JavaScript

独立的语言，浏览器具有 js 解释器

JavaScript 代码存在形式：

- Head 中

```
<script>
    //javascript 代码
    alert(123);
</script>

<script type="text/javascript">
    //javascript 代码
    alert(123);
</script>
```

- 文件

```
<script src='js 文件路径'> </script>
```

PS: JS 代码需要放置在 <body>标签内部的最下方

注释

当行注释 //

多行注释 /* */

变量：

python：

```
name = 'alex'
```

JavaScript:

```
name = 'alex'        # 全局变量
var name = 'eric' # 局部变量
```

写 Js 代码：

- html 文件中编写
- 临时，浏览器的终端 console

基本数据类型

数字

```
a = 18;
```

字符串

```
a = "alex"
a.chartAt(索引位置)
a.substring(起始位置，结束位置)
a.length      获取当前字符串长度
...
```

列表(数组)

```
a = [11,22,33]
```

字典

a = {'k1':'v1','k2':'v2'}

布尔类型

小写

for 循环

1. 循环时，循环的元素是索引

```
a = [11,22,33,44]
for(var item in a){
    console.log(item);
}

a = {'k1':'v1','k2':'v2'}
for(var item in a){
    console.log(item);
}
```

2.

```
for(var i=0;i<10;i=i+1){

}

a = [11,22,33,44]
for(var i=0;i<a.length;i=i+1){

}
```

不支持字典的循环

条件语句

```
if(条件){

}else if(条件){

}else if(条件){

}else{

}
swutch(w)
{
    case 0:   break;
    case 1:   break;
    default: break;
}
```

== 值相等

=== 值和类型都相等
&& and
|| or

函数:
普通函数:
```
function func(argv){
    console.log(argv);
}
func(1) //  调用
```

匿名函数:
```
setinterval(function(argv){
    console.log(argv);
}, 5000);
```

自执行函数:
```
(function(argv){
    console.log(argv);
})(1)
```

序列化:
c = SON.stringify(a) 将对象 a 转变为字符串 c
a = JSON.parse(c) 将字符串 c 还原成对象 a

转义:
客户端（cookie） 》 服务器
将数据经过转义后保存在 kookie

eval
python 中：var = eval(表达式) 只能识别简单的表达式，不能解释复杂的代码，可用 exec(执行代码)，但是却不能像 eval 有返回值
JavaScrip 中的 eval：即可执行代码，又可以后返回值

时间
Date 类
var d = new Date()
d.getXXX 获取
d.setXXX 设置

作用域
JavaScrip 中以函数为作用域

作用域一个示例:
```
var myTag = document.getElementByld('i1');
for(var i=0; i<mytag.legth; i++)
{
    // myTag[i].style.color = 'red'; //  错误
```

```
            this.style.color = 'red'; // this 代表当前事件的标签
        }
```

JavaScrip 面向对象

Dom

1、找到标签

获取单个元素　　document.getElementById('i1')

获取多个元素（列表）document.getElementsByTagName('div')

获取多个元素（列表）document.getElementsByClassName('c1')

a. 直接找

```
document.getElementById                根据 ID 获取一个标签
document.getElementsByName             根据 name 属性获取标签集合
document.getElementsByClassName        根据 class 属性获取标签集合
document.getElementsByTagName          根据标签名获取标签集合
```

b. 间接

```
tag = document.getElementById('i1');

parentElement                 // 父节点标签元素
children                      // 所有子标签
firstElementChild             // 第一个子标签元素
lastElementChild              // 最后一个子标签元素
nextElementtSibling           // 下一个兄弟标签元素
previousElementSibling        // 上一个兄弟标签元素
```

2、操作标签

a. innerText

获取标签中的文本内容

标签.innerText

对标签内部文本进行重新赋值

标签.innerText = ""

b. className：类属性操作

tag.className =》 直接整体做操作

tag.classList.add('样式名')　添加指定样式

tag.classList.remove('样式名')　删除指定样式

PS:

```
<div onclick='func();'>点我</div>
<script>
    function func(){
```

```
        }

    </script>
```

c. style：样式操作

```
tag.style.color = 'red';
```

d. 属性操作

```
attributes
getAttribute('');
removeAttribute('');
```

checkbox

```
获取值
checkbox 对象.checked
设置值
checkbox 对象.checked = true
```

3.创建标签并添加到 HTML 中

字符串方式：

```
var tag = '<div style='color: green';>test</div>';
document.document.getElementByld('id').insertAdjacentHTML('beforeEnd', tag);
```

对象方式：

```
var tag = document.createElement('div');
tag.innerText = "test";
t.style.color = 'green';
```

4.提交表单

5.事件

onclick，单击时触发事件
ondblclick，双击触发事件
onchange，内容修改时触发事件
onfocus，获取焦点时触发事件
onblur，失去焦点触发事件

绑定事件的三种方式：

a. test

```
function func(this){
    this.xxx // this 代表当前事件的标签
}
```

b. test

```
document.getElementByld('i1').onclick = function (){
    this.xxx // this 代表当前事件的标签
}
```

c. 事件的冒泡与捕捉

6.词法分析