JavaScript

> 和其他语言一样，它有算术运算符、逻辑运算符，和CSS一样，它拥有先后的引入顺序

基础语法

连接js文件

```
<script src="path/to/js.js"></script>
```

激活严格模式

```
'use strict';
```

简单用法

```
alert("String") //弹出警告窗口
console.log("Output log") //在控制台输出log
```

变量

> 它支持和Golang一样的显式声明变量

变量类型

- 整数
- 小数
- 字符
- 布尔
- Undefined (只声明)
- Null
- BigInt (long long int)

```
let a = 17 //int
let b = 1.1 //float
let c = false //bool
let d = 'a' //char
let e = "string" //string
```

> `let`和`var`的区别

let只支持变量进行一次声明，如果重复let就会报错，而var却可以，他们声明变量之后的默认值都是0，var是一种比较旧的声明方式，现在多用的是let

- let作用于块
- var作用于函数

条件语句

```
if (condition) {
    codeblock
} else{
    codeblock
}
```

查看变量类型

```
console.log(typeof varieties)
console.log(variable1 , variable2) //同行多个输出，可以做算术运算
```

幂运算符

```
const a = 2 ** 3;  //2*2*2 = 8
```

字符串连接

```
const year = 2024;
let birthYear = 2004;
job = 'student';
const firstName = 'Sf';
const lastName = 'Tian';
console.log(firstName + ' ' + lastName);
const jonasNew = `I'm ${firstName},
a ${year - birthYear} year old ${job}!`;
console.log(jonasNew); //注意！这里用的是反引号 ` `
console.log('hello \n\
world');//换行符\n\只能用在引号中而不是反引号中
```

类型强制转换

```
Number(variable);
Boolean(variable);
String(variable);
//除了'+'在字符串中可以做拼接把其他类型转换成字符串以外，其他的运算符均会转换成数字
```

比较运算符

> ===和==的区别

=== 是强制执行，它必须保证比较的双方的 类型和数值 是相同的才会返回 true 否则就会返回false

```
const age = 18;
if(age === 18) console.log('adult');
```

控制台输入

```
prompt("string");
const test = prompt("type a string"); //输入的类型都是字符串
const test1 = Number(prompt('type a number')); //输入的类型强制转换成数字
```

函数

```
//创建函数
function logger(){
    console.log("test");
}
function fruit(apples,oranges){
    console.log(apples, oranges);
    let juice = `Juice with ${apples} apples and ${oranges} oranges`;
    return juice; //有返回值的
}
//调用函数
logger();
fruit(5, 0);
```

箭头函数(匿名函数,没有this这个关键字)

> 更简洁的函数定义方式，可以直接返回内容不需要再写return，可以直接被调用

```
//单行定义
const calcAge = birthyear => 2037 - birthyear;
const age = calcAge(1991);
console.log(age);
//多行定义 同function yeartoretire (birthyear, firstName){}
const yeartoretire = (birthyear, firstName) => {
    const age = 2037 - birthyear;
    const retirement = 65  - age;
    return `${firstName} retires in ${retirement} years`;
}
console.log(yeartoretire(1991, 'Jonas'));
```

数组

```
const friends = ['SfTian', 'Job', 'Steve'];
const years = new Array[100](1991,1992,2000,1884,2020); //调用函数式创建
console.log(friends[0]);
const len = years.length;
const person = ['SfTian', 'Male', 2024-2004, false]; //数组可以存储不同类型的内容
friends.push('Jay');//类C++的vector队尾添加
friends.unshift('John');//在数组队头添加
friends.pop();//弹出队尾元素
friends.shift();//弹出队头元素
console.log(friends.indexOf('SfTian'));//返回指定的内容在第几个元素如果没有就返回-1
console.log(friends.includes('Jay'));//返回指定内容，如果在数组里找到了就返回true否则返回
false
//他们都不会进行类型强制转换
```

对象

```
const jonas = {
    firstName: 'SfTian',
    lastName: 'SfTian',
    age: 2024 - 2004,
    birthYear: 2004,
    job: 'Student',
    friends: ['Bob', 'Jay', 'Wills'],
    calcAge: function (){ //函数也可以做对象内容
        return 2037 - this.birthYear; //this关键字可以直接访问对象内的属性，这里等价于
jonas.birthYear
    },
    calc: function (){
        this.age = 2024-this.birthYear;
        return this.age;
    }
}
console.log(jonas.calcAge());//调用对象中的函数
console.log(jonas['calc']());
console.log(jonas.firstName); //获取对象内容，类似C语言的结构体
console.log(jonas.friends[1]); //数组只能通过这个方式来显示
console.log(jonas['firstName']); //放在方括号里可以用任何表达式而点不可以

const nameKey = 'Name'; //做拼接
console.log(jonas['first' + nameKey]); //合成之后是firstName
console.log(jonas['last' + nameKey]);
//通过输入来获取
const interested = prompt('what do you want to know about Jonas? firstName,
lastName, age, job and friends');
console.log(jonas[interested]);
//可以用在if表达式中来判断是否有真值
if(jonas[interested]){
    console.log(jonas[interested]);
} else{
    console.log("Wrong request");
}
//追加对象的定义内容
jonas.location = 'CHN';
jonas['qq'] = '22120';
```

循环结构

```
for(let i=0;i<100;i++){
    console.log('第'+i+'次');
}
//遍历数组和插入数组
const years = [1991,2007,1969,2020];
const age = [];
for (let i=0;i<years.length;i++){
    age.push(2024-years[i]);
}
console.log(age);
//continue;
```

```
//break;
let rep=1;
while(rep<=10){
    console.log(`you rep ${rep}次`);
    rep++;
}
```

可能会用的函数

```
let a=Math.random();//随机数
let b=Math.trunc();//只保留整数(不会四舍五入)
```

与HTML和CSS相配合

选择器

```
//获取元素内容
document.getElementById('id');
document.getElementsByClassName('');
document.getElementsByName('');
//选择器获取
document.querySelector('#message'); //传入一个ID是message的选择器
document.querySelector('.msg').textContent //传入一个class类名是msg的选择器的文本内容
//对其进行操作
document.querySelector('.message').textContent = 'Correct!';
//获取input框的内容
document.querySelector('.guess'.value);
//监听click事件
const displayMessage = function (message) {
  document.querySelector('.message').textContent = message;
};
document.querySelector('.click').addEventListener('click',function(){
  console.log(document.querySelector('.guess').value);
});
//获取文档中class='example'的所有元素
document.querySelectorAll('.example');
//用数组遍历被选中的元素
const qexp = document.querySelectorAll('.example');
for (let i=0;i<qexp.length;i++){
  console.log(qexp[i].textContent);
}
//管理CSS
//格式: document.querySelector('body').style.type = content;
document.querySelector('body').style.backgroundColor = '#60b347';
document.querySelector('.click').setAttribute('disabled', true);//添加标签的元素
document.querySelector('.click').removeAttribute('disabled', true);//删除标签元素
//添加class='exp'
document.querySelector('.exm').classList.add('.exp');
//删除class='exp'
document.querySelector('.exm').classList.remove('.exp');
//确认是否包含给定的类
document.querySelector('.exm').classList.contains('.exp');
//切换类，如果存在就删除，不存在就添加
```

```
document.querySelector('.exm').classList.toggle('.exp');
//替换类，把exp替换成example
document.querySelector('.exm').classList.replace('exp','example');
//监听click事件可以用以下方式实现
const closeModal = function(){
  modal.classList.add('hidden');
  overlay.classList.add('hidden');
}
btnCloseModal.addEventListener('click',closeModal);//推荐
overlay.addEventListener('click',function(){
  closeModal();
})
```

监听键盘

```
//按下任意键时
document.addEventListener('keydown', function(){

})
//查看按下的键
document.addEventListener('keydown', function(e){
  console.log(e);
  console.log(e.key);
  if(e.key === 'Escape'){
    console.log("你按下了Esc键");
  }
})
```

类目	功能
keypress	按住连续触发
keydown	按下触发
keyup	按下松开后触发

其他

```
//修改图片src
document.querySelector('.demo').src = `dice`;
```