

# 实 验 报 告

(与程序设计有关)

课程名称: Web 前端开发技术

实验题目: 实验一 Web 基础实验

班级学号: 2203050320

姓 名: 闻家尉

成 绩: \_\_\_\_\_

沈 阳 理 工 大 学

2023 年 3 月 25 日

## 实验目的及要求：

学习并掌握 html 和 css 的部分功能，并完成实验一网页设计和骰子实验

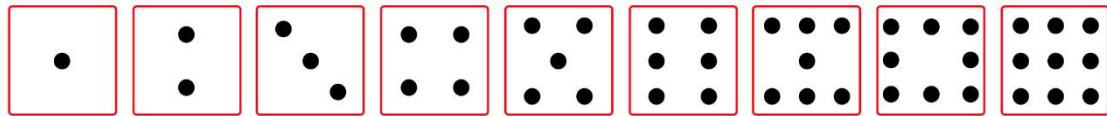
## 软硬件环境：

- 1、硬件环境：PC 机一台
- 2、软件环境：Windows 7/8/10、Visual Studio Code

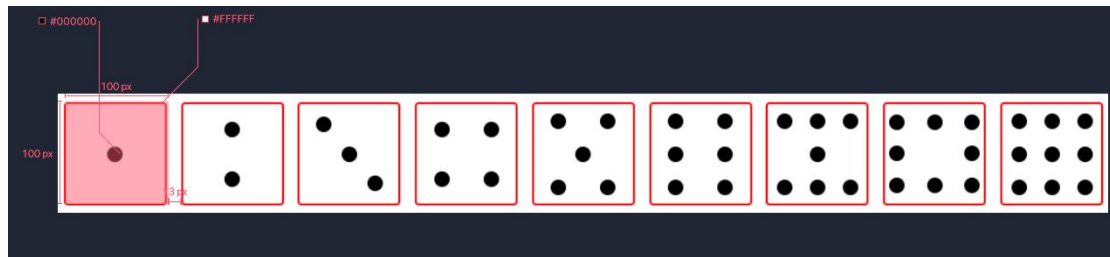
## 算法或原理分析（实验内容）：

实验一：网页设计主要针对我们对 HTML 和 css 联合使用时对 css 控制 HTML 的程序表现，用 css 来控制

实验二：根据试题需求中要求的属性进行页面布局，无需过多样式设置。按以上要求完成以下效果（页面整体宽度 1024px）：



关键尺寸批注如下：（可以通过在图片上右键-》“在新标签页打开图片”，或在右键-》复制图片地址后到浏览器新标签页打开并放大图片，查看关键尺寸）



## 程序代码或实现过程：

### 实验一：

```
1 // 实验一：实现一个简单的计算器
2 #include <iostream>
3 #include <string>
4 #include <vector>
5 #include <stack>
6 #include <map>
7 #include <algorithm>
8 using namespace std;
9
10 // 中缀表达式转后缀表达式
11 string infixToPostfix(string infix) {
12     stack<char> s;
13     string postfix;
14     map<char, int> precedence;
15     precedence['+'] = 1;
16     precedence['-'] = 1;
17     precedence['*'] = 2;
18     precedence['/'] = 2;
19     precedence['('] = 0;
20     precedence[')'] = 3;
21
22     for (int i = 0; i < infix.length(); i++) {
23         char c = infix[i];
24         if (c == '(') {
25             s.push(c);
26         } else if (c == ')') {
27             while (s.top() != '(') {
28                 postfix += s.top();
29                 s.pop();
30             }
31             s.pop();
32         } else if (c == '+' || c == '-' || c == '*' || c == '/') {
33             while (!s.empty() && precedence[s.top()] >= precedence[c]) {
34                 postfix += s.top();
35                 s.pop();
36             }
37             s.push(c);
38         }
39     }
40     while (!s.empty()) {
41         postfix += s.top();
42         s.pop();
43     }
44     return postfix;
45 }
46
47 // 后缀表达式求值
48 double postfixEval(string postfix) {
49     stack<double> s;
50     for (int i = 0; i < postfix.length(); i++) {
51         char c = postfix[i];
52         if (c == '+' || c == '-' || c == '*' || c == '/') {
53             double b = s.top(); s.pop();
54             double a = s.top(); s.pop();
55             double res;
56             if (c == '+') res = a + b;
57             if (c == '-') res = a - b;
58             if (c == '*') res = a * b;
59             if (c == '/') res = a / b;
60             s.push(res);
61         } else {
62             s.push(c - '0');
63         }
64     }
65     return s.top();
66 }
```


### 实验二：

```
1 // 实验二：实现一个简单的表达式求值器
2 #include <iostream>
3 #include <string>
4 #include <vector>
5 #include <stack>
6 #include <map>
7 #include <algorithm>
8 using namespace std;
9
10 // 中缀表达式转后缀表达式
11 string infixToPostfix(string infix) {
12     stack<char> s;
13     string postfix;
14     map<char, int> precedence;
15     precedence['+'] = 1;
16     precedence['-'] = 1;
17     precedence['*'] = 2;
18     precedence['/'] = 2;
19     precedence['('] = 0;
20     precedence[')'] = 3;
21
22     for (int i = 0; i < infix.length(); i++) {
23         char c = infix[i];
24         if (c == '(') {
25             s.push(c);
26         } else if (c == ')') {
27             while (s.top() != '(') {
28                 postfix += s.top();
29                 s.pop();
30             }
31             s.pop();
32         } else if (c == '+' || c == '-' || c == '*' || c == '/') {
33             while (!s.empty() && precedence[s.top()] >= precedence[c]) {
34                 postfix += s.top();
35                 s.pop();
36             }
37             s.push(c);
38         }
39     }
40     while (!s.empty()) {
41         postfix += s.top();
42         s.pop();
43     }
44     return postfix;
45 }
46
47 // 后缀表达式求值
48 double postfixEval(string postfix) {
49     stack<double> s;
50     for (int i = 0; i < postfix.length(); i++) {
51         char c = postfix[i];
52         if (c == '+' || c == '-' || c == '*' || c == '/') {
53             double b = s.top(); s.pop();
54             double a = s.top(); s.pop();
55             double res;
56             if (c == '+') res = a + b;
57             if (c == '-') res = a - b;
58             if (c == '*') res = a * b;
59             if (c == '/') res = a / b;
60             s.push(res);
61         } else {
62             s.push(c - '0');
63         }
64     }
65     return s.top();
66 }
```

```
1 // 实验二：实现一个简单的表达式求值器
2 #include <iostream>
3 #include <string>
4 #include <vector>
5 #include <stack>
6 #include <map>
7 #include <algorithm>
8 using namespace std;
9
10 // 中缀表达式转后缀表达式
11 string infixToPostfix(string infix) {
12     stack<char> s;
13     string postfix;
14     map<char, int> precedence;
15     precedence['+'] = 1;
16     precedence['-'] = 1;
17     precedence['*'] = 2;
18     precedence['/'] = 2;
19     precedence['('] = 0;
20     precedence[')'] = 3;
21
22     for (int i = 0; i < infix.length(); i++) {
23         char c = infix[i];
24         if (c == '(') {
25             s.push(c);
26         } else if (c == ')') {
27             while (s.top() != '(') {
28                 postfix += s.top();
29                 s.pop();
30             }
31             s.pop();
32         } else if (c == '+' || c == '-' || c == '*' || c == '/') {
33             while (!s.empty() && precedence[s.top()] >= precedence[c]) {
34                 postfix += s.top();
35                 s.pop();
36             }
37             s.push(c);
38         }
39     }
40     while (!s.empty()) {
41         postfix += s.top();
42         s.pop();
43     }
44     return postfix;
45 }
46
47 // 后缀表达式求值
48 double postfixEval(string postfix) {
49     stack<double> s;
50     for (int i = 0; i < postfix.length(); i++) {
51         char c = postfix[i];
52         if (c == '+' || c == '-' || c == '*' || c == '/') {
53             double b = s.top(); s.pop();
54             double a = s.top(); s.pop();
55             double res;
56             if (c == '+') res = a + b;
57             if (c == '-') res = a - b;
58             if (c == '*') res = a * b;
59             if (c == '/') res = a / b;
60             s.push(res);
61         } else {
62             s.push(c - '0');
63         }
64     }
65     return s.top();
66 }
```

结果分析：

返回



### 请登录

用户名

密码

登录 重置

注册 | 忘记密码

1.

2.

教师签字	冯杜	日期	
------	----	----	--