

JavaScript

-

- `<input type = 'button' value = '[]' onclick = 'alert('OK')'>`

- ☐ ☐ ☐

```
o <script>
    // alert()
    alert('NO')
</script>
```

- □□□□□□□□□□□□□□□□

□□□□

- alert()
- console.log()
- prompt()
- document.write()

```
// alert
alert("");

// console.log
console.log("hello world");

// prompt
prompt('请输入姓名');

// document.write
document.write('<h1>hello world</h1>');
```

--	--	--	--	--	--	--

javascript

- `□□□□□//`
- `□□□□□/* */`
- `□□□□□□□□ ; □□`

10/10/2019

```
// 変数宣言
var item = 45;
let item = 45;
const item = 45;
```

5□□□□□□

1. number

2. string 字符串
3. boolean 布尔 true/false
4. undefined 未定义
5. null 空
6. object 对象
 - 普通对象
 - 数组 特殊对象

typeof 运算符

```
// 变量
var num = 20;
var string = '11';
var unData;
// 对象
var operson = {
  name: 'itcast',
  age: 12
};
// 检测
alert(typeof(num)); //number
alert(typeof(string)); //string
alert(typeof(unData)); //undefined
alert(typeof(unData)); //object

// 对象 检测
console.log(operson.name);
console.log(operson.age);
```

模板字符串

- 字符串`变量`\${}`

```
let name = 'whs';
let content = `字符串${name}`;
console.log(content) //字符串whs
```

函数

function 函数

```
// 函数
function fnAlert(){
  alert('hello!');
};
function add(num1,num2){
  var irs = num1 + num2;
  return irs
};
```

```
// 测试
fnAlert();
var result = add(3,4);
alert(result); // 测试result
```

运算符

1. 赋值

运算符	说明	示例
=	赋值	x = 8>false
===	严格相等	x === 5>true;x === "5"false

```
var score = 79;
if(score < 60){
    alert('不及格');
}else{
    alert('及格');
}
```

运算符	说明
&&	与
	或
!	非

2. 查找

方法	说明
getElementByClassName()	通过类名查找
getElementById()	通过id查找
getElementByTagName()	通过标签名查找
getElementByName()	通过name属性查找

```
// 通过类名查找
let class = document.getElementsByClassName('my_class');
// 通过id查找
let id = document.getElementById('my_id');
// 通过标签名查找
let tagname = document.getElementsByTagName('div');
// 通过name属性查找
let name = document.getElementsByName('my_name');
```

- **document**의 **getElementById** 메소드는 id 속성을 가진 html 요소를 반환한다

- onload 메소드
- onclick 메소드

```
// id를 가진 btn 요소
var oBth = document.getElementById('btn');
alert(oBth);
// onload 메소드
function fnload(){
    var oBth = document.getElementById('btn');
    alert(oBth);
}
// onload 메소드
window.onload = fnload;
// onclick 메소드
window.onload = function(){
    var oBth = document.getElementById('btn');
    alert(oBth);
}

// onclick 메소드
<input type = 'button' value = '확인' id = 'bin' onclick = 'fnload()'>
```

CSS 선택자

선택자	메소드
querySelector()	첫 번째 요소
querySelectorAll()	모든 요소

```
// class 선택자
let div = document.querySelector('div');
// class 선택자
let cla = document.querySelector('.class');
// id 선택자
let id = document.querySelector('#id');
// 모든 요소
let divs = document.querySelectorAll('div');
```

getElementById

예제

```
// getElementById
var obth = document.getElementById('bin');
// alert
```

```
obtn.value = 'user_name';
// 設置背景
obtn.style.background = 'red';

<input type = 'button' value = '提交' id = 'bin'>
```

DOM操作js

1. html元素js操作class 'className' class 'className'
2. 'style'操作font-size 'fontSize' style.fontSize

innerHTML和innerText

- **innerHTML**操作html内容
- **innerText**操作html内容

```
window.onload = function(){
    // 获取div
    var odiv = document.getElementById('mydiv');
    // 获取内容
    alert(odiv.innerHTML);
    // 设置内容
    odiv.innerHTML = '123';
}

<div id = 'mydiv'>123</div>
```

```
// 获取内容
let content = document.querySelector('.class').innerText
content = '123'

<div class = 'class'> 123 </div>
```

数组

数组的创建

- 创建

```
// 创建数组
var alist = new Array(1,2,3);

// 创建数组
var alist2 = [1,2,3,'asd'];

// 创建
var alist3 = [[1,2,3],[4,5,6]];
```

- 访问

```
var alist = [1,2,3,4];
alert(alist.length); // 4
```

- 数组长度

```
var alist = [1,2,3,4];
alert(alist[0]); // 1
```

- 数组索引

```
var alist = [1,2,3,4];
alist.push(5); // 添加
alert(alist); // 1 2 3 4 5
alist.pop(); // 删除
alert(alist) // 1 2 3 4
```

- 数组 splice
 - `arr.splice(start,num,element1,...elementN)`
 - start 开始索引
 - num 删除元素个数
 - elementN 从 start 索引开始插入的元素

start 开始索引 num 删除元素个数 elementN 从 start 索引开始插入的元素

```
// 数组 splice
var array = [10,20,30,40];
array.splice(0,1,'itcast');
console.log(array); // ['itcast',20,30,40]
```

数据类型

""+""

```
var inum1 = 10;
var inum2 = 11.1;
var str = 'abc';

result = inum1 + inum2;
alert(result); // 21.1

result = inum2 + str; // 11.1abc
alert(result); // 11.1abc
```

类型

数据类型转换

- js 数据类型转换

- func函数
- delay延迟时间0
- param1参数

函数	说明
setTimeout(func[,delay,param1,param2])	在指定的延迟时间后执行函数
setInterval(func[,delay,param1,param2])	在指定的延迟时间后，每隔指定的时间间隔重复执行函数

- 清除定时器
 - clearTimeout(timeoutID)清除定时器
 - clearInterval(timeoutID)清除定时器
 - **timeoutID**定时器ID

```
// 清除定时器
function fnshow(name){
    alert(name);
    clearTimeout(Id);
}

var Id = setTimeout(fnshow,2000,'itcast');
```

jQuery

jQuery是JavaScript的库

jQuery的使用

```
// 引入jQuery
<script src = 'jQuery'> </script>
```

jQuery的初始化

jQuery的ready事件

jQuery

```
// 初始化
$(document).ready(function(){
    ...
});

// 初始化
$(function(){
    ...
});
```

jQuery的使用

jQuery 过滤器

1. 过滤器
2. 过滤器
3. id 过滤器
4. 过滤器
5. 过滤器

```
$('#myid') // id=myid 的  
$('#.myclass') // class=myclass 的  
$('li') // li 的  
$('#ul1 li span') // id=ul1 下的 li 下的 span 的  
$('input[name=first]') // name=first 的 input 的
```

jQuery 过滤器 \$

```
$(function(){  
    var $myp = $("p");  
    alert($myp.length); // p 的数量  
});  
  
$(function(){  
    var $myobject = $("div p");  
    alert($myobject.length); // div 下的 p 的数量  
});  
  
$(function(){  
    var $myclass = $(".myclass");  
    alert($myclass.length); // class=myclass 的数量  
});
```

过滤器

- **has()** 过滤器
- **eq()** 过滤器

```
$(function(){  
    // 过滤器  
    var $myobject = $("div");  
    // 过滤器  
    $myobject.css({"height":"100px"});  
    // has  
    $myobject.has("#bin").css({"background":"red"});  
    // eq  
    $myobject.eq(0).css({"background":"red"}); // 第一个 div  
});
```

```
<div> <input type = 'button' value = '测试'> </div>  
<div> <input type = 'button' value = '测试_'> </div>
```


jQuery

jQuery 的 DOM 操作

jQuery

- `$("#box").prev();` 返回 id 为 box 的前一个元素
- `$("#box").prevAll();` 返回 id 为 box 的所有前一个元素
- `$("#box").next();` 返回 id 为 box 的下一个元素
- `$("#box").nextAll();` 返回 id 为 box 的所有下一个元素
- `$("#box").parent();` 返回 id 为 box 的父元素
- `$("#box").children();` 返回 id 为 box 的所有子元素
- `$("#box").siblings();` 返回 id 为 box 的所有兄弟元素
- `$("#box").find('.myclass');` 返回 id 为 box 的所有 class 为 myclass 的元素

```
// jQuery
var $mydiv = $('#div');
// 返回前一个元素
alert($mydiv.prev());
// 返回所有前一个元素
alert($mydiv.prevAll());
// 返回下一个元素
alert($mydiv.next());
// 返回所有下一个元素
alert($mydiv.nextAll());
// 返回父元素
alert($mydiv.parent());
// 返回所有子元素
alert($mydiv.children());
// 返回 class 为 myclass 的元素
alert($mydiv.find('.myclass'));
```

jQuery 的 DOM 操作

- **html()**
 - `.html()` 返回元素的 HTML 内容
 - `.append()` 向元素添加内容

```
// jQuery
var $mydiv = $('#div');
// 返回 HTML 内容
var $result = $mydiv.html();
alert($result);
// 向元素添加内容
$mydiv.html('');
// 向元素添加内容
$mydiv.append('');
```

jQuery

- `height()``weight()``font-size()`
 - 通过 CSS 属性来操作
- `type()``value()``id()`
 - 通过 `prop` 属性来操作

```
$(function(){
    // 获取元素
    var $my = $("p");
    // 设置样式
    var $style = $my.css('font-size');
    // 设置背景色
    $my.css({"background": 'red'});
    // 获取属性
    var $myt = $('#btn');
    // 设置属性
    var $result = $myt.prop('type');
    alert($result);
    // 设置值
    $myt.prop({'value': 'name'});
    // 获取值
    $myt.val('name');
})

<p>itcast</p>
<input type = 'button' value = '按钮' id = 'btn'>
```

jQuery

jQuery

- `click()` 单击
- `blur()` 失去焦点
- `focus()` 获得焦点
- `mouseover()` 鼠标移入
- `mouseout()` 鼠标移出
- `ready()` DOM 加载完成

```
// ready
$(function(){
    // 获取元素
    var $mytext = $('#text1');
    var $mybutton = $('#btn1');
    // click
    $mybutton.click(function(){
        alert('successfully')
    });
    // focus
    $mytext.focus(function(){
```

```

        // this은 $mytext
        $(this).css({"background": "red"});
    });
    // blur()
    $mytext.blur(function(){
        $(this).css({"background": "white"});
    });
    // mouseover()
    $mydiv = $('div');
    $mydiv.mouseover(function(){
        $(this).css({"background": "red"});
    });
    // mouseout()
    $mydiv.mouseout(function(){
        $(this).css({"background": "white"});
    });
})
<div>
    <input type = 'text' id = 'text1'>
    <input type = 'button' value = ' ' id = 'btn1'>
</div>

```

jQuery

- jQuery는 DOM(문서 객체 모델)을 조작하는 라이브러리
- jQuery는 CSS를 사용하여 스타일을 변경할 수 있다
- jQuery는 delegate를 사용하여 이벤트를 처리할 수 있다
 - delegate(childSelector, event, function)
 - childSelector: 선택자
 - event: 이벤트
 - function: 실행될 함수

```

$(function(){
    var $myul = $('ul')
    $myul.delegate('li', 'click', function(){
        $(this).css({"background": "red"})
    });
});

```

javascript

객체

- 객체는 object로 생성된다
 - `var person = new object();`

```

// 객체
person.name = 'tom';
person.age = '25';

```

```
// 测试
person.sayname = function(){
    alert(this.name);
}

// 测试
alert(person.age);
person.sayname();
```

- 对象字面量

```
◦ var person = {
    name: 'tom';
    age: '26';
    sayname: function(){
        alert(this.name);
    }
}

// 测试
alert(person.name);
person.sayname();
```

Json

JSON(JavaScript Object Notation) 一种轻量级的数据交换格式。

- 特点
 - 纯文本，json 格式的数据通过 key-value 键值对的形式表示

- 创建 JSON 对象

```
◦ {
    "name": "tom",
    "age": 18
}
```

- 解析 JSON 对象

- 使用 eval 函数

```
◦ [{"name": "tom", "age": 18}, {"name": "tom", "age": 23}]
```

-

- JSON 对象与 JavaScript 对象

- 使用 JSON.parse() 方法

- ```
var sjson = '{
 "name": "tom",
 "age": 18
}';
var json = ' [{"name": "tom", "age": 18}, {"name": "tom", "age": 23}] ';
```

```
// 將JSON字符串转换为JavaScript对象
var operson = JSON.parse(sjson);

// 将JSON字符串转换为JavaScript对象
var list = JSON.parse();
```

## ajax

- 使用JavaScript通过http向服务器发送请求并接收数据
  - 使用XMLHttpRequest对象
- **ajax**
  - 使用 \$.ajax() 方法

## Axios

安装Axios

安装Axiosjs

```
// 安装
axios({
 url: '',
 method: 'GET',
}).then((result) =>{
 console.log('成功');
}).catch((err) =>{
 console.log(err);
})

// 使用
// 使用axios.方法(url,[data[,config]])
axios.get('').then((result) =>{
 console.log('成功');
}).catch((err)=>{
 console.log(err);
})
```

## async/await

- 使用async/await
- Promise的rejected/resolved

📄

```
async search(){
 let result = await axios.get('');
 this.searchbox = result.data.data;
}
```

Vue📄📄📄

|               |     |
|---------------|-----|
| 📄             | 📄📄📄 |
| beforeCreate  | 📄📄  |
| created       | 📄📄  |
| beforeMount   | 📄📄  |
| mounted       | 📄📄📄 |
| beforeUpdate  | 📄📄📄 |
| updated       | 📄📄📄 |
| beforeUnmount | 📄📄📄 |
| unmounted     | 📄📄📄 |