匠人学院
JR ACADEMY

Web开发进阶班第1期

Lesson 6 JavaScript

2023.08.01

匠人学院
JR ACADEMY

1. JS概念介绍
2. JS数据类型
3. 运算符、基本逻辑
4. 函数、参数

1. JS概念介绍

HTML : Content Structure

CSS : Content Presentation (Layout and Design)

JS : Content Behaviour

Interaction with user, User behaviour
tracking, Post and get(fetch) data (aka AJAX)

HTML + CSS + JS

1. JS概念介绍

What is JavaScript

JavaScript is a high-level, weakly typed, object-based interpreted programming language.

JavaScript is one of the three core technologies of web development.

Client-side: Browser

Server-side: Node.js

1. JS概念介绍

在1994年，当时的**网景公司**（Netscape）凭借**Navigator**这个浏览器成为了Web时代开启最著名的第一代的互联网公司。当时所用的第一版本的浏览器就是下图中的浏览器，相信有很多年轻朋友都没有见过这种浏览器。如果大家回想十年前或是十五年前，大家用的浏览器应该是**IE浏览器**，那个时候的浏览器跟现在的比如说**谷歌浏览器、火狐浏览器**等相比而言，那就是天壤之别了。

当时整个浏览器都是静态的，也就是用**HTML**和**CSS**写的，并没有像今天的各种浏览器一样具有各种各样的动态效果了，比如像网页的图片轮播、鼠标悬浮切换等效果。
网景公司就想在原来的静态页面的基础上添加一些动态的效果，这时候网景公司出来一个很牛的人物叫**布兰登·艾奇**，他用不到两周的时间就设计出了能在网页上实现动态效果的编程语言。

为什么会命名为JavaScript呢？原因是在当时，Java非常火，网景公司希望借用Java在当时的名气来进行推广。其实事实上呢，JavaScript除了语法上有点像Java外，别的地方都跟Java没有任何关系。

1. JS概念介绍

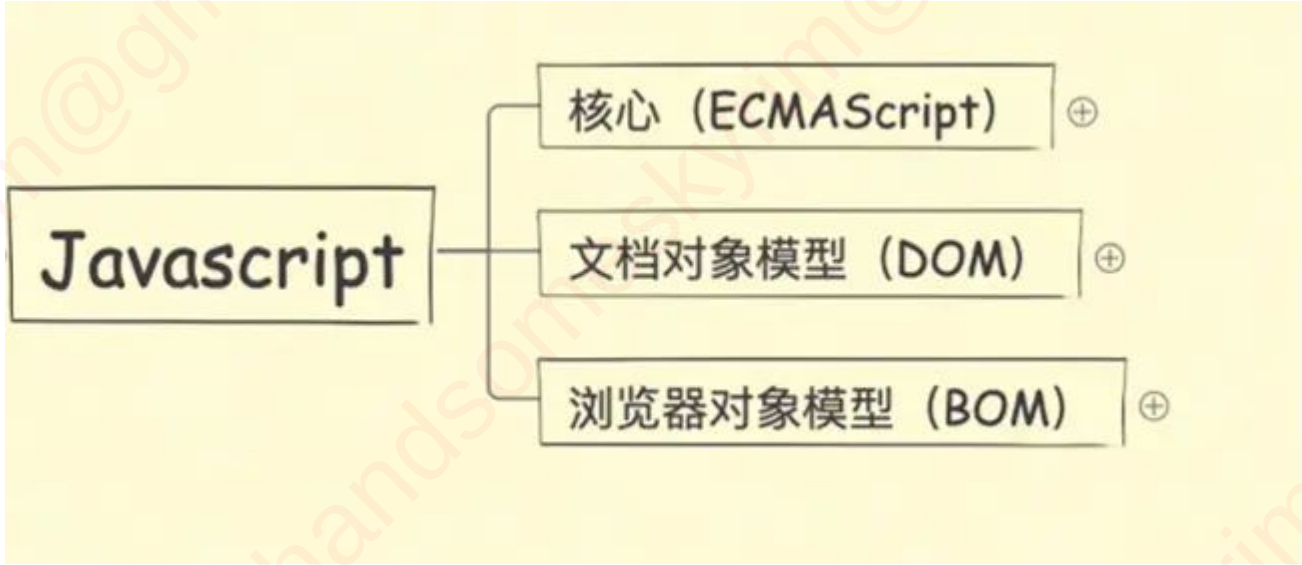
匠人学院 JR ACADEMY

JavaScript和ECMAScript的关系

从上面讲的JavaScript的由来中，我们就知道JavaScript由网景公司的布兰登·艾奇开发出来的，一年后，微软又模仿JavaScript开发出了一种编程语言叫**JScript**,再后来，陆续又有别的商家推出JavaScript的不同实现语言。这就导致JavaScript的语法和特性日益混乱，其标准化问题被提上日程。最终由欧洲计算机制造商协会（ECMA）以JavaScript1.1为蓝本，制定了【ECMA-262】标准，并由此标准定义了一种新脚本语言ECMAScript。随后，ISO也采用ECMAScript作为标准，各浏览器厂商便纷纷开始将ECMAScript作为各自JavaScript实现的基础。

那到底**JavaScript**和**ECMAScript**有什么关系呢？ECMAScript其实并不等同于JavaScript，它只是JavaScript的核心标准（语法、类型、语句、关键字、保留字、操作符、对象），而JavaScript还包括文档对象模型（DOM）和浏览器对象模型（BOM）等。其中各主流浏览器对ECMAScript的支持都还不错，但对DOM的支持相差较大，对于BOM一直没有相关标准。最后再简单总结一下就是：**ECMAScript是一种语言标准，JavaScript是对ECMAScript的一种实现**。

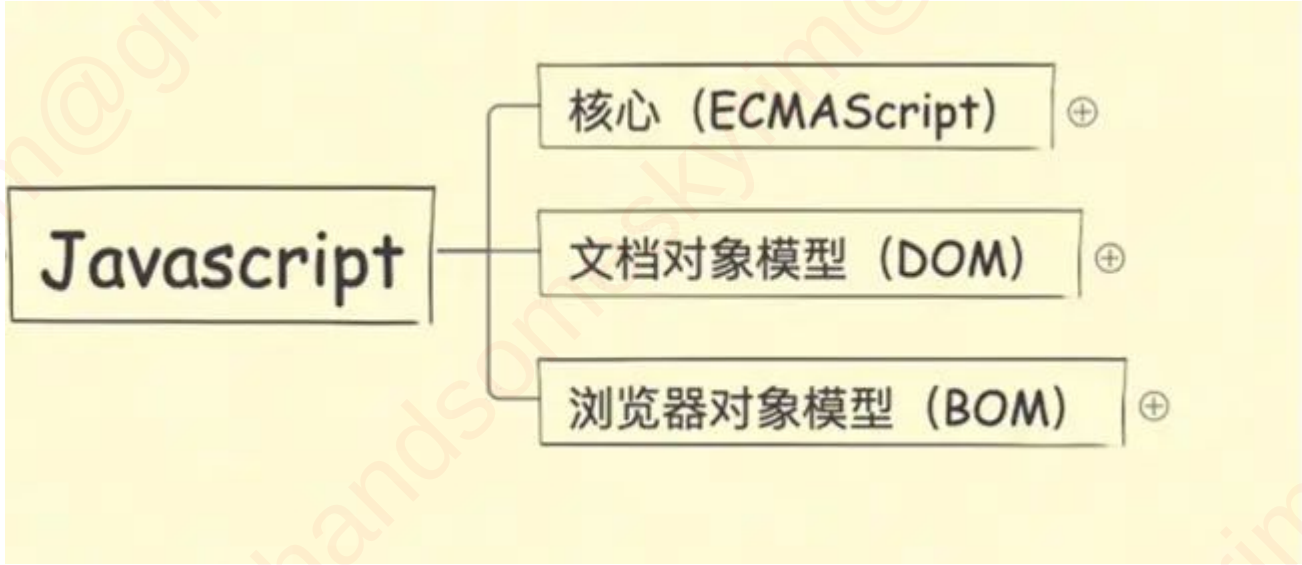
1. JS概念介绍



ECMAScript(核心)

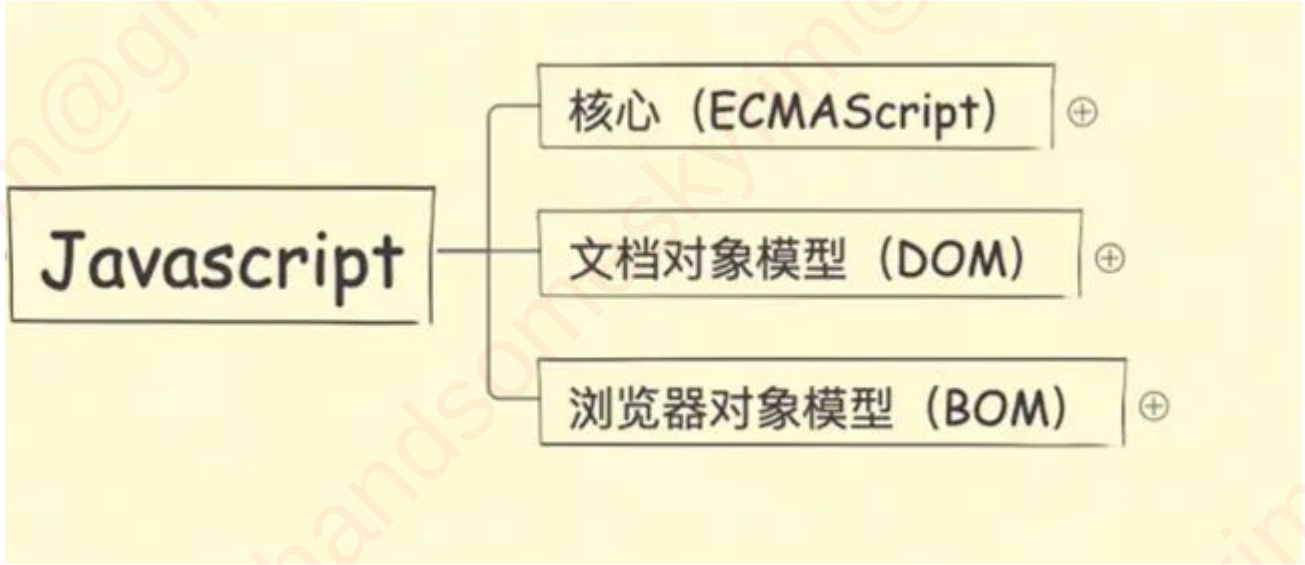
【ECMA-262】并没有参照web浏览器，规定了语言的组成部分，其具体内容包括语法、类型、语言、关键字、保留字、操作符、对象等。

1. JS概念介绍



文档对象模型(DOM)是针对XML但经过扩展用于HTML的应用程序编程接口（API）。DOM把整个页面映射为一个多层次节点结构。HTML或者XML页面中的每个组成部分都是某种类型的节点，这些节点又包含着不同类型的数据。

1. JS概念介绍



BOM（浏览器对象模型）

浏览器对象模型(BOM)是处理**浏览器窗口**和**框架**，我们习惯上把所有针对浏览器的JavaScript扩展算作是BOM的一部分。包括以下：

- 弹出新浏览器窗口的功能。
- 移动、缩放和关闭浏览器窗口的功能。
- 提供浏览器所加载页面的详细信息的navigator对象。
- 提供浏览器所加载页面的详细信息的location对象。
- 提供用户分辨率详细信息的screen对象。
- 对cookies的支持。
- 像XMLHttpRequest和IE的ActionXobject这样的自定义对象。

1. JS概念介绍

Before we go

- Open Chrome
- Right-click(F12) on any page element and select Inspect Element.
- Move to Console

```
console.log("Hello World”);
console.warn(“This is a warning message");
console.error(“This is an error message”);
```

1. JS概念介绍

Now Try in Console

```
2 + 2;

4 + 0.1;

10 % 3;

10 / 3;

2 ** 4;

0.1 + 0.2
```

2. JS数据类型

变量

一个变量，就是一个用于存放数值的容器。这个数值可能是一个用于累加计算的数字，或者是一个句子中的字符串。变量的独特之处在于它存放的数值是可以改变的。让我们看一个简单的例子：

```
var myVariable;
定义一个变量，你可以赋值

myVariable = ‘jiangren’;

也可以将这些操作写在一行
var myVariable = 'jiangren';

读取变量
myVariable;

可以改变变量的值
var myVariable = 'Billy';
myVariable = 'Lightman';
```

2. JS数据类型

变量

JavaScript 是一种动态语言，也就是说，变量的类型没有限制，可以赋予各种类型的值：

```
1. var a = 0 ;
2. a = ‘String’;
3. a = false;
4. let b = 1;
5. b = ‘kevin’;
```

2. JS数据类型

变量

var 与 let (const)的区别

原因是有些历史性的。回到最初创建 JavaScript 时，是只有 var 的。在大多数情况下，这种方法可以接受，但有时在工作方式上会有一些问题——它的设计会令人困惑或令人讨厌。因此，let 是在现代版本中的 JavaScript 创建的一个新的关键字，用于创建与 var 工作方式有些不同的变量，解决了过程中的问题。

首先，如果你编写一个声明并初始化变量的多行 JavaScript 程序，你可以在初始化一个变量之后用 var 声明它，它仍然可以工作。例如：

```
myName = "Chris";

function logName() {
  console.log(myName);
}

logName();

var myName;
```

但提升操作不再适用于 let 。如果将上面例子中的 var 替换成 let 将不起作用并引起一个错误。这是一件好事——因为初始化后再声明一个变量会使代码变得混乱和难以理解。

2. JS数据类型

变量
let和const的区别

let: 声明的是变量

let声明的变量可以改变，值和类型都可以改变；

const: 声明的是常量
const声明的常量不可以改变，这意味着，const一旦声明，就必须赋予初始化值，不能以后再赋予一个新的值(对象)

```
const PI = 3.1415;
PI // 3.1415

PI = 3;
// TypeError: Assignment to constant variable.
```

2. JS数据类型

命名

1. UPPERCASE, **camelCase**, PascalCase, under_score, hy-phen

2. Starting character:

    no number, no symbol (except for _ and $)

3. Reserved words:

    if, else, instanceof, true, switch ... more

2. JS数据类型

命名

```
age
myAge
init
initialColor
finalOutputValue
audio1
audio2
```

```
1
a
_12
myage
MYAGE
var
Document
skjfndskjfnbdskjfb
thisisareallylongstupidvariablenameman
```

2. JS数据类型

匠人学院 JR ACADEMY

数值Number
你可以在变量中存储数字，不论这些数字是像 30（也叫整数）这样，或者像 2.456 这样的小数（也叫做浮点数）。与其他编程语言不同，在 JavaScript 中你不需要声明一个变量的类型。当你给一个变量数字赋值时，不需要用引号括起来。

```
let myAge = 17;
```

字符串String
字符串是文本的一部分。当你给一个变量赋值为字符串时，你需要用单引号或者双引号把值给包起来，否则 JavaScript 将会把这个字符串值理解成别的变量名。

```
let dolphinGoodbye = "So long and thanks for all the fish";
```

布尔值 Boolean:
Boolean 的值有 2 种：true 或 false。它们通常被用于在适当的代码之后，测试条件是否成立。举个例子，一个简单的示例如下：

```
let iAmAlive = true;
```

undefined：表示“未定义”或者不存在，表示没有定义所以此处暂时没有任何值
null: 表示无值，即此处的值的是“无”的状态
数组Array
数组是一个单个对象，其中包含很多值，方括号括起来，并用逗号分隔。尝试在您的控制台输入以下行：

```
let myNameArray = ["Chris", "Bob", "Jim"];
let myNumberArray = [10, 15, 40];
```

对象 Object：各种值的集合
可以有一个代表一个人的对象，并包含有关他们的名字，身高，体重，他们说什么语言，如何说 你好，他们，等等。

```
let dog = { name: "Spot", breed: "Dalmatian" };
```

2. JS数据类型

匠人学院 JR ACADEMY

数值Number
你可以在变量中存储数字，不论这些数字是像 30（也叫整数）这样，或者像 2.456 这样的小数（也叫做浮点数）。与其他编程语言不同，在 JavaScript 中你不需要声明一个变量的类型。当你给一个变量数字赋值时，不需要用引号括起来。

```
let myAge = 17;
```

字符串String
字符串是文本的一部分。当你给一个变量赋值为字符串时，你需要用单引号或者双引号把值给包起来，否则 JavaScript 将会把这个字符串值理解成别的变量名。

```
let dolphinGoodbye = "So long and thanks for all the fish";
```

布尔值 Boolean:
Boolean 的值有 2 种：true 或 false。它们通常被用于在适当的代码之后，测试条件是否成立

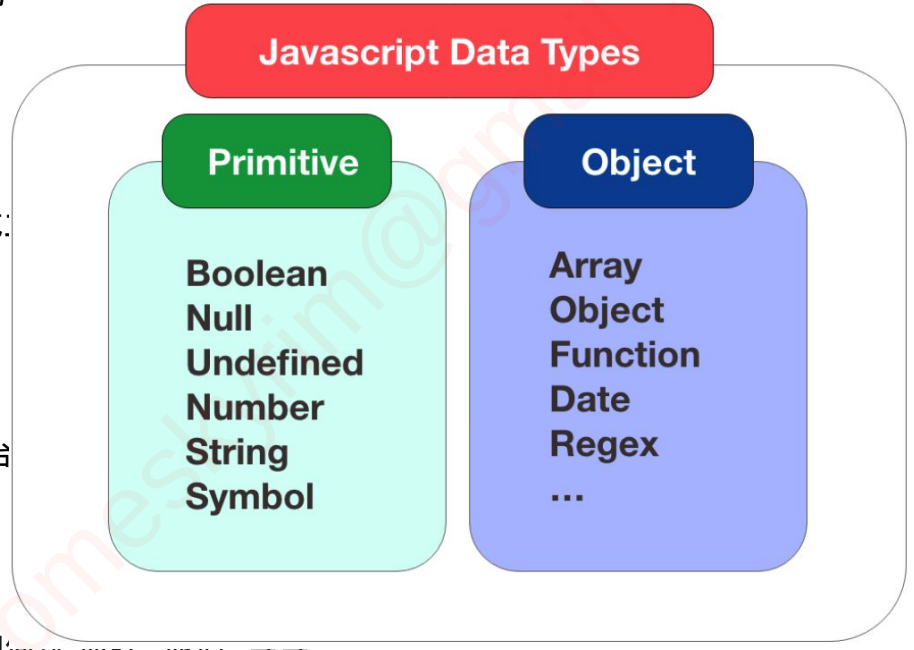
```
let iAmAlive = true;
```

undefined：表示“未定义”或者不存在，表示没有定义所以此处暂时没有任何值
null: 表示无值，即此处的值的是“无”的状态
数组Array
数组是一个单个对象，其中包含很多值，方括号括起来，并用逗号分隔。尝试在您的控制台

```
let myNameArray = ["Chris", "Bob", "Jim"];
let myNumberArray = [10, 15, 40];
```

对象 Object：各种值的集合
可以有一个代表一个人的对象，并包含有关他们的名字，身高，体重，他们说什么语言，如何称呼对方，他们，爸爸。

```
let dog = { name: "Spot", breed: "Dalmatian" };
```



2. JS数据类型

数组

```
let arr = [‘x’,’y’,’z’];

console.log(arr.length) //3

arr[4] = ‘b’; // arr = [‘x’,’y’,’z’, , ‘b’]

console.log(arr.length) 5

//get the last element by index
myArray[(myArray.length - 1)]

// push an element to the end of array
myArray.push("tail elememt")

// put an element to the head of array
myArray.push("head elememt")

// get the index of an element
myArray.indexOf(true)

// check an element exists in the array
myArray.includes("new elememt")
```

2. JS数据类型

匠人学院 JR ACADEMY

Now try in console

```
let myArray = [‘a’,’b’ , 10, false];
console.log(myArray[0]);
myArray[100] = “101st item”;
console.log(myArray);

let a = [[1, 3], [7, 9]];
// first row second column
console.log(a[0][1])
// second row second column
console.log(a[1][1])
```

2. JS数据类型

匠人学院 JR ACADEMY

对象Object

对象（object）是JavaScript的核心概念，也是最重要的数据类型。
JavaScript的所有数据都可以被视为对象。
所谓对象，就是一种无序的数据集合，由若干个“键值对”（key-value）构成。

```
var obj = {};

obj.key = 'string value';
obj.myArray = [1,2,3,4];
obj.myArrayLength : = function() { return this.myArray.length()};
```

2. 运算符、基本逻辑

算术运算符

算术运算符是我们用来做和的基本运算符：

运算符	名称	作用	示例
`+`	加法	两个数相加。	`6 + 9`
`-`	减法	从左边减去右边的数。	`20 - 15`
`*`	乘法	两个数相乘。	`3 * 7`
`/`	除法	用右边的数除左边的数	`10 / 5`
`%`	求余 (有时候也叫取模)	在你将左边的数分成同右边数字相同的若干整数部分后，返回剩下的余数	`8 % 3` (返回 2，8 除以 3 的倍数，余下 2。)
`**`	幂	取底数的指数次方，即指数所指定的底数相乘。它在 EcmaScript 2016 中首次引入。	`5 ** 5` (返回 3125，相当于 `5 * 5 * 5 * 5 * 5` 。)

自增和自减运算符

有时候，您需要反复把一个变量加 1 或减 1。这可以方便地使用增量（`++`）和递减（`--`）运算符来完成。

```
guessCount++;
```

运算符	名称	作用	示例	等价于
`+=`	加法赋值	右边的数值加上左边的变量，然后再返回新的变量。	`x = 3; x += 4;`	`x = 3; x = x + 4;`
`-=`	减法赋值	左边的变量减去右边的数值，然后再返回新的变量。	`x = 6; x -= 3;`	`x = 6; x = x - 3;`
`*=`	乘法赋值	左边的变量乘以右边的数值，然后再返回新的变量。	`x = 2; x *= 3;`	`x = 2; x = x * 3;`
`/=`	除法赋值	左边的变量除以右边的数值，然后再返回新的变量。	`x = 10; x /= 5;`	`x = 10; x = x / 5;`

比较运算符

有时，我们将要运行真/假测试，然后根据该测试的结果进行相应的操作 - 为此，我们使用比较运算符。

运算符	名称	作用	示例
`===`	严格等于	测试左右值是否相同	`5 === 2 + 4`
`!==`	严格不等于	测试左右值是否不相同	`5 !== 2 + 3`
`<`	小于	测试左值是否小于右值。	`10 < 6`
`>`	大于	测试左值是否大于右值	`10 > 20`
<=	小于或等于	测试左值是否小于或等于右值。	`3 <= 2`
>=	大于或等于	测试左值是否大于或等于正确值。	`5 >= 4`

2. 运算符、基本逻辑

== 相等
=== 严格相等
!= 不相等
!== 严格不相等
< 小于
<= 小于或等于
> 大于
>= 大于或等

1 === '1' //fasle
1 == '1' // true
[1] === '1' //fasle
[1] == 1 //true

只有检测 null/undefined 的时候可以使用 x == null
var a = undefined;
!a // true
a == null//false

绝大多数场合应该使用
===

2. 运算符、基本逻辑

Now try in console

4 + 3 + “3”

"4" + 3 + 3

0.1 + 0.2

1 + null

1 + undefined

2. 运算符、基本逻辑

匠人学院 JR ACADEMY

条件Condition

1. 条件语句提供一种语法构造，只有满足某个条件，才会执行相应的语句。
   JavaScript

```
var a = 0;
if(a === 0) { // a === is expression
  a = a + 100;  //run this statement while the expression is true
} else {
  a = a - 100; // run this statement while the expression is false
}
```

2. 三元运算符 ?:

```
(condition) ? expr1 : expr2
```

等同於

```
if(condition) {
  expr 1
} else {
  expr 2
}
```

2. 运算符、基本逻辑

循环Loop

for语句是循环命令的另一种形式，可以指定循环的起点、终点和终止条件。它的格式如下。

```
for (initialize; test; increment) {
  statement
}
```

```
for (var i = 0; i < 100 ; i++) {
 console.log('i Now is  : ' + i);
}
```

* i 由 0 开始 ，到 99

2. 运算符、基本逻辑

循环Loop

While语句包括一个循环条件和一段代码块，只要条件为真，就不断循环执行代码块。

```
while (expression) {
  statement;
}

var i = 0;
while (i < 100) {
  console.log('i Now is  : ' + i);
  i = i + 1;
}
```

* i 由 0 开始 ，到 99

2. 运算符、基本逻辑

循环Loop

break语句和continue语句都具有跳转作用，可以让代码不按既有的顺序执行。
break语句用于跳出代码块或循环。

```
var i = 0;

while(i < 100) {
  console.log('Now is：' + i);
  i++;
  if (i === 10) break; //到10 离开這個block
}
```

2. 运算符、基本逻辑

循环Loop

continue语句用于立即终止本轮循环，返回循环结构的头部，开始下一轮循环。

```
var i = 0;

while (i < 100){
  i++;
  if (i%3 === 0) continue;
  console.log('i当前为：' + i);
}
```

上面代码有在i为的数时时，不才会输出i的值。直接进入下一轮循环。

2. 函数，参数

匠人学院 JR ACADEMY

函数Function

function命令声明的代码区块，就是一个函数。function命令后面是函数名，函数名后面是一对圆括号，里面是传入函数的参数。函数体放在大括号里面。

```
function print(parameter) {
  console.log(parameter);
}

print('String text') // String text
```

2. 函数，参数

函数Function

函数体内部的return语句，表示返回
JavaScript引擎遇到return语句，就直接返回return后面的那个表达式的值，后面即使还有语句，也不会得到执行。也就是说，return语句所带的那个表达式，就是函数的返回值。

```
function add(a,b) {
  return a + b;
  console.log('I will not be executed');
}

add(5,10) // 10
```

2. 函数，参数

函数Function

函数体内部的return语句，表示返回
JavaScript引擎遇到return语句，就直接返回return后面的那个表达式的值，后面即使还有语句，也不会得到执行。也就是说，return语句所带的那个表达式，就是函数的返回值。

```
function isEven (num) {
  if(num%2 === 0) {
    return true;
  }
  return false;
}
```

```
function isEven (num) {
  var result;
  if(num%2 === 0) {
    result = true;
  } else {
    result = false;
  }
  return result;
}
```

2. 函数，参数

Practice

在0-N里面找出前十个，是4的倍数，但不是10的倍数

write a function that receives N as parameter, find first 10 numbers, which is multiples of 4, but not multiples of 10

expression1 && expression2 (true && true => true)
expression1 || expression2 (true || false => true)

first 10 of array => slice an array => array.slice(0, 10)

2. 函数，参数

匠人学院 JR ACADEMY

Homework:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

匠人学院
JR ACADEMY

Q & A