

JavaScript基础

- JS 简介 JS 概要
- JS 基本语法 JS 基本文法
- JQuery 基本语法 JQuery 基本文法
- Ajax 简介 Ajax 概要



1

JS 简介

[JS 概要](#)

2

JS 基本语法

[JS 基本文法](#)

3

JQuery 基本语法

[JQuery 基本文法](#)

4

Ajax 简介

[Ajax 概要](#)

JavaScript 简介

JavaScript 概要

- JavaScript 是互联网上最流行的脚本语言[スクリプト言語]，这门语言可用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。
- 脚本语言[スクリプト言語]：又被称为扩建的语言，或者动态语言，是一种编程语言，用来控制软件应用程序，脚本通常以文本（如ASCII）保存，只在被调用时进行解释[インタープリタ]或编译[コンパイル]。



为什么要学习JavaScript?

- **JavaScript**是所有Web开发人员必须学习的三种语言之一：
 - 1. [HTML](#)定义网页内容
 - 2. [CSS](#)指定网页的布局
 - 3. 使用JavaScript编写网页行为
- **JavaScript**不只用在做网页。许多桌面和服务器程序都使用JavaScript。[Node.js](#)是最著名的例子。一些数据库[データベース]，例如MongoDB和CouchDB，也是用JavaScript编写的。



大多数人的疑惑

- JavaScript和[Java](#)在概念和设计上都是完全不同的语言。
- JavaScript由Brendan Eich于1995年发明，并于1997年成为ECMA标准。ECMA-262是该标准的正式名称。ECMAScript是该语言的正式名称。

Java is to Javascript what Car is to Carpet.

"The language's name is the result of a co-marketing deal between Netscape and Sun, in exchange for Netscape bundling Sun's Java runtime with their then-dominant browser."

JS在网页中可以做什么

- JS可以设计用户与网页的“**互动**”，例如：
 - 点击按钮(<button></button>)后触发事件
 - 可以更改HTML内容，属性或CSS
 - 可以控制显示/隐藏HTML元素
 -



JS代码放在什么位置

- 在HTML中，JavaScript代码插入在<script></script>标签之间。这个标签可以在<head>，<body>里面。

```
<script>
document.getElementById("demo").innerHTML = "My First
JavaScript";
</script>
```

- 脚本也可以放在**外部文件**中：

外部文件：myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
```

- 当在许多不同的网页中使用相同的代码时，推荐使用外部脚本。
- JavaScript文件的文件扩展名为.js。
- 要使用外部脚本，请将脚本文件的名称放在<script>的src（源）属性中：

```
<script src="myScript.js"></script>
```

JS代码放在外部的优势

- 将脚本放在外部文件中具有以下一些优点：
 - HTML和代码分开
 - 使HTML和JavaScript易于阅读和维护
 - 缓存的JavaScript文件可以加快页面加载速度
- 要将多个脚本文件添加到一页中，请使用多个脚本标签：

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

- 调用网络上的脚本：

```
<script src="https://www.w3schools.com/js/myScript1.js">
</script>
```

- 调用本地的脚本：

```
<script src="myScript1.js"></script>
```

JavaScript参考

- 该参考包含所有属性，方法[メソッド]和事件[イベント]的示例，并根据最新的Web标准不断进行更新。
- [完整的JavaScript参考](#)

w3schools.com

Q & A

You Have Questions
We Have Answers



1 JS 简介
[JS 概要](#)

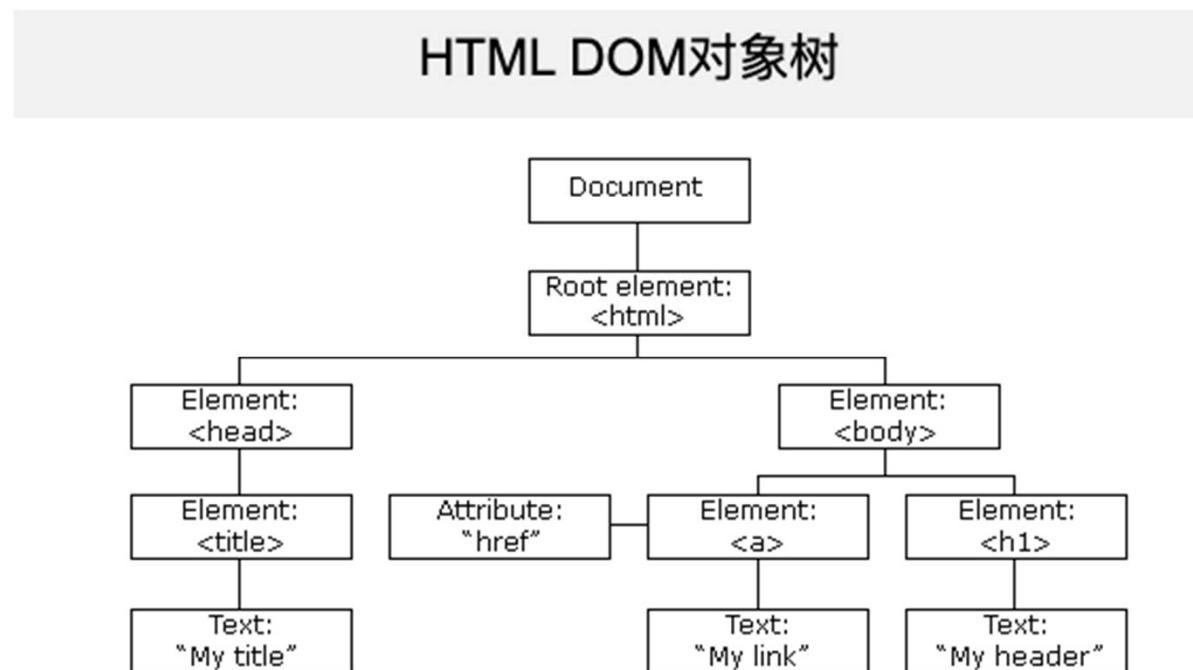
2 JS 基本语法
JS 基本文法

3 JQuery 基本语法
[JQuery 基本文法](#)

4 Ajax 简介
[Ajax 概要](#)

JavaScript HTML DOM (文档对象模型)

- 当网页加载时，浏览器会创建一个Document Object Model。
- 使用HTML DOM，JavaScript可以访问和更改HTML文档的所有元素。



- 简言之，HTML DOM是如何获取，更改，添加或删除HTML元素的标准。

JavaScript- HTML DOM方法[メソッド]

- DOM编程接口：
 - 可以使用JavaScript（和其他编程语言）访问HTML DOM。
 - 在DOM中，所有HTML元素都定义为对象[オブジェクト]（objects）。
 - 编程接口是每个对象的属性和方法[メソッド]。
 - 属性是你可以获取或设置的值（如更改HTML元素的内容）。
 - 方法是你可以做的动作（如添加或删除HTML元素）。
- 以下示例使用id="demo"更改元素

的内容（innerHTML）：

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

getElementById是方法，
innerHTML是属性。

尝试代码：dom_method.html

JavaScript HTML DOM document

- HTML DOM document 对象[オブジェクト]是网页中所有其他对象的所有者。
- document 对象代表整个网页。
- 如果要访问HTML页面中的任何元素，始终从访问document对象开始。
- 下面是三个document对象查找HTML元素的方法[メソッド]：

Method	Description
<code>document.getElementById(id)</code>	Find an element by element id
<code>document.getElementsByTagName(name)</code>	Find elements by tag name
<code>document.getElementsByClassName(name)</code>	Find elements by class name

- document还有很多其他的属性跟方法[メソッド]：

https://www.w3schools.com/js/js_htmldom_document.asp

JavaScript HTML DOM--更改HTML

- HTML DOM允许JavaScript更改HTML元素的内容。
- 在JavaScript中，`document.write()`可用于直接写入HTML输出流。但它会覆盖整个html，请谨慎使用。
- 修改HTML元素内容的最简单方法是使用`innerHTML`属性：

```
document.getElementById(id).innerHTML = new HTML
```

- 要更改HTML属性的值，请使用以下语法：

```
document.getElementById(id).attribute = new value
```

尝试代码：[dom_modify.html](#)

JavaScript HTML DOM事件[イベント]

- HTML DOM允许JavaScript对HTML事件[イベント]做出反应。
- 事件[イベント]发生时可以执行JavaScript，例如用户单击HTML元素的时候。
- HTML事件[イベント]对象[オブジェクト]的示例：

属性	说明	属性	说明
onabort	图像的加载被中断。	onmousedown	鼠标按钮被按下。
onblur	元素失去焦点。	onmousemove	鼠标被移动。
onchange	域的内容被改变。	onmouseout	鼠标从某元素移开。
onclick	当用户点击某个对象时调用的事件句柄。	onmouseover	鼠标移到某元素之上。
ondblclick	当用户双击某个对象时调用的事件句柄。	onmouseup	鼠标按键被松开。
onerror	在加载文档或图像时发生错误。	onreset	重置按钮被点击。
onfocus	元素获得焦点。	onresize	窗口或框架被重新调整大小。
onkeydown	某个键盘按键被按下。	onselect	文本被选中。
onkeypress	某个键盘按键被按下并松开。	onsubmit	确认按钮被点击。
onkeyup	某个键盘按键被松开。	onunload	用户退出页面。
onload	页面或一幅图像完成加载。		

- 完整的HTML DOM事件[イベント]参考：

https://www.w3schools.com/jsref/dom_obj_event.asp

使用HTML DOM 事件[イベント]

- HTML DOM 允许 JavaScript 将事件[イベント]添加给 HTML 元素。
- 将 onclick 事件[イベント]添加给按钮元素：

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

尝试代码：dom_events.html

JavaScript HTML DOM 事件[イベント]监听器

- addEventListener()方法[メソッド]将事件[イベント]处理程序附加到指定的元素。并且不会覆盖现有的事件[イベント]处理程序。
- 可以将多个事件处理程序添加到一个元素。
- 可以使用removeEventListener()方法[メソッド]删除事件监听器。
- 句法：

```
element.addEventListener(event, function, useCapture);
```

- event: 事件[イベント]的类型（例如“click”或“mousedown”或任何其他[HTML DOM Event](#)。）
- function: 事件发生时我们要调用的函数。
- useCapture: 是一个布尔值，指定是使用事件冒泡还是使用事件捕获。此参数[引数]是可加可不加的。
- 尝试代码：[dom_eventListener.html](#)

Q & A

You Have Questions
We Have Answers

JavaScript 显示运行结果

- JavaScript可以通过不同的方式“显示”数据：
 - 使用innerHTML写入HTML元素。
 - 使用document.write()写入HTML输出。
 - 使用window.alert()写入警报框。
 - 使用console.log()写入浏览器控制台。

- 尝试代码：firstJS.html
- 用浏览器打开之后，打开浏览器控制台console
- Mac打开控制台快捷键：cmd+option+j
- Win：win +shift +j

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript 语句

- JavaScript语句由以下组成：
 - 值，运算符[演算子]，表达式，关键字和注释[コメント]。
- 分号分隔JavaScript语句。在每个可执行语句的末尾添加分号：

```
var a, b, c;      // Declare 3 variables
a = 5;            // Assign the value 5 to a
b = 6;            // Assign the value 6 to b
c = a + b;        // Assign the sum of a and b to c
```

- 当用分号分隔时，允许在一行上使用多个语句：

```
a = 5; b = 6; c = a + b;
```

- 可以在脚本中添加空格以使其更具可读性。一个好的做法是在运算符 (= +-* /) 周围放置空格。

JavaScript代码块

- JavaScript语句可以在大括号{}内的代码块中分组在一起。
- 代码块的目的是定义要一起执行的语句。
- 在JavaScript函数中，可以以代码块分组：

```
function myFunction() {  
    document.getElementById("demo1").innerHTML = "Hello Dolly!";  
    document.getElementById("demo2").innerHTML = "How are you?";  
}
```

JavaScript关键字

- JavaScript语句通常以关键字开头，以标识要执行的JavaScript操作。
- 看[JavaScript关键字](#)的完整列表。
- JavaScript关键字是保留字[\[予約語\]](#)。保留字[\[予約語\]](#)不能用作变量的名称。

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

JavaScript 注释[コメント]

- JavaScript注释[コメント]可用于解释JavaScript代码并使其更具可读性
 - 测试代码时，JavaScript注释[コメント]还可用于避免执行。
 - 单行注释单行注释以//开头。
 - 本示例在每个代码行之前使用单行注释：

```
// Change heading:  
document.getElementById("myH").innerHTML = "My First Page";  
  
// Change paragraph:  
document.getElementById("myP").innerHTML = "My first paragraph.;"
```

- 本示例在每行末尾使用一行注释来解释代码：

```
var x = 5;      // Declare x, give it the value of 5  
var y = x + 2; // Declare y, give it the value of x + 2
```

JavaScript 注释[コメント]--多行注释

- 多行注释以/*开头和*/结尾。
- /*和*/之间的任何文本都将被JavaScript忽略。
- 本示例使用多行注释（注释块）来解释代码：

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

- 最常见的是使用单行注释。多行注释通常用于正式文档。

JavaScript 变量[变数]

- JavaScript变量是用于存储数值的容器。
- 在这个例子中，x, y, 和z, 是变量：

- x存储5 `var x = 5;`
- y存储6 `var y = 6;`
- z存储11 `var z = x + y;`

- 编程中，就像在代数中一样，使用变量（例如x）来保存值。在表达式中使用变量（ $z = x + y$ ）。
- 那么z就会存储11

JavaScript 变量--标识符

- 所有JavaScript 变量必须 使用唯一的名称标识。
- 这些唯一的名称称为标识符。
- 标识符可以是短名称（如x和y），也可以是更具描述性的名称（age, sum, totalVolume）。
- 起变量名称（唯一标识符）的一般规则是：
 - 名称可以包含字母，数字，下划线和\$符号。
 - 名称必须以字母开头
 - 名称也可以以\$和_开头（但在本课件中不用它们）
 - 名称区分大小写（y和Y是不同的变量）
 - 保留字^[子約語]（如JavaScript关键字）不能用作名称

JavaScript 变量--JavaScript 保留字[[予約語](#)]

- 在JavaScript中，不能将这些保留字[[予約語](#)]用作变量，标签或函数名：

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

JavaScript 运算符—赋值运算符 [代入演算子]

- 在JavaScript中，等号（=）是“赋值”运算符 [演算子]，而不是“等于”运算符 [演算子]。
- 与代数不同。比如以下内容在代数中没有意义：

```
x = x + 5
```

- 但是，在JavaScript中，这很有意义：它将 $x + 5$ 的值赋给 x 。
- （它计算 $x + 5$ 的值并将结果存入 x 。即 x 的值增加5。）
- 在JavaScript中，“等于”运算符的编写方式类似于 $==$ 。

JavaScript 变量--数据类型[データタイプ]

- JavaScript变量可以包含数字（如100）和文本值（如“John Doe”）。
- 在编程中，文本值称为文本字符串。
- JavaScript可以处理多种类型的数据，目前仅考虑数字和字符串。
- 字符串用双引号或单引号引起，数字不带引号。
- 如果将数字加引号，它将被视为文本字符串。

```
var pi = 3.14;  
var person = "John Doe";  
var answer = 'Yes I am!';
```

声明[宣言] JavaScript变量

- 在JavaScript中创建变量称为“声明[宣言]”变量。
- 使用var关键字声明一个JavaScript变量：

```
var carName;
```

- 声明后，变量没有值（从技术上讲，其值为undefined）。
- 使用等号指定值给变量：

```
carName = "Volvo";
```

- 还可以在声明变量时为其赋值：

```
var carName = "Volvo";
```

- 在脚本开头声明所有变量是一种好的编程习惯。

声明[宣言] JavaScript变量

- 如果重新声明一个JavaScript变量，它将不会丢失其值。
- carName执行以下语句后，变量将仍然具有值“Volvo”：

```
var carName = "Volvo";
var carName;
```

JavaScript 运算符--算术运算符 [算術演算子]

- 算术运算符用于对数字执行算术运算：

尝试代码：[js_operators.html](#)

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript 运算符—赋值运算符 [代入演算子]

- 赋值运算符 [代入演算子] 将值分配给 JavaScript 变量

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>
**=	<code>x **= y</code>	<code>x = x ** y</code>

JavaScript 运算符—字符串运算符 [文字列演算子]

- +运算符还可以用于添加（连接）字符串。当在字符串上使用时，+运算符称为串联运算符。

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

- txt3的结果是：“John Doe”
- 相加一个数字和一个字符串将返回一个字符串：

```
var x = 5 + 5;
var y = "5" + 5;
var z = "Hello" + 5;
```

- x, y, z的结果分别是：10, “55”, “Hello5”
- JavaScript从左到右评估表达式。不同的序列可以产生不同的结果：

```
var x = 16 + 4 + "Volvo";
```

- 结果：20Volvo

```
var x = "Volvo" + 16 + 4;
```

- Volvo164

JavaScript 运算符—比较运算符 [関係演算子]

Operator	Description
<code>==</code>	equal to
<code>===</code>	equal value and equal type
<code>!=</code>	not equal
<code>!==</code>	not equal value or not equal type
<code>></code>	greater than
<code><</code>	less than
<code>>=</code>	greater than or equal to
<code><=</code>	less than or equal to
<code>?</code>	ternary operator

JavaScript 运算符—逻辑运算符 [論理演算子]

Operator	Description
&&	logical and
	logical or
!	logical not

JavaScript 运算符—类型运算符

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JavaScript 运算符—位运算符 [ビット演算子]

- 位运算符处理32位数字。
- 该操作中的任何数字操作数都将转换为32位数字。结果转换回JavaScript编号。

Operator	Description	Example	Same as	Result	Decimal
&	AND	$5 \& 1$	$0101 \& 0001$	0001	1
	OR	$5 1$	$0101 0001$	0101	5
~	NOT	~ 5	~ 0101	1010	10
^	XOR	$5 ^ 1$	$0101 ^ 0001$	0100	4
<<	Zero fill left shift	$5 << 1$	$0101 << 1$	1010	10
>>	Signed right shift	$5 >> 1$	$0101 >> 1$	0010	2
>>>	Zero fill right shift	$5 >>> 1$	$0101 >>> 1$	0010	2

JavaScript 运算符优先级 [演算子の優先順位]



Value	Operator	Description	Example
20	()	Expression grouping	(3 + 4)
19	.	Member	person.name
19	[]	Member	person["name"]
19	()	Function call	myFunction()
19	new	Create	new Date()
17	++	Postfix Increment	i++
17	--	Postfix Decrement	i--
16	++	Prefix Increment	++i
16	--	Prefix Decrement	--i
16	!	Logical not	!(x==y)
16	typeof	Type	typeof x
15	**	Exponentiation (ES2016)	10 ** 2

14	*	Multiplication	10 * 5
14	/	Division	10 / 5
14	%	Division Remainder	10 % 5
13	+	Addition	10 + 5
13	-	Subtraction	10 - 5
12	<<	Shift left	x << 2
12	>>	Shift right	x >> 2
12	>>>	Shift right (unsigned)	x >>> 2
11	<	Less than	x < y
11	<=	Less than or equal	x <= y
11	>	Greater than	x > y
11	>=	Greater than or equal	x >= y
11	in	Property in Object	"PI" in Math
11	instanceof	Instance of Object	instanceof Array
10	==	Equal	x == y
10	==	Strict equal	x === y

JavaScript 运算符优先级

10	<code>!=</code>	Unequal	<code>x != y</code>
10	<code>!==</code>	Strict unequal	<code>x !== y</code>
9	<code>&</code>	Bitwise AND	<code>x & y</code>
8	<code>^</code>	Bitwise XOR	<code>x ^ y</code>
7	<code> </code>	Bitwise OR	<code>x y</code>
6	<code>&&</code>	Logical AND	<code>x && y</code>
5	<code> </code>	Logical OR	<code>x y</code>
4	<code>? :</code>	Condition	<code>? "Yes" : "No"</code>
3	<code>+=</code>	Assignment	<code>x += y</code>
3	<code>/=</code>	Assignment	<code>x /= y</code>
3	<code>-=</code>	Assignment	<code>x -= y</code>
3	<code>*=</code>	Assignment	<code>x *= y</code>
3	<code>%=</code>	Assignment	<code>x %= y</code>
3	<code><<=</code>	Assignment	<code>x <<= y</code>
3	<code>>>=</code>	Assignment	<code>x >>= y</code>
3	<code>>>>=</code>	Assignment	<code>x >>>= y</code>
3	<code>&=</code>	Assignment	<code>x &= y</code>
3	<code>^=</code>	Assignment	<code>x ^= y</code>
3	<code> =</code>	Assignment	<code>x = y</code>

2	<code>yield</code>	Pause Function	<code>yield x</code>
1	,	Comma	<code>5 , 6</code>

JavaScript 数据类型[データタイプ]

- JavaScript 变量可以包含许多数据类型：数字，字符串[文字列]，对象[オブジェクト]等：

```
var length = 16;                      // Number
var lastName = "Johnson";              // String
var x = {firstName:"John", lastName:"Doe"}; // Object
```

- 在编程中，数据类型是一个重要的概念。
- 为了能够对变量进行操作，了解有关类型的知识很重要。
- JavaScript 具有动态类型。这意味着可以使用同一变量来保存不同的数据类型：

```
var x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

JavaScript 数据类型[データタイプ]—字符串[文字列]

- 字符串（或文本字符串）是一系列字符，例如“John Doe”。
- 字符串用引号引起来。可以使用单引号或双引号：

```
var carName1 = "Volvo XC60";    // Using double quotes
var carName2 = 'Volvo XC60';    // Using single quotes
```

JavaScript 数据类型[データタイプ]—数字

- JavaScript 只有一种数字类型。
- 数字可以带或不带小数：

```
var x1 = 34.00;      // Written with decimals
var x2 = 34;         // Written without decimals
```

- 可以使用科学（指数）表示法：

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

JavaScript 数据类型[データタイプ]—布尔值[ブーリアン型]

- 布尔值只有两个值： true或false。

```
var x = 5;  
var y = 5;  
var z = 6;  
(x == y)      // Returns true  
(x == z)      // Returns false
```

- 在条件测试中经常使用布尔值。

JavaScript 数据类型[データタイプ]—数组[配列]

- JavaScript数组用方括号括起来。
- 数组元素用逗号分隔。
- 以下代码声明（创建）一个名为cars的数组，其中包含三个元素（汽车名称）：

```
var cars = ["Saab", "Volvo", "BMW"];
```

- 数组索引从0开始，这意味着第一项为[0]，第二项为[1]，依此类推。

JavaScript 数据类型[データタイプ]—对象[オブジェクト]

- JavaScript 对象[オブジェクト]用花括号括起来{}。
- 对象[オブジェクト]属性写为name: value对，以逗号分隔。

```
var person = {firstName:"John", lastName:"Doe", age:50,  
eyeColor:"blue"};
```

- 上例中的对象[オブジェクト] (person) 具有4个属性：firstName, lastName, age和eyeColor。

运算符[演算子] typeof

- 你可以使用JavaScript typeof运算符查看JavaScript变量的类型
- typeof返回一个变量或一个表达式的类型：

```
typeof ""           // Returns "string"
typeof "John"       // Returns "string"
typeof "John Doe"  // Returns "string"

typeof 0            // Returns "number"
typeof 314          // Returns "number"
typeof 3.14         // Returns "number"
typeof (3)          // Returns "number"
typeof (3 + 4)      // Returns "number"
```

JavaScript 数据类型[データタイプ]--Null

- 在JavaScript中，null表示“nothing”。数据类型null是一个对象[オブジェクト]。

```
var person = {firstName:"John", lastName:"Doe", age:50,  
eyeColor:"blue"};  
person = null; // Now value is null, but type is still an object
```

- undefined和null之间的区别：

```
typeof undefined          // undefined  
typeof null              // object  
  
null === undefined       // false  
null == undefined        // true
```

JavaScript 数据类型[データタイプ]-- Primitive Data (原始数据)

- 原始数据值是一个简单的数据值，没有其他属性和方法[メソッド]。
- `typeof`可以返回这些原始类型中的一种：

```
typeof "John"           // Returns "string"
typeof 3.14             // Returns "number"
typeof true              // Returns "boolean"
typeof false             // Returns "boolean"
typeof x                 // Returns "undefined" (if x has no
value)
```

JavaScript 数据类型[データタイプ]-- Complex Data (复杂数据)

- `typeof`可以返回两个复杂类型中的一种：
 - `function`
 - `object`
- `typeof`不会为函数返回“对象[オブジェクト]”。

```
typeof {name:'John', age:34} // Returns "object"
typeof [1,2,3,4]           // Returns "object" (not "array", see
note below)
typeof null                // Returns "object"
typeof function myFunc() {} // Returns "function"
```

Q & A

You Have Questions
We Have Answers

JavaScript 语法—条件语句 [条件分歧]

- 通常在编写代码时，你会希望针对不同的情况执行不同的操作。
- JavaScript具有以下条件语句：
 - 如果一个指定的条件是true，使用if指定的代码块将被执行
 - 如果上面的条件为假，使用else指定的代码块将被执行
 - 如果第一个条件为假，使用else if指定一个新的条件进行筛选

The screenshot shows a terminal window with the following content:

```
JS test.js  ×  
JS test.js > ...  
1 var num = 15;  
2 ↴if(num > 10){  
3   console.log("嘻嘻嘻");  
4 }  
5  
6  
问题  输出  调试控制台  终端  
C:\Users\lu\Desktop\DemoXKD>node test.js  
嘻嘻嘻  输出
```

The terminal output "嘻嘻嘻" is highlighted with a red box.

条件语句—if语句

- 如果条件为真，则使if语句指定要执行的JavaScript代码块。

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

尝试代码：[if.html](#)

条件语句—else语句

- 如果条件为假，则使用else语句指定要执行的代码块。

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

尝试代码：else.html

条件语句—else if语句

- 如果第一个条件为false，则使用该语句指定新条件。

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false  
and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false  
and condition2 is false  
}
```

- 尝试代码：[else_if.html](#)

JavaScript 比较运算符 [関係演算子]

- 在逻辑语句中使用比较运算符来确定变量或值之间的相等性或差异性。
- 现有 $x = 5$, 则右表解释了比较运算符:

- 可以在条件语句中使用比较运算符来比较值并根据结果采取措施:

```
if (age < 18) text = "Too young";
```

将字符串与数字进行比较时, 在进行比较时JavaScript会将字符串转换为数字。空字符串将转换为0。非数字字符串将转换为NaN, 始终为false。

$==$	equal to	$x == 8$	false
		$x == 5$	true
		$x == "5"$	true
$====$	equal value and equal type	$x === 5$	true
		$x === "5"$	false
$!=$	not equal	$x != 8$	true
$!==$	not equal value or not equal type	$x !== 5$	false
		$x !== "5"$	true
		$x !== 8$	true
$>$	greater than	$x > 8$	false
$<$	less than	$x < 8$	true
$>=$	greater than or equal to	$x >= 8$	false
$<=$	less than or equal to	$x <= 8$	true

JavaScript 逻辑运算符 [論理演算子]

- 逻辑运算符用于确定变量或值之间的逻辑关系。
- 鉴于 $x = 6$ 和 $y = 3$ ，下表解释了逻辑运算符：

Operator	Description	Example
<code>&&</code>	and	$(x < 10 \&\& y > 1)$ is true
<code> </code>	or	$(x == 5 y == 5)$ is false
<code>!</code>	not	$!(x == y)$ is true

JavaScript 条件（三元）运算符 [三項演算子]

- JavaScript还包含一个条件运算符，该运算符可根据某些条件将值分配给变量。
- 句法： variablename = (condition) ? value1:value2
- 如果age的值小于18，则voteable的值将为“Too young”，voteable的值将为“Old enough”。

```
var voteable = (age < 18) ? "Too young": "Old enough";
```

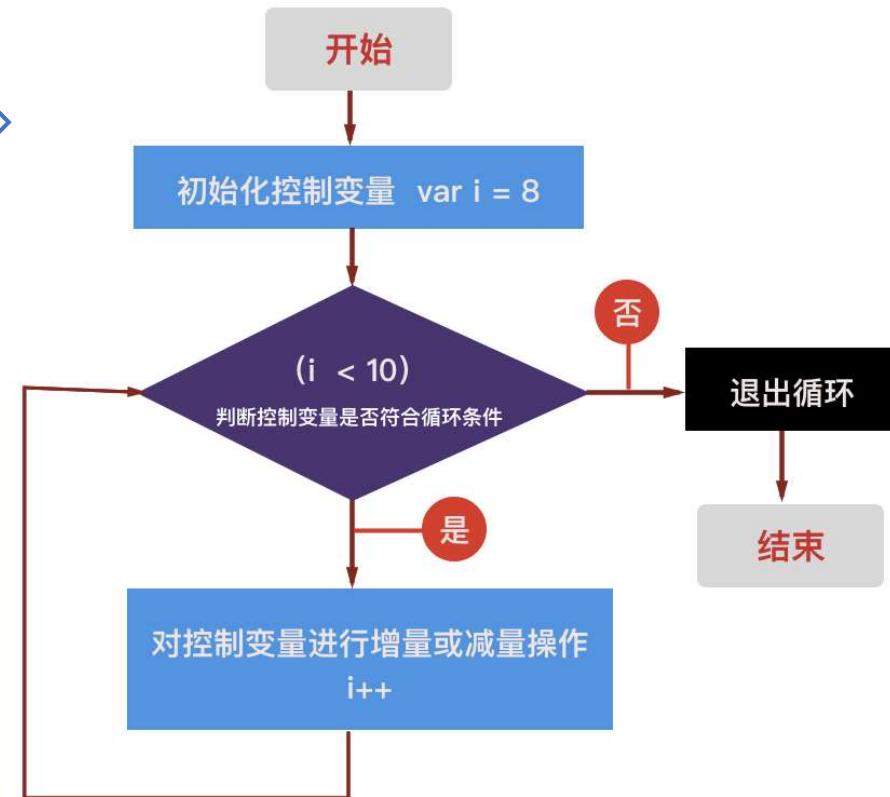
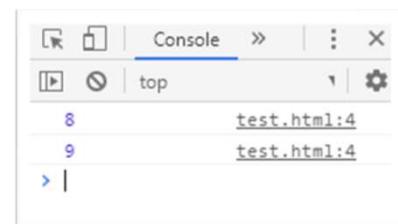
JavaScript 循环语句 [繰り返し文]

- 循环可以多次执行一个代码块。
- JavaScript 支持各种循环：
 - **for - 遍历代码块多次**
 - **for/in - 遍历对象的属性**
 - **for/of - 遍历可迭代对象的值**
 - **while - 在指定条件为真时，循环遍历代码块**
 - **do/while - 在指定条件为true时，也循环遍历代码块**

```

1 <script>
2 var i=8
3 for (; i<10; i++) {
4   console.log(i)
5 }
6 </script>

```



JavaScript 循环语句—for循环

- for循环的语法如下：

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

- statement1在执行代码块之前（一次）执行。
- statement2定义了执行代码块的条件。
- statement3在每次执行代码块后都会执行（每次）。

```
for (i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}
```

- 语句1在循环开始之前设置了一个变量（var i = 0）。
- 语句2定义了循环运行的条件（i必须小于5）。
- 每次执行循环中的代码块时，语句3都会增加一个值（i ++）。

尝试代码：for.html

JavaScript 循环语句--For / In循环

- JavaScript for/in语句遍历对象[オブジェクト]的属性：

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

JavaScript 循环语句--For / Of循环

- JavaScript for/of语句遍历可迭代对象[オブジェクト]的值
- for/of 可以遍历可迭代的数据结构，例如数组[配列]，字符串[文字列]，映射， NodeList等。
- for/of循环的语法如下：

```
for (variable of iterable) {  
    // code block to be executed  
}
```

- 遍历数组：

```
var cars = ['BMW', 'Volvo', 'Mini'];  
var x;  
  
for (x of cars) {  
    document.write(x + "<br >");  
}
```

JavaScript 循环语句--While循环

- 只要指定条件为真，while循环就可以执行代码块：

```
while (condition) {  
    // code block to be executed  
}
```

- 在以下示例中，只要变量（i）小于10，循环中的代码就会一次又一次地运行：

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```

- 如果忘记增加条件下使用的变量，循环将永远不会结束。这将使你的浏览器崩溃

尝试代码：[while.html](#)

JavaScript 循环语句—Do/While循环

- do/while循环是while循环的变体。在检查条件是否为真之前，此循环将执行一次代码块，然后只要条件为真，它将重复循环。

```
do {  
    // code block to be executed  
}  
while (condition);
```

- 下面的示例使用do/while循环。即使条件为假，循环也将至少执行一次，因为代码块是在条件测试之前执行的：

```
do {  
    text += "The number is " + i;  
    i++;  
}  
while (i < 10);
```

- 不要忘记增加条件中使用的变量，否则循环将永远不会结束！

JavaScript break和continue

- **break**会“跳出”全部循环， **continue**会“跳过”本次循环。
- **break**语句中断循环，并在循环后继续执行代码（如果有的话）：

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { break; }  
    text += "The number is " + i + "<br>";  
}
```

- 如果出现指定条件， **continue**语句将中断一次迭代（在循环中），并在循环中继续进行下一次迭代。

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { continue; }  
    text += "The number is " + i + "<br>";  
}
```

- 此示例会跳过数字3

Q & A

You Have Questions
We Have Answers

JavaScript 数组 [配列] Array

- 数组是一个特殊变量，一次可以容纳多个值。
- 如果你有一个项目列表（例如，汽车名称列表），则将汽车存储在单个变量中，像这样：

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

- 但是，如果你想遍历汽车并找到特定的汽车怎么办？或者如果不是3辆车，而是300辆车怎么办？
- 解决方案是数组！
- 数组可以用一个变量名称保存许多值，并且你可以通过引用索引号来访问这些值。

创建一个数组

- 语法: `var array_name = [item1, item2, ...];`
- 比如: `var cars = ["Saab", "Volvo", "BMW"];`
- 也可以使用JavaScript关键字new:

```
var cars = new Array("Saab", "Volvo", "BMW");
```

- 上面的两个示例完全相同。无需使用`new Array()`。
为了简单起见，提高可读性和执行速度，请使用第一个。

访问数组的元素

- 你可以通过引用索引号来访问数组元素。
- 此语句访问中的第一个元素的值cars：

```
var name = cars[0];
```

- 注意：数组索引从0开始的。 [0]是第一个元素。 [1]是第二个元素。
 - 更改数组元素： cars[0] = "Opel";
 - 也可以通过引用数组名称来访问整个数组
-
- 尝试代码：[array.html](#)

数组元素可以是对象[オブジェクト]

- 可以在同一Array中使用不同类型的变量。
- 可以在数组中包含对象[オブジェクト]、函数、甚至是数组：

```
myArray[0] = Date.now;  
myArray[1] = myFunction;  
myArray[2] = myCars;
```

数组的属性和方法[メソッド]

- JavaScript数组的真正优势在于内置的数组属性和方法[メソッド]：
 - 返回数组的长度：length属性（数组元素的数量）length属性始终比最高数组索引大1。
 - 向数组添加新元素：push()
 - 从数组中删除最后一个元素：pop()
 - 将数组转换为字符串：toString()
- 尝试代码：[array_attr.html](#)

JavaScript 函数

- JavaScript函数旨在执行特定任务的代码块。
- 当“某个东西”将其调用时，将执行JavaScript函数。

```
function myFunction(p1, p2) {  
    return p1 * p2;    // The function returns the product of p1  
and p2  
}
```

JavaScript函数语法

- JavaScript函数是用function关键字定义的，其后是名称，然后是括号 ()。
- 函数名称可以包含字母，数字，下划线和 \$ 符号（与变量相同的规则）。
- 括号中可能包含用逗号分隔的参数[引数]名称：
(parameter1, parameter2, ...)
- 该函数要执行的代码放在大括号内： {}

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

- 函数参数[引数]在函数定义的括号 () 中列出。
- 函数参数[引数]是函数被调用时收到的值。
- 在函数内部，参数[引数]表现为局部变量。

JavaScript函数语法

- 当“某物”调用该函数时，该函数内部的代码将执行：
 - 事件[イベント]发生时（如用户单击按钮时）
 - 从JavaScript代码调用时
 - 自动（自行调用）
- 当JavaScript到达一条return语句时，该函数将停止执行。
- 如果从语句调用了该函数，则JavaScript将在调用语句之后“返回”，以执行代码。
- 函数通常计算返回值[戻り値]。返回值[戻り値]被“返回”到“调用者”：

尝试代码：function.html

JavaScript 函数参数[引数]

- 函数参数[引数]是 传递给函数（并由函数接收）的实际值。
- 参数[引数]规则：
 - JavaScript 函数定义未指定参数[引数]的数据类型。
 - JavaScript 函数不对传递的参数[引数]执行类型检查。
 - JavaScript 函数不检查接收到的参数[引数]数量。

```
function functionName(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

JavaScript 闭包

- JavaScript 变量作用范围有本地范围或全局范围。
- 可以使用闭包将全局变量设为局部（私有）变量。
- function 可以访问函数内部定义的所有变量，如下所示：

```
function myFunction() {  
    var a = 4;  
    return a * a;  
}
```

- 但 function 也可以访问在函数外部定义的变量，如下所示：

```
var a = 4;  
function myFunction() {  
    return a * a;  
}
```

- 第一个示例的 a 是局部变量，第二个是全局变量，在网页中，全局变量属于 window 对象 [オブジェクト]。局部变量只能在定义它的函数内部使用。具有相同名称的全局变量和局部变量是不同的变量。

JavaScript HTML事件[イベント]

- HTML事件[イベント]可以是浏览器执行的操作，也可以是用户执行的操作，JavaScript可以对这些事件[イベント]“做出反应”。
- 以下是一些HTML事件[イベント]的示例：
 - HTML网页已完成加载
 - HTML输入字段已更改
 - 单击了HTML按钮
- HTML允许使用JavaScript代码将事件[イベント]处理程序属性添加到HTML元素。
- 在以下示例中，将onclick属性（带有代码）添加到<button>元素：

```
<button onclick="document.getElementById('demo').innerHTML =  
Date()">The time is?</button>
```

- 在上面的示例中，JavaScript代码使用id = “ demo”更改了元素的内容。
- 在下一个示例中，代码更改了自己元素的内容（使用this.innerHTML）：

```
<button onclick="this.innerHTML = Date()">The time is?  
</button>
```

常见的HTML事件 [イベント]

- 以下是一些常见的HTML事件 [イベント] 的列表：

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

- 尝试代码：events.html

JavaScript 可以做什么？

- 事件[イベント]处理程序可用于处理和验证用户输入，用户操作和浏览器操作：
 - 每次页面加载时应该做的事情
 - 关闭页面时应该做的事情
 - 用户单击按钮时应执行的操作
 - 用户输入数据时应验证的内容
 - more ...
- 可以使用许多不同的方法[メソッド]来使JavaScript处理事件[イベント]：
 - HTML事件属性可以直接执行JavaScript代码
 - HTML事件属性可以调用JavaScript函数
 - 可以将自己的事件处理函数分配给HTML元素
 - 可以防止事件被发送或处理
 - more ...

Q & A

You Have Questions
We Have Answers



1 JS 简介
[JS 概要](#)

2 JS 基本语法
[JS 基本文法](#)

3 **JQuery 基本语法**
[JQuery 基本文法](#)

4 Ajax 简介
[Ajax 概要](#)

什么是库[ライブラリー]

- 库：别人用这个语言写好的一些方便使用的方法[メソッド]、类等代码，后人想实现同样功能的时候就可以直接调用库里面的方法[メソッド]，不必重复造轮子。
- 典型的库：jQuery, java.util, C的stdlib, python的numpy, pandas等等。



TensorFlow



jQuery 简介

- jQuery是一个轻量级的“少写，多做”的JavaScript库。它大大简化了JavaScript编程。
- jQuery执行许多需要许多行JavaScript代码才能完成的常见任务，并将它们包装到可以用一行代码调用的方法[メソッド]中。
- Web上许多最大的公司都使用jQuery，例如：
 - 谷歌
 - 微软
 - IBM
 - 网飞

jQuery Get Started

- 下载jQuery
 - 有两种版本的jQuery可下载：
 - 生产版本-压缩版本，占用空间小适用于你的实时网站
 - 开发版本-用于测试和开发（未压缩，可读的代码）
 - 两种功能完全一样，这两个版本都可以从[jQuery.com](https://jquery.com)下载。
 - jQuery库就是一个JavaScript文件，可以使用HTML <script>标签对其进行引用（注意，该<script>标签应位于<head>中）：

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

- 提示：将下载的文件放置在与你希望使用的页面相同的目录中。

jQuery 语法

- 基本语法为: `$(selector).action()`
 - `$`是定义/访问jQuery的符号
 - `(selector)`用来“查询”HTML元素
 - `action()`用来对元素执行动作
- 例子:
 - `$(this).hide()` -隐藏当前元素。
 - `$("p").hide()` -隐藏所有`<p>`元素。
 - `$(".test").hide()` -隐藏所有带有`class = “ test”`的元素。 (.表示class)
 - `$("#test").hide()` -隐藏`id = “ test”`的元素。 (#表示id)

文件准备活动

- 为了防止文档完成加载（准备就绪）之前运行任何jQuery代码，要把所有jQuery方法[メソッド]都在文档就绪事件[イベント]内部。

```
$ (document) . ready (function () {  
    // jQuery methods go here...  
});  
  
$ (function () {  
    // jQuery methods go here...  
});
```

jQuery Selectors (选择器)

- jQuery选择器是jQuery库中最重要的部分之一。
 - jQuery选择器允许你选择和操作HTML元素。
 - jQuery选择器用于根据其name, id, class, type, 属性, 属性值等等来“查找”(或选择) HTML元素。它基于现有的[CSS选择器](#), 此外, 它还具有一些自定义选择器。
 - jQuery中的所有选择器都以美元符号和括号开头: \$ ()。
-
- 尝试代码: [jquery_selectors.html](#)

jQuery Selectors—元素选择器

- jQuery元素选择器根据元素名称选择元素。
- 如右所示，你可以选择页面上的所有

元素： \$("p")
- 例，当用户单击按钮时，所有

元素都将被隐藏：

```
$ (document) .ready(function () {  
    $("button") .click(function () {  
        $("p") .hide();  
    });  
});
```

jQuery Selectors—#id选择器

- #id选择器使用HTML标记的id属性来查找特定元素。
- id在页面中是唯一的，因此当你要查找单个唯一元素时，应该使用#id选择器。
- 要查找具有特定id的元素，请先写一个#，后跟HTML元素的id名称：\$("#test")
- 例，当用户单击按钮时，具有id =“ test”的元素将被隐藏：

```
$ (document) .ready(function () {  
    $("button") .click(function () {  
        $("#test") .hide();  
    });  
});
```

jQuery Selectors—.class选择器

- .class选择器查找具有特定class (类) 的元素。
- 要查找具有特定类的元素, 请写一个句点字符, 后跟类的名称:
 `$(".test")`
- 例, 当用户单击按钮时, 具有class =“ test”的元素将被隐藏:

```
$ (document).ready(function() {
    $("button").click(function() {
        $(".test").hide();
    });
});
```

- jQuery选择器的更多示例:

https://www.w3school.com.cn/jquery/jquery_ref_selectors.asp

jQuery 事件方法

- 在jQuery中，大多数DOM事件[イベント]都有一个等效的jQuery方法[メソッド]。
- 要将点击事件[イベント]分配给页面上的所有段落，你可以执行以下操作：

```
$("p").click();
```

- 下一步是定义事件[イベント]触发时应调用的方法[メソッド]。所以必须将一个函数传递给事件[イベント]：

```
$("p").click(function() {  
    // action goes here!!  
});
```

尝试代码：jquery_events.html

常用的jQuery事件[イベント]方法[メソッド]

- `$(document).ready ()`
 - 该`$(document).ready()`方法[メソッド]允许我们在文档完全加载后再执行功能。
- `click ()`
 - 当用户单击HTML元素时，执行动作。
- `on ()`
 - 该`on()`方法[メソッド]为所选元素附加一个或多个事件[イベント]处理程序
 -
- `dblclick ()`
 - 用户双击HTML元素时，执行动作

尝试代码：jquery_events.html

jQuery DOM操作

- 三种简单但很有用的进行DOM操作的jQuery方法[メソッド]：
 - text() - 设置或返回所选元素的文本内容
 - html() - 设置或返回所选元素的内容（包括HTML标签）
 - val() - 设置或返回表单字段的值
- jQuery attr()方法[メソッド]用于获取属性值：

```
$("button").click(function() {
    alert($("#w3s").attr("href"));
});
```

- 尝试代码：jquery_dom.html

jQuery- 设置内容和属性

- 可以用前面的text () , html () 和val () 方法[メソッド]设置内容:

```
$("#btn1").click(function() {
    $("#test1").text("Hello world!");
});
$("#btn2").click(function() {
    $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function() {
    $("#test3").val("Dolly Duck");
});
```

- jQuery attr()方法[メソッド]也用于设置/更改属性值:

```
$("button").click(function() {
    $("#w3s").attr({
        "href" : "https://www.w3schools.com/jquery/",
        "title" : "W3Schools jQuery Tutorial"
    });
});
```

尝试代码：jquery_dom.html

jQuery--添加，删除元素

- 用于添加新内容的四种jQuery方法[メソッド]：
 - append() - 在元素的末尾插入内容
 - prepend() - 在元素的开头插入内容
 - after() - 在元素之后插入内容
 - before() - 在元素之前插入内容
- 要删除元素和内容，主要有两种jQuery方法[メソッド]：
 - remove() - 删除选定的元素（及其子元素）
 - empty() - 从元素中删除子元素
- jQuery remove()方法[メソッド]还接受一个参数[引数]，该参数[引数]过滤要删除的元素。
- 该参数[引数]可以是任何jQuery选择器语法。
- 以下示例删除带有class="test"或class="demo"的所有<p>元素：

```
$("p").remove(".test, .demo");
```

- 尝试代码：jquery_addDelete.html

jQuery 效果方法

- [效果方法参考](#)
- 尝试代码：[jquery_effects.html](#)

Q & A

You Have Questions
We Have Answers



1 JS 简介
[JS 概要](#)

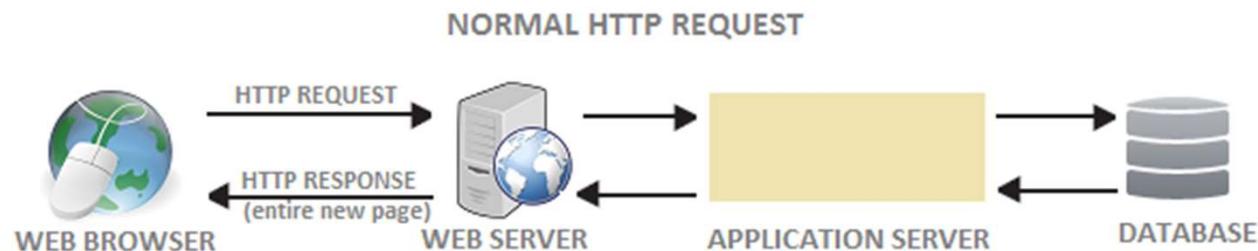
2 JS 基本语法
[JS 基本文法](#)

3 JQuery 基本语法
[JQuery 基本文法](#)

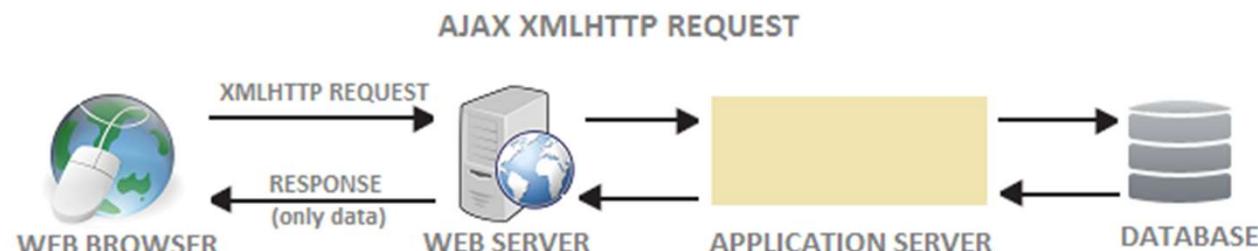
4 Ajax 简介
[Ajax 概要](#)

AJAX简介

- AJAX = Asynchronous JavaScript and XML. (异步JavaScript和XML)
- AJAX是与服务器交换数据并更新部分网页的技术--无需重新加载整个页面。



@DotNetCurry.com



jQuery AJAX常用方法[メソッド]—load ()

- load()方法[メソッド]从服务器加载数据，并将返回的数据放入选定的元素：`$(selector).load(URL,data,callback);`
 - URL参数[引数]指定你要加载的URL
 - 可选的data参数[引数]指定要与请求一起发送的一组查询字符串键/值对
 - 可选的回调参数[引数]是load()方法[メソッド]完成后要执行的函数的名称
- 尝试代码：[ajax_load.html](#)

jQuery AJAX常用方法[メソッド]—get () post ()

- get () 和post () 方法[メソッド]用于通过HTTP GET或POST请求从服务器请求数据。
- `$.get(URL,callback);`
- `$.post(URL,data,callback);`
- [所有方法参考](#)

Q & A

You Have Questions
We Have Answers

THANK YOU