مقدمه

در این تمرین از شما خواسته شده است که با استفاده از XML،JavaScript، JQuery ،XSLT، Schema XML و Ajax نسبت به پیاده سازی بازی **Mine Sweeper** (مین روب) اقدام کنید ☺

یک نسخه ی ایستا از سایت این بازی در قالب HTML و CSS در اختیارتان قرار داده شده است که می تونید از اون برای شروع پروژه ی خودتون استفاده فرمایید. در واقع شما باید تنها با استفاده از JavaScript و توابعی که در اختیارتان قرارداده شده اقدام به پردازش داده ها و اجرای روند بازی نمایید. در قسمتی از تمرین نیز لازم است که مطابق با ساختاری که در اختیارتان قرار داده ایم، یک XML معتبر تولید کرده و برای اعتبارسنجی و دریافت پاسخ به سرور ارسال کنید. (عملیات های ارسال به سرور نیازی به پیاده سازی ندارند و توابعی به جای آنها در اختیار شما قرارداده شده)

صورت مسأله

همان طور که در مقدمه بیان شد، هدف از این تمرین، پیاده سازی یک بازی ساده و پویا تحت وب و با استفاده از جاوا اسکریپت هست. این سایت دارای تنها و تنها یک صفحه HTML و یک یا چند فایل JS برای پیاده سازی خود بازی هست.

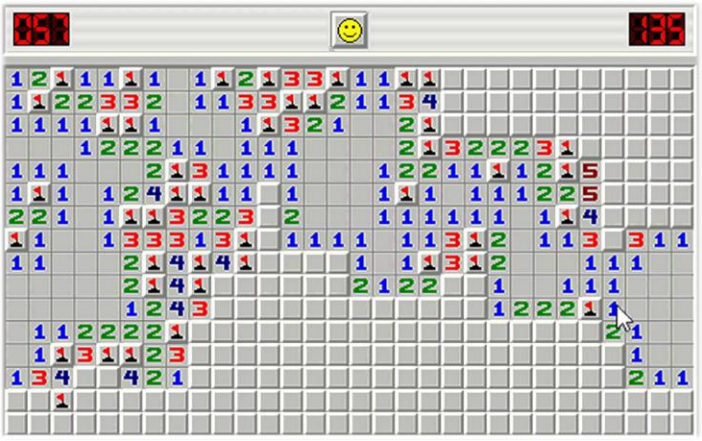
لازم به ذکر است که آدرس صفحه به هیچ وجه **نباید** تغییر کند و همه ی موارد باید با حذف و اضافه کردن محتوای جدید در صفحه بازی انجام شوند. محتوای جدید در هر مرحله از پردازش های صورت گرفته بر روی XML دریافتی تولید می شود .

توضیحات تکمیلی و نیازمندی های کامل بخش های مختلف تمرین در ادامه به تفصیل آورده شده است.

بازی مین روب

بازی مین روب (یا Minesweeper) یک بازی یک نفره پازلی است که هدف این بازی خنثی کردن مین ها داخل یک صفحه با مین های مخفی بدون ترکیدن هیچکدوم از آنها است. (کار خنثی کردن با کمک تعداد همسایه های مین داری که خانه های همسایه هر فیلد نمایان می کنند انجام می شود.) این بازی اولین بار در سال ۱۹۶۰ نوشته شده و نسخه هایی از آن روی پلتفرم های گوناگون از جمله OS2، Windows و KMines و Gnomine (لینوکس) منتشر شده. (منبع)

با شروع بازی یک صفحه با خانه های مربعی و یکسان بر روی صفحه برای بازیکن ظاهر می شود. تعدادی از خانه های تصادفی که برای کاربر نامعلوم است حاوی یک مین هستند. قبل از شروع بازی بازیکن مقادیر طول و عرض زمین و تعداد مین های کارگذاشته شده را مشخص می کند.



بازیکن در طول بازی با کلیک می تواند یک خانه را آشکار سازد یا مشخص کند که این خانه حاوی مین هست. (اگر به اشتباه خانه حاوی مین را آشکار سازد بازیکن بازنده است) اگر خانه آشکار شود و مینی در آن نباشد، به جای آن یک عدد داخل آن خانه نمایش داده می شود که تعداد خانه های همسایه حاوی مین را نمایان می سازد. اگر هیچ مین در خانه های اطراف آن وجود نداشته باشد هیچ عددی نشان داده نمی شود **و همه ی خانه های اطراف آن مادامی که مین نیستند به طور خودکار و بازگشتی نمایان می شوند.**

کاربر از اطلاعات عدد هایی که بر روی خانه های نمایان شده است برای حدس زدن محتوای خانه های دیگر استفاده می کند و می تواند با اطمینان یک خانه جدید را آشکار سازد یا آن را به عنوان مین پرچم گذاری کند.

⚠ امتیازی کاربر می تواند خانه هایی را هم با **علامت سوال** علامت گذاری کند که به کاربر برای تصمیم گیری های منطقی کمک می کند.

کاربر می تواند وقتی به تعداد مشخص شده اطراف یک خانه آشکار مین علامت گذاری کرد، با کلیک تمام خانه های اطراف آن را آشکار کند. (که ممکن است اگر اشتباه علامت گذاری شده باشند منجر به باخت کاربر شود!!)

بازی با خنثی کردن تمام خانه های خالی از مین به اتمام می رسد.

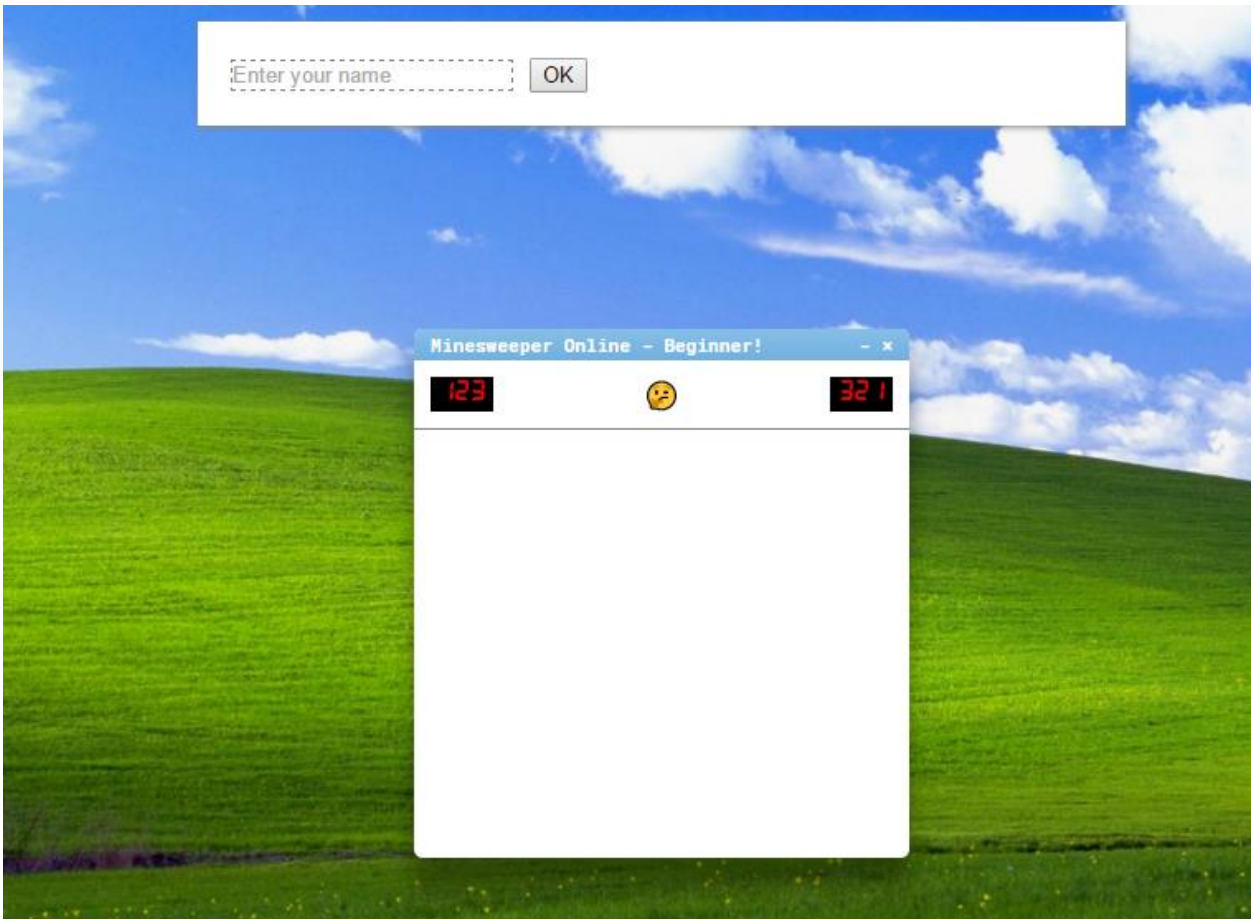
روند انجام تمرین

فرآیند کلی اجرایی در این تمرین به قرار زیر است (جزئیات بیشتر را در ادامه خواهید دید)

دقت کنید که تمامی کارهایی که جلوتر انجام میدیم فقط و فقط باید داخل فایل mineswipper.js باشد و فایل های دیگه حق تغییر ندارند.

گام اول – تولید Element های اصلی

هنگام شروع بازی باید با استفاده createElement اقدام به ساختن المنت های اصلی صفحه کنید. (برای مرجع می تونید فایل example.html رو مشاهده کنید.) (محتویات grid در مرحله های بعدی تولید می شوند.) باید خروجی مشابه تصویر زیر داشته باشید:



گام دوم – پردازش اطلاعات بازی

اطلاعات بازی و سطوح مختلف (شامل طول و عرض، زمان مجاز و تعداد مین ها و ...) از فایل XML در اختیار گذاشته شده بارگذاری و توسط JavaScript پردازش می شوند. برای اینکار تابع getGameXML رو در اختیار شما قرار داده ایم. نحوه استفاده:

```
// Please note that getGameXML can be only called once!

getGameXML(function callback(xml_str)){
    // TODO: xml_str is string response from server, parse and use it.
}
```

برای مرجع می توانید به فایل های game.xsd و game.example.xml در پوشه ی schema مراجعه کنید.

game مشخص کننده اطلاعات مرحله های بازی

id="minesweeper" این مقدار مشخص می کند که این فایل مختص بازی minesweeper می باشد و درصورتی که مقداری متفاوت با این بود باید در مرحله validate کردن با پیام مناسبی به کاربر اطلاع داده شود.

title="Minesweeper Online" عنوان بازی . این مقدار باید توسط JavaScript برای عنوان بازی جایگزین شود.

Levels مراحل مختلف بازی داخل این تگ مشخص می شوند.

default = 1 شناسه ی مرحله ی پیشفرضی که با آغار بازی انتخاب می شود.

level تگ آغازین مشخص کننده هر مرحله

id = 1 شناسه هر مرحله که برای ذخیره سازی اطلاعات آن و مراحل بعدی مورد استفاده قرار می گیرد.

title = "Beginner! " عنوان مرحله که در منو نمایش داده می شود.

time – mines – cols – Rows

به ترتیب نمایانگر سطر ها، ستون ها، تعداد مین های تصادفی و زمان مجاز برای بازی هستند.

دقت کنید که فرآیند درخواست و پردازش فایل XML تنها یک بار صورت می گیرد و باید این اطلاعات تا پایان کار نگه داری شوند.

پیشنهاد میشود که اطلاعات XML بارگذاری شده در یک Array معین ذخیره و نگه داری شوند. برای نمونه :

```
var game_title = "Minesweeper Online",
var game_id = "minesweeper",
var levels = [
    {
        id: 1,
        title : "Beginner!",
        timer : true,
        rows: 10,
        cols: 10,
        mines: 5,
        time: 120,
    },
];
```

✓ **نکته** بدون تست کردن کدهایی که تا اینجا پیاده سازی کردین مراحل بعدی رو ادامه ندهید! تست کردن مرحله به مرحله کمک خیلی زیادی در عیب یابی و داشتن یک پروژه خوب میکنه. برای تست و دیباگ کردن می تونین از کنسول Inspector مرورگر خورتون و تابع console.log("Message") استفاده کنید. نهایتا تا اینجا باید با استفاده از دیباگ از نحوه صحیح پردازش فایل XML مطمن شوید.

گام سوم – NewGame

در مرحله ی بعد تابعی به اسم newGame رو پیاده سازی می کنیم. این تابع با دو رویداد فراخوانی می شود: ۱- اولین بار که صفحه بازی اجرا می شود. ۲- با کلیک برروی دکمه newGame (شکلک بالای صفحه)

شما باید با تولید یک درخواست مناسب با فرمت XML که در فایل schema/new_game.xsd توصیف شده است، و فراخوانی توسط تابع getNewGame اطلاعات بازی با فرمت XML را دریافت نمایید و سپس با پردازش آن توسط XSLT که توصیف می کنید المان های مناسب برای تولید بخش grid روی صفحه رو تولید می کند. (دقت کنید اینکار حتما باید با استفاده از XSLTProcessor انجام شود.) . جزیات این گام در زیر توضیح داده شده.

مرحله اول : شما باید یک xml مناسب (به صورت string) را مطابق new_game.xsd تولید نمایید. این xml حاوی فیلدهای مورد نیاز برای دریافت اطلاعات یک بازی جدید است.

مرحله دوم : با فراخوانی تابع آماده `getNewGame` به همراه xml قبلی، اطلاعات level جدید را دریافت می کنید.

برای مرجع می توانید به فایل های level.xsd و level.example.xml در پوشه ی schema مراجعه کنید.

مرحله سوم: بعد از دریافت xml، نیاز هست توسط کلاس XSLTProcessor المان های مناسب تولید و به صفحه افزوده شوند.

> ✓ **نکته** بهتر است برای پیدا کردن راحت تر خانه های بازی در مراحل بعدی هنگام تولید div ها به آنها attribute های مناسب بدهید.

مرحله ی چهارم: Event Listener های مناسب را برای Element های ساخته شده تعریف می کنیم

نمونه پیاده سازی این گام :

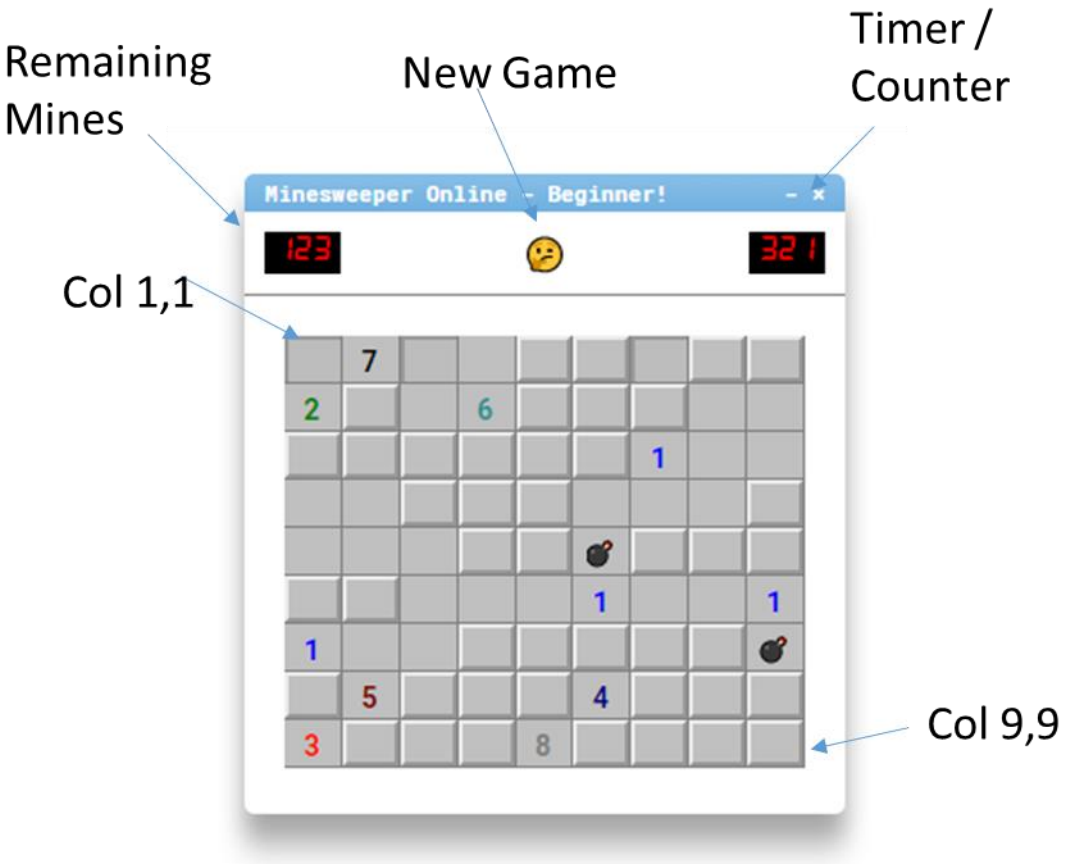
```
function makeXSL() {
    // This XSL Should Convert level.xml to
    // appreciate DOM elements for #grid.
    return `
        <?xml version="1.0" encoding="UTF-8"?>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
            <xsl:template match="/">
                <!-- TODO -->
            </xsl:template>
        </xsl:stylesheet>
    `;
}

function newGame() {
    // TODO: make string according to new_game.xsd schema
    var requestXML="";

    getNewGame(requestXML, function(xmlStr) {
        // Process and convert xmlStr to DOM using XSLTProcessor
        xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet(makeXSL());
        resultDocument = xsltProcessor.transformToFragment(xmlStr, document);
        document.getElementById('grid').appendChild(resultDocument);
    });
}
```

گام چهارم – پیاده سازی منطق بازی برپایه Event ها

- با کلیک روی نخستین خانه بازی (درصورت فعال بودن تایمر) تایمر بازی شروع به کار می کند. و یا درصورت عدم فعال بودن تایمر این شمارنده باید تعداد کلیک های انجام شده برای آشکار سازی را مشخص کند.
- با اتمام تایمر و یا درصورتی که کاربر روی یکی از خانه های حاوی مین کلیک کند بازی با پیام خطا به اتمام میرسد و شکلک باید به حالت متفاوتی دربیاید.
- در سمت چپ شمارنده counter نمایانگر تفاضل تعداد مین های بازی منهای تعداد خانه های علامت گذاری شده به عنوان مین (درواقع مین های خنثی نشده) هست.
- رویداد های مناسب برای کلیک کاربر باید پیاده سازی شوند (از جمله mouseDown و mouseUp) و کلاس های مناسب به col ها نسبت داده شود.
- درصورت راست کلیک کردن روی یک خانه آشکار نشده باید پرچم فلگ روی آن فعال یا غیر فعال شود و شمارنده ی بالای صفحه تغییر کند. بدیهی است مادامی که یک خانه پرچم گذاری شده باشد امکان آشکار سازی آن (کلیک) نباید باشد. (برای جلوگیری از نمایش منو هنگام راست کلیک می توانید از attribute ، oncontextmenu استفاده کنید)
- درصورتی که روی یک خانه بدون مین کلیک شد باید با فراخوانی تابع revealNeighbors به صورت recursive تمامی همسایه هایی که حاوی مین نیستند آشکار شوند و عدد های مناسب که بیانگر تعداد همسایه های حاوی مین آن هاست توسط data- value روی آنها نشان داده شود.
- هنگامی که روی شکلک کلیک شد، با یک پیام مناسب از کاربر سطح مرحله برای بازی جدید را دریافت کند.
- با کلیک روی یک خانه آشکار شده درصورتی که به تعداد مناسب خانه ی اطراف فلگ گذاری شده بود تمامی خانه های دیگر آشکار شوند (که ممکن است منجر به انفجار شود!)
- دریافت نام کاربر داخل Modal و عدم نمایش modal بعد از کلیک روی Ok. نکته: نام کاربر تنها میتواند شامل حروف a تا z باشد و کاراکتر های دیگر امکان وارد کردن را نباید داشته باشند.
- ⚠ **امتیازی** امکان تغییر سایز پنجره بازی توسط گوشه پایین سمت راست
- ⚠ **امتیازی** امکان جابه جایی پنجره بازی توسط نوار title bar

توضیحات صفحه بازی

نکته: دقت کنید شماره گذاری سلول ها از ۱و۱ و گوشه ی بالا سمت چپ شروش می شود. و عنصرهای اول و دوم به ترتیب نمایانگر شماره سطر و ستون سلول هستند!



نکات انجام و تحویل تمرین

- این تمرین به صورت انفرادی انجام می شود. در صورتی که تمرین تحویلی دو دانشجو و یا با یکی از نسخه های در دسترس بر روی اینترنت م شابه هم با شد، برای هر دو نفر نمره صفر در نظر گرفته می شود. (کد ها تو سط نرم افزار به صورت دقیق با سایرین و نسخه های آنلاین این بازی مقایسه خواهند شد)
- مشابه تمرین قبل، حق استفاده از هیچ گونه از کدهای آماده یا کتابخانه های مختلف را ندارید و تمامی کدها باید توسط خودتان نوشته شود. بخش های مورد نیاز برای دریافت فایل های XML در اختیار شما قرار داده شده است.
- مهلت انجام تمرین تا روز **چهار شنبه ۱۰ آذر** می با شد. این زمان به هیچ عنوان تمدید نخواهد شد و در ازای هرروز تاخیر ۱۵٪ از نمره شما کسر خواهد شد. پیشنهاد می شود برای آمادگی هر چه بیشتر برای امتحان و فرصت کافی برای رفع ابهامات انجام تمرین کار رو هر چه زودتر شروع کنید. ☺

- فایل minesweeper.js تکمیل شده خودتون رو با فرمت zip فشرده کرده و آن را مطابق الگوی زیر نام گذاری کنید:
HW3-StudentNumber-FirstNameLastName.zip
  - درصورتی که فایل های دیگر از جمله stylesheet ها یا عکس ها بنا به نیاز تغییر داده شده هستند و یا کد ها در چند فایل js نوشته شده اند، لطفا پروژه را به صورت کامل ذخیره سازی و ارسال کنید.
  - محتوای داخل body فایل فایل index.html به هیچ عنوان تغییر نکند!
  - استفاده از کامنت گذاری و فرمت زیبای کد کار پسندیده ای است ☺

- تمامی منابع مورد نیاز جهت انجام پروژه برروی آدرس زیر قابل دسترس می باشد:

https://github.com/AUT-CEIT/ie/tree/master/2016/fall/HW-3

- در صورت هرگونه ابهام، سوال و نیاز راهنمایی در مورد انجام تمرین از طریق ایجاد Issue اقدام کنید ☺

موارد تمرین امتیازی

- بخش های امتیازی با علامت ⚠ امتیازی در تعریف پروژه مشخص شده اند.
- این تمرین در مجموع ۲۰ نمره امتیازی دارد!