值,由一对引号“ ”引起来.

元素的属性

```
<a href=“https://www.mozilla.org/” title=“链接元素”></a>
```

元素<a>是锚，它使被标签包裹的内容成为一个超链接。

此元素也可以添加大量的属性，其中几个如下：

- **href:** 这个属性声明超链接的web地址，当这个链接被点击，浏览器会跳转至href声明的web地址。例如：href=“https://www.mozilla.org/”。
- **title:** 标题title属性为超链接声明额外的信息，比如你将链接至那个页面。例如：title=“The Mozilla homepage”。当鼠标悬浮时，将出现一个工具提示。
- **target:** 目标target属性指定将用于显示链接的浏览上下文。例如，target=“_blank”将在新标签页中显示链接。如果你希望在目前标签页显示链接，只需忽略这个属性。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<!DOCTYPE html>：声明文档类型。
类型声明类似于链接，规定了HTML页面必须遵从的良好规则，能自动检测错误和其他有用的东西。你只需要知道<!DOCTYPE html>是最短的有效的文档声明。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<html></html>: <html>元素。
这个元素包裹了整个完整的页面，是一个根元素。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<head></head>: <head>元素。
这个元素是一个容器，它包含了所有你想包含在HTML页面中，但不想在HTML页面中显示的内容。这些内容包括CSS样式，字符集声明等等。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<meta charset="utf-8">: 这个元素设置文档使用utf-8字符集编码。
utf-8字符集包含了人类大部分的文字。基本上能识别你放上去的所有文本内容。
毫无疑问要使用它，并且它能在以后避免很多其他问题。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<title></title>: 页面标题。
出现在浏览器标签上，当你标记/收藏页面时它可用来描述页面。

分析HTML文档

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>我的测试站点</title>
6    </head>
7    <body>
8      <p>这是我的页面</p>
9    </body>
10 </html>
```

<body></body>: <body>元素。
包含了你访问页面时所有显示在页面上的内容，文本，图片，音频，游戏等等。

特殊字符

在HTML中，字符 < > “ ‘ & 是特殊字符，它们是HTML语法自身的一部分,。
如何将这些字符包含进你的文本中呢？

——使用字符引用, 每个字符引用以符号&开始, 以分号(;)结束。

原义字符	等价字符引用
<	<
>	>
"	"
'	'
&	&

特殊字符

```
1 <p>HTML 中用 <p> 来定义段落元素。</p>
2
3 <p>HTML 中用 &lt;p&gt; 来定义段落元素</p>
```

HTML 中用

来定义段落元素。

HTML 中用 <p> 来定义段落元素

上面的代码实例和结果展示中，我们可以看出：
第一行是错误的，因为浏览器会认为第二个<p>是开始一个新的段落。
第三行是正确的，因为我们用字符引用来代替了角括号（‘<’和‘>’符号）。

HTML注释

如同大部分的编程语言一样，在HTML中有一种可用的机制来在代码中书写注释。

为了将一段HTML中的内容置为注释，你需要将其用特殊的记号<!--和-->包括起来：比如：

```
<!-- <p>我在注释内！</p> -->
```

块级元素和内联元素

在HTML中有两种元素类别，块级元素和内联元素：

•块级元素

- 在页面中以块的形式展现。相对与其前面的内容它会出现在新的一行，其后的内容也会被挤到下一行展现。
- 通常用于展示页面上结构化的内容，例如段落、列表、导航菜单。
- 一个以block形式展现的块级元素不会被嵌套进内联元素中，但可以嵌套在其它块级元素中。

•内联元素

- 通常出现在块级元素中并包裹文档内容的一小部分，而不是一整个段落或者一组内容。
- 内联元素不会导致文本换行。
- 通常出现在一堆文字之间，例如超链接元素<a>。

块级元素和内联元素

```
1  <em>第一</em><em>第二</em><em>第三</em>
2
3  <p>第四</p><p>第五</p><p>第六</p>
```

HTML语句

 是一个内联元素，所以第一行代码中的三个元素都没有间隙的展示在了同一行。<p>是一个块级元素，所以第二行代码中的每个元素分别都另起了新的一行展现，并且每个段落间都有一些间隔。（这是因为默认的浏览器有着默认的展示<p>元素的CSS styling）。

展示结果

第一第二第三

第四

第五

第六

块级元素和内联元素

常见的块级元素：

div , dl , form , h1 , h2 , h3 , h4 , h5 , h6 , menu , ol , p , table , ul , li

块级元素的宽度始终是与浏览器宽度一样，与内容无关；

常用的内联元素：

a , em , strong , font , img , input , label , select , span , textarea , cite , dfn

内联元素的宽度随着内容增加，高度随字体大小而改变。
内联元素可以设置外边界，但是外边界不对上下起作用，只能对左右起作用。

HTML的CSS样式

当浏览器读到一个样式表，它就会按照这个样式表来对文档进行格式化。
有以下三种方式来插入样式表：

1. 外部样式表

当样式需要被应用到很多页面的时候，外部样式表将是理想的选择。
使用外部样式表，你就可以通过更改一个文件来改变整个站点的外观。

```
<head>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

HTML的CSS样式

2.内部样式表

当单个文件需要特别样式时，就可以使用内部样式表。
可以在 head 部分通过 <style> 标签定义内部样式表。

```
<head>
    <style type="text/css">
        body {background-color: red}
        p {margin-left: 20px}
    </style>
</head>
```

3.内联样式

当特殊的样式需要应用到个别元素时，就可以使用内联样式。
使用内联样式的方法是在相关的标签中使用样式属性，可以包含任何 CSS 属性。

```
<p style="color: red; margin-left: 20px">
    This is a paragraph
</p>
```

学习网址

- http://www.w3school.com.cn/html/html_getstarted.asp
- https://developer.mozilla.org/zh-CN/docs/Learn/HTML/Introduction_to_HTML/Getting_started
- http://www.ruanyifeng.com/blog/2009/05/guide_to_semantic_html_elements.html

CSS

- CSS简介
- 层叠次序
- CSS语法
- CSS的选择器
- 学习网址

CSS简介

- CSS 指层叠样式表 (Cascading Style Sheets)，定义如何显示 HTML 元素。
- 样式表允许以多种方式规定样式信息。样式可以规定在单个HTML元素中，在HTML页头元素中，在一个外部CSS文件中，甚至可以在同一个HTML文档内部引用多个外部样式表。
- 更多情况下，样式保存在外部的 .css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观，极大提高工作效率。
- 多个样式定义可层叠为一。

层叠次序

当同一个 HTML 元素被不止一个样式定义时，会使用哪个样式呢？

一般而言，所有的样式会根据下面的规则层叠于一个新的虚拟样式表中，
其中数字 4 拥有最高的优先权。

1.浏览器缺省设置。

2.外部样式表。

3.内部样式表（位于 <head> 标签内部）。

4.内联样式（在 HTML 元素内部）。

因此，内联样式（在 HTML 元素内部）拥有最高的优先权，
这意味着它将优先于以下的样式声明：

•<head> 标签中的样式声明

•外部样式表中的样式声明

•或者浏览器中的样式声明（缺省值）。

CSS语法

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明。

```
selector {declaration1; declaration2; ... declarationN }
```

选择器通常是您需要改变样式的 HTML 元素。

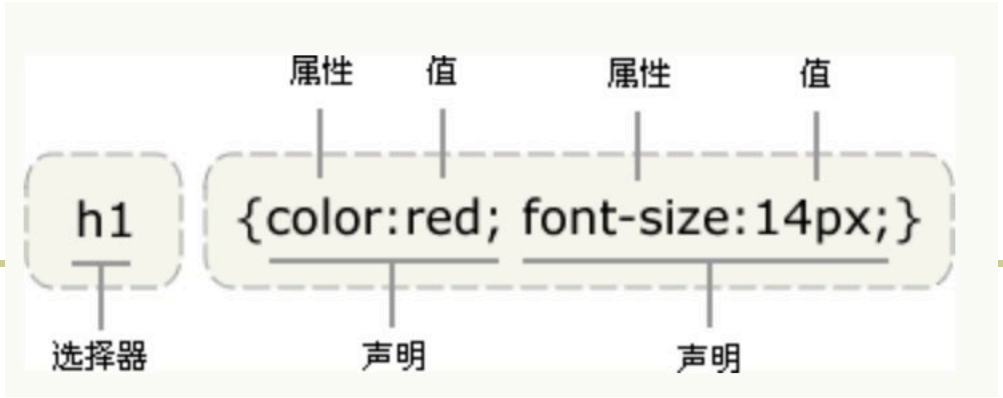
每条声明由一个属性和一个值组成。属性和值被冒号分开。

属性（property）是希望设置的样式属性（style attribute）。

```
selector {property: value}
```

下面代码是将 h1 元素内的文字颜色定义为红色，同时将字体大小设置为 14 像素。

```
h1 {color:red; font-size:14px;}
```



CSS的选择器

- **派生选择器（上下文选择器）**
  - 依据元素在其位置的上下文关系来定义样式。
  - 通过合理地使用派生选择器，我们可以使 HTML 代码变得更加整洁。

比如，希望列表中的 strong 元素变为斜体字，而不是通常的粗体字，
可以这样定义一个派生选择器：

```
li strong {
    font-style: italic;
    font-weight: normal;
}
```

CSS的选择器

- **id 选择器**
  - 可以为标有特定 id 的 HTML 元素指定特定的样式。
  - 以 "#" 来定义。

可以这样定义一个id选择器：下面的两个 id 选择器，
第一个可以定义元素的颜色为红色，第二个定义元素的颜色为绿色：

```
#red {color:red;}
#green {color:green;}
```

CSS的选择器

- **属性选择器**
  - 对带有指定属性的 HTML 元素设置样式。
  - 只有在规定了 !DOCTYPE 时，IE7 和 IE8 才支持属性选择器。在 IE6 及更低的版本中，不支持属性选择。

可以这样定义一个属性选择器：
下面的例子为带有 title 属性的所有元素设置样式：

```
[title]
{
color:red;
}
```

学习网址

- http://www.w3school.com.cn/css/css_jianjie.asp
- https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Introduction_to_CSS
- http://www.ruanyifeng.com/blog/2010/03/css_cookbook.html

JavaScript

- JavaScript简介
- JavaScript使用
- JavaScript基本语法
- JavaScript异常处理
- 学习网址

JavaScript简介

JavaScript 是一门跨平台、面向对象的脚本语言，它能够让网页具有交互性（例如具有复杂的动画，可点击的按钮，通俗的菜单等）。另外还有高级的服务端JavaScript版本，例如Node.js，它可以让你在网页上添加更多功能，不仅仅是下载文件（例如在多台电脑之间的协同合作）。

JavaScript 内置了一些对象的标准库，比如数组（Array），日期（Date），数学（Math）和一套核心语句，包括运算符，流程控制符以及申明方式等。JavaScript 的核心部分可以通过添加对象来扩展语言以适应不同用途。

JavaScript简介

- 客户端的 JavaScript 通过提供控制浏览器及其 DOM 对象来扩展语言核心。例如：客户端版本直接支持应用将元素放在HTML表单中并且支持响应用户事件，比如鼠标点击、表单提交和页面导航。

- 服务端的 JavaScript 则通过提供有关在服务器上运行 JavaScript 的对象来可扩展语言核心。例如：服务端版本直接支持应用和数据库通信，提供应用不同调用间的信息连续性，或者在服务器上执行文件操作。

JavaScript使用

如需在 HTML 页面中插入 JavaScript，请使用 <script> 标签。

<script> 和 </script> 之间的代码行包含了 JavaScript：

```
<script>
    alert("My First JavaScript");
</script>
```

HTML 中的脚本必须位于 <script> 与 </script> 标签之间。脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

```
<!DOCTYPE html>
<html>
<body>
    <script src="myScript.js"></script>
</body>
</html>
```

JavaScript基本语法

JavaScript 是区分大小写的，并使用 Unicode 字符集。
举个例子，可以将单词 Früh （在德语中意思是“早”）用作变量名。

```
var Früh = "foobar";
```

但是，由于 JavaScript 是大小写敏感的，因此变量 früh 和 Früh 则是两个不同的变量。

在 JavaScript 中，指令被称为语句 Statement，并用分号（;）进行分隔。
如果一条语句独占一行的话，那么分号是可以省略的。
但如果一行中有多条语句，那么这些语句必须以分号分开。
虽然不是必需的，但是在一条语句的末尾加上分号可以大大减少代码中产生 bug 的可能性。

JavaScript基本语法

注释

Javascript 注释的语法和 C++ 或许多其他语言类似：

```
1 | // 单行注释
2 |
3 | /* 这是一个更长的，
4 |    多行注释
5 | */
6 |
7 | /* 然而，你不能，/* 嵌套注释 */ 语法错误 */
```

在代码执行过程中，注释将被自动跳过（不执行）。

JavaScript基本语法

变量

在应用程序中，使用变量来作为值的符号名。变量的名字又叫做标识符，其需要遵守一定的规则。一个 JavaScript 标识符必须以字母、下划线（_）或者美元符号（$）开头，后续的字符也可以是数字（0-9）。因为 JavaScript 语言是区分大小写的，所以字母可以是从“A”到“Z”的大写字母和从“a”到“z”的小写字母。

变量声明

Javascript 有三种声明方式：

- var：声明一个变量，可选初始化一个值。
- let：声明一个块作用域的局部变量，可选初始化一个值。
- const:声明一个块作用域的只读常量。

JavaScript基本语法

数据类型

- 数字
- 字符串
- 布尔类型
- 数组
- 对象
- Undefined 和 Null

JavaScript基本语法

声明变量类型

当声明新变量时，可以使用关键词 "new" 来声明其类型：

```
var carname = new String;
var x = new Number;
var y = new Boolean;
var cars = new Array;
var person = new Object;
```

JavaScript 变量均为对象。当声明一个变量时，就创建了一个新的对象。

JavaScript基本语法

变量的作用域

在函数之外声明的变量，叫做全局变量，因为它可被当前文档中的任何其他代码所访问。

在函数内部声明的变量，叫做局部变量，因为它只能在当前函数的内部访问。

变量提升

JavaScript 变量的另一个不同寻常的地方是，你可以先使用变量稍后再声明变量而不会引发异常。这一概念称为变量提升；JavaScript 变量感觉上是被“提升”或移到了函数或语句的最前面。但是，提升后的变量将返回 undefined 值。因此在使用或引用某个变量之后进行声明和初始化操作，这个被提升的变量仍将返回 undefined 值。

JavaScript基本语法

变量提升

左边的例子也可写作右边：

```
/**
 * 例子1
 */
console.log(x === undefined); // true
var x = 3;


/**
 * 例子2
 */
// will return a value of undefined
var myvar = "my value";

(function() {
  console.log(myvar); // undefined
  var myvar = "local value";
})();
```

```
/**
 * 例子1
 */
var x;
console.log(x === undefined); // true
x = 3;

/**
 * 例子2
 */
var myvar = "my value";

(function() {
  var myvar;
  console.log(myvar); // undefined
  myvar = "local value";
})();
```

JavaScript异常处理

可以用 throw 语句抛出一个异常并且用 try...catch 语句捕获处理它。

throw语句

使用throw语句抛出一个异常。当你抛出异常，你规定一个含有值的表达式要被抛出。

你可以抛出任意表达式而不是特定一种类型的表达式。

下面的代码抛出了几个不同类型的表达式：

```
throw "Error2";   // String type
throw 42;         // Number type
throw true;       // Boolean type
throw {toString: function() { return "I'm an object!"; } };
```

JavaScript异常处理

try...catch语句

下面的例子使用了try...catch语句。

示例调用了一个函数用于从一个数组中根据传递值来获取一个月份名称。

如果该值与月份数值不相符，会抛出一个带有“InvalidMonthNo”值的异常，

然后在捕捉块语句中设置monthName变量为unknown。

```
function getMonthName(mo) {
  mo = mo - 1; // Adjust month number for array index (1 = Jan, 12 = Dec)
  var months = ["Jan","Feb","Mar","Apr","May","Jun","Jul",
                "Aug","Sep","Oct","Nov","Dec"];
  if (months[mo]) {
    return months[mo];
  } else {
    throw "InvalidMonthNo"; //throw keyword is used here
  }
}

try { // statements to try
  monthName = getMonthName(myMonth); // function could throw exception
}
catch (e) {
  monthName = "unknown";
  logMyErrors(e); // pass exception object to error handler -> your own function
}
```

JavaScript异常处理

finally块

finally块包含了在try和catch块完成后、下面接着try...catch的语句之前执行的语句。finally块无论是否抛出异常都会执行。

下面的例子中，如果在文件打开时有异常抛出，finally块会在脚本错误之前关闭文件。

```
openMyFile();
try {
    writeMyFile(theData); //This may throw a error
}catch(e){
    handleError(e); // If we got a error we handle it
}finally {
    closeMyFile(); // always close the resource
}
```

学习网址

- http://www.w3school.com.cn/js/js_intro.asp
- https://developer.mozilla.org/zh-CN/docs/Learn/JavaScript
- http://javascript.ruanyifeng.com/introduction/intro.html