

Web Project Report of mine sweeping game in JavaScript

何沛霖(HE PEILIN)

StudentNum : 1809853U-I011-0078

Contents

1. Problem Restatement.....	1
2. Functional Analysis.....	1
3. Programming Designment	3
3.1 Constructor Module	3
3.2 Essential Function Realizing Minesweeping	4
3.3 Draw Mine Area	5
3.4 Bind Click Events.....	7
3.5 Initialization	8
3.6 HTML and CSS Technologies Application	8
Reference	9
Appendices	10
a) demo.html.....	10
b) mine.css.....	11
c) mine.js.....	13
d) main.js.....	20

1. Problem Restatement

Imitating windows mine sweeping game, I plan to develop a simple mine sweeping web page game by using technologies I have learned this term. Importantly I focus attention on the Document Object Model (DOM) and Cascading Style Sheets (CSS) to defining how JavaScript programs can access, manipulate and provides a great deal of control over the presentation of the HTML document currently displayed by browser Chrome & IE without installing additional plugins.

2. Functional Analysis

General Game Process Analysis:

A mine area is displayed in the center of the screen, and a certain number of randomly distributed mines are embedded in the mine area in advance. The player controls the cursor to move among the small squares of the mine area and marks them through the left and right mouse buttons. If all mines in the mine area can be correctly

marked, the game wins; otherwise, the game fails.

More Specific Functional Analysis:

A. Beginning:

First select the game level, and then generate different levels of minefield interface. The game level is divided into three levels: the number of squares in each level is - primary level: 10×10, middle level: 15×15, senior level: 20×20. Next click on button "Start Game" by Left mouse button to play. The number of mines at all levels is equal to the total number of squares / 5. There is one mine buried under each square in the minefield, or no mine.

B. Sweeping:

Move the cursor to a certain cell, click the left mouse button to open it. If there is a mine under the opened square, step on the mine, at this time, all the blocks with mines are marked, and the game fails; if there is a number on the cell, it represents how many mines there are in the eight cells around it.

C. Automatic Sweeping:

When sweeping a cell which display null number except gray background color, it will automatic sweep the eight cells around it.

D. Marking Mines:

Click the right mouse button on a cell where the cursor is, then the landmine buried under this block will be marked (in fact, it may be mislabeled), and it will be displayed as a picture of flag. For each landmine marked, the number of mines is reduced by 1.

E. Marking Question:

If you click the right mouse button by twice on a cell where the cursor is, you will mark a question mark (?) on it, which means you are not sure whether it has a mine. The cells marked as "?" can be dug or marked as mines on the right time by clicking the right mouse button again.

F. Timing Counter:

In a section of web page, it can display how many seconds you spend during a round of game. It will begin to count while clicking on button "Start Game" until games over. Furthermore, you can clear counter by exchange the level of difficulty or restart a new game.

G. Output:

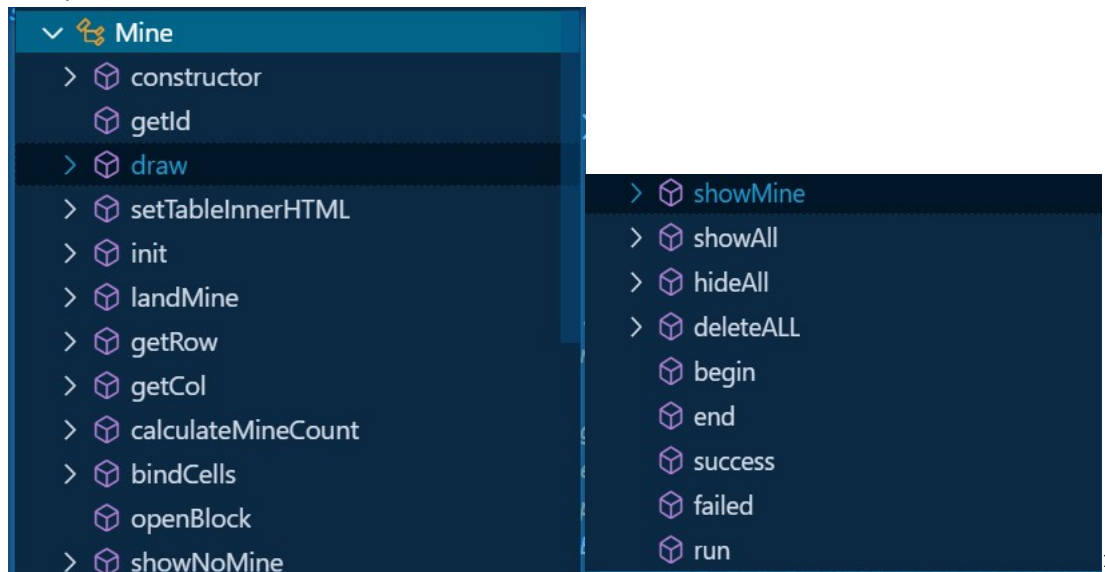
Prompt whether the game is successful or not.

Hit:

Click "start game" during the game to start a new game; The cells marking mines or question couldn't click on the left mouse button unless clicking the right mouse button again to return to normal; Not supported click two mouse buttons at same times.

3. Programming Designment

The important code is almost in file "mine.js". You clearly see them in those two pictures blew.



Besides, the file "main.js" is just a script of API to call the class to realize function. The file "demo.html" is to use layout tables and tracing images to design a web page, and to use to start my layout. And the file "mine.css" provides a rich document style appearance, as well as the ability to set text and background attributes and to create borders for corresponding element. Meanwhile it can adjust the distance between element borders and other elements, or the distance between element borders and element content.

3.1 Constructor Module

I use a class declaration² to take replace of constructor function to do a favor to initialize different variable properties in class "Mine". The specific meaning of variables you can easily understand by parts of comments.

To emphasize some import properties, I would introduce the meaning of some code. In No.2 line, "this instanceof Mine" is used to determine whether the keyword "new" is used so that to construct a scope safe constructor to prevent users from entering global objects without this binding. In No.19 line, I return Boolean "false" to "this.doc.oncontextmenu" in order to forbidden the right mouse button. Because the property "GlobalEventHandlers.oncontextmenu" is that opening the context menu, when the user clicks the right mouse button in the client area. Then we can bind the corresponding click events.

¹ All method in class "Mine".

² Constructor function can convert to ES2015 class

```

JS mine.js > Mine > constructor
1  class Mine{
2      constructor(id, row, col){
3          if(!(this instanceof Mine)){
4              return new Mine(id, row, col);
5          }
6          this.doc = document;
7          this.row = row || 10;
8          this.col = col || 10;
9          this.mineCount = this.row * this.col / 5;
10         this.markMineCount = 0; //marked number of mine
11         this.arr = [];
12         this.beginTime = null; //beginning time
13         this.endTime = null; //ending time
14         this.stepCount = 0; //current steps
15         this.table = this.doc.getElementById(id); //table for calls
16         this.endCallBack = null; //callback function at the end of the game
17         this.LandMineCallBack = null;
18         //callback function to update the number of remaining mines when marked as mines
19         this.doc.oncontextmenu = function(){
20             return false;
21         };
22
23         this.draw();

```

3.2 Essential Function Realizing Minesweeping

I plan to specifically clarify these methods to realize the function of minesweeping game. I would introduce important method to realize minesweeping. And you can read the detailed code in appendices. To set the value of the array item that is mine to 9, I define method "landMine()". The action of method "calculateMineCount()" is calculating the numbers in other cells. To show mine-areas and no-mine-areas, I create method "showMine()" and "showNoMine()" to distinguish and realize. Besides, method "hideAll()" and "deleteALL()" which will be called in interface method could hide all information of cells and delete binding event of cells.

The flow-chart diagram of my coding general thinking below³:

³ flow-chart diagram is to describe the main idea of realization rather than calling all of concrete methods. Because of lengthy code, I wouldn't introduce all of methods. You can still find in appendices.



3.3 Draw Mine Area

To realize the function of drawing the table within mines, I write two methods, "draw()" and "setTableInnterHTML()".

The code of method "draw()" below:

```

30 draw() {
31     var tdArr = []; // Record the <td> with id of all cells
32     if (window.ActiveXObject && parseInt(navigator.userAgent.match(/msie ([\d.]+)/i)[1]) < 8) {
33         var css = '#main table td{background-color:#888;}',
34             head = this.doc.getElementsByTagName("head")[0],
35             style = this.doc.createElement("style");
36         style.type = "text/css";
37         if (style.styleSheet) {
38             style.styleSheet.cssText = css;
39         } else {
40             style.appendChild(this.doc.createTextNode(css));
41         }
42         head.appendChild(style);
43     }
44     for (var i = 0; i < this.row; i++) {
45         tdArr.push("<tr>");
46         for (var j = 0; j < this.col; j++) {
47             tdArr.push("<td id='" + i + " " + j + "'></td>");
48         }
49         tdArr.push("</tr>");
50     }
51     this.setTableInnerHTML(this.table, tdArr.join(""));
52 }
53

```

First of all, we judge the browser whether is less than IE8 or others by using regular expression and "ActiveXObject" control. Then we create essential CSS of the element table, head by retrieving a collection of objects based on the specified element name, and style by defining an instance of the element for the specified tag. After that, judge whether getting the two elements of the CSS stylesheet object. The result shows that this type of not IE uses sheet property, IE uses stylesheet. When getting corresponding elements. Otherwise, we first use method "createTextNode" to create a text string from the specified value with the parameter "css" – String that specifies the "nodeValue" property of the text node. Then we use method "appendChild" to add a new child to the end of the node's child list so that we can append the elements style into head. Finally, we create a string array to push ID of all the cells by traversing the table for next part.

The code of method "setTableInnerHTML()" below:

```

54 //Add HTML to Table
55 setTableInnerHTML(table, html) {
56     if (navigator && navigator.userAgent.match(/msie/i)) {
57         var temp = table.ownerDocument.createElement('div');
58         temp.innerHTML = '<table><tbody>' + html + '</tbody></table>';
59         if (table.tBodies.length == 0) {
60             var tbody = document.createElement("tbody");
61             table.appendChild(tbody);
62         }
63         table.replaceChild(temp.firstChild.firstChild, table.tBodies[0]);
64     } else {
65         table.innerHTML = html;
66     }
67 }

```

This method is to add the corresponding HTML to the HTML event "table". Specifically, I use DOM technology to change the information of tag "div" and "tbody" by replacing its first child nodes as an array which includes the information of table body.

3.4 Bind Click Events

To realize the function of binding click events, I write two methods, "bindCells()" and "openBlock()".

The code of method "bindCells()" below:

```
139     bindCells() {
140         var self = this;
141         for (var i = 0; i < this.row; i++) {
142             for (var j = 0; j < this.col; j++) {
143                 (function (row, col) {
144                     self.getId( i + " " + j).onmousedown = function (e) {
145                         e = e || window.event;
146                         var mouseNum = e.button;
147                         var className = this.className;
148                         if (mouseNum == 2) {
149                             if (className == "flag") {
150                                 this.className = "question";
151                                 self.markMineCount--;
152                             }
153                             else if(className == "question"){
154                                 this.className = "";
155                             }
156                             else {
157                                 this.className = "flag";
158                                 self.markMineCount++;
159                             }
160                             if (self.LandMineCallBack) {
161                                 self.LandMineCallBack(self.mineCount - self.markMineCount);
162                             }
163                         } else if (className != "flag" && className != "question") {
164                             self.openBlock.call(self, this, row, col);
165                         }
166                     };
167                 })(i,j);
168             }
169         }
170     }
```

I assume a variable "self" as a local variable to distinguish the GlobalEventHandlers "this.className". Depending on property "onmousedown", I write a function to judge the left or right mouse button. For example, No.149 line means judgment of the right mouse button. And judge whether chosen cell has been planted a flag, if there is a flag, you can dig it on the right time by clicking the right mouse button again. Meanwhile callback function will timely update surplus number of mines. If you click on the left button, it will display the correct number of mins around 8 cells or trigger the mine booming. The cells marked flag or question cannot click on left mouse button.

The code of method "openBlock()" below:

```
173     //display
174     openBlock(obj, x, y) {
175         if (this.arr[x][y] != 9) {
176             this.stepsCount++;
177             if (this.arr[x][y] != 0) {
178                 obj.innerHTML = this.arr[x][y];
179             }
180             obj.className = "normal";
181             if (this.stepsCount + this.mineCount == this.row * this.col) {
182                 this.success();
183             }
184             obj.onmousedown = null;
185             if (this.arr[x][y] == 0) {
186                 this.showNoMine.call(this, x, y);
187             }
188         } else {
189             this.failed();
190         }
191     }
```


This method is to display the chosen cells and set class name "normal" to correct cells so as to change color in CSS. The No.185 line is designed to realize automatic sweeping function that can show no-mine area. And I calculate the sum of the counts of steps and mines to check player whether sweep all mines to get success.

3.5 Initialization

In this part, you pay attention to JavaScript file "main.js". After the browser loads the object "window.onload" immediately gets a collection of object based on the value of the NAME attribute to initialize the level of difficulty. I will emphasize method "findBegin()".

The code of method "findBegin()" below:

```
19 function findBegin(row, col) {
20     var countElement = document.getElementById("mineCount");
21     var timeShow = document.getElementById("costTime");
22     var beginButton = document.getElementById("begin");

23     if (mine != null) {
24         clearInterval(timeHandle);
25         timeShow.innerHTML = 0;
26         countElement.innerHTML = 0;
27     }

28     mine = new Mine("landMine", row, col);
29     // set the callback function
30     mine.endCallBack = function () {
31         clearInterval(timeHandle);
32     };
33     mine.LandMineCallBack = function (m) {
34         countElement.innerHTML = m;
35     };
36     //Binding event for Button "Start Game"
37     beginButton.onclick = function () {
38         mine.run();//Initialization
39         //Show surplus number of mines
40         countElement.innerHTML = mine.mineCount;
41         mine.begin();
42         //Updating spent time
43         timeHandle = setInterval(function () {
44             timeShow.innerHTML = parseInt((new Date() - mine.beginTime) / 1000);
45         }, 1000);
46     };
47 }
48
49
50
51
52
53 }
```

In No.31 line, I define callback function to clear the mines and time when finishing a round of game. Then we can regenerate a new game randomly.

In No.49 line, I use method "setInterval()" to set the updating spent time. In HTML file we can follow ID "costTime" to be seen in web page. The setInterval() method calls a function or evaluates an expression in milliseconds over a specified period. The setInterval() method calls the function until clearInterval() is called or the window is closed. The ID value returned by setInterval() can be used as an argument to the clearInterval() method.

3.6 HTML and CSS Technologies Application

In file "mine.css", I define different class to display mines, flag and question mark. And I assume the global page is belonged to class "global" which includes other elements like "table", "fieldset", "label", and different type of "input".

To show different background, I use “: hover” selector in order to select elements on which the mouse pointer floats. the code below:













```

45 .global table td:hover {
46     background-color: beige;
47 }
48
49 .global table td.normal:hover {
50     background-color: gray;
51 }
52

```

In file “demo.html”, all of dynamic script and data blocks in their documents will be added in. And the code file represents content for the user.

The sample of web page below⁴:

						1	1	1	
	1	1	1			1		2	1
	1		1			2	2	3	
	1	1	1			1		2	1
						1	2	2	2
						2		3	3
	1	1	1			2			3
	1		2	1	2	3	3	2	1
	1	2		1	2		2	1	1
		1	1	1	2		2	1	

Hit:

1. Click "start game" to start
2. Click "start game" during the game to start a new game

Surplus number of mines: **17**

Duration: **4**

Level: ☐ primary (10*10)
☐ middle (15*15)
☐ senior (20*20)

Start Game

Reference

- [1] Jackson, J. (2007). Web technologies : a computer science perspective / Jeffrey C. Jackson. (Pearson internation ed.). Upper Saddle River, NJ: Pearson Prentice Hall.
- [2] <https://blog.csdn.net/justBeHerHero/article/details/89917613>

⁴ You can find the .gif file in the folder.

Appendices

a) demo.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mine Sweeping</title>
  <link rel="stylesheet" type="text/css" href="mine.css"/>
  <script type="text/javascript" src="mine.js"></script>
  <script type="text/javascript" src="main.js"></script>
  <script src="https://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js
"></script>
</head>
<body>
  <div id="main" class="global">
    <table id="landMine"></table>
    <div id="op">
      <div style="text-align:left">Hit:
        <ol>
          <li> Click "start game" to start</li>
          <li> Click "start game" during the game to start a
new game</li>
        </ol>
      </div>
      <div class="tip">Surplus number of mines:
<span style="color: red; font-
size: 20px;" id="mineCount">0</span> </div>
      <div class="tip">Duration: <span style="color: orange; font
-size: 20px;" id="costTime">0</span> </div>
      <fieldset >
        <legend>Level: </legend>
        <input type="radio" name="level" id="pLevel" checked="c
hecked" value="10" /><label for="pLevel">primary (10*10) </label><br />
        <input type="radio" name="level" id="mLevel" value="15
" /><label for="mLevel">middle (15*15) </label><br />
        <input type="radio" name="level" id="sLevel" value="20
" /><label for="sLevel">senior (20*20) </label><br />
      </fieldset>
      <input type="button" id="begin" value="Start Game" /><br />
    </div>
  </div>
```

```
    </div>
</body>
</html>
```

b) mine.css

```
.global {
    margin:10px auto;
    padding:20px;
    background:white;
    width:600px;
    zoom:1;
}

.global input[type="button"] {
    padding:2px 10px;
    margin:5px;
    font-size:20px;
    cursor:pointer;
}

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

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

.global table {
    background:white;
    float:left;
}

.global table td {
    border:2px outset ghostwhite;
    font-size:20px;
    width:32px;
    height:32px;
    text-align:center;
    cursor:pointer;
}
```

```

        .global table td.normal {
            border:2px solid ghostwhite;
            background-color:gray;
        }

        .global table td.hover {
            background-color:beige;
        }

        .global table td.normal:hover {
            background-color:gray;
        }

.global #op {
    width:180px;
    float:right;
    text-align:center;
}

.global .tip {
    font-size:14px;
    margin:5px;
}

        .global .tip ol li {
            margin:5px 0;
            line-height:20px;
        }

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

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

```

```

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

```

c) mine.js

```

class Mine{
    constructor(id, row, col){
        if(!(this instanceof Mine)){
            return new Mine(id, row, col);
        }
        this.doc = document;
        this.row = row || 10;
        this.col = col || 10;
        this.mineCount = this.row * this.col / 5;
        this.markMineCount = 0;//marked number of mine
        this.arr = [];
        this.beginTime = null;//beginning time
        this.endTime = null;//ending time
        this.stepCount = 0;//current steps
        this.table = this.doc.getElementById(id);//table for calls
        this.endCallBack = null;//callback function at the end of the game
        this.LandMineCallBack = null;
        //callback function to update the number of remaining mines when marked as mines
        this.doc.oncontextmenu = function(){
            return false;
        };

        this.draw();
    }

    getId(id){
        return document.getElementById(id);
    }

    draw() {
        var tdArr = [];//Record the <td> with id of all cells

```

```

        if (window.ActiveXObject && parseInt(navigator.userAgent.match(
/msie ([\d.]+)/i)[1]) < 8) {
            var css = '#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.row; i++) {
            tdArr.push("<tr>");
            for (var j = 0; j < this.col; j++) {
                tdArr.push("<td id='" + i + " " + j + "'></td>");
            }
            tdArr.push("</tr>");
        }
        this.setTableInnerHTML(this.table, tdArr.join(""));
    }

    //Add HTML to Table
    setTableInnerHTML(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;
        }
    }

    init() {
        for (var i = 0; i < this.row; i++) {
            this.arr[i] = [];
            for (var j = 0; j < this.col; j++) {

```

```

        this.arr[i][j] = 0;
    }
}
this.mineCount = this.row * this.col / 5;
this.markMineCount = 0;
this.beginTime = null;
this.endTime = null;
this.stepsCount= 0;
}

//Set the value of the array item that is mine to 9
landMine() {
    var allCount = this.row * this.col;
    var tempArr = [];
    for (var i = 0; i < this.mineCount; i++) {
        var randomNum = Math.floor(Math.random()*allCount)+1;
        var Row = this.getRow(randomNum);
        var Col = this.getCol(randomNum);
        if (randomNum in tempArr) {
            i--;
            continue;
        }
        if(Row != 0 && Col != 0){
            this.arr[Row][Col] = 9;
        }

        tempArr[randomNum] = randomNum;
    }
}

//get the corresponding row or column by value
getRow(val) {
    var row = parseInt(val / this.col);
    return row;
}

getCol(val){
    var col = val % this.col;
    return col;
}

//Calculate the numbers in other cells
calculateMineCount() {
    var dir=[0,1,-1];

```



```

        for (var i = 0; i < this.row; i++) {
            for (var j = 0; j < this.col; j++) {
                if (this.arr[i][j] == 9)
                    continue;
                for(var m = 0; m < 3; m++){
                    for(var n = 0; n < 3; n++){
                        if(i + dir[m] >= 1 && j + dir[n] >=1
                            && i + dir[m] <= this.row - 1 && j + dir[n]
<= this.col-1
                            && !(dir[m] == 0 && dir[n] == 0)
                            && this.arr[i + dir[m]][j + dir[n]] != -1){
                            if(this.arr[i + dir[m]][j + dir[n]] ==
9){
                                this.arr[i][j]++;
                            }
                        }
                    }
                }
            }
        }
    }

    //Bind click events (left and right) to each cell
    bindCells() {
        var self = this;
        for (var i = 0; i < this.row; i++) {
            for (var j = 0; j < this.col; j++) {
                (function (row, col) {
                    self.getId( 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 = "question";
                                self.markMineCount--;
                            }
                            else if(className == "question"){
                                this.className = "";
                            }
                            else {
                                this.className = "flag";

```

```

                self.markMineCount++;
            }
            if (self.LandMineCallBack) {
                self.LandMineCallBack(self.mineCount -
self.markMineCount);
            }
        } else if (className != "flag" && className !=
"question") {
            self.openBlock.call(self, this, row, col);
        }
    };
    })(i,j);
}
}
}

//display
openBlock(obj, x, y) {
    if (this.arr[x][y] != 9) {
        this.stepsCount++;
        if (this.arr[x][y] != 0) {
            obj.innerHTML = this.arr[x][y];
        }
        obj.className = "normal";
        if (this.stepsCount + this.mineCount == this.row * this.col
) {
            this.success();
        }
        obj.onmousedown = null;
        if (this.arr[x][y] == 0) {
            this.showNoMine.call(this, x, y);
        }
    } else {
        this.failed();
    }
}

//show no-mine-areas
showNoMine(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.getId( i + " " + j);
            }
        }
    }
}

```

```

        if (ele && ele.className == "") {
            this.openBlock.call(this, ele, i, j);
        }
    }
}

//show mine-areas
showMine(){
    for (var i = 0; i < this.row; i++) {
        for (var j = 0; j < this.col; j++) {
            if (this.arr[i][j] == 9) {
                this.getId( i + " " + j).className = "landMine";
            }
        }
    }
}

//show all areas
showAll() {
    for (var i = 0; i < this.row; i++) {
        for (var j = 0; j < this.col; j++) {
            if (this.arr[i][j] == 9) {
                this.getId( i + " " + j).className = "landMine";
            } else {
                var ele=this.getId( i + " " + j);
                if (this.arr[i][j] != 0)
                    ele.innerHTML = this.arr[i][j];
                ele.className = "normal";
            }
        }
    }
}

//Hide all information of cells
hideAll() {
    for (var i = 0; i < this.row; i++) {
        for (var j = 0; j < this.col; j++) {
            var tdCell = this.getId( i + " " + j);
            tdCell.className = "";
            tdCell.innerHTML = "";
        }
    }
}
}

```

```

//Delete binding event of cells
deleteALL(){
    for (var i = 0; i < this.row; i++) {
        for (var j = 0; j < this.col; j++) {
            var tdCell = this.getId( i + " " + j);
            tdCell.onmousedown = null;
        }
    }
}

begin(){
    this.stepsCount = 0;//number of steps clearing
    this.markMineCount = 0;
    this.beginTime = new Date();//beginning time
    this.hideAll();
    this.bindCells();
}

end(){
    this.endTime = new Date();//ending time
    if (this.endCallBack) { //Call if there is a callback function
        this.endCallBack();
    }
}

success(){
    this.end();
    this.showALL();
    this.deleteALL();
    alert("YOU WIN!! ");
}

failed(){
    this.end();
    this.showALL();
    this.deleteALL();
    alert("GAME OVER!! ");
}

run(){

```

```

        this.init();
        this.landMine();
        this.calculateMineCount();
    }
}

```

d) main.js

```

var mine = null;
var timeHandle = null;
window.onload = function () {
    var radios = document.getElementsByName("level");
    for (var i = 0, j = radios.length; i < j; i++) {
        radios[i].onclick = function () {
            if (mine != null)
                if (mine.mineCountElement > 0)
                    if (!confirm("end? "))
                        return false;
            var value = this.value;
            findBegin(value, value);
            document.getElementById("main").style.width = value * 40 +
180 + 60 + "px";
        }
    }
    findBegin(10, 10);
};

function findBegin(row, col) {
    var countElement = document.getElementById("mineCount");
    var timeShow = document.getElementById("costTime");
    var beginButton = document.getElementById("begin");
    if (mine != null) {
        clearInterval(timeHandle);
        timeShow.innerHTML = 0;
        countElement.innerHTML = 0;
    }

    mine = new Mine("landMine", row, col);
    // set the callback function
    mine.endCallBack = function () {
        clearInterval(timeHandle);
    };

    mine.landMineCallBack = function (m) {
        countElement.innerHTML = m;
    }
}

```

```

};

//Binding event for Button "Start Game"
beginButton.onclick = function () {
    mine.run();//Initialization

    //Show surplus number of mines
    countElement.innerHTML = mine.mineCount;

    mine.begin();

    //Updating spent time
    timeHandle = setInterval(function () {
        timeShow.innerHTML = parseInt((new Date() - mine.beginTime)
/ 1000);
    }, 1000);
};
}

```