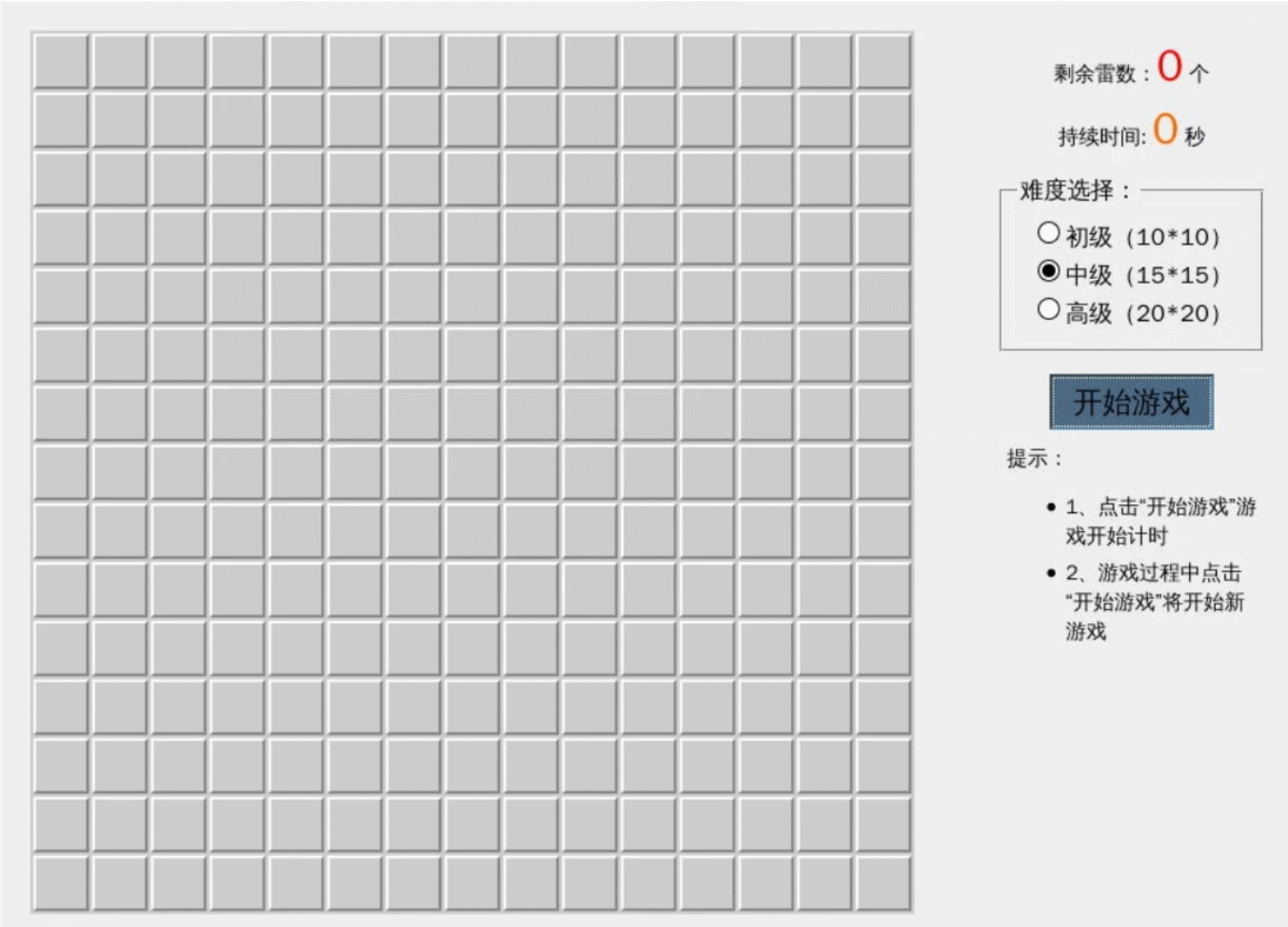
在线实验，请到PC端体验

网页版扫雷

一、实验介绍

1.1 实验内容

相信大家都玩过扫雷这个经典的小游戏，它规则简单但耐玩。你有没有想过自己动手开发一个呢？今天我们来做一个网页版的扫雷，先上一张效果截图：



1.2 实验知识点

- HTML/CSS
- JavaScript

1.3 实验环境

本实验环境采用带桌面的 Ubuntu Linux 环境，实验中会用到桌面上的程序：

1. Xfce 终端: Linux 命令行终端，打开后会进入 Bash 环境，可以使用 Linux 命令
2. Firefox：浏览器，可以用在需要前端界面的课程里，只需要打开环境里写的 HTML/JS 页面即可
3. GVim：非常好用的编辑器，最简单的用法可以参考课程 Vim编辑器 (http://www.shiyanlou.com/courses/2)

1.4 适合人群

本课程难度中等，需要你有一定的前端基础(HTML+CSS+JavaScript).

动手实践是学习 IT 技术最有效的方式！ 开始实验

1.5 代码获取

```
$ git clone https://github.com/shiyanlou/js-minesweeper
```

二、实验原理

在开始开发之前，我们先来设计一下游戏算法。

扫雷游戏的规则很简单：

> 游戏面板上有一些格子，每个格子中有一个数字（空白表示数字为 0）或是地雷，格子中的数字表示格子周围格子中地雷的数量。玩家要做的就是把数字格子找出来，时间花的越少越好。

除边界上的格子外，每个格子周围有 8 个格子：上、下、左、右、4 个斜角。所以数字范围是 0~8。

所以我们的算法如下:

> 根据用户选择的难易程度（有初、中、高三个级别，级别越高地雷和格子数量越多），随机产生一定个数的地雷并随机放在格子中。然后遍历格子，计算每个格子中的数字，标记在格子上。玩家左键点击格子时显示格子内容（如果遇到地雷则挑战失败，游戏结束），右键点击格子时标记格子为地雷，真到正确标记所有地雷并打开所有非地雷格子，挑战成功，游戏结束。

小技巧：由于格子中数字范围是 0~8，所以为了便于计算，我们可以把地雷所在的格子中的数字记为9。

三、开发准备

先建立下面的目录结构：

```
minesweeper
    |__index.html
     |__index.css
    |__index.js
     |__jms.js
```

然后我们需要在项目目录中放入实验要用到的地雷和旗帜的图片， 通过下面的命令下载图片：

```
$ wget http://labfile.oss.aliyuncs.com/courses/144/mine.png
$ wget http://labfile.oss.aliyuncs.com/courses/144/flag.png
```

四、实验步骤

4.1 页面布局

首先我们需要有一个面板来显示游戏信息，包括剩余地雷个数、所用时间、难度级别等。因为格子数量不是固定的，所以我们先不画格子，放在 JS 代码中绘制。

创建 index.html 文件

```
$ vim index.html
```

添加如下代码并保存：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>JavaScript版扫雷</title>
    <link rel="stylesheet" type="text/css" href="index.css"/>
</head>
<body>
    <div id="JMS_main" class="main">
        <table id="landmine"></table>
        <div id="operation">
            <div class="tip">剩余雷数：<span class="light red" id="landMineCount">0</span> 个</div>
            <div class="tip">持续时间：<span class="light f60" id="costTime">0</span> 秒</div>
            <fieldset>
                <legend>难度选择：</legend>
                <input type="radio" name="level" id="llevel" checked="checked" value="10" /><label
for="llevel">初级（10*10）</label><br />
                <input type="radio" name="level" id="mlevel" value="15" /><label for="mlevel">中级
（15*15）</label><br />
                <input type="radio" name="level" id="hlevel" value="20" /><label for="hlevel">高级
（20*20）</label><br />
            </fieldset>
            <input type="button" id="begin" value="开始游戏" /><br />
            <div class="tip txtleft">提示：
                <ul>
                    <li>1、点击“开始游戏”游戏开始计时</li>
                    <li>2、游戏过程中点击“开始游戏”将开始新游戏</li>
                </ul>
            </div>
        </div>
    </div>

    <script src="jms.js"></script>
    <script src="index.js"></script>
</body>
</html>
```


然后需要调整面板上游戏信息的位置，添加一些样式，在 index.css 中添加如下代码并保存：

```
.main {
    margin:10px auto;
    padding:20px;
    background:#EEE;
    width:600px;
    zoom:1;
}
.main table {
    background:#CCC;
    float:left;
}

.main table td {
    border:2px outset #EEE;
    font-size:20px;
    width:32px;
    height:32px;
    text-align:center;
    cursor:pointer;
}

.main table td:hover {
    background-color:#AAA;
}
.main #operation {
    width:180px;
    float:right;
    text-align:center;
}

.landMine {
    background-image:url(mine.png);
    background-position:center;
    background-repeat:no-repeat;
}

.main table td.normal {
    border:2px solid #EEE;
    background-color:#AAA;
}

.main table td.normal:hover {
    background-color:#AAA;
}

.flag {
    background-image:url(flag.png);
    background-position:center;
    background-repeat:no-repeat;
}

.main:after {
    clear: both;
    display: block;
    content: "";
    line-height: 0;
    height: 0;
    visibility:hidden;
}
.main .tip {
    font-size:14px;
    margin:5px;
}

.main .tip ul {

}

.main .tip ul li {
    margin:5px 0;
    line-height:20px;
}
.main .light{
    font-size:30px;
}
.main .red {
    color:red;
```

```
}
.main .f60 {
    color:#F60;
}
.main input[type=button] {
    padding:2px 10px;
    margin:5px;
    font-size:20px;
    cursor:pointer;
}
.main .txtleft {
    text-align:left;
}

.main input[type='radio'],
.main fieldset label {
    cursor:pointer;
}

.main fieldset {
    margin:10px 0;
    line-height:25px;
}
```

动手实践是学习 IT 技术最有效的方式！ 开始实验

完成这步后，在浏览器中打开 index.html，效果如下：

左边空白处用于显示格子。

4.2绘制格子

完成上面的步骤后，下面就要画格子了，为了让代码更清晰，我们把游戏实现部分和调用部分分开，游戏实现部分放在跟 index.html 同目录下的 jms.js 中，游戏调用部分放在同目录下的 index.js 中。

画格子需要传入一些参数，如放格子的表格的id，格子的数量（用行数和列数表示），另外，游戏的其他数据也要进行初始化。

在 `jms.js` 中添加如下代码并保存：

```
//在jms.js中
(function () {
    var JMS = function (id,rowCount,colCount, minLandMineCount, maxLandMineCount) {
        if (!(this instanceof JMS)) {
            return new JMS(id, rowCount, colCount, minLandMineCount, maxLandMineCount);
        this.doc = document;
        this.table = this.doc.getElementById(id);//画格子的表格
        this.cells = this.table.getElementsByTagName("td");//小格子
        this.rowCount = rowCount || 10;//格子行数
        this.colCount = colCount || 10;//格子列数
        this.landMineCount = 0;//地雷个数
        this.markLandMineCount = 0;//标记的地雷个数
        this.minLandMineCount = minLandMineCount || 10;//地雷最少个数
        this.maxLandMineCount = maxLandMineCount || 20;//地雷最多个数
        this.arrs = [];//格子对应的数组
        this.beginTime = null;//游戏开始时间
        this.endTime = null;//游戏结束时间
        this.currentSetpCount = 0;//当前走的步数
        this.endCallBack = null;//游戏结束时的回调函数
        this.landMineCallBack = null;//标记为地雷时更新剩余地雷个数的回调函数
        this.doc.oncontextmenu = function () {//禁用右键菜单
            return false;
        };
        this.drawMap();
    };

    JMS.prototype = {
        //画格子
        drawMap: function () {
            var tds = [];
            if (window.ActiveXObject && parseInt(navigator.userAgent.match(/msie ([\d.]+)/i)[1]) < 8) {
                var css = '#JMS_main table td{background-color:#888;}',
                    head = this.doc.getElementsByTagName("head")[0],
                    style = this.doc.createElement("style");
                style.type = "text/css";
                if (style.styleSheet) {
                    style.styleSheet.cssText = css;
                } else {
                    style.appendChild(this.doc.createTextNode(css));
                }
                head.appendChild(style);
            }
            for (var i = 0; i < this.rowCount; i++) {
                tds.push("<tr>");
                for (var j = 0; j < this.colCount; j++) {
                    tds.push("<td id='m_" + i + "_" + j + "'></td>");
                }
                tds.push("</td>");
            }
            this.setTableInnerHTML(this.table, tds.join(""));
        },
        //添加HTML到Table
        setTableInnerHTML: function (table, html) {
            if (navigator && navigator.userAgent.match(/msie/i)) {
                var temp = table.ownerDocument.createElement('div');
                temp.innerHTML = '<table><tbody>' + html + '</tbody></table>';
                if (table.tBodies.length == 0) {
                    var tbody = document.createElement("tbody");
                    table.appendChild(tbody);
                }
                table.replaceChild(temp.firstChild.firstChild, table.tBodies[0]);
            } else {
                table.innerHTML = html;
            }
        }
    };

    window.JMS = JMS;
})();
```


上面的代码中，部分代码是为了兼容 IE 浏览器，可忽略。

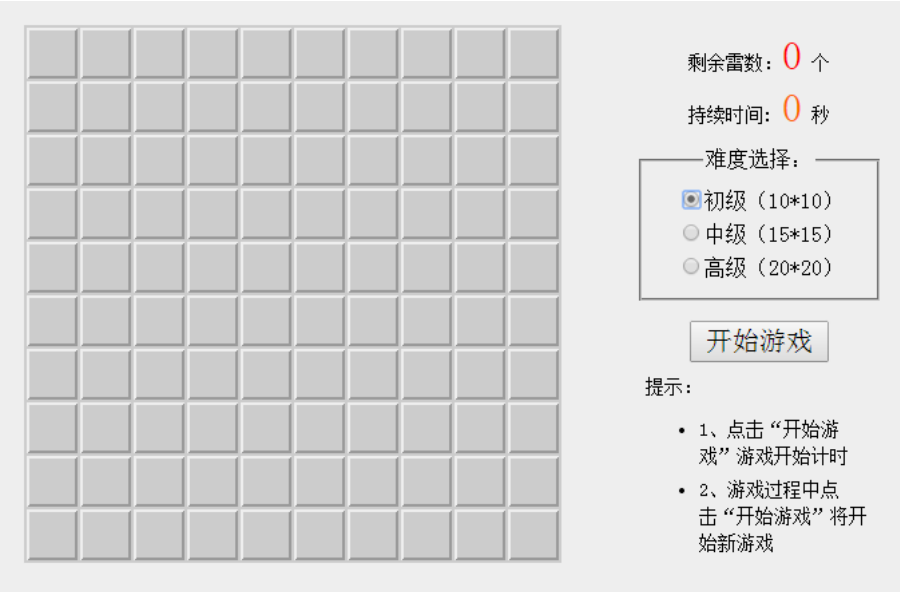
在 index.js 中的调用代码中，我们需要绑定难度选择按钮的事件，然后调用上面定义的 JMS，开始绘制格子。

在 index.js 中添加如下代码并保存：

```
//在index.js中
var jms = null,
    timeHandle = null;
window.onload = function ()动手实践是学习 IT 技术最有效的方式!                开始实验
    var radios = document.getElementsByName("level");
    for (var i = 0, j = radios.length; i < j; i++) {
        radios[i].onclick = function () {
            if (jms != null)
                if (jms.landMineCount > 0)
                    if (!confirm("确定结束当前游戏? "))
                        return false;
            var value = this.value;
            init(value, value, value * value / 5 - value, value * value / 5);
            document.getElementById("JMS_main").style.width = value * 40 + 180 + 60 + "px";
        }
    }
    init(10, 10);
};

function init(rowCount, colCount, minLandMineCount, maxLandMineCount) {
    var doc = document,
        landMineCountElement = doc.getElementById("landMineCount"),
        timeShow = doc.getElementById("costTime"),
        beginButton = doc.getElementById("begin");
    if (jms != null) {
        clearInterval(timeHandle);
        timeShow.innerHTML = 0;
        landMineCountElement.innerHTML = 0;
    }
    jms = JMS("landmine", rowCount, colCount, minLandMineCount, maxLandMineCount);
}
```

然后在浏览器中打开 index.html，格子已经可以显示出来了，效果如下：



点击右边的难度选择，可以看到格子的数量变化。

4.3 游戏初始化

现在，我们开始对游戏初始化，主要分三步：

1. 把所有格子（代码中用一个数组表示）初始化为0
2. 随机生成地雷个数，把地雷随机放到数组中，数组项值设置为9
3. 计算其他格子中的数字，值放入数组中

在 jms.js 中 JMS.prototype 内加入如下代码：

```
//在jms.js中JMS.prototype内加入
//初始化，一是设置数组默认值为0，二是确定地雷个数
init: function () {
    for (var i = 0; i < this.rowCount; i++) {
        this.arrs[i] = [];
        for (var j = 0; j < this.colCount; j++) {
            this.arrs[i][j] = 0;
        }
    }
    this.landMineCount = this.selectFrom(this.minLandMineCount, this.maxLandMineCount);
    this.markLandMineCount = 0;
    this.beginTime = null;
    this.endTime = null;
    this.currentSetpCount = 0;
},
//把是地雷的数组项的值设置为9
landMine: function () {
    var allCount = this.rowCount * this.colCount - 1,
        tempArr = {};
    for (var i = 0; i < this.landMineCount; i++) {
        var randomNum = this.selectFrom(0, allCount),
            rowCol = this.getRowCol(randomNum);
        if (randomNum in tempArr) {
            i--;
            continue;
        }
        this.arrs[rowCol.row][rowCol.col] = 9;
        tempArr[randomNum] = randomNum;
    }
},
//计算其他格子中的数字
calculateNoLandMineCount: function () {
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            if (this.arrs[i][j] == 9)
                continue;
            if (i > 0 && j > 0) {
                if (this.arrs[i - 1][j - 1] == 9)
                    this.arrs[i][j]++;
            }
            if (i > 0) {
                if (this.arrs[i - 1][j] == 9)
                    this.arrs[i][j]++;
            }
            if (i > 0 && j < this.colCount - 1) {
                if (this.arrs[i - 1][j + 1] == 9)
                    this.arrs[i][j]++;
            }
            if (j > 0) {
                if (this.arrs[i][j - 1] == 9)
                    this.arrs[i][j]++;
            }
            if (j < this.colCount - 1) {
                if (this.arrs[i][j + 1] == 9)
                    this.arrs[i][j]++;
            }
            if (i < this.rowCount - 1 && j > 0) {
                if (this.arrs[i + 1][j - 1] == 9)
                    this.arrs[i][j]++;
            }
            if (i < this.rowCount - 1) {
                if (this.arrs[i + 1][j] == 9)
                    this.arrs[i][j]++;
            }
            if (i < this.rowCount - 1 && j < this.colCount - 1) {
                if (this.arrs[i + 1][j + 1] == 9)
                    this.arrs[i][j]++;
            }
        }
    }
},
//获取一个随机数
selectFrom: function (iFirstValue, iLastValue) {
    var iChoices = iLastValue - iFirstValue + 1;
    return Math.floor(Math.random() * iChoices + iFirstValue);
},
```

```
//通数值找到行数和列数
getRowCol: function (val) {
    return {
        row: parseInt(val / this.colCount),
        col: val % this.colCount
    };
},
```


4.4 给格子添加点击事件

现在，该给格子添加点击事件了，当左键点击时，显示出格子中的数字（如果是地雷就挑战失败，结束游戏），右键点击时标记为地雷。

另外，第一次点击格子（往往带有运气成份）如果周围有空白区域会直接展开。

`jms.js` 中的这部分代码如下：

```
//在jms.js中JMS.prototype内加入
//获取元素
$: function (id) {
    return this.doc.getElementById(id);
},
//给每个格子绑定点击事件（左键和右键）
bindCells: function () {
    var self = this;
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            (function (row, col) {
                self.$("m_" + i + "_" + j).onmousedown = function (e) {
                    e = e || window.event;
                    var mouseNum = e.button;
                    var className = this.className;
                    if (mouseNum == 2) {
                        if (className == "flag") {
                            this.className = "";
                            self.markLandMineCount--;
                        } else {
                            this.className = "flag";
                            self.markLandMineCount++;
                        }
                        if (self.landMineCallBack) {
                            self.landMineCallBack(self.landMineCount - self.markLandMineCount);
                        }
                    } else if (className != "flag") {
                        self.openBlock.call(self, this, row, col);
                    }
                };
            })(i,j);
        }
    }
},
//展开无雷区域
showNoLandMine: function (x, y) {
    for (var i = x - 1; i < x + 2; i++)
        for (var j = y - 1; j < y + 2; j++) {
            if (!(i == x && j == y)) {
                var ele = this.$("m_" + i + "_" + j);
                if (ele && ele.className == "") {
                    this.openBlock.call(this, ele, i, j);
                }
            }
        }
},
//显示
openBlock: function (obj, x, y) {
    if (this.arrs[x][y] != 9) {
        this.currentSetpCount++;
        if (this.arrs[x][y] != 0) {
            obj.innerHTML = this.arrs[x][y];
        }
        obj.className = "normal";
        if (this.currentSetpCount + this.landMineCount == this.rowCount * this.colCount) {
            this.success();
        }
        obj.onmousedown = null;
        if (this.arrs[x][y] == 0) {
            this.showNoLandMine.call(this, x, y);
        }
    } else {
        this.failed();
    }
},
//显示地雷
showLandMine: function () {
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            if (this.arrs[i][j] == 9) {
                this.$("m_" + i + "_" + j).className = "landMine";
            }
        }
    }
},
//显示所有格子信息
showAll: function () {
```

```
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            if (this.arrs[i][j] == 9) {
                this.$("m_" + i + "_" + j).className = "landMine";
            } else {
                var ele=this.$("m_" + i + "_" + j);
                if (this.arrs[i][j] != 0)
                    ele.innerHTML = this.arrs[i][j];
                ele.className = "normal";
            }
        }
    }
},
//清除显示的格子信息
hideAll: function () {
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            var tdCell = this.$("m_" + i + "_" + j);
            tdCell.className = "";
            tdCell.innerHTML = "";
        }
    }
},
//删除格子绑定的事件
disableAll: function () {
    for (var i = 0; i < this.rowCount; i++) {
        for (var j = 0; j < this.colCount; j++) {
            var tdCell = this.$("m_" + i + "_" + j);
            tdCell.onmousedown = null;
        }
    }
},
```


4.5 添加游戏控制功能

到现在为止，游戏主要部分已经完成，下面要做的就是添加游戏控制功能，让游戏正常运行起来，主要有以下几步：

1. 给游戏开始按钮添加点击事件，重置游戏参数
2. 游戏开始后开始记时
3. 游戏结束停止记时并提示

在 jms.js 中添加游戏入口和开始功能：

```
//在 jms.js 中 JMS.prototype 内加入
//游戏开始
begin: function () {
    this.currentSetpCount = 0;//开始的步数清零
    this.markLandMineCount = 0;
    this.beginTime = new Date();//游戏开始时间
    this.hideAll();
    this.bindCells();
},
//游戏结束
end: function () {
    this.endTime = new Date();//游戏结束时间
    if (this.endCallBack) {//如果有回调函数则调用
        this.endCallBack();
    }
},
//游戏成功
success: function () {
    this.end();
    this.showAll();
    this.disableAll();
    alert("Congratulation! ");
},
//游戏失败
failed: function () {
    this.end();
    this.showAll();
    this.disableAll();
    alert("GAME OVER! ");
},
//入口
play: function () {
    this.init();
    this.landMine();
    this.calculateNoLandMineCount();
}
```


在 index.js 中给开始按钮添加事件，开始游戏，并显示游戏时间和剩余地雷个数：

```
//在index.js的init中加入
jms.endCallBack = function () {
    clearInterval(timeHandle);
};
jms.landMineCallBack = function (count) {
    landMineCountElement.innerHTML = count;
};

//为“开始游戏”按钮绑定事件
beginButton.onclick = function () {
    jms.play();//初始化

    //显示地雷个数
    landMineCountElement.innerHTML = jms.landMineCount;

    //开始
    jms.begin();

    //更新花费的时间
    timeHandle = setInterval(function () {
        timeShow.innerHTML = parseInt((new Date() - jms.beginTime) / 1000);
    }, 1000);
};
```

到此我们的网页版扫雷就完成了。用浏览器打开 index.html 文件，就可以开始游戏了。

开始实验

五、实验总结

本实主要使用 JavaScript 实现了经典小游戏扫雷的网页版。相信通过完成本实验，可以提高你对 JavaScript 的理解与运用能力。此外，你也可以学习到如何使用 JavaScript 语言抽象、封装游戏中的对象。

六、思考题

完成基本的扫雷版本后可以考虑为游戏增加一些难度，比如增加限时等功能。

欢迎把你的思考发布到实验报告，与其他同学交流。

课程教师

atwal

共发布过**2**门课程

查看老师的所有课程 > (/teacher/21657)

前置课程

HTML基础入门 (/courses/19)

Javascript基础（新版） (/courses/21)

CSS速成教程 (/courses/53)

进阶课程

网页版别踩白块游戏 (/courses/306)

网页版2048 (/courses/62)