Web 开发 基础班

HTML + CSS + JS

HTML : Content Structure

CSS : Content Presentation (Layout and Design)

JS : Content Behaviour

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

What is JavaScript

JavaScript is a **dynamic**, **weakly typed** programming language.

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

Client-side: Browser

Server-side: Node.js

Javascript历史

1994年10月，NCSA的一个主要程序员Marc Andreessen联合风险投资家Jim Clark，成立了Mosaic通信公司（Mosaic Communications），不久后改名为Netscape。这家公司的方向，就是在Mosaic的基础上，开发面向普通用户的新一代的浏览器Netscape Navigator。

Netscape公司很快发现，Navigator浏览器需要一种可以嵌入网页的脚本语言，用来控制浏览器行为。当时，网速很慢而且上网费很贵，有些操作不宜在服务器端完成。比如，如果用户忘记填写“用户名”，就点了“发送”按钮，到服务器再发现这一点就有点太晚了，最好能在用户发出数据之前，就告诉用户“请填写用户名”。这就需要在网页中嵌入小程序，让浏览器检查每一栏是否都填写了。

1995年，Netscape公司雇佣了程序员Brendan Eich开发这种网页脚本语言。

1995年5月，Brendan Eich只用了**10天**，就设计完成了这种语言的第一版。它是一个**大杂烩**

* Brendan 是Mozilla 公司的首席技术官

Before we go

- Open Chrome
- Right-click 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”);
```

Now Try in Console

```
2 + 2;

4 + 0.1;

10 % 3;

10 / 3;

2 ** 4;
```

变量 Variable

Variables 是你存储数据的容器。要声明一个变量你需要使用关键字 **var**

```
var myVariable;
```

定义一个变量之后，你可以赋予它一个值：

```
myVariable = ‘jiangren’;
```

你也可以将这些操作写在一行：

```
var myVariable = 'jiangren';
```

你可以通过变量名称读取变量：

```
myVariable;
```

在给变量赋值之后，你可以改变变量的值：

```
var myVariable = 'Billy';
myVariable = 'Lightmen';
```

变量 Variable

1. JavaScript 是一种动态类型(Dynamic Type)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;

2. JavaScript引擎的工作方式是，先解析代码，获取所有被声明的变量，然后再一行一行地运行。这造成的结果，就是所有的变量的声明语句，都会被提升到代码的头部，这就叫做变量提升（hoisting)
   1. console.log(myVariable);
   2. var myVariable; // Unlike other programming language, No error will occur

命名规则 Naming convention

1. Cases:
   FINAL CONSTANT
   react component
   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

数据类型 Data type

Not a number
function isNaN

数值（number）：整数和小数（比如1, 3.14 , NaN, infinity ）
字符串（string）：字符组成的文本（比如”jiangren”）
布尔值（boolean）：true（真）和false（假）两个特定值
undefined：表示“未定义”或不存在，即由于目前没有定义，所以此处暂时没有任何值
null：表示无值，即此处的值就是“无”的状态。
对象（object）：各种值组成的集合
数组(Array)：一种允许你存储多个值在一个引用里的结构。

Map
Set

运算符	解释	符号	示例
加/连接	用来相加两个数字，或者连接两个字符串。	+	6 + 9; "Hello " + "world!";

减、乘、除	这些运算符操作将与你期望它们在基础数学中所做的一样。	-, *, /	9 - 3;
			8 * 2; // JS中的乘是一个"*"号;
			9 / 3;

赋值运算符	你之前已经见过这个符号了：它将一个值赋给一个变量	=	var myVariable = 'Bob';
相等	它将测试两个值是否相等，而且会返回一个 true/false （布尔型）值。	===	var myVariable = 3; myVariable === 4;

非，不等	经常与相等运算一起使用，非运算符在JS中表示逻辑非——它也返回一个布尔值。	!, !==	原本的值是 true ，但是返回了 false 因为之后我们做了非运算：
			var myVariable = 3;
			!myVariable === 3;
			这里我们测试了"我的 myVariable 是否等于 3". 这里返回了 false, 因为它等于 3.
			var myVariable = 3;
			myVariable !== 3;

[Operator precedence]

Now Try in Console

4 + 3 + “3”

"4" + 3 + 3

0.1 + 0.2

1 + null

1 + undefined

Comparator 比较运算符

== 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

var a = undefined;
!a // true

绝大多数场合应该使用
===

Array 数组

数组（array）是按次序排列的一组值。每个值的位置都有编号（从0开始），整个数组用方括号表示。

var arr = [‘x’,’y’,’z’];

console.log(arr.length) //3

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

console.log(arr.length) 5

Now Try in Console

```
var anArray = [‘a’,’b’ , 10, false];
console.log(anArray[0]);
anArray[100] = “100th item”;
console.log(anArray);

anArray[101] = ‘hi’;
anArray.length


var a = [[1, 3], [7, 9]];
console.log(a[0][1])
console.log(a[1][1])
```

条件语句 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
}
```

循环语句 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

循环语句 Loop

Nested Loop

```
for (var i = 0; i < 100 ; i++ ) {
  for(var j = 0 ; j < 100 ; j++) {
    console.log(“i : ” + i “ and j:” + j);
  }
}
```

循环语句 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

循环语句 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
}
```

循环语句 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。

quiz

在0-100之间，找出前十个，是4的倍数，但不是5的倍数的数字。

Function 函数

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
```

Argument

Function 函数

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
```

Function 函数

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;
}
```

Function declaration and function expression

Function declaration

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

Function expression

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

Almost identical, but hoisting!

Function 函数

作用域（scope）指的是变量存在的范围。Javascript只有两种作用域：一种是全局作用域，变量在整个程序中一直存在，所有地方都可以读取；另一种是函数作用域，变量只在函数内部存在。
在函数外部声明的变量就是全局变量（global variable），它可以在函数内部读取。
在函数内部定义的变量，外部无法读取，称为“局部变量”（local variable）。

```
var globalVar =“I am Global variable”;

function f() {
  var localVar = “I am local variable”;
  console.log (globalVar) // Ok
}
console.log(localVar) //ReferenceError: localVar is not defined
```

Function 函数

内部变量提升 与全局作用域一样，函数作用域内部也会产生“变量提升”现象。var命令声明的变量，不管在什么位置，变量声明都会被提升到函数体的头部。

```
function foo(x) {
  if (x > 100) {
    var localVar = 1000;
  }
}
```

=

```
function foo(x) {
  var localVar
  if (x > 100) {
    localVar = 1000;
  }
}
```

Function 函数

有时需要提供外部数据，不同的外部数据会得到不同的结果，这种外部数据就叫参数 (Parameter) 。

```
function add(x,y) {
  return x + y;
}
```

Javascript允许省略参数

```
function abc(a,b,c){
  return(a);
}

abc(1,2,3)//1
abc(1,2)//1
abc() //undefined
```

Function 函数

函数参数如果是原始类型的值（数值、字符串、布尔值），传递方式是传值传递（传值传递（passes by value）。这意味着，在函数体内修改参数值，不会影响到函数外部。

```
var a = 100;

function square(num) {
  num = num * num; // num is 10000
}

square (a);//a is 100
```

Function 函数

但是，如果函数参数是复合类型的值（数组、对象、其他函数），传递方式是传址传递（pass by reference）。也就是说，传入函数的原始值的地址，因此在函数内部修改参数，将会影响到原始值。

```
var a = { key:100 };

function squareFuc(args) {
  args.key = args.key * args.key;
}

squareFuc(a);
console.log(args.key);//10000
```

Object 对象

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

console.log(obj.key); // string value
console.log(obj.myArray[0])
```

Object 对象

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()};
```

Object 对象

```
var car =  {
    name  :“Toyota”,
    door : 2,
    model : “86 GT”,
    year : “2014”,
    fullName :  function() {
      return this.name + “ ” this.model +“ - “ +this.year
    }
}

car.fullName();//Toyota 86GT - 2014
```

Object 对象

如果不同的变量名指向同一个对象，那么它们都是这个对象的引用，也就是说指向同一个内存地址。修改其中一个变量，会影响到其他所有变量。

```
var billy = { a : 100} ;
var lightmen = billy;

lightmen.a = 200;
console.log(billy.a);
```

Object 对象

for...in循环用来遍历一个对象的全部属性。

```
var o = {a: 1, b: 2, c: 3};

for (var i in o) {
  console.log(o[i]);
}

// 1
// 2
// 3
```

Quiz

在一个升序,无重复数的数组中，给定一个目标值，找出哪两个数的和为这个目标值(任意一个组合），并返回这两个数的序号（index）
比如：
Array = [1,3,4,6,7,8,10,14,15]
Target = 14
要求：写一个function， 输入为一个数组和一个数字，输出为长度为2的数组

DOM

DOM是JavaScript操作网页的接口，全称为“文档对象模型”（Document Object Model）。它的作用是将网页转为一个JavaScript对象，从而可以用脚本进行各种操作（比如增删内容）。
浏览器会根据DOM模型，将结构化文档（比如HTML和XML）解析成一系列的节点，再由这些节点组成一个树状结构（DOM Tree）。所有的节点和最终的树状结构，都有规范的对外接口。所以，DOM可以理解成网页的编程接口

DOM

DOM的最小组成单位叫做节点（node）。文档的树形结构（DOM树），就是由各种不同类型的节点组成。每个节点可以看作是文档树的一片叶子。

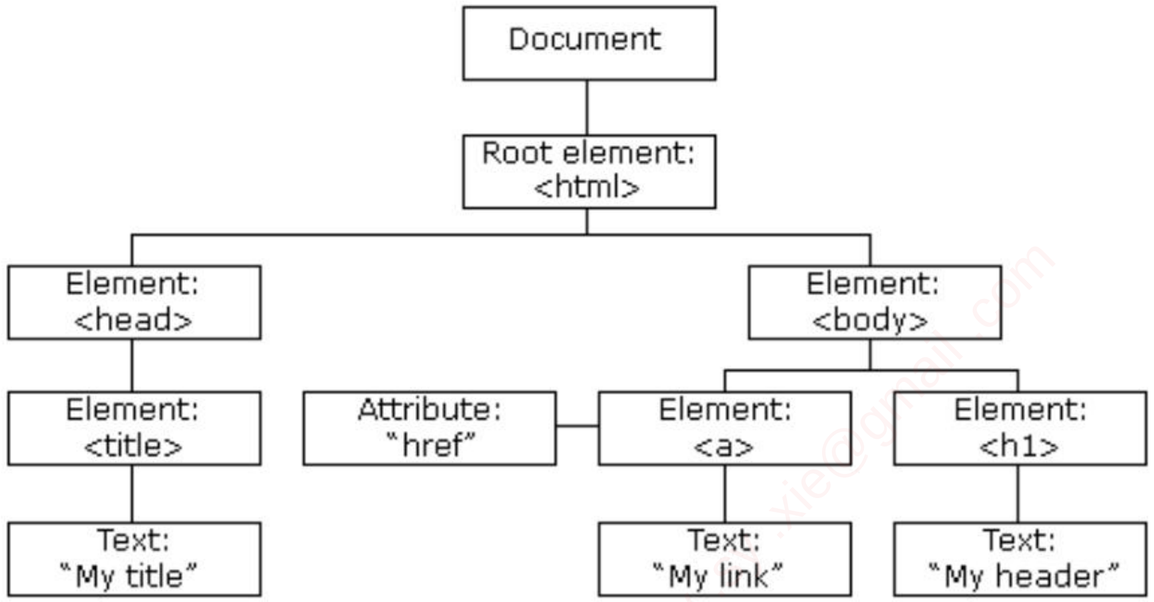
Document：整个文档树的顶层节点

Element：网页的各种HTML标签（比如<body>、<a>等）

Attribute：网页元素的属性（比如class="right"）

Text：标签之间或标签包含的文本

Comment：注释



Common JS methods for DOM Manipulation

1. querySelector
   return the first element found or null
2. querySelectorAll
   return all elements as NodeList object, or empty object
3. addEventListener
   assign function to certain event
4. removeEventListener
5. createElement
   create a new HTML element using the name of HTML tag
6. appendChild
   add an element as the last child to the element that is invoking this method
7. removeChild
8. insertBefore
   add an element before a child element
9. setAttribute
   add a new attribute to an HTML element

More Methods
Events

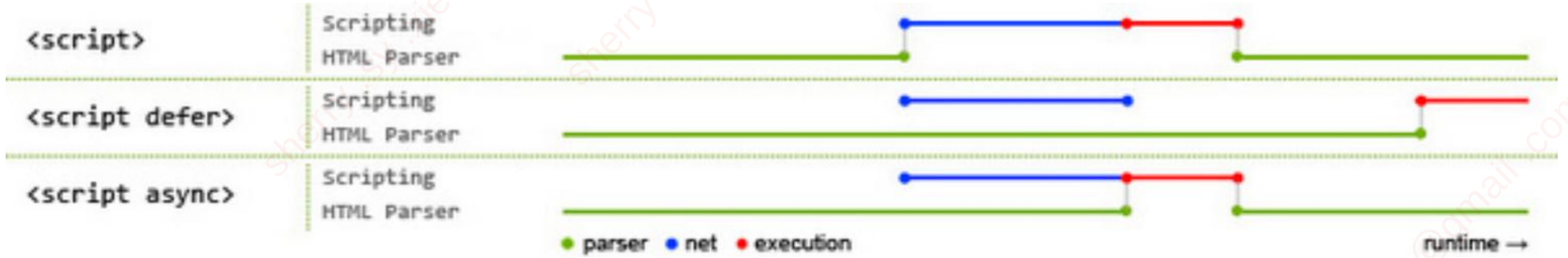
JS 异步加载

通常情况下，我们会把script tag放在head里。考虑这种情况，如果某一个script文件需要花费很久时间下载，执行。我们的页面会是什么情况。

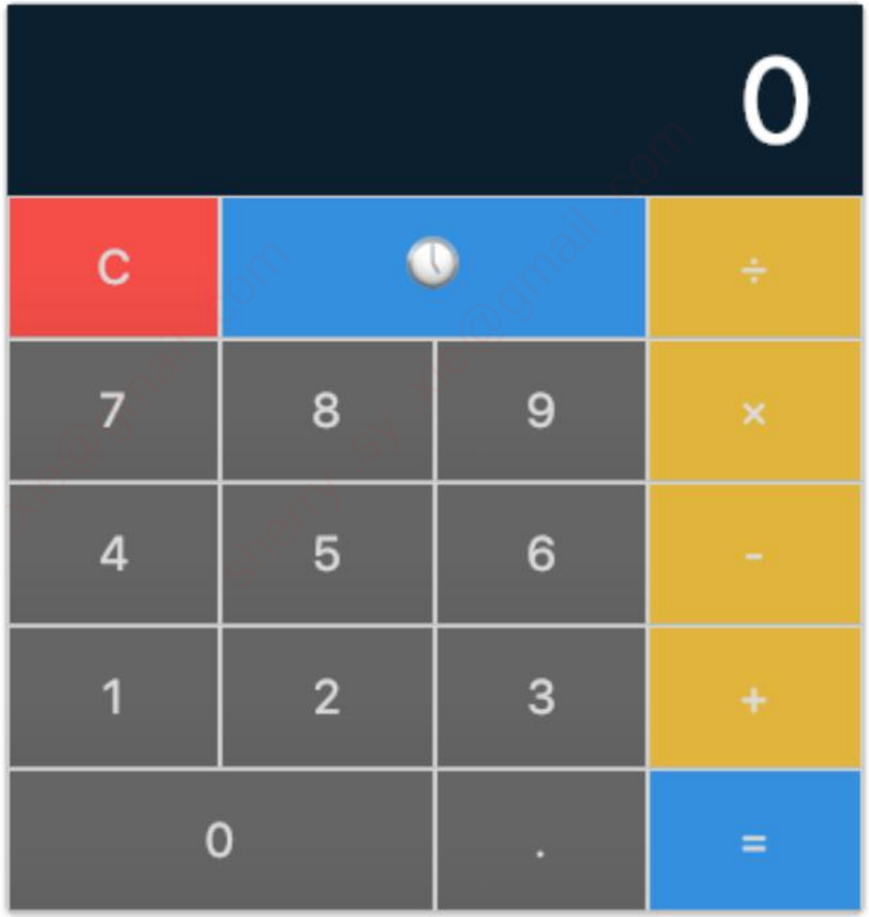
How to solve?

Put script in the body?

Or ‘async’ and ‘defer’



Build a simple calculator



新的社交媒介

微信公共号企业号

jiangren.com.au

facebook.com/jiangrenquan

weibo.com/ozitquan

meetup.com/en-AU/itgroup/