JavaScript

千锋教育
QIAN FENG EDUCATION

做真实的自己，用良心做教育

张骏
H5教学部

JavaScript基础

课程概要：

什么是JavaScript
JavaScript的发展史
JavaScript的实现
JavaScript和HTML5的关系
JavaScript的特点
JavaScript运行环境
浏览器内核
JavaScript语法规范
变量
命名规范
JavaScript数据类型
JavaScript运算符
数据类型转换

什么是JavaScript

- javaScript是一种基于对象和事件驱动并具有相对安全性并广泛用于客户端网页开发的脚本语言，同时也是一种广泛用于客户端Web开发的脚本语言。

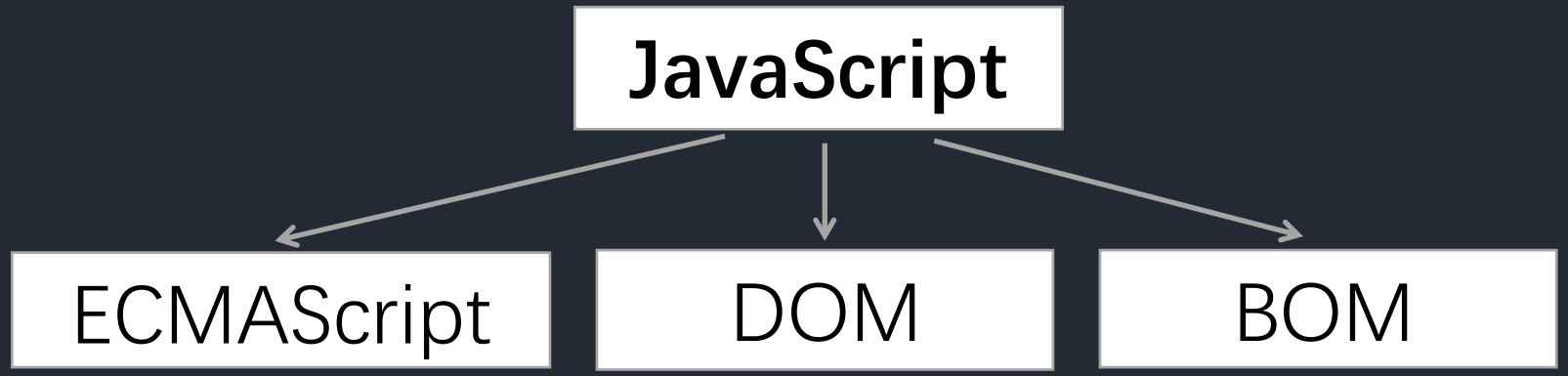
- JavaScript的核心部分相当精简，只包括两个部分：基本的语法构造（比如操作符、控制结构、语句）和标准库（就是一系列具有各种功能的对象比如Array、Date、Math等）。

JavaScript的发展史

- 1992 年一家称作 Nombas 的公司为自己的软件开发了一种叫做ScriptEase的脚本语言可以嵌入在网页当中
- 1995年，Netscape公司为自己的Navigator2.0浏览器开发了另一种客户端脚本语言livescript为了赶时髦改成了JavaScript但是跟Java没有任何关系
- 1996年Microsoft公司为了进军浏览器的市场，在IE3.0浏览器发布了一个JavaScript的克隆版，称为JScript
- 早期的JavaScript指只能运行于Navigator浏览器中的脚本语言 后来JavaScript被提交给ECMA(European Computer Manufacturers Association)，成为ECMAScript标准，由各大浏览器厂家来实现。

JavaScript的实现

- 完整的JavaScript由三部分组成：
- JavaScript的核心(ECMAScript);
- 文档对象模型(DOM , Document Object Model);
- 浏览器对象模型(BOM , Browser Object Model);



JavaScript和HTML5的关系

HTML5 “就是” JavaScript

HTML 5 与 HTML 关系不大，它其实是 JavaScript。那么，HTML 本身有什么变化？它不过是一些很好理解的新标签而已。HTML 5 的威力在于让你能用 JavaScript 来创建这些标签。如果没有后台代码的支持，来创建一个动画效果、游戏、或是可视化工具，那么，图形画布（Canvas）毫无用处。自从出现支持 Canvas 的浏览器，我们看到了Asteroids 游戏的上百个实现，这些都是开发人员为了熟悉HTML 5的新特性。有的比较粗糙，而有的则极其精致。这些工作完全归功于 JavaScript。

JavaScript 的特点

- 代码可以使用任何文本编辑工具编写
- 无需编译，由JavaScript引擎解释执行
- 弱类型语言
- 基于对象和原型

JavaScript运行环境

- JavaScript运行在浏览器内核中的JavaScript解释器内
  - 哪些地方可以执行JavaScript
    - 浏览器的控制台(console) Chrome控制台快捷键 ctrl+shift+J
    - 将js脚本嵌入在html页面中执行
      - html元素事件
      - script元素
      - 外部脚本文件

浏览器的内核

- 浏览器的内核负责页面内容的渲染，主要由两部分组成：
  - 内容排版----解析HTML/CSS
  - 脚本解释----解析JavaScript

浏览器	内核名称	脚本解释引擎
Microsoft IE	Trident	Chakra
Mozilia FireFox	Gecko	猴子系列
Apple Safari	Webkit	Nitro
Google Chrome	Blink	V8
Opera	Blink	Blink

html元素事件

- 要执行的脚本语句直接编写在body中

```
<body>
    <button onclick="console.log('hello world')">打印</button>
</body>
```

<script>元素

- 内部脚本块中可以执行多条语句：

```
<script>
        document.write('<b>你好</b>');
        console.log('脚本执行完成');
</script>
```

外部脚本文件

- 将JavaScript代码写入一个单独的文件，并保存为后缀名为js的文件
  - 为纯文本文件
  - 文件中，不需要包含<script>标签，直接书写js代码
  - 在<head>中添加<script>标签
  - 设置<script>标签的“src”属性，以指定js文件的url

JavaScript调试

- JavaScript的代码错误：解释型语言，若某代码错误，则解释器会终止执行，但不会影响后续块的执行，以及HTML的解析；

- 使用浏览器的开发者工具查看JavaScript中的错误

JavaScript的语法规范

- 语句----会被JavaScript引擎解释执行的代码
  - 由表达式、关键字、运算符组成；
  - 大小写敏感；
  - 使用分号或者换行结束

注释不会被JavaScript引擎解释执行的代码

单行注释：//

多行注释：/* */

JavaScript的大小写敏感

在JavaScript程序中大小写是敏感的
uname、UNAME、uName是不同的

基本输出语句：

```
console.log();//控制台输出
document.write();//页面输出
alert();//弹出提示框
```

变量

ECMAScript的变量是松散类型的，所谓松散类型就是可以用来保存任何类型的数据。定义变量时要使用var操作符(var是一个关键字)，后跟变量名(即一个标识符)。

当声明一个变量时，就创建了一个新的对象。

什么是变量

变量是存储信息的容器。

例如：
```
x = 1;
y = 3;
sum = x + y;
```

在JavaScript中，这些字母被成为变量。

变量的声明

- 使用关键字 var 如：

  var name；

  使用 "=" 为变量赋值

  var price = 25；

  没有初始化的变量自动取值为undefined

  var date；

  console.log(date);

一条语句中声明多个变量

- 可以在一条语句中声明多个变量，变量名使用“,”分隔
  - var name1,name2,name3;
  - var age1,age2 = 30;

命名规范

- 标识符命名规范
  - 不能使用关键字和保留字
  - 可以包含字母、数字、下划线、美元符号
  - 不能以数字开头
  - 常用于表示函数、变量等名称
  - 名称最好有明确的含义
  - 可以采用 小驼峰命名法，大驼峰命名法

关键字和保留字

ECMAScript 关键字（共25个）				
break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with

ECMAScript 保留字（共31个）				
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile	let	yield		

保留字在某种意义上是为将来的关键字而保留的单词，因此保留字也不能被用作变量名或函数名。

注意：如果将保留字用作变量名或函数名，那么除非将来的浏览器实现了该保留字，否则很可能收不到任何错误消息。当浏览器将其实现后，该单词将被看做关键字，如此将出现关键字错误。

变量的使用

- 变量可以先声明再赋值
  - var a；
  - a = 1；
- 可以对变量中的值进行修改

  获取变量的值
  - var a = "1234"；
  - var b = a;

  重新设置变量的值

  var age = 1；

  age = 2；

JavaScript数据类型

- ECMAScript中有5种基本数据类型：
- Undefined
- Null
- Boolean
- Number
- String
- 还有一种复杂的数据类型——Object，Object本质上是由一组无序的名值对组成的。

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（未赋值），这个变量的值就是undefined。
- var message;
- alert(message == undefined); //true

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。
- null值表示一个变量以后可能会指向某个对象，但目前什么都没有指向
- var car = null;
- alert(typeof car); // “object”
- 从逻辑角度来看，null值表示一个空对象指针，而这也正是使用typeof操作符检测null值时会返回“object”的原因。

- alert(null == undefined); //true

Number类型

- JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：
- var x1=34.00; //使用小数点来写
- var x2=34;　　//不使用小数点来写
- NaN(Not a Number)是一个特殊的数值，这个数值用于表示一个本来要返回数值的操作数未返回数字的情况。

- isNaN() 函数

  这个函数接受一个参数，该参数可以是任何类型，而函数会帮我们确定这个参数是否“不是数字”。并返回一个Boolean值。

String类型

- 字符串是存储字符的变量，字符串可以是引号中的任意文本。
- 可以使用单引号或双引号
- var name = "zhangsan"; //申明并直接赋值

字面量	含义	字面量	含义
\n	换行	\\	斜杠
\t	制表	\’	单引号
\b	空格	\”	双引号
\r	回车		

Boolean类型

- 布尔类型
  - 仅有两个值：true 和 false
  - 也代表 1 和 0
  - 实际运算中 true == 1，false == 0

Object 类型

ECMAScript 中的对象其实就是一组数据和功能的集合。对象可以通过执行new操作符后跟要创建的对象类型的名称来创建。而创建Object类型的实例并为其添加属性和(或)方法，就可以创建自定义对象：

```
var o = new Object();
o.name = "zhangsan";
o.age = 18;
```

扩展：基本包装类型

- 为了便于操作基本类型值，ECMAScript 还提供了 3 个特殊的引用类型：Boolean、Number 和String。基本包装类型具有与各自的基本类型相应的特殊行为。实际上，每当读取一个基本类型值得时候，后台就会创建一个对应的基本包装类型的对象，从而让我们能够调用一些方法来操作这些数据。

JavaScript运算符

- 算数运算 + - * / % ++ --
- 关系运算> < >= <= == === != !==
- 逻辑运算 && || !
- 赋值运算 = += -= *= /= %=
- 字符链接 +
- 条件（三目）运算 ? :

算数运算符

运算符	名称	表达式	示例	运算结果
+	加法运算符	5+5	var a=5+5;	10
-	减法运算符	5-5	var a=5-5;	0
*	乘法运算符	5*5	var a=5*5;	25
/	除法运算符	5/5	var a=5/5;	1
%	取余运算符	8%5	var a=8%5;	3

一元运算符

++

--

```
var num = 10;
num++;  //相当于num = num + 1
num--;  //相当于num = num - 1
```

关系运算

- 关系运算用于判断数据之间的大小关系
- 关系表达式的值为boolean类型（true或false）

操作符	名称	表达式	返回结果
>	大于	10>20	false
<	小于	5<6	true
>=	大于等于	3>=3	true
<=	小于等于	5<=8	true

相等操作符

确定两个变量是否相等是编程中的一个非常重要的操作。在比较字符串、数值和布尔值的相等性时，问题还是比较简单。但在涉及到对象的比较时，问题就变得复杂了。最早的ECMAScript中的相等和不等操作符会在执行比较之前，现将对象转换成相似的类型。后来，有人提出这种类型转换道理是否合理的质疑。最后，ECMAScript的解决方案就是提供两组操作符

相等和不相等 —— 先转换再比较

全等和不全等 —— 仅比较而不转换

1.相等和不相等

- ECMAScript中的相等操作符由两个等号(==)表示，如果两个操作数相等，则返回true。
- 而不相等操作符由叹号后跟等于号(!=)表示，如果两个操作数不相等，则返回true。
- 这两个操作符都会先转换操作数(通常称为强行转型)，然后再比较它们的相等性。

在转换不同的数据类型时，相等和不相等操作符遵循下列基本规则：

- 如果有一个操作数是布尔值，则在比较相等性之前将其转换为数值

  false转换为0，而true转换为1；

- 如果一个操作数是字符串，另外一个操作数是数值，在比较相等性之前

  先将字符串转换为数值；

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		

2.全等和不全等

- 除了在比较之前不转换操作数之外，全等和不全等操作符与相等和不相
- 等操作符没有什么区别。全等操作符由3个等于号(===)表示，它只在两
- 个操作数未经转换就相等的情况下返回true。
- var b1 = (“55” == 55); //true,应为转换后相等
- var b2 = (“55” === 55); //false,应为不同的数据类型不相等
- 全不等操作符由一个叹号后跟两个等号(!==)表示，它在两个操作数未经
- 转换就不相等的情况下返回true。
- var b1 = (“55” != 55)//false,转换后相等
- var b2 = (“55” !== 55)//true,应为不同数据类型不相等

布尔操作符

1.逻辑非

逻辑非操作符由一个叹号(!)表示，可以应用于ECMAScript中的任何值。无论这个值是什么数据类型，这个操作符都会返回一个布尔值。逻辑非操作符首先会将它的操作数据转换为一个布尔值，然后再对其求反。也就是说，逻辑非操作符遵循下列规则：

操作数	返回值
对象	false
空字符串	true
非空字符串	false
数值0	true
任意非0数值	false
null	true
NaN	true
undefined	true

2.逻辑与

逻辑与操作符由两个和号(&&)表示，有两个操作数。

var t = true && false; //false

一个个操作数	第二个操作数	结果
true	true	true
true	false	false
false	true	false
false	false	false

3.逻辑或

逻辑或操作符由两个竖线符号(||)表示，有两个操作数。

var t = true || false; //true

一个个操作数	第二个操作数	结果
true	true	true
true	false	true
false	true	true
false	false	false

扩展：短路逻辑

- **短路求值**（Short-circuit evaluation，又称**最小化求值**），是一种逻辑运算符的求值策略。只有当第一个运算数的值无法确定逻辑运算的结果时，才对第二个运算数进行求值。例如，当AND的第一个运算数的值为false时，其结果必定为false；当OR的第一个运算数为true时，最后结果必定为true，在这种情况下，就不需要知道第二个运算数的具体值。

- 在JavaScript中
  - AND的一个运算数的值为true时,返回第二个运算数
  - OR的第一个运算数的值为true时,返回第一个运算数

赋值运算

- 使用 “=” 进行赋值运算
- var add = 100;
- 赋值表达式

  += -= *= /= %=

字符链接运算

- 使用 “+” 进行字符串链接
  - 作用于两个数字时，表示算数运算
  - 作用与字符串时，表示字符串拼接运算

条件（三目）运算

- 三目运算符（？：）--需要对三个表达式进行运算
  - 表达式1？表达式2：表达式3
  - 其中表达式1的值是boolean类型，表达式的含义：
    若表达式1为true，则整个表达式的值为表达式2的值；
    若表达式1为false，则整个表达式的值为表达式3的值；

```
var age = 20；
var msg = age > 18？"成年人"："未成年";
```

数据类型转换

- 隐式转换
  - 变量在声明时不需要指定数据类型
  - 变量由赋值操作确定数据类型
  - 不同类型数据在计算过程中会自动进行转换

数据类型转换函数

- Number()
  - 把一个string解析为number
  - 如果包含非法字符，则返回NaN
- parseFloat()
  - 解析出一个string或number的浮点数部分
  - 如果没有可转换的部分，则返回NaN
- parseInt()
  - 解析出一个string或number的整数部分
  - 如果没有可转换的部分，则返回NaN
- toString()
  - 转换成字符串，所有数据类型均可转换为string类型
- Boolean()
  - 将操作数转换成布尔值

THANK YOU

千锋教育
QIANFENG EDUCATION

做真实的自己，用良心做教育