JavaScript简介

讲师：李立超

尚硅谷
www.atguigu.com

什么是语言

- 计算机就是一个由人来控制的机器，人让它干嘛，它就得干嘛。
- 我们要学习的语言就是人和计算机交流的工具，人类通过语言来控制、操作计算机。
- 编程语言和我们说的中文、英文本质上没有区别，只是语法比较特殊。
- 语言的发展：
  - 纸带机：机器语言
  - 汇编语言：符号语言
  - 现代语言：高级语言

起源

- JavaScript诞生于1995年，它的出现主要是用于处理网页中的前端验证。
- 所谓的前端验证，就是指检查用户输入的内容是否符合一定的规则。
- 比如：用户名的长度，密码的长度，邮箱的格式等。

尚硅谷
www.atguigu.com

简史

- JavaScript是由网景公司发明，起初命名为LiveScript，后来由于SUN公司的介入更名为了JavaScript。
- 1996年微软公司在其最新的IE3浏览器中引入了自己对JavaScript的实现JScript。
- 于是在市面上存在两个版本的JavaScript，一个网景公司的JavaScript和微软的JScript。
- 为了确保不同的浏览器上运行的JavaScript标准一致，所以几个公司共同定制了JS的标准名命名为ECMAScript。

尚硅谷
www.atguigu.com

时间表

年份	事件
1995 年	网景公司开发了 JavaScript
1996 年	微软发布了和 JavaScript 兼容的 JScript
1997 年	ECMAScript 第 1 版（ECMA-262）
1998 年	ECMAScript 第 2 版
1998 年	DOM Level1 的制定
1998 年	新型语言 DHTML 登场
1999 年	ECMAScript 第 3 版
2000 年	DOM Level2 的制定
2002 年	ISO/ IEC 16262:2002 的确立
2004 年	DOM Level3 的制定
2005 年	新型语言 AJAX 登场
2009 年	ECMAScript 第 5 版
2009 年	新型语言 HTML5 登场

尚硅谷
www.atguigu.com

实现

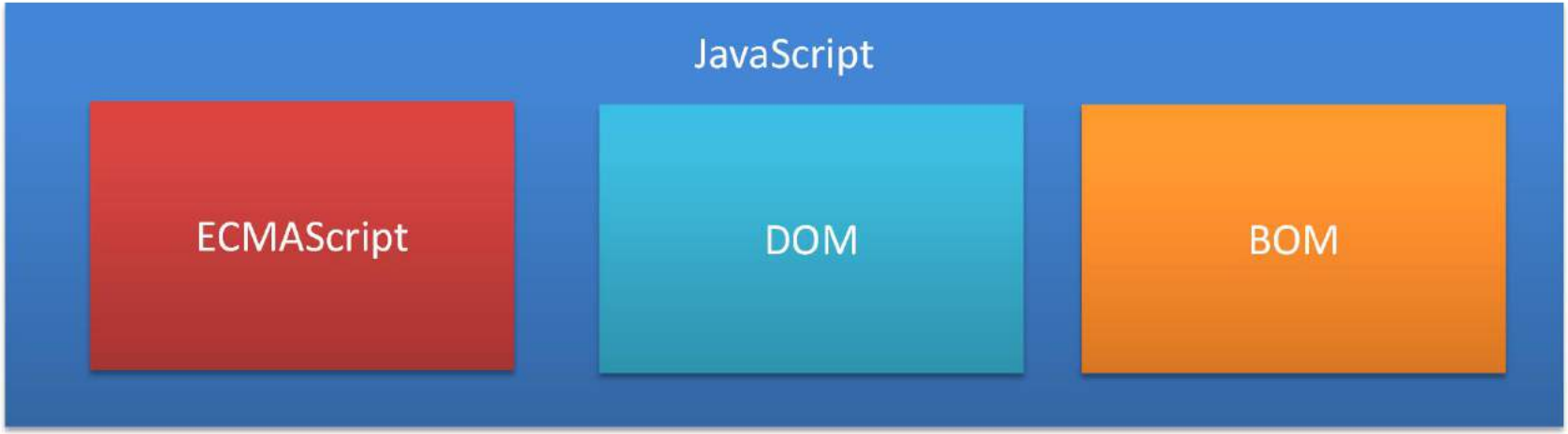
- ECMAScript是一个标准，而这个标准需要由各个厂商去实现。
- 不同的浏览器厂商对该标准会有不同的实现。

浏览器	JavaScript实现方式
FireFox	SpiderMonkey
Internet Explorer	JScript/Chakra
Safari	JavaScriptCore
Chrome	v8
Carakan	Carakan

尚硅谷
www.atguigu.com

实现

- 我们已经知道ECMAScript是JavaScript标准，所以一般情况下这两个词我们认为是一个意思。
- 但是实际上JavaScript的含义却要更大一些。
- 一个完整的JavaScript实现应该由以下三个部分构成：



尚硅谷
www.atguigu.com

学习内容

- 我们已经知道了一个完整的JavaScript实现包含了三个部分：ECMAScript、DOM和BOM。
- 由此我们也知道了我们所要学习的内容就是这三部分。
  - ECMAScript
  - DOM
  - BOM

尚硅谷
www.atguigu.com

特点

- JS的特点
  - 解释型语言
  - 类似于 C 和 Java 的语法结构
  - 动态语言
  - 基于原型的面向对象

尚硅谷
www.atguigu.com

解释型语言

- JavaScript是一门解释型语言，所谓解释型值语言不需要被编译为机器码在执行，而是直接执行。
- 由于少了编译这一步骤，所以解释型语言开发起来尤为轻松，但是解释型语言运行较慢也是它的劣势。
- 不过解释型语言中使用了JIT技术，使得运行速度得以改善。

尚硅谷
www.atguigu.com

类似于 C 和 Java 的语法结构

- JavaScript的语法结构与C和Java很像，向for、if、while等语句和Java的基本上是一模一样的。
- 所以有过C和Java基础的同学学习起来会轻松很多。
- 不过JavaScript和与Java的关系也仅仅是看起来像而已。

尚硅谷
www.atguigu.com

动态语言

- JavaScript是一门动态语言，所谓的动态语言可以暂时理解为在语言中的一切内容都是不确定的。比如一个变量，这一时刻是个整型，下一时刻可能会变成字符串了。当然这个问题我们以后再谈。
- 不过在补充一句动态语言相比静态语言性能上要差一些，不过由于JavaScript中应用的JIT技术，所以JS可能是运行速度最快的动态语言了。

尚硅谷
www.atguigu.com

基于原型的面向对象

- JavaScript是一门面向对象的语言。啥是对象？下次聊。
- Java也是一门面向对象的语言，但是与Java不同JavaScript是基于原型的面向对象。啥是原型？下次聊。


尚硅谷
www.atguigu.com

基本语法

讲师：李立超

编写位置

- 我们目前学习的JS全都是客户端的JS，也就是说全都是需要在浏览器中运行的，所以我们我们的JS代码全都需要在网页中编写。
- 我们的JS代码需要编写到<script>标签中。
- 我们一般将script标签写到head中。（和style标签有点像）
- 属性：
  - type：默认值text/javascript可以不写，不写也是这个值。
  - src：当需要引入一个外部的js文件时，使用该属性指向文件的地址。

Hello World

- 创建一个html文件。
- 在html文件的的head标签中创建一个script标签，并编写如下代码。

```
<script type="text/javascript">
    console.log("Hello World");
</script>
```

尚硅谷
www.atguigu.com

严格区分大小写

- JavaScript是严格区分大小写的，也就是abc和Abc会被解析器认为是两个不同的东西。
- 所以在编写上边的HelloWorld时，一定要注意区分大小写。

尚硅谷
www.atguigu.com

注释

- 注释中的内容不会被解析器解析执行，但是会在源码中显示，我们一般会使用注释对程序中的内容进行解释。
- JS中的注释和Java的的一致，分为两种：
  - 单行注释：//注释内容
  - 多行注释：/*注释内容*/

尚硅谷
www.atguigu.com

标识符

- 所谓标识符，就是指变量、函数、属性的名字，或函数的参数。
- 标识符可以是按照下列格式规则组合起来的一或多个字符：
  - 第一个字符必须是一个字母、下划线（ _ ）或一个美元符号（ $ ）。
  - 其他字符可以是字母、下划线、美元符号或数字。
- 按照惯例，ECMAScript 标识符采用驼峰命名法。
- 但是要注意的是JavaScript中的标识符不能是关键字和保留字符。

关键字和保留字符

- 关键字

break	do	instanceof	typeof	case
else	new	var	catch	finally
return	void	continue	for	switch
while	default	if	throw	delete
in	try	function	this	with
debugger	false	true	null	

- 保留字符

class	enum	extends	super	const	export
import	implements	let	private	public	yield
interface	package	protected	static		

尚硅谷 www.atguigu.com

其他不建议使用的标识符

abstract	double	goto	native	static	boolean
enum	implements	package	super	byte	export
import	private	synchronize	char	extends	int
protected	throws	class	final	interface	public
transient	const	float	long	short	volatile
arguments	encodeURI	Infinity	Number	RegExp	undefined
isFinite	Object	String	Boolean	Error	RangeError
parseFloat	SyntaxError	Date	eval	JSON	ReferenceError
TypeError	decodeURI	EvalError	Math	URIError	decodeURIComponent
Function	NaN	isNaN	parseInt	Array	encodeURICOmponent

尚硅谷
www.atguigu.com

变量

- 变量的作用是给某一个值或对象标注名称。
- 比如我们的程序中有一个值123，这个值我们是需要反复使用的，这个时候我们最好将123这个值赋值给一个变量，然后通过变量去使用123这个值。
- 变量的声明：
  - 使用var关键字声明一个变量。
  - var a;
- 变量的赋值：
  - 使用=为变量赋值。
  - a=123;
- 声明和赋值同时进行：
  - var a = 123;

尚硅谷
www.atguigu.com

数据类型

- 数据类型决定了一个数据的特征，比如：123和”123”，直观上看这两个数据都是123，但实际上前者是一个数字，而后者是一个字符串。
- 对于不同的数据类型我们在进行操作时会有很大的不同。
- JavaScript中一共有5种基本数据类型：
  - 字符串型（String）
  - 数值型（Number）
  - 布尔型（Boolean）
  - null型（Null）
  - undefined型（Undefined）
- 这5种之外的类型都称为Object，所以总的来看JavaScript中共有六种数据类型。

尚硅谷
www.atguigu.com

typeof运算符

- 使用typeof操作符可以用来检查一个变量的数据类型。
- 使用方式：typeof 数据，例如 typeof 123。
- 返回结果：
  - typeof 数值 number
  - typeof 字符串 string
  - typeof 布尔型 boolean
  - typeof undefined undefined
  - typeof null object

String

- String用于表示一个字符序列，即字符串。
- 字符串需要使用’ 或 “ 括起来。
- 转义字符：

转义字符	含义	转义字符	含义
\n	换行	\\	斜杠
\t	制表	\’	单引号
\b	空格	\”	双引号
\r	回车		

- 将其他数值转换为字符串有三种方式：toString()、String()、拼串。

尚硅谷
www.atguigu.com

Number

- Number 类型用来表示整数和浮点数，最常用的功能就是用来表示10进制的整数和浮点数。
- Number表示的数字大小是有限的，范围是：
  - ± 1.7976931348623157e+308
  - 如果超过了这个范围，则会返回± Infinity。
- NaN，即非数值（Not a Number）是一个特殊的数值，JS中当对数值进行计算时没有结果返回，则返回NaN。

尚硅谷
www.atguigu.com

数值的转换

- 有三个函数可以把非数值转换为数值：Number()、parseInt()和parseFloat()。
- Number()可以用来转换任意类型的数据，而后两者只能用于转换字符串。
- parseInt()只会将字符串转换为整数，而parseFloat()可以转换为浮点数。

Boolean(布尔型)

- 布尔型也被称为逻辑值类型或者真假值类型。
- 布尔型只能够取真（true）和假（false）两种数值。除此以外，其他的值都不被支持。
- 其他的数据类型也可以通过Boolean()函数转换为布尔类型。
- 转换规则：

数据类型	转换为true	转换为false
Boolean	true	false
String	任何非空字符串	“”（空字符串）
Number	任何非0数字	0和NaN
Object	任何对象	null
Undefined	n/a	undefined

尚硅谷
www.atguigu.com

Undefined

- Undefined 类型只有一个值，即特殊的 undefined 。
- 在使用 var 声明变量但未对其加以初始化时，这个变量的值就是 undefined。例如：
  - var message;
  - message 的值就是 undefined。
- 需要注意的是typeof对没有初始化和没有声明的变量都会返回undefined。

尚硅谷
www.atguigu.com

Null

- Null 类型是第二个只有一个值的数据类型，这个特殊的值是 null 。
- 从语义上看null表示的是一个空的对象。所以使用typeof检查null会返回一个Object。
- undefined值实际上是由null值衍生出来的，所以如果比较undefined和null是否相等，会返回true；

尚硅谷
www.atguigu.com

运算符

- JS中为我们定义了一套对数据进行运算的运算符。
- 这其中包括：算数运算符、 位运算符、关系运算符等。

算数运算符

- 算数运算符顾名思义就是进行算数操作的运算符。
- JS中为我们提供了多种算数运算符。
- 算数运算符：

运算符	说明	运算符	说明
+	加法	++（前置）	自增
-	减法	++（后置）	自增
*	乘法	--（前置）	自减
/	除法	--（后置）	自减
%	取模	+	符号不变
		-	符号反转

尚硅谷
www.atguigu.com

自增和自减

- 自增 ++ 自减 --
  - 自增和自减分为前置运算和后置元素。
  - 所谓的前置元素就是将元素符放到变量的前边，而后置将元素符放到变量的后边。
  - 例子：
    - 前置自增：++a
    - 后置自减：a—
  - 运算符在前置时，表达式值等于变量原值。
  - 运算符在后置是，表达式值等于变量变更以后的值。

逻辑操作符

- 一般情况下使用逻辑运算符会返回一个布尔值。
- 逻辑运算符主要有三个：非、与、或。
- 在进行逻辑操作时如果操作数不是布尔类型则会将其转换布尔类型在进行计算。
- 非使用符号 ! 表示，与使用 && 表示，或使用 || 表示。

运算符	说明	短路规则
!	逻辑非（NOT）	无
&&	逻辑与（AND）	若左值为假，则不运算右值
||	逻辑或（OR）	若左值为真，则不运算右值

尚硅谷
www.atguigu.com

非

- 非运算符使用！表示。
- 非运算符可以应用于任意值，无论值是什么类型，这个运算符都会返回一个布尔值。
- 非运算符会对原值取反，比如原值是true使用非运算符会返回false，原值为false使用非运算符会返回true。

与

- 与运算符使用 && 表示。
- 与运算符可以应用于任何数据类型，且不一定返回布尔值。
- 对于非布尔值运算，会先将非布尔值转换为布尔值。
- 对布尔值做运算时，如果两个值都为true则返回true，否则返回false。
- 非布尔值时：如果两个都为true，则返回第二个值，如果两个值中有false则返回靠前的false的值。

尚硅谷
www.atguigu.com

或

- 或运算符使用 || 表示。
- 或运算符可以应用于任何数据类型，且不一定返回布尔值。
- 对于非布尔值运算，会先将非布尔值转换为布尔值。
- 对布尔值进行运算时，如果两个值都为false则返回false，否则返回true。
- 非布尔值时：如果两个都为false ，则返回第二个值，否则返回靠前true的值。

尚硅谷
www.atguigu.com

赋值运算符

- 简单的赋值操作符由等于号（ = ）表示，其作用就是把右侧的值赋给左侧的变量。
- 如果在等于号左边添加加减乘除等运算符，就可以完成复合赋值操作。
- +=、*=、-=、/=、%=
- 比如：a+=10和a=a+10是一样的。

尚硅谷
www.atguigu.com

关系运算符

- 小于（<）、大于（>）、小于等于（<=）和大于等于（>=）这几个关系运算符用于对两个值进行比较，比较的规则与我们在数学课上所学的一样。
- 这几个运算符都返回一个布尔值。用来表示两个值之间的关系是否成立。
  - 5 > 10 false
  - 5 < 10 true
  - 5 <= 10 true
  - 5 >= 10 false

尚硅谷 www.atguigu.com

相等

- JS中使用==来判断两个值是否相等，如果相等则返回true。
- 使用!=来表示两个值是否不相等，如果不等则返回true。
- 注意：null和undefined使用==判断时是相等的。

表达式	值	表达式	值
null == undefined	true	true == 1	true
“NaN” == NaN	false	true == 2	false
5 == NaN	false	undefined == 0	false
NaN == NaN	false	null == 0	false
NaN != NaN	true	“5” == 5	true
false == 0	true		

尚硅谷
www.atguigu.com

全等

- 除了==以外，JS中还提供了===。
- ===表示全等，他和==基本一致，不过==在判断两个值时会进行自动的类型转换，而===不会。
- 也就是说"55"==55会返回true，而"55"===55会返回false；
- 同样我们还有!==表示不全等，同样比较时不会自动转型。
- 也就是说"55"!=55会返回false，而"55"!==55会返回true；

尚硅谷
www.atguigu.com

逗号

- 使用逗号可以在一条语句中执行多次操作。
- 比如：var num1=1, num2=2, num3=3;
- 使用逗号运算符分隔的语句会从左到右顺序依次执行。

尚硅谷
www.atguigu.com

条件运算符

- 条件运算符也称为三元运算符。通常运算符写为?:。
- 这个运算符需要三个操作数，第一个操作数在?之前，第二个操作数在?和:之间，第三个操作数在:之后。
- 例如：x > 0 ? x : -x // 求x的绝对值
- 上边的例子，首先会执行x>0，如果返回true则执行冒号左边的代码，并将结果返回，这里就是返回x本身，如果返回false则执行冒号右边的代码，并将结果返回。

尚硅谷
www.atguigu.com

运算符的优先级

- .、[]、 new
- ()
- ++、 --
- !、~、+(单目)、-(单目)、typeof、void、delete
- %、*、/
- +(双目)、-(双目)
- << 、 >>、 >>>
- <、<=、>、>=
- ==、!==、===
- &
- ^
- |
- &&
- ||
- ?:
- =、+=、-=、*=、/=、%=、<<=、>>=、>>>=、&=、^=、|=
- ,

尚硅谷
www.atguigu.com

语句

- 前边我所说表达式和运算符等内容可以理解成是我们一门语言中的单词，短语。
- 而语句（statement）就是我们这个语言中一句一句完整的话了。
- 语句是一个程序的基本单位，JS的程序就是由一条一条语句构成的，每一条语句使用;结尾。
- JS中的语句默认是由上至下顺序执行的，但是我们也可以通过一些流程控制语句来控制语句的执行顺序。

尚硅谷
www.atguigu.com

代码块

- 代码块是在大括号 {} 中所写的语句，以此将多条语句的集合视为一条语句来使用。
- 例如：

```
{
    var a = 123;
    a++;
    alert(a);
}
```

- 我们一般使用代码块将需要一起执行的语句进行分组，需要注意的是，代码块结尾不需要加分号。

尚硅谷
www.atguigu.com

条件语句

- 条件语句是通过判断指定表达式的值来决定执行还是跳过某些语句。
- 最基本的条件语句：
  - if...else
  - switch...case

if...else语句

- if...else语句是一种最基本的控制语句，它让JavaScript可以有条件的执行语句。
- 第一种形式:
```
if(expression)
    statement
```
- 第二种形式:
```
if(expression)
    statement
else
    statement
```
- 除了if和else还可以使用 else if 来创建多个条件分支。

if语句例子

- 例1

```
if(age >= 18){
    alert("您已经成年！");
}
```

- 例2

```
if(age >= 18){
    alert("您已经成年！");
}else{
    alert("你还未成年！");
}
```

- 例3

```
if(age < 18){
    alert("你还未成年！");
}else if(age <= 30){
    alert("您已经是个青年了！")
}else{
    alert("你已经是个中年了！");
}
```

switch...case语句

- switch...case是另一种流程控制语句。
- switch语句更适用于多条分支使用同一条语句的情况。
- 语法：

```
switch(语句){
    case 表达式1:
        语句...
    case 表达式2:
        语句...
    default:
        语句...
}
```

- 需要注意的是case语句只是标识的程序运行的起点，并不是终点，所以一旦符合case的条件程序会一直运行到结束。所以我们一般会在case中添加break作为语句的结束。

尚硅谷
www.atguigu.com

循环语句

- 和条件语句一样，循环语句也是基本的控制语句。
- 循环中的语句只要满足一定的条件将会一直执行。

尚硅谷
www.atguigu.com

while

- while语句是一个最基本的循环语句。
- while语句也被称为while循环。
- 语法：

```
while(条件表达式){
    语句...
}
```

- 和if一样while中的条件表达式将会被转换为布尔类型，只要该值为真，则代码块将会一直重复执行。
- 代码块每执行一次，条件表达式将会重新计算。

do...while

- do...while和while非常类似，只不过它会在循环的尾部而不是顶部检查表达式的值。
- do...while循环会至少执行一次。
- 语法：

```
do{
    语句...
}while(条件表达式);
```

- 相比于while，do...while的使用情况并不是很多。

尚硅谷
www.atguigu.com

for

- for语句也是循环控制语句，我们也称它为for循环。
- 大部分循环都会有一个计数器用以控制循环执行的次数，计数器的三个关键操作是初始化、检测和更新。for语句就将这三步操作明确为了语法的一部分。
- 语法：

```
for(初始化表达式 ; 条件表达式 ; 更新表达式){
    语句...
}
```

尚硅谷
www.atguigu.com

break和continue

- break 和 continue 语句用于在循环中精确地控制代码的执行。
- 使用break语句会使程序立刻退出最近的循环，强制执行循环后边的语句。
- break和continue语句只在循环和switch语句中使用。
- 使用continue语句会使程序跳过当次循环，继续执行下一次循环，并不会结束整个循环。
- continue只能在循环中使用，不能出现在其他的结构中。

label

- 使用 label 语句可以在代码中添加标签，以便将来使用。
- 语法：
  - label: statement
- 例子：

```
start: for (var i=0; i < count; i++) {
    alert(i);
}
```

- 这个例子中定义的 start 标签可以在将来由 break 或 continue 语句引用。加标签的语句一般都要与 for 语句等循环语句配合使用。


尚硅谷
www.atguigu.com

对象

讲师：李立超

尚硅谷
www.atguigu.com

Object对象

- Object类型，我们也称为一个对象。是JavaScript中的引用数据类型。
- 它是一种复合值，它将很多值聚合到一起，可以通过名字访问这些值。
- 对象也可以看做是属性的无序集合，每个属性都是一个名/值对。
- 对象除了可以创建自有属性，还可以通过从一个名为原型的对象那里继承属性。
- 除了字符串、数字、true、false、null和undefined之外，JS中的值都是对象。

尚硅谷
www.atguigu.com

创建对象

- 创建对象有两种方式：
  - 第一种

```
var person = new Object();
person.name = "孙悟空";
person.age = 18;
```

  - 第二种

```
var person = {
        name:"孙悟空",
        age:18
};
```

尚硅谷
www.atguigu.com

对象属性的访问

- 访问属性的两种方式：
  - .访问
    - 对象.属性名
  - []访问
    - 对象[‘属性名’]

尚硅谷
www.atguigu.com

基本数据类型

- JS中的变量可能包含两种不同数据类型的值：基本数据类型和引用数据类型。
- JS中一共有5种基本数据类型：String、Number、Boolean、Undefined、Null。
- 基本数据类型的值是无法修改的，是不可变的。
- 基本数据类型的比较是值的比较，也就是只要两个变量的值相等，我们就认为这两个变量相等。

尚硅谷
www.atguigu.com

引用数据类型

- 引用类型的值是保存在内存中的对象。
- 当一个变量是一个对象时，实际上变量中保存的并不是对象本身，而是对象的引用。
- 当从一个变量向另一个变量复制引用类型的值时，会将对象的引用复制到变量中，并不是创建一个新的对象。
- 这时，两个变量指向的是同一个对象。因此，改变其中一个变量会影响另一个。

尚硅谷
www.atguigu.com

栈和堆

- JavaScript在运行时数据是保存到栈内存和堆内存当中的。
- 简单来说栈内存用来保存变量和基本类型。堆内存用来保存对象。
- 我们在声明一个变量时实际上就是在栈内存中创建了一个空间用来保存变量。
- 如果是基本类型则在栈内存中直接保存，
- 如果是引用类型则会在堆内存中保存，变量中保存的实际上对象在堆内存中的地址。

栈和堆

```
var a = 123;
var b = true;
var c = "hello";
var d = {name:'sunwukong',age:18};
```

栈内存	
d	0x000
c	hello
b	true
a	123

堆内存	
0x000	name = ‘sunwukong’ age = 18

尚硅谷
www.atguigu.com

数组

- 数组也是对象的一种。
- 数组是一种用于表达有顺序关系的值的集合的语言结构。
- 创建数组：
  - var array = [1,44,33];
- 数组内的各个值被称作元素。每一个元素都可以通过索引（下标）来快速读取。索引是从零开始的整数。

尚硅谷
www.atguigu.com

函数

- 函数是由一连串的子程序（语句的集合）所组成的，可以被外部程序调用。向函数传递参数之后，函数可以返回一定的值。
- 通常情况下，JavaScript 代码是自上而下执行的，不过函数体内部的代码则不是这样。如果只是对函数进行了声明，其中的代码并不会执行。只有在调用函数时才会执行函数体内部的代码。
- 这里要注意的是JavaScript中的函数也是一个对象。

尚硅谷
www.atguigu.com

函数的声明（一）

- 首先明确一点函数也是一个对象，所以函数也是在堆内存中保存的。
- 函数声明比较特殊，需要使用function关键字声明。

```
var sum = function(a,b){return a+b};
```

- 上边的例子就是创建了一个函数对象，并将函数对象赋值给了sum这个变量。其中()中的内容表示执行函数时需要的参数，{}中的内容表示函数的主体。

尚硅谷
www.atguigu.com

函数的调用

- 调用函数时，传递给函数的参数称为实参（实际参数）。
- 如果想调用我们上边定义的sum函数，可以这样写：

```
var result = sum(123,456);
```

  - 这样表示调用sum这个函数，并将123和456作为实参传递给函数，函数中会将两个参数求和并赋值给result。

函数的声明（二）

- 可以通过函数声明语句来定义一个函数。函数声明语句以关键字 function 开始，其后跟有函数名、参数列表和函数体。其语法如下所示：

```
function 函数名(参数,参数,参数...){
        函数体
}
```

- 例如:

```
function sum(a,b){
        return a+b;
}
```

  - 上边我们定义了一个函数名为sum，两个参数a和b。函数声明时设置的参数称为形参（形式参数），这个函数对两个参数做了加法运算并将结果返回。

尚硅谷
www.atguigu.com

传递参数

- JS中的所有的参数传递都是按值传递的。也就是说把函数外部的值赋值给函数内部的参数，就和把值从一个变量赋值给另一个变量是一样的。

尚硅谷
www.atguigu.com

执行环境

- 执行环境定义了变量或函数有权访问的其他数据，决定了它们各自的行为。
- 每个执行环境都有一个与之关联的变量对象，环境中定义的所有变量和函数都保存在这个对象中。
- 全局执行环境是最外围的一个执行环境。在 Web 浏览器中，全局执行环境被认为是 window 对象，因此所有全局变量和函数都是作为 window 对象的属性和方法创建的。
- 某个执行环境中的所有代码执行完毕后，该环境被销毁，保存在其中的所有变量和函数定义也随之销毁。
- 在内部环境可以读取外部环境的变量，反之则不行。

尚硅谷
www.atguigu.com

函数内部属性

- 在函数内部，有两个特殊的对象：
  - arguments
    - 该对象实际上是一个数组，用于保存函数的参数。
    - 同时该对象还有一个属性callee来表示当前函数。
  - this
    - this 引用的是一个对象。对于最外层代码与函数内部的情况，其引用目标是不同的。
    - 此外，即使在函数内部，根据函数调用方式的不同，引用对象也会有所不同。需要注意的是，this 引用会根据代码的上下文语境自动改变其引用对象。

this 引用的规则

- 在最外层代码中，this 引用的是全局对象。
- 在函数内，this 根据函数调用方式的不同而有所不同：

函数的调用方式	this引用的对象
构造函数	所生成的对象
调用对象的方法	当前对象
apply或call调用	参数指定的对象
其他方式	全局对象（window）

构造函数

- 构造函数是用于生成对象的函数，像之前调用的Object()就是一个构造函数。
- 创建一个构造函数：

```
function MyClass(x,y) {
    this.x = x;
    this.y = y;
}
```

- 调用构造函数：
  - 构造函数本身和普通的函数声明形式相同。
  - 构造函数通过 new 关键字来调用，new 关键字会新创建一个对象并返回。
  - 通过 new关键字调用的构造函数内的 this 引用引用了（被新生成的）对象。

尚硅谷
www.atguigu.com

new关键字

- 使用new关键字执行一个构造函数时：
  - 首先，会先创建一个空的对象。
  - 然后，会执行相应的构造函数。构造函数中的this将会引用这个新对象。
  - 最后，将对象作为执行结果返回。
- 构造函数总是由new关键字调用。
- 构造函数和普通函数的区别就在于调用方式的不同。
- 任何函数都可以通过new来调用，所以函数都可以是构造函数。
- 在开发中，通常会区分用于执行的函数和构造函数。
- 构造函数的首字母要大写。

尚硅谷
www.atguigu.com

属性的访问

- 在对象中保存的数据或者说是变量，我们称为是一个对象的属性。
- 读取对象的属性有两种方式：
  - 对象.属性名
  - 对象['属性名']
- 修改属性值也很简单：
  - 对象.属性名 = 属性值
- 删除属性
  - delete 对象.属性名
- constructor
  - 每个对象中都有一个constructor属性，它引用了当前对象的构造函数。

尚硅谷
www.atguigu.com

垃圾回收

- 不再使用的对象的内存将会自动回收，这种功能称作垃圾回收。
- 所谓不再使用的对象，指的是没有被任何一个属性（变量）引用的对象。
- 垃圾回收的目的是，使开发者不必为对象的生命周期管理花费太多精力。

尚硅谷
www.atguigu.com

原型继承

- JS是一门面向对象的语言，而且它还是一个基于原型的面向对象的语言。
- 所谓的原型实际上指的是，在构造函数中存在着一个名为原型的(prototype)对象，这个对象中保存着一些属性，凡是通过该构造函数创建的对象都可以访问存在于原型中的属性。
- 最典型的原型中的属性就是toString()函数，实际上我们的对象中并没有定义这个函数，但是却可以调用，那是因为这个函数存在于Object对应的原型中。

尚硅谷
www.atguigu.com

设置原型

- 原型就是一个对象，和其他对象没有任何区别，可以通过构造函数来获取原型对象。
  - 构造函数. prototype
- 和其他对象一样我们可以添加修改删除原型中的属性，也可以修改原型对象的引用。
- 需要注意的是prototype属性只存在于函数对象中，其他对象是没有prototype属性的。
- 每一个对象都有原型，包括原型对象也有原型。特殊的是Object的原型对象没有原型。

尚硅谷
www.atguigu.com

获取原型对象的方法

- 除了可以通过构造函数获取原型对象以外，还可以通过具体的对象来获取原型对象。
  - Object.getPrototypeOf(对象)
  - 对象.__proto__
  - 对象. constructor.prototype
- 需要注意的是，我们可以获取到Object的原型对象，也可以对它的属性进行操作，但是我们不能修改Object原型对象的引用。

尚硅谷
www.atguigu.com

原型链

- 基于我们上边所说的，每个对象都有原型对象，原型对象也有原型对象。
- 由此，我们的对象，和对象的原型，以及原型的原型，就构成了一个原型链。
- 比如这么一个对象：
  - var mc = new MyClass(123,456);
  - 这个对象本身，原型MyClass.proprototype原型对象的原型对象是Object，Object对象还有其原型。这组对象就构成了一个原型链。
  - 这个链的次序是：mc对象、mc对象原型、原型的原型（Object）、Object的原型
- 当从一个对象中获取属性时，会首先从当前对象中查找，如果没有则顺着向上查找原型对象，直到找到Object对象的原型位置，找到则返回，找不到则返回undefined。

尚硅谷
www.atguigu.com

instanceof

- 之前学习基本数据类型时我们学习了typeof用来检查一个变量的类型。
- 但是typeof对于对象来说却不是那么好用，因为任何对象使用typeof都会返回Object。而我们想要获取的是对象的具体类型。
- 这时就需要使用instanceof运算符了，它主要用来检查一个对象的具体类型。
- 语法：
  - var result = 变量 instanceof 类型

尚硅谷
www.atguigu.com

引用类型

- 上边我们说到JS中除了5种基本数据类型以外其余的全都是对象，也就是引用数据类型。
- 但是虽然全都是对象，但是对象的种类却是非常繁多的。比如我们说过的Array（数组），Function（函数）这些都是不同的类型对象。
- 实际上在JavaScript中还提供了多种不同类型的对象。

尚硅谷
www.atguigu.com

Object

- 目前为止，我们看到的最多的类型就是Object，它也是我们在JS中使用的最多的对象。
- 虽然Object对象中并没有为我们提供太多的功能，但是我们会经常会用途来存储和传输数据。
- 创建Object对象有两种方式：
  - var obj = new Object();
  - var obj = {}
- 上边的两种方式都可以返回一个Object对象。
- 但是第一种我们使用了一个new关键字和一个Object()函数。
- 这个函数就是专门用来创建一个Object对象并返回的，像这种函数我们称为构造函数。

尚硅谷
www.atguigu.com

Array

- Array用于表示一个有序的数组。
- JS的数组中可以保存任意类型的数据。
- 创建一个数组的方式有两种：
  - 使用构造器：
    - var arr = new Array(数组的长度);
    - var arr = new Array(123,’hello’,true);
  - 使用[]
    - var arr = [];
    - var arr = [123,’hello’,false];
- 读取数组中的值使用数组[索引]的方式，注意索引是从0开始的。

尚硅谷
www.atguigu.com

Date

- Date类型用来表示一个时间。
- Date采取的是时间戳的形式表示时间，所谓的时间戳指的是从1970年1月1日0时0秒0分开始经过的毫秒数来计算时间。
- 直接使用new Date()就可以创建一个Date对象。
- 创造对象时不传参数默认创建当前时间。可以传递一个毫秒数用来创建具体的时间。
- 也可以传递一个日期的字符串，来创建一个时间。
  - 格式为：月份/日/年 时:分:秒
  - 例如：06/13/2004 12:12:12

尚硅谷
www.atguigu.com

Function

- Function类型代表一个函数，每一个函数都是一个Function类型的对象。而且都与其他引用类型一样具有属性和方法。
- 由于函数是对象，因此函数名实际上也是一个指向函数对象的指针，不会与某个函数绑定。
- 函数的声明有两种方式：
  - function sum(){}
  - var sum = function(){};
- 由于存在函数声明提升的过程，第一种方式在函数声明之前就可以调用函数，而第二种不行。

尚硅谷
www.atguigu.com

函数也可以作为参数

- 函数也是一个对象，所以函数和其他对象一样也可以作为一个参数传递给另外一个函数。
- 但是要注意的是使用函数作为参数时，变量后边千万不要加()，不加()表示将函数本身作为参数，加上以后表示将函数执行的结果作为参数。

尚硅谷
www.atguigu.com

函数对象的方法

- 每个函数都有两个方法call()和apply()。
- call()和apply()都可以指定一个函数的运行环境对象，换句话说就是设置函数执行时的this值。
- 使用方式：
  - 函数对象.call(this对象,参数数组)
  - 函数对象.apply(this对象,参数1,参数2,参数N)

尚硅谷
www.atguigu.com

闭包（closure）

- 闭包是JS一个非常重要的特性，这意味着当前作用域总是能够访问外部作用域中的变量。因为函数是JS中唯一拥有自身作用域的结构，因此闭包的创建依赖于函数。
- 也可以将闭包的特征理解为，其相关的局部变量在函数调用结束之后将会继续存在。

尚硅谷
www.atguigu.com

基本包装类型

- 基本数据类型是不能去调用方法的，所以JS中还提供了3个特殊的引用类型：
  - Boolean
  - Number
  - String
- 这三个类型分别包装了Boolean、Number、String并扩展了许多实用的方法。
- 他们的使用方式和普通的对象一样。
- 要注意的是使用typeof检查这些包装类型时返回的都是object。

尚硅谷
www.atguigu.com

Boolean

- Boolean 类型是与布尔值对应的引用类型。
- 可以采用这种方式创建：
  - var booleanObject = new Boolean(true);
- 我们最好永远不要使用Boolean包装类。

尚硅谷
www.atguigu.com

Number

- Number是数值对应的引用数据类型。创建Number对象只需要在调用构造函数时传递一个数值：
  - var num = new Numbaer(20);
- 使用数值时我们建议使用基本数值，而不建议使用包装类。

尚硅谷
www.atguigu.com

String

- String 类型是字符串的对象包装类型，可以像下面这样使用 String 构造函数来创建。
  - var str = new String("hello world");
- 可以使用length属性来获取字符串的长度。

尚硅谷
www.atguigu.com

Math

- JS 还为保存数学公式和信息提供了一个公共位置，即 Math 对象。
- 与我们在 JavaScript 直接编写的计算功能相比， Math 对象提供的计算功能执行起来要快得多。 Math 对象中还提供了辅助完成这些计算的属性和方法。

尚硅谷
www.atguigu.com

Math对象的属性

属性	说明
Math.E	自然对数的底数，即常量 e 的值
Math.LN10	10的自然对数
Math.LN2	2的自然对数
Math.LOG2E	以2为底 e 的对数
Math.LOG10E	以10为底 e 的对数
Math.PI	π的值
Math.SQRT1_2	1/2的平方根（即2的平方根的倒数）
Math.SQRT2	2的平方根

尚硅谷
www.atguigu.com

Math的方法

- 最大最小值
  - Math.max()获取最大值
  - Math.min()获取最小值
- 舍入：
  - 向上舍 Math.ceil()
  - 向下舍 Math.floor()
  - 四舍五入 Math.round()
- 随机数： Math.random()
  - 选取某个范围内的随机值：
    - 值 = Math.floor(Math.random() * 可能值的总数 + 第一个可能的值)


尚硅谷
www.atguigu.com

BOM

讲师：李立超

尚硅谷
www.atguigu.com

什么是BOM

- ECMAScript无疑是JavaScript的核心，但是要想在浏览器中使用JavaScript，那么BOM（浏览器对象模型）才是真正的核心。
- BOM 提供了很多对象，用于访问浏览器的功能，这些功能与任何网页内容无关。
- BOM将浏览器中的各个部分转换成了一个一个的对象，我们通过修改这些对象的属性，调用他们的方法，从而控制浏览器的各种行为。

尚硅谷
www.atguigu.com

window对象

- window对象是BOM的核心，它表示一个浏览器的实例。
- 在浏览器中我们可以通过window对象来访问操作浏览器，同时window也是作为全局对象存在的。
- 全局作用域：
  - window对象是浏览器中的全局对象，因此所有在全局作用域中声明的变量、对象、函数都会变成window对象的属性和方法。

尚硅谷
www.atguigu.com

窗口大小

- 浏览器中提供了四个属性用来确定窗口的大小：
  - 网页窗口的大小
    - innerWidth
    - innerHeight
  - 浏览器本身的尺寸
    - outerWidth
    - outerHeight

尚硅谷
www.atguigu.com

打开窗口

- 使用 window.open() 方法既可以导航到一个特定的 URL，也可以打开一个新的浏览器窗口。
- 这个方法需要四个参数：
  - 需要加载的url地址
  - 窗口的目标
  - 一个特性的字符串
  - 是否创建新的历史记录

尚硅谷
www.atguigu.com

超时调用

- 超时调用：
  - setTimeout()
  - 超过一定时间以后执行指定函数
  - 需要连个参数：
    - 要执行的内容
    - 超过的时间
- 取消超时调用
  - clearTimeout()
- 超时调用都是在全局作用域中执行的。

尚硅谷
www.atguigu.com

间歇调用

- 间歇调用：
  - setInterval()
  - 每隔一段时间执行指定代码
  - 需要两个参数：
    - 要执行的代码
    - 间隔的时间
- 取消间隔调用：
  - clearInterval()

尚硅谷
www.atguigu.com

系统对话框

- 浏览器通过 alert() 、 confirm() 和 prompt() 方法可以调用系统对话框向用户显示消息。
- 它们的外观由操作系统及（或）浏览器设置决定，而不是由 CSS 决定。
- 显示系统对话框时会导致程序终止，当关闭对话框程序会恢复执行。

alert

- alert()接收一个字符串并显示给用户。调用alert()方法会向用户显示一个包含一个确认按钮的对话框。
- 例如：
  - alert("Hello World");

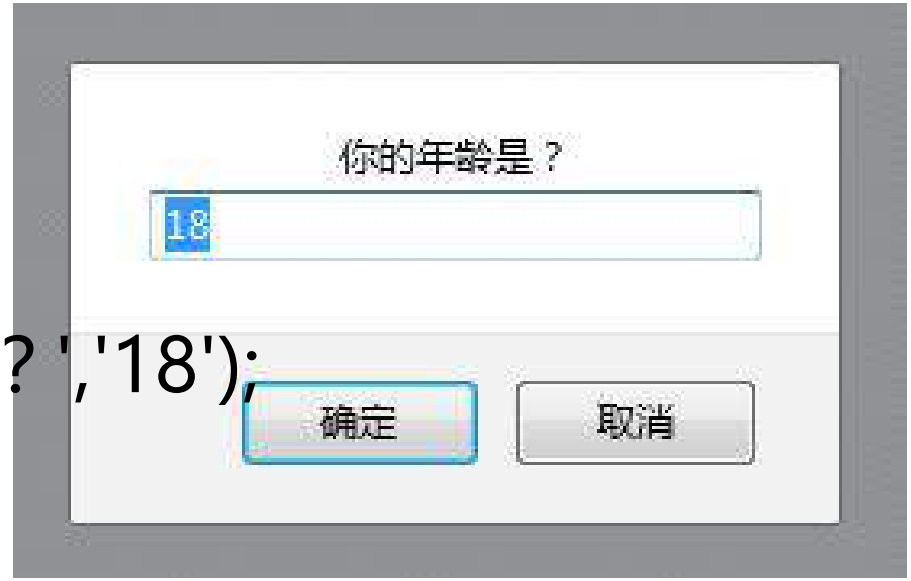
confirm

- confirm和alert类似，只不过confirm弹出的对话框有一个确认和取消按钮。用户可以通过按钮来确认是否执行操作。
- 例如：
  - confirm('你确定吗？');

- 这个函数的执行会返回一个布尔值，如果选择确定则返回true，如果点击取消则返回false。

prompt

- prompt会弹出一个带输入框的提示框，并可以将用户输入的内容返回。
- 它需要两个值作为参数：
  - 显示的提示文字
  - 文本框中的默认值
- 例子：
  - prompt('你的年龄是？','18');



尚硅谷
www.atguigu.com

location对象

- location对象提供了与当前窗口中加载的文档有关的信息，还提供了一些导航功能。
- href属性：
  - href属性可以获取或修改当前页面的完整的URL地址，使浏览器跳转到指定页面。
- assign() 方法
  - 所用和href一样，使浏览器跳转页面，新地址错误参数传递到assign ()方法中
- replace()方法
  - 功能一样，只不过使用replace方法跳转地址不会体现到历史记录中。
- reload() 方法
  - 用于强制刷新当前页面

尚硅谷
www.atguigu.com

navigator对象

- navigator 对象包含了浏览器的版本、浏览器所支持的插件、浏览器所使用的语言等各种与浏览器相关的信息。
- 我们有时会使用navigator的userAgent属性来检查用户浏览器的版本。

尚硅谷
www.atguigu.com

screen对象

- screen 对象基本上只用来表明客户端的能力，其中包括浏览器窗口外部的显示器的信息，如像素宽度和高度等。
- 该对象作用不大，我们一般不太使用。

尚硅谷
www.atguigu.com

history对象

- history 对象保存着用户上网的历史记录，从窗口被打开的那一刻算起。
- go()
  - 使用 go() 方法可以在用户的历史记录中任意跳转，可以向后也可以向前。
- back()
  - 向后跳转
- forward()
  - 向前跳转

尚硅谷
www.atguigu.com

document

- document对象也是window的一个属性，这个对象代表的是整个网页的文档对象。
- 我们对网页的大部分操作都需要以document对象作为起点。
- 关于document对象的内容，我们后边还要具体讲解。


尚硅谷
www.atguigu.com

DOM

讲师：李立超

尚硅谷
www.atguigu.com

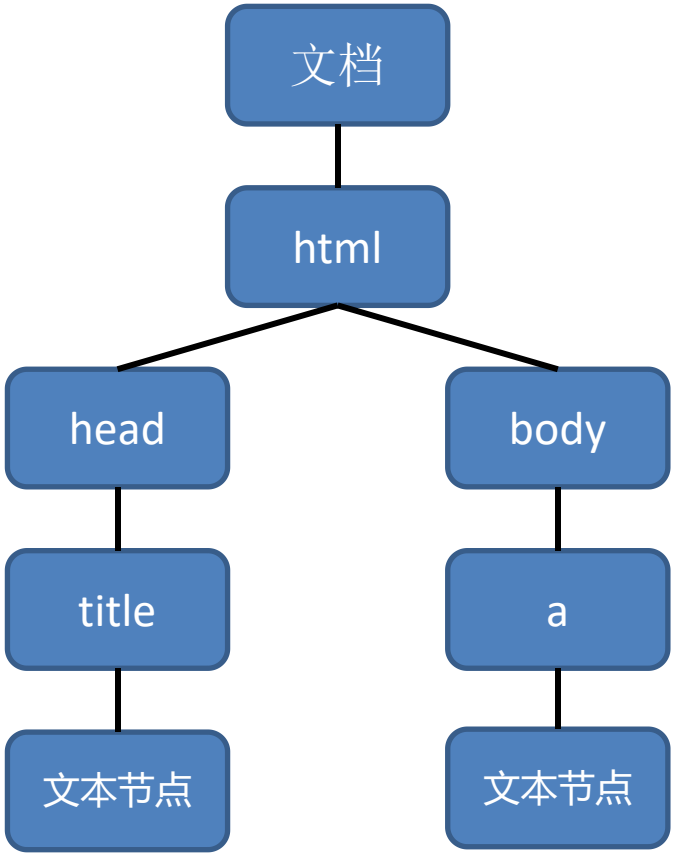
什么是DOM

- DOM，全称Document Object Model文档对象模型。
- JS中通过DOM来对HTML文档进行操作。只要理解了DOM就可以随心所欲的操作WEB页面。
- 文档
  - 文档表示的就是整个的HTML网页文档
- 对象
  - 对象表示将网页中的每一个部分都转换为了一个对象。
- 模型
  - 使用模型来表示对象之间的关系，这样方便我们获取对象。

模型

1.html

```
<html>
        <head>
                <title>网页的标题</title>
        </head>
        <body>
                <a href="1.html" >超连接</a>
        </body>
</html>
```



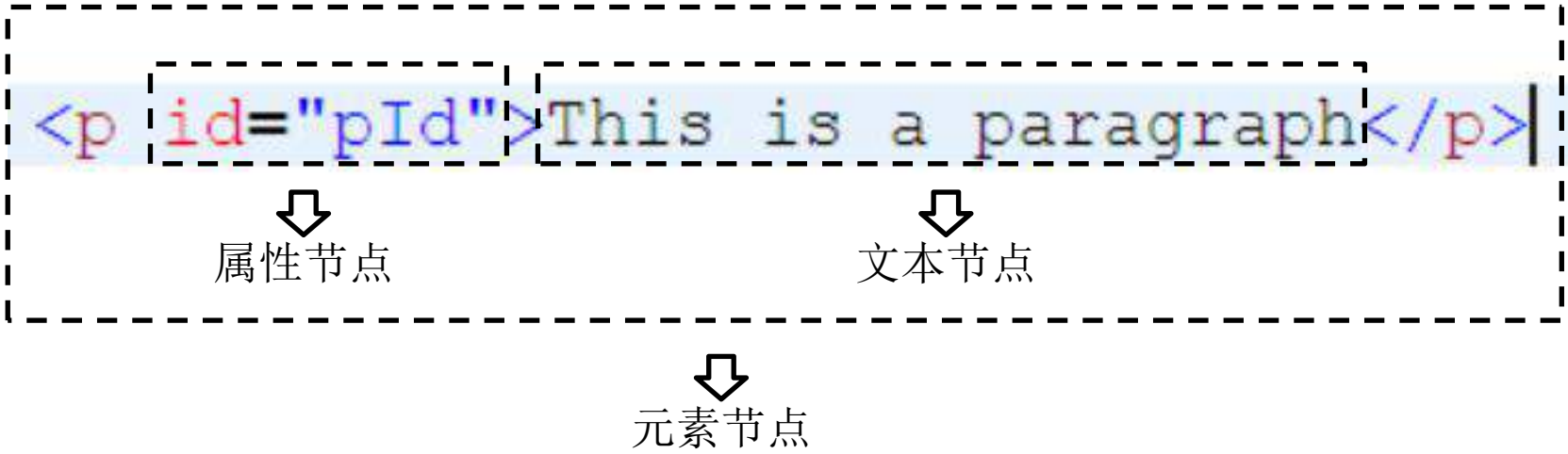
尚硅谷
www.atguigu.com

节点

- 节点Node，是构成我们网页的最基本的组成部分，网页中的每一个部分都可以称为是一个节点。
- 比如：html标签、属性、文本、注释、整个文档等都是一个节点。
- 虽然都是节点，但是实际上他们的具体类型是不同的。
- 比如：标签我们称为元素节点、属性称为属性节点、文本称为文本节点、文档称为文档节点。
- 节点的类型不同，属性和方法也都不尽相同。

节点

- 节点：Node——构成HTML文档最基本的单元。
- 常用节点分为四类
  - **文档节点**：整个HTML文档
  - **元素节点**：HTML文档中的HTML标签
  - **属性节点**：元素的属性
  - **文本节点**：HTML标签中的文本内容



尚硅谷
www.atguigu.com

节点的属性

	nodeName	nodeType	nodeValue
文档节点	#document	9	null
元素节点	标签名	1	null
属性节点	属性名	2	属性值
文本节点	#text	3	★文本内容

尚硅谷
www.atguigu.com

文档节点（document）

- 文档节点document，代表的是整个HTML文档，网页中的所有节点都是它的子节点。
- document对象作为window对象的属性存在的，我们不用获取可以直接使用。
- 通过该对象我们可以在整个文档访问内查找节点对象，并可以通过该对象创建各种节点对象。

尚硅谷
www.atguigu.com

元素节点（Element）

- HTML中的各种标签都是元素节点，这也是我们最常用的一个节点。
- 浏览器会将页面中所有的标签都转换为一个元素节点，我们可以通过document的方法来获取元素节点。
- 比如：
  - document.getElementById()
  - 根据id属性值获取一个元素节点对象。

尚硅谷
www.atguigu.com

文本节点（Text）

- 文本节点表示的是HTML标签以外的文本内容，任意非HTML的文本都是文本节点。
- 它包括可以字面解释的纯文本内容。
- 文本节点一般是作为元素节点的子节点存在的。
- 获取文本节点时，一般先要获取元素节点。在通过元素节点获取文本节点。
- 例如：
  - 元素节点.firstChild;
  - 获取元素节点的第一个子节点，一般为文本节点

尚硅谷
www.atguigu.com

属性节点（Attr）

- 属性节点表示的是标签中的一个一个的属性，这里要注意的是属性节点并非是元素节点的子节点，而是元素节点的一部分。
- 可以通过元素节点来获取指定的属性节点。
- 例如：
  - 元素节点.getAttributeNode("属性名");
- 注意：我们一般不使用属性节点。

尚硅谷
www.atguigu.com

事件

- 事件，就是文档或浏览器窗口中发生的一些特定的交互瞬间。
- JavaScript 与 HTML 之间的交互是通过事件实现的。
- 对于 Web 应用来说，有下面这些代表性的事件：点击某个元素、将鼠标移动至某个元素上方、按下键盘上某个键，等等。

获取元素节点

- 通过document对象调用

1. getElementById()
   - 通过**id**属性获取**一个**元素节点对象
2. getElementsByTagName()
   - 通过**标签名**获取**一组**元素节点对象
3. getElementsByName()
   - 通过**name**属性获取**一组**元素节点对象

尚硅谷
www.atguigu.com

获取元素节点的子节点

- 通过具体的元素节点调用

1. getElementsByTagName()
   - 方法，返回当前节点的指定标签名后代节点
2. childNodes
   - 属性，表示当前节点的所有子节点
3. firstChild
   - 属性，表示当前节点的第一个子节点
4. lastChild
   - 属性，表示当前节点的最后一个子节点

尚硅谷
www.atguigu.com

获取父节点和兄弟节点

- 通过具体的节点调用

1. parentNode
   - 属性，表示当前节点的父节点
2. previousSibling
   - 属性，表示当前节点的前一个兄弟节点
3. nextSibling
   - 属性，表示当前节点的后一个兄弟节点

元素节点的属性

- 获取，元素对象.属性名

例：element.value
element.id
element.className

- 设置，元素对象.属性名=新的值

例：element.value = “hello”
element.id = “id01”
element.className = “newClass”

尚硅谷
www.atguigu.com

其他属性

- nodeValue
  - 文本节点可以通过nodeValue属性获取和设置文本节点的内容
- innerHTML
  - 元素节点通过该属性获取和设置标签内部的html代码

尚硅谷
www.atguigu.com

使用CSS选择器进行查询

- querySelector()
- querySelectorAll()
- 这两个方法都是用document对象来调用，两个方法使用相同，都是传递一个选择器字符串作为参数，方法会自动根据选择器字符串去网页中查找元素。
- 不同的地方是querySelector()只会返回找到的第一个元素，而querySelectorAll()会返回所有符合条件的元素。

尚硅谷
www.atguigu.com

节点的修改

- 这里的修改我们主要指对元素节点的操作。
- 创建节点
  - document.createElement(标签名)
- 删除节点
  - 父节点.removeChild(子节点)
- 替换节点
  - 父节点.replaceChild(新节点 , 旧节点)
- 插入节点
  - 父节点.appendChild(子节点)
  - 父节点.insertBefore(新节点 , 旧节点)


尚硅谷
www.atguigu.com