JS整理笔记-runoob-23-9

📌 TIPS：1. 字体颜色含义：绿色文字：常の理解　红色文字：常不理解　黑色文字：常瞎写的

橘色文字：有点屌的摘抄

2.荧光部分：notice或者引导理解

3.关于图标：📌:常认为琐碎的杂知识 和 TIPS

❓：常不明白

❗：常警觉

关键词var创建变量

无值变量值和数据类型为undefined。

定义：obj->{} （obj指向对象）

键名：`foo` 键值：`Hello` 键值对：`foo: 'Hello'`

```
var obj = {
  foo: 'Hello',
  bar: 'World'
};
```

看见{}：表达式 要不 语句

函数外部无法读取内部声明变量

函数内部可直接读取全局变量

圆括号（）出现在函数名后，表示调用该函数

IIFE：（立即调用函数表达式）是一个在定义时就会立即执行的 JavaScript 函数。

```
(function () {
  // …
```

```
})();

(() => {
  // …
})();

(async () => {
  // …
})();

```

第一部分是一个具有词法作用域的匿名函数，并且用圆括号运算符 `()` 运算符闭合起来。这样不但阻止了外界访问 IIFE 中的变量，而且不会污染全局作用域。

第二部分创建了一个立即执行函数表达式 `()`，通过它，JavaScript 引擎将立即执行该函数。

以下，本里记得：

```
 (() => {
  // 初始化代码
  let firstVariable;
  let secondVariable;
})();

// firstVariable 和 secondVariable 变量在函数执行后会被丢弃

```

eval函数：计算 JavaScript 字符串，并把它作为脚本代码来执行。 语法：eval(string)

参数为表达式，执行表达式；参数js语句，则执行语句

```
eval("x=10;y=20;document.write(x*y)");
document.write("<br>" + eval("2+2"));
document.write("<br>" + eval(x+17));
```

输出结果：

```
200
4
27
```

从JS中访问html元素：`document.getElementById(id)`

a. id标识html元素

b. `innweHTML` 获取插入元素内容

文档加载完成后执行 `docuument.write` 原有被覆盖

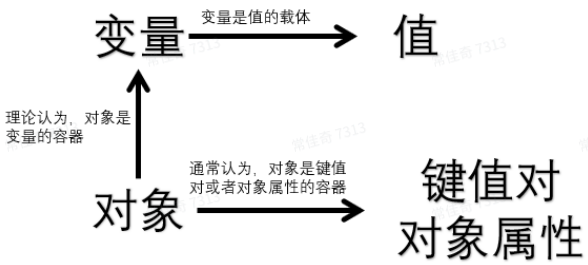
代码 - 依次执行　代码块 - 一并执行

type操作符 查看数据类型

```
typeof.[1,2,3,4] //返回object
typeof{name:"joho",age:"34"}//返回object
```

声明新变量时，可使用关键词 `new` 声明数据类型

访问变量属性 `person.lastName` or `person["lastName"]`



📌：`()` 调用方法

使用 `return` 函数停止，返回指定值，但是JS不会停止

代码修改自身元素内容

```
<button onClick = "this.innerHTML=Date()">现在的时间</button>
```

ES6模板字符串：

```
const message = `my name is ${name}`;
```

for 循环

```
for (var i = 0; v < 5 ; i++)
```

`var i=0` 为设置变量 `i<5` 为定义循环条件 `i++` 为已执行后操作

For in 循环 遍历对象属性

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

var text = "";
var x;
for (x in person) {
    text += person[x] + " ";
}
//输出结果为: John Doe 25
```

while循环

```
while（条件）
{
    需执行的代码
}
```

do ... while循环（while循环变体）

```
do
{
    需执行的代码
}
while（条件）
```

📌：break 与 continue

`break` :跳出循环

`continue` :跳过循环中的一个迭代（肯蹄牛谁就没有谁）

📌：其他-表格

catch 语句块	在 try 语句块执行出错时执行 catch 语句块。
function	定义一个函数
if ... else	用于基于不同的条件来执行不同的动作。
return	退出函数
switch	用于基于不同的条件来执行不同的动作
throw	抛出（生成）错误 。
try	实现错误处理，与 catch 一同使用。
var	声明一个变量。
while	当条件语句为 true 时，执行语句块。

📌：`undefined` 是没有设置值的变量

全局方法和局部方法(表格形式)

全局方法	String（false）	String（123）
Boolean方法	false.toString()	-
Number方法	-	(123).toString()

`d=new Date();` = `Number(d)=d.get.time`

数据类型返回

```
typeof "John"                 // 返回 string
typeof 3.14                   // 返回 number
typeof NaN                    // 返回 number
typeof false                  // 返回 boolean
typeof [1,2,3,4]              // 返回 object
typeof {name:'John', age:34} // 返回 object
typeof new Date()             // 返回 object
typeof function () {}         // 返回 function
```

```
typeof myCar                    // 返回 undefined (如果 myCar 没有声明)
typeof null                     // 返回 object
```

```
"John".constructor                 // 返回函数 String()  { [native code] }
(3.14).constructor                 // 返回函数 Number()  { [native code] }
false.constructor                  // 返回函数 Boolean() { [native code] }
[1,2,3,4].constructor              // 返回函数 Array()   { [native code] }
{name:'John', age:34}.constructor  // 返回函数 Object()  { [native code] }
new Date().constructor             // 返回函数 Date()    { [native code] }
function () {}.constructor         // 返回函数 Function(){ [native code] }
```

正则用于：

a. `search()`

b. `replace()`

❗ 正则表达式（我的评价是寄）

❗ `try catch` （寄 again）

`use strict` 严格模式 （基本用不到，活）

📌：关于 `=`

`=` : 赋值运算符　`==` : 比较运算符　`===` :严格比较运算符

null和undefined

`null` 用于对象

`undefined` 用于属性，变量，方法

❗ 验证API（我的评价是寄）

❗ this（寄 again）

`let` 就是在 `{}` 里有用 能解决重新声明变量问题

📌: JSON

JSON： 存储和传输数据的格式

服务端 —Data→ 网页
（使用JSON）

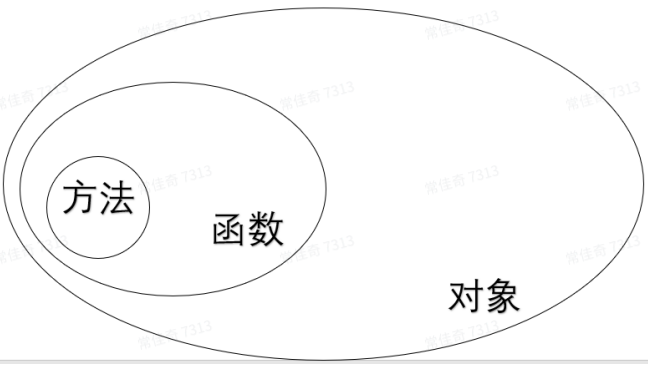
从服务器中读取JSON数据
并在网页中显示数据

- 独立语言
- 轻量级数据交换格式

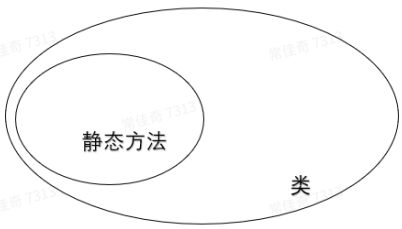
❗ 异步AJAX（我的评价是寄）

❗ Promise（寄 again）

📌：方法 函数 对象关系图解（附赠静态方法和类的）



- 静态方法是使用 static 关键字修饰的方法
- 又叫类方法
- 属于类的，但不属于对象
- 在实例化对象之前可以通过 **类名.方法名** 调用静态方法



❗ 原型链继承（zhei个好像寄一半）

❗ 冒泡与捕获（我的评价是寄）

❗ HTMLCollection（寄 again）

📌：常见HTML事件-表格

onchange	HTML 元素改变
onclick	用户点击 HTML 元素
onmouseover	鼠标指针移动到指定的元素上时发生
onmouseout	用户从一个 HTML 元素上移开鼠标时发生
onkeydown	用户按下键盘按键
onload	浏览器已完成页面的加载

📌：转义字符-表格

\'	单引号
\\"	双引号
\\	反斜杠
\n	换行
\r	回车
\t	tab（制表符）
\b	退格符
\f	换页符

📌：字符串方法

📌：

📌：

📌：

📌：

📌：

学完于：2023-09-05

写于：2023-09-08