第 3 章 ES6（网页计算器）

3.1 实验目标

（1）掌握 ES6 变量和常量的用法。
（2）掌握 ES6 Class 类的定义、声明等基本方法。
（3）掌握 ES6 数组的基本用法。
（4）掌握 ES6 数值的基本用法。
（5）掌握 ES6 函数的基本用法。
（6）综合应用 ES6 新规范，开发“网页计算器”。
知识地图如图 3-1 所示。

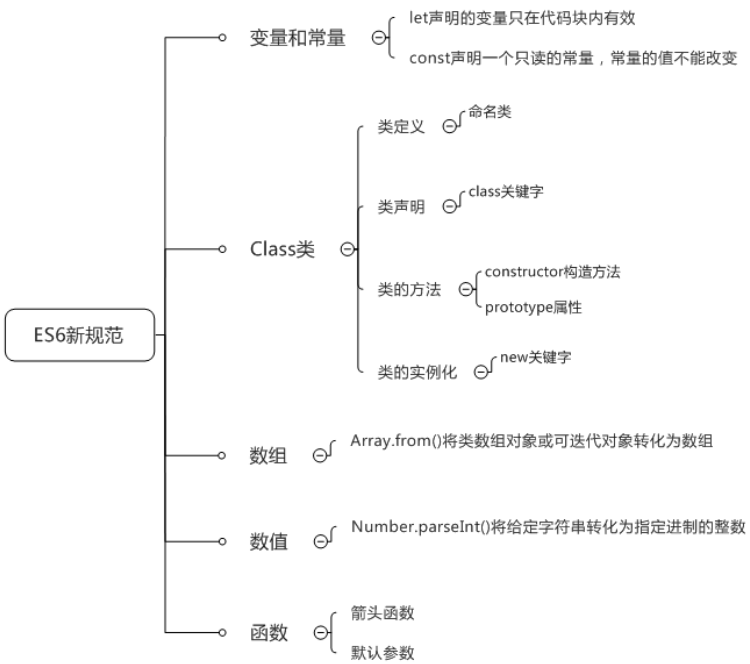

图 3-1

3.2 实验任务

构建网页计算器页面，实现简单的四则运算，运算逻辑使用 ES6 语法实现。

（1）制作网页计算器页面，如图 3-2 所示。

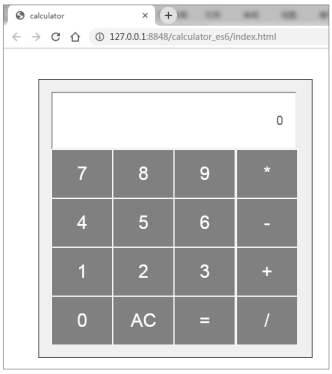

图 3-2

（2）输入：单击数字和符号，相应的数字和符号会展示在显示区，如图 3-3 所示。

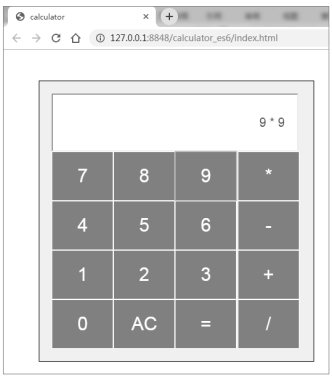

图 3-3

（3）计算：单击“=”，会对显示区中的算术式进行计算，并将计算后的结果展示在显示区，如图 3-4 所示。

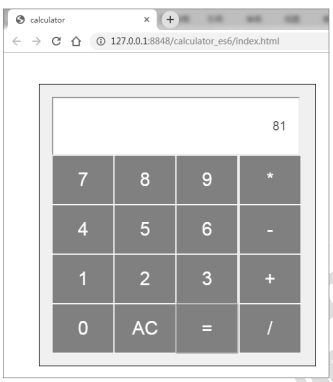

图 3-4

（4）清空：单击“AC”，会清空显示区，如图 3-5 所示。

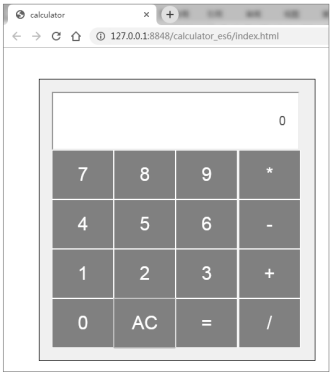

图 3-5

3.3 设计思路

（1）创建工程 calculator，在工程目录中创建 index.html 文件、index.css 文件和 index.js 文件，如表 3-1 所示。

表 3-1

类　型	文　件	说　明
html 文件	index.html	网页计算器主页面文件
css 文件	index.css	网页计算器主页面样式文件
js 文件	index.js	网页计算器类和功能逻辑文件

（2）页面结构：在 index.html 文件中编写网页计算器主页面，创建 0～9、+、−、*、/、=和 AC 按钮。

（3）页面样式：在 index.css 文件中编写网页计算器页面样式，设计显示区、功能按钮等样式。

（4）在 index.js 文件中，用 JavaScript 创建实例对象，命名为 calculator。

- constructor()：构造函数，默认参数为 null，给 number 和 result 赋值。
- number：实例属性，为一维数组，用于存放算术式，数组元素为数字和符号。
- result：实例属性，用于存放计算结果，数组元素为数字。
- compute()：计算方法。
- back()：返回计算结果的方法，此方法使用 prototype 属性创建。

（5）在 index.js 文件中创建如下方法。

- 创建 get()和 set()方法，get()方法用于获取显示区中的算术式，set()方法用于将值输出到显示区中。
- 创建 numberClick()方法，给数字 0～9 绑定 numberClick()方法，单击数字，相应的数字会输出到显示区中。
- 创建 operatorClick()方法，给符号“+”“−”“*”“/”绑定 operatorClick()方法，单击符号，相应的符号会输出到显示区中。
- 创建 cleanClick()方法，给“AC”绑定 cleanClick()方法，单击“AC”，会清空显示区。

- 创建 equalClick()方法，给符号“=”绑定 equalClick()方法，单击“=”，会调用 get()方法获取显示区中的算术式，并判断算术式是否为空，若为空则结束操作，否则使用 new 创建 calculator 对象，并将算术式作为构造函数的参数，通过 calculator 对象调用 compute()方法进行计算，调用 back()方法获取计算后的值，使用 Number.parseInt()方法对值取整，最后调用 set()方法将值输出到显示区中。

3.4 实验实施（跟我做）

3.4.1 步骤一：创建项目和文件

1．创建项目

项目名称为 calculator。

2．创建文件

- index.html：网页计算器主页面文件。
- index.css：网页计算器主页面样式文件。
- index.js：网页计算器类和功能逻辑文件。

目录结构如图 3-6 所示。

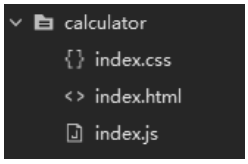

图 3-6

3.4.2 步骤二：制作 HTML 页面

引入 index.css 文件和 index.js 文件，在 index.html 文件的<body>标签中编写按钮<input>、输入框<input>等标签，并绑定相应的 onClick 点击事件，制作 HTML 页面的代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>calculator</title>
        <link rel="stylesheet" type="text/css" href="index.css">
        <script type="text/javascript" src="index.js"></script>
    </head>
    <body>
        <div class="calculator">
            <input class="output" value="0" id="iputNum"
disabled="disabled"></input>
        <div class="numbers">
            <input type="button" value="7" onclick="numberClick(value)">
            <input type="button" value="8" onclick="numberClick(value)">
            <input type="button" value="9" onclick="numberClick(value)">
            <input type="button" value="4" onclick="numberClick(value)">
```

```
            <input type="button" value="5" onclick="numberClick(value)">
            <input type="button" value="6" onclick="numberClick(value)">
            <input type="button" value="1" onclick="numberClick(value)">
            <input type="button" value="2" onclick="numberClick(value)">
            <input type="button" value="3" onclick="numberClick(value)">
            <input type="button" value="0" onclick="numberClick(value)">
            <input type="button" value="AC" onclick="cleanClick(value)">
            <input type="button" value="=" onclick="equalClick()">
        </div>
        <div class="operators">
            <input type="button" value="*" onclick="operatorClick(value)">
            <input type="button" value="-" onclick="operatorClick(value)">
            <input type="button" value="+" onclick="operatorClick(value)">
            <input type="button" value="/" onclick="operatorClick('/')">
        </div>
    </div>
    </body>
</html>
```

3.4.3 步骤三：制作 CSS 样式

在 index.css 文件中编写页面标签的样式，制作 CSS 样式的代码如下：

```
.calculator {
    width: 405px;
    border: solid 1px;
    background: #ffefd5;
    margin: 50px;
    padding: 20px;
}
.output {
    width: 356px;;
    padding: 20px;
    height: 50px;
    font-size: 20px;
    text-align: right;
    background: white;
}
.numbers {
    width: 300px;
    display: -webkit-inline-box;
    display: -ms-inline-flexbox;
    display: inline-flex;
    -ms-flex-wrap: wrap;
    flex-wrap: wrap;
}
input[type=button] {
    border: solid 1px white;
    background: none;
    width: 100px;
    height: 80px;
    background: grey;
    cursor: pointer;
    color: white;
    font-size: 30px;
}
```

```
.operators {
    display: -webkit-inline-box;
    display: -ms-inline-flexbox;
    display: inline-flex;
    width: 100px;
    -ms-flex-wrap: wrap;
    flex-wrap: wrap;
    position: relative;
    left: -3px;
}
```

3.4.4 步骤四：编写网页计算器类

在 index.js 文件中编写如下代码：

```
class calculator {

        constructor(value = null){
            //分割算术数组
            this.number = value;
            this.result = 0;
        }
        compute() {
            //计算乘除
            //计算加减
        }
}
```

（1）类声明：声明类 calculator。

（2）构造器：使用构造器初始化 number 和 result 属性。

（3）默认参数：给 value 属性设置默认参数为 null。

3.4.5 步骤五：编写乘除运算逻辑

在 index.js 文件中编写如下代码：

```
this.result = Array.from(this.number);
//计算乘除
for(let index = 0; index < this.result.length; index++){
    if(this.result[index] == "*" || this.result[index] == "/"){
        //若最后输入的字符为运算字符，默认在最后加上 1
        if(this.result[index + 1] == ""){
            this.result[index + 1] = 1;
        }
        if(this.result[index] == "*"){
            //删除数组内已计算的数字，并添加计算后的数字
            this.result.splice(+index - 1, 3, +this.result[index - 1] *
+this.result[+index + 1]);
        }else if(this.result[index] == "/"){
            //删除数组内已计算的数字，并添加计算后的数字
            this.result.splice(+index - 1, 3, +this.result[index - 1] / +this.
result[+index + 1]);
        }
        index--;
    }
}
```

3.4.6 步骤六：编写加减运算逻辑

在 index.js 文件中编写如下代码：

```
//计算加减
for(let index = 0; index < this.result.length; index++){
    if(this.result[index] == "+" || this.result[index] == "-"){
        if(this.result[index] == "+"){
            //删除数组内已计算的数字，并添加计算后的数字
            this.result.splice(+index - 1, 3, +this.result[index - 1] +
+this.result[+index + 1]);
        }else if(this.result[index] == "-"){
            //删除数组内已计算的数字，并添加计算后的数字
            this.result.splice(+index - 1, 3, +this.result[index - 1] -
+this.result[+index + 1]);
        }
        index--;
    }
}
```

3.4.7 步骤七：编写数据返回方法

使用 prototype 属性添加 back()方法，返回计算结果，在 index.js 文件中编写如下代码：

```
/* 添加方法 */
Object.assign(calculator.prototype,{
    back() {
        return this.result;
    }
})
```

3.4.8 步骤八：页面交互逻辑

在 index.js 文件中编写代码。

（1）编写 get()和 set()方法，分别用于获得输入框的值和给输入框赋值，代码如下：

```
/* 获得输入框的值 */
const get = ()=>{
    return document.getElementById("iputNum").value;
}
/* 给输入框赋值 */
const set = (value)=>{
    document.getElementById("iputNum").value = value;
}
```

（2）编写 numberClick()和 operatorClick()方法，分别用于输入数字和输入字符时的交互逻辑。针对输入数字时的交互逻辑，编写代码如下：

```
/* 输入数字 */
function numberClick(value) {
    let val = get();
    //显示框为 0 时，输入 0 无效
    if(value == "0" && val == "0"){
        return;
```

```
    }
    if(val == "0"){
        //计算结果为 0 时，删除 0
        set(value);
    }else{
        //在显示框显示对应字符
        set(val + value);
    }
}
```

针对输入字符时的交互逻辑，编写代码如下：

```
/* 输入字符 */
function operatorClick(value) {
    let val = get();
    //不可连续输入运算字符
    if(val[val.length - 1] == " "){
        return;
    }
    //在显示框显示对应字符
    set(val + " " + value + " ");
}
```

（3）清空输入框。

单击“AC”，将输入框的内容置为 0，代码如下：

```
/* 清空数据 */
function cleanClick() {
    set("0");
}
```

（4）运算。

在运算时，先判断输入框是否为空，若为空则结束操作，否则实例化类，给 number 属性赋值，调用 compute()方法进行运算，调用 back()方法获取计算结果，并通过 Number.parseInt()方法将运算后的值取整，最后调用 set()方法将计算结果输出到显示区，代码如下：

```
/* 计算 */
function equalClick() {
    if(get() == ""){
        return;
    }else{
        let cal = new calculator();
        cal.number = get().split(' ');
        cal.compute();
        set(Number.parseInt(cal.back()));
    }
}
```

3.4.9 步骤九：运行效果

（1）访问网页计算器页面，如图 3-7 所示。

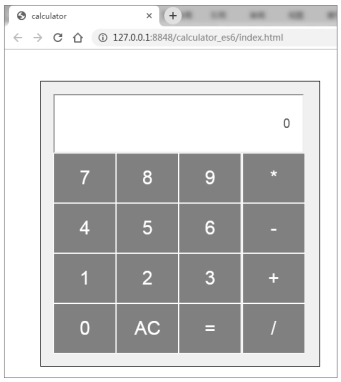

图 3-7

（2）输入数字和符号，如图 3-8 所示。

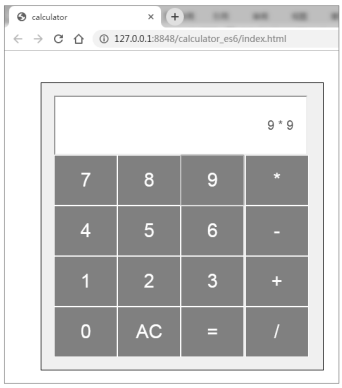

图 3-8

（3）单击“=”，如图 3-9 所示。

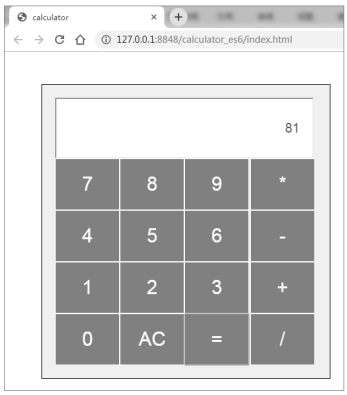

图 3-9

（4）单击“AC”，如图 3-10 所示。

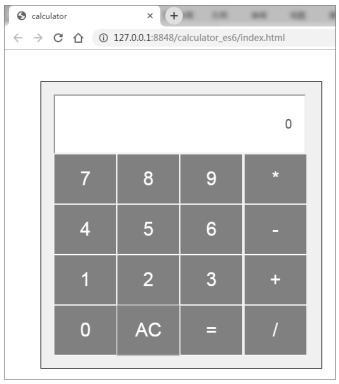

图 3-10