# 前言

前面同学们掌握了JavaScript基础语法和程序结构，函数、数组、面向对象、事件的定义和使用与DOM操作，为了巩固和熟练前面所学的知识，接下来我们结合应用JavaScript编程技术，一起开发“网页计算机”页面。

# 知识技能篇

## 单元一 了解计算器的功能

### 学习目标

了解计算机页面包括哪些内容，界面的输入规则，各个按键之间的关系。还有怎么直接在输出文本框中输出内容并计算

### 任务1.1 了解计算机页面内容等

(1) 计算器页面包括数字按键、运算符按键、计算按键和清空按键， 以及计算区域输 出文本框，并设置计算区域输出文本框为不可编辑。

(2) 界面输入规则：输入一次数字，再输入一次运算符，然后输入一次数字，接着输入一次运算符，如此往复，形成类似"3+5x6"的形式。

(3) 单击"="按键触发JS函数， 进行计算。

(4) 计算结果显示至页面计算区域输出文本框。

(5) 单击"AC"按键清空计算区域输出文本框值。

(6) 扩展功能：将计算区域输出文本框改为可编辑文本框，可直接输入算式，如图所示， 单击"="按键使用正则表达式验证文本框输入内容并进行计算。



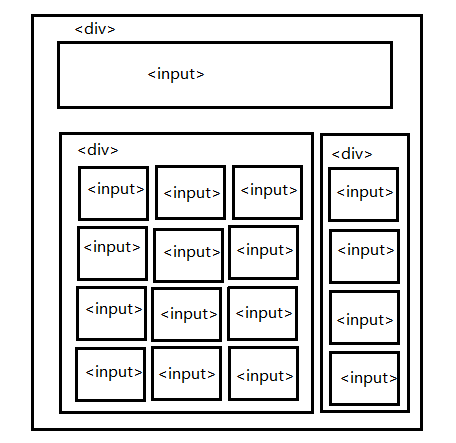
## 单元二 了解计算器的HTML和CSS布局

### 学习目标

了解计算机的页面布局，以及所触发的点击事件，函数。

### 任务2.1 页面布局情况

1．用HTML和CSS布局网页计算器的界面。（如图所示）



2. 创建JavaScript文件：index.js。

在index.js文件中创建1个calculator的对象，并定义一个“number数组”和4个计算函数。

2.1 number数组：存储输入的数字和运算符。

2.2 numberClick：输入数字函数，通过数学按键得到输入的数字，并显示到计算区域输出文本框。

2.3 operatorClick：输入运算函数，通过运算符按键得到输入的运算符，并显示到计算区域输出文本框。

2.4 equalClick：获取计算区域输出文本框数字和运算符，解析之后保存到“number数组”，计算并显示结果。

2.5 cleanClick：清空数据。

3. 为每个按键的<input>标签绑定click事件。

4. 输入数字和运算符，每次click事件，都更新计算区域输出文本框，将输入的内容添加到计算区域输出文本框中。

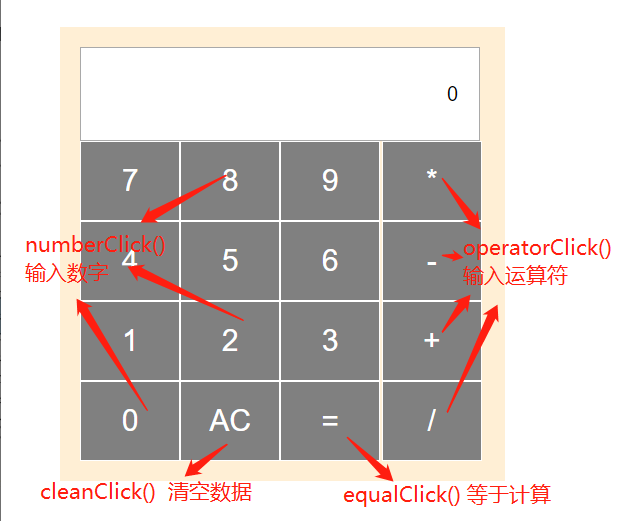
5. 编写equalClick（）计算函数，响应“=”click事件，计算步骤如下：

5.1 获取显示框字符串。

5.2 将字符串分割并赋给数组，然后对数组进行计算。

5.3 将计算结果显示在input显示框上。

各个按键对应的函数如图所示



6. 针对扩展功能，设计reg（）函数，并进行计算。

6.1当单击“=”按键时，先调用reg（）函数验证键盘输入内容的格式。

6.2 如果验证不通过，则弹出提示，并置空文本框。

6.3 如果验证通过则进行计算。

# 技能应用篇

## 单元三 完成网页计算器的开发

### 学习目标

学生独自完成一个网页计算器的开发。

### 任务3.1 HTML布局

1.创建计算器页面，命名为calculator.html。

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>计算器</title>

<link rel="stylesheet" type="text/css" href="calculator.css">

</head>

<body>

2.按照计算器样式添加<input>进行按键布局。

2.1 添加数字输入区域和文本框，文本框id属性为output,class属性为 output, 设置文本框不可编辑。

<div class="calculator">

<input class="output" value="0" id="output" disabled/>

</div>

2.2 添加运算符输入区域。

2.3 加入数字按键、 归零(AC)按键和"="按键。

<div class="numbers">

<input type="button" value="7">

<input type="button" value="8">

<input type="button" value="9">

<input type="button" value="4">

<input type="button" value="5">

<input type="button" value="6">

<input type="button" value="1">

<input type="button" value="2">

<input type="button" value="3">

<input type="button" value="0">

<input type="button" value="AC">

<input type="button" value="=">

</div>

2.4 加入运算符按键。

<div class="operators">

<input type="button" value="\*" onclick="calculator.operatorClick(value)">

<input type="button" value="-" onclick="calculator.operatorClick(value)">

<input type="button" value="+" onclick="calculator.operatorClick(value)">

<input type="button" value="/" onclick="calculator.operatorClick(value)">

</div>

### 任务3.2 CSS添加样式

1. 创建calculator.css 文件。

2. 在页面中引入 calculator.css 文件。

3. 编辑 calculator.css 文件， 为页面添加样式。

3.1 设置计算区域样式。

/\*设置计算区域的宽度、边框、背景色、边距\*/

.calculator{

width: 405px;

border: 1px solid white;

background: #ffefd5;

margin: 50px;

padding: 20px;

}

3.2 设置输出区域样式。

/\*设置计算区域文本框宽度、高度、边距、字体大小、文本右对齐、背景颜色\*/

.output{

width: 356px;

height: 50px;

padding: 20px;

font-size: 20px;

text-align: right;

background: white;

}

3.3 设置输出按键样式。

/\*设置按键样式\*/

input[type=button]{

border: 1px solid white; /\*边框\*/

width: 100px;

height: 80px;

background: grey;

cursor: pointer; /\*光标形状：手型\*/

color: white;

font-size: 30px;

}

3.4 设置输出数字样式。

/\*数字样式\*/

.numbers{

width: 300px;

/\*弹性布局\*/

display: inline-flex;

flex-wrap: wrap;

}

3.5 设置输出符号样式。

.operators{

width: 100px;

position: relative;

left: -3px;

/\*弹性布局\*/

display: inline-flex;

flex-wrap: wrap;

}

### 任务3.3 JavaScript计算

1.创建calculator.js文件。

2.导入JS文件。

<head>

<meta charset="utf-8">

<title>计算器</title>

<link rel="stylesheet" type="text/css" href="calculator.css">

<script type="text/javascript" src="calculator.js"></script>

</head>

3.创建calculator.js文件，实现计算功能。

3.1定义1个变量和1个对象，对象中包括1个计算数组和4个计算方法。

//计算对象

var calculator = {

//用于保存输入的数字和符号数据

number:[],

//计算方法

numberClick: numberClick,

operatorClick: operatorClick,

equalClick: equalClick,

cleanClick: cleanClick

};

3.2 数字按键 click 事件调用方法。

/\*---数字按键 click 事件调用方法。---\*/

//输入数字方法

var numberClick = function (value){

var val = document.getElementById("output").value;

//显示框为0时，输入0无效

if(value == "0" && val == "0"){

return;

};

if(val == "0"){

//如果显示框为0, 则去掉0, 只显示输入值

document.getElementById("output").value = value;

}else{

//在显示框显示对应字符

document.getElementById("output").value = val + value;

}

}

3.3 运算符按键 click 事件调用方法。

//输入运算符方法

var operatorClick = function(value){

var val = document.getElementById("output").value;

//判断是否连续输入了两次运算符，运算符后面输入数字，不能连续输入多个运算符

if(val[val.length - 1] == ""){

return;

// console.log("111");

}

//在显示框显示对应运算符

document.getElementById("output").value = val + " " +value+ " ";

}

3.4 调用数字和运算符按键 click 事件。

<div class="numbers">

<input type="button" value="7" onclick="calculator.numberClick(7)">

<input type="button" value="8" onclick="calculator.numberClick(8)">

<input type="button" value="9" onclick="calculator.numberClick(9)">

<input type="button" value="4" onclick="calculator.numberClick(4)">

<input type="button" value="5" onclick="calculator.numberClick(5)">

<input type="button" value="6" onclick="calculator.numberClick(6)">

<input type="button" value="1" onclick="calculator.numberClick(1)">

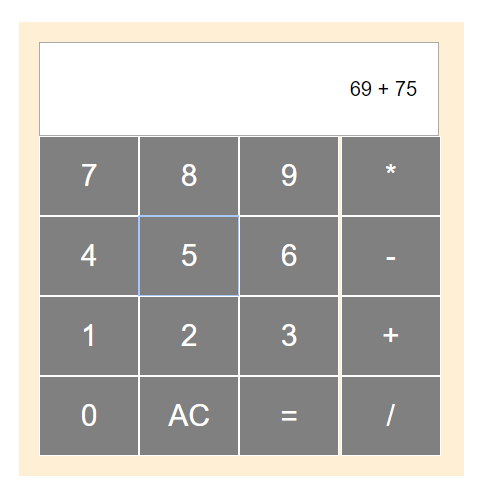
<input type="button" value="2" onclick="calculator.numberClick(2)">

<input type="button" value="3" onclick="calculator.numberClick(3)">

<input type="button" value="0" onclick="calculator.numberClick(0)">

</div>

3.5 输入算式，效果如图所示。



3.6 计算并显示结果

//计算并显示结果

var equalClick = function(){

//分割算术数组

this.number = document.getElementById("output").value.split(" ");

console.log(this.number);

//计算乘除

for (var index = 0; index < this.number.length; index++) {

//若输入的字符最后为 “乘”或“除“ 运算符， 则在最后面加1

if(this.number[index + 1] == ""){

this.number[index + 1] = 1;

}

if(this.number[index] == "\*") {

//删除数组内已计算数字， 并添加计算后数字

var index\_num = Number(index);

console.log(index\_num);

var firstNum = Number(this.number[index\_num - 1]);

var secondNum = Number(this.number[index\_num + 1]);

var result = firstNum \* secondNum;

this.number.splice(index\_num - 1, 3,result);

}else if(this.number[index] == "/") {

//删除数组内已计算数字， 并添加计算后数字

var index\_num = Number(index);

var firstNum = Number(this.number[index\_num - 1]);

var secondNum = Number(this.number[index\_num + 1]);

var result= firstNum / secondNum;

this.number.splice(index\_num - 1, 3, result);

}

}

//计算加减

for(var index = 0; index < this.number.length; index++) {

if(this.number[index] == "+" || this.number[index] == "-") {

if(this.number[index] == "+") {

//删除数组内巳计算数字，并添加计算后数字

var index\_num = Number (index);

var firstNum = Number(this.number[index\_num - 1]);

var secondNum = Number(this.number[index\_num + 1]);

var result= firstNum + secondNum;

this.number.splice(index\_num - 1, 3, result);

}else if(this.number[index] == "-") {

//删除数组内巳计算数字，并添加计算后数字

var index\_num = Number (index);

var firstNum = Number(this.number[index\_num - 1]);

var secondNum = Number (this.number[index\_num + 1]);

var result = firstNum - secondNum;

this.number.splice(index\_num - 1, 3, result);

}

}

}

document.getElementById("output").value = this.number[0];

}

3.7 清空计算。

//清空计算

//清空数据

var cleanClick=function(){

document.getElementById("output") .value = "";

}

3.8 "="按键和清空按键注册click事件。

<input type="button" value="AC" onclick="calculator.cleanClick()">

<input type="button" value="=" onclick="calculator.equalClick()">

### 任务3.4 扩展功能（验证正则表达式）

1.计算器的文本框可以手动输入运算表达式，根据表达式进行计算。

<input class="output" value="0" id="output" onblur="fn()" />

2.编写正则表达式。

2.1 计算区域输入格式的要求主要包括以下几点。

2.2 开头为数字（数字至少为一位），对应的正则表达式为 "\\d+"。

2.3 接下来为一个运算符("+"或"-"或"\*"或"/")和一个数字（数字至少为一位），对应的正则表达式为 "[+\*/-][\\d](file:///\\d)+"。

2.4 上一条要求中的符号和数字组合可以出现多次， "([+\*/-][\\d+)](file:///\\d+))+"。

2.5正则表达式以"^"开头，以"$"结尾，结合第一点要求和第三点要求中的表达式， 最终的正则表达式为 "^\\d+([+\*/-]\\ct+)+$"。

3.应用正则表达式进行验证。

3.1在fn（）函数中使用正则表达式验证文本框输入内容是否为运算表达式格式：如果是，则单击"="按键计算结果；反之，则重新输入。

var fn = function(){

var val = document.getElementById("output").value;

var reg = new RegExp("^\\d+([+\*/-]\\d+)+$");

if(!reg.test(val)){ //如果验证不通过， 则弹出提示，并置空文本框

alert("请输入正确的计算表达式");

document.getElementById("output").value = "";

return false;

}else{

//如果验证通过，则进行计算

//获取运算符

var reg1 = /[+\*/-]/g;

var str = (val.match(reg1));

//获取数字

var reg2 = /\d+/g;

var str2 = (val.match(reg2));

var str1 = [];

var res = "";

//在运算符和数字之间加入一个空格符号

for(var i = 0; i < str.length; i++){

str1[i] = " " + str[i] + " ";

res += str2[i] + str1[i];

}

var res1 = res + str2[str2.length-1];

document.getElementById("output").value = res1;

}

}

3.2正则表达式验证不通过的运行效果如图所示。

