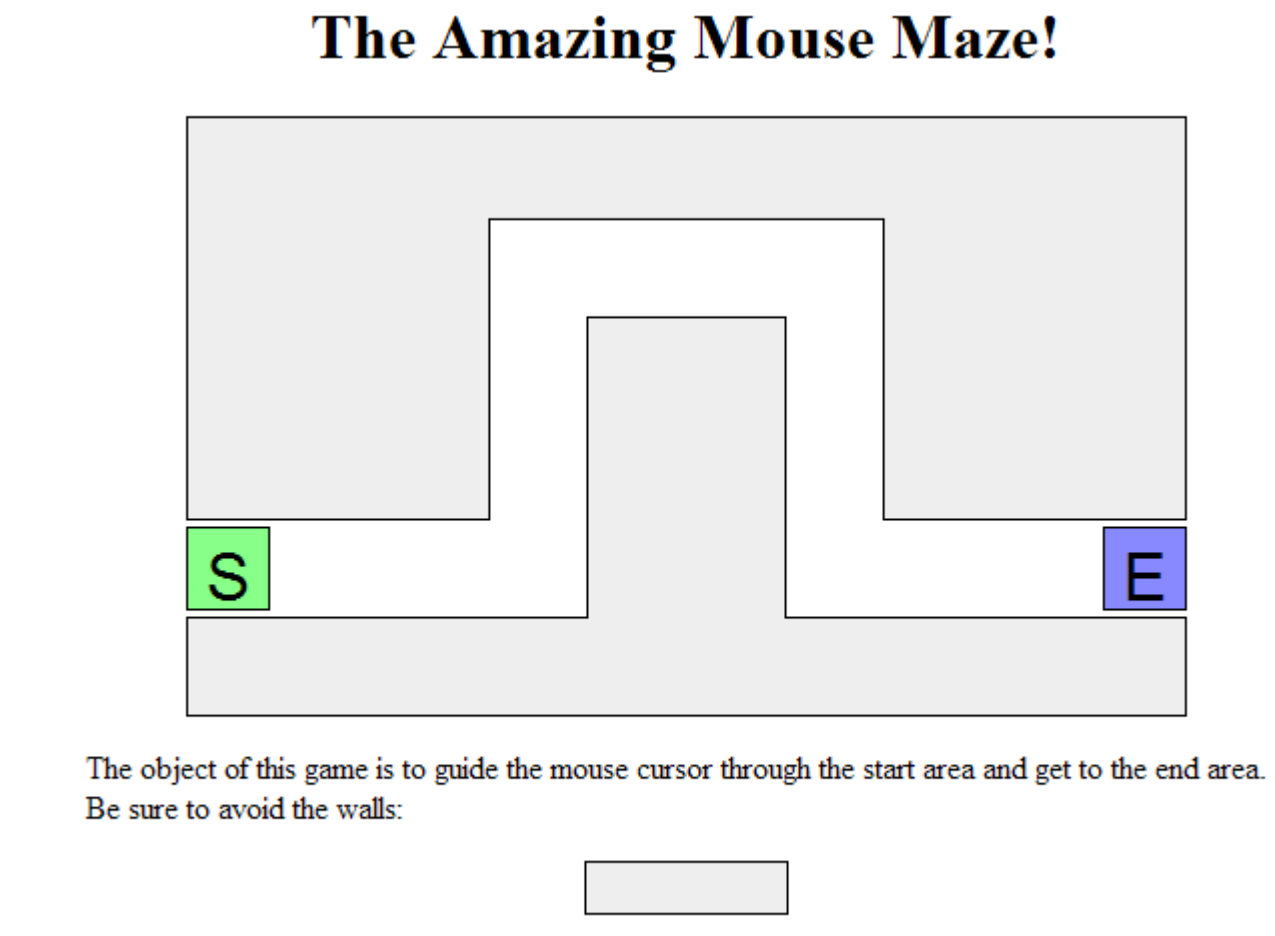
Assignment

Interactive Mouse Maze

The theme of this lab is that we'll be writing a page with a simple "maze" that the user must navigate with the mouse. The difficulty is not in finding the exit, because the maze is trivial, but rather in having the dexterity to move the mouse through the maze without touching any of the walls. When the user's mouse cursor touches one of the maze walls, the walls turn red and a "You lose" message is shown. Touching the Start button with the mouse will remove the red coloring from the maze walls.



You're given files maze.html and maze.css. You have the XHTML, but do not modify it. You should write a script file maze.js in the same directory as these files that provide all event handling and behavior to make the maze work as specified below.

The maze walls are implemented as 5 `div` elements on the page. Our provided CSS code already positions these `div` elements into their proper places on the page, but they do nothing when you hover the mouse over them. The relevant section of the `maze.html` code in the page's `body` that contains these elements is the following:

```
<div id="maze">
    <div id="start">S</div>
```

```
    <div class="boundary" id="boundary1"></div>
    <div class="boundary"></div>
    <div class="boundary"></div>
    <div class="boundary"></div>
    <div class="boundary"></div>
    <div id="end">E</div>
</div>
```

Exercises for Today:

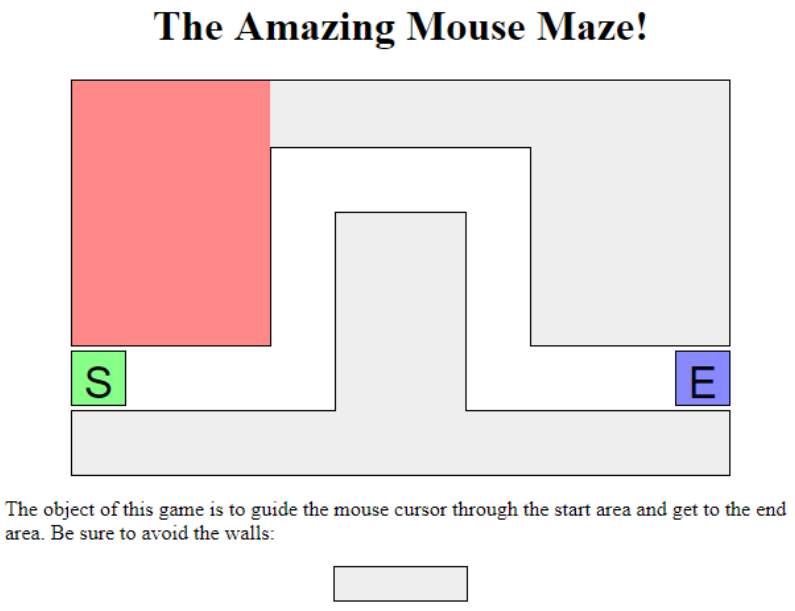
1. Single Boundary Turns Red
2. All Boundaries Glow Red on Hover
3. Alert on Successful Completion of Maze
4. Restartable Maze
5. JSLint / Upload Your Page
6. On-Page Status Updates
7. Disallow Cheating

Exercise 1: Single boundary turns red (roughly 15 minutes)

The first task is to write event-handling code so that when the user moves the mouse onto a single one of the maze's walls, that wall will turn red. Write your JavaScript code *unobtrusively*, without modifying the maze.html page. You will probably want to choose the first (top-left) wall; it is easier to access using the DOM, since it has an `id` attribute. You may want to turn the wall red by setting it to have the provided CSS class youlose, using Prototype's addClassName method.

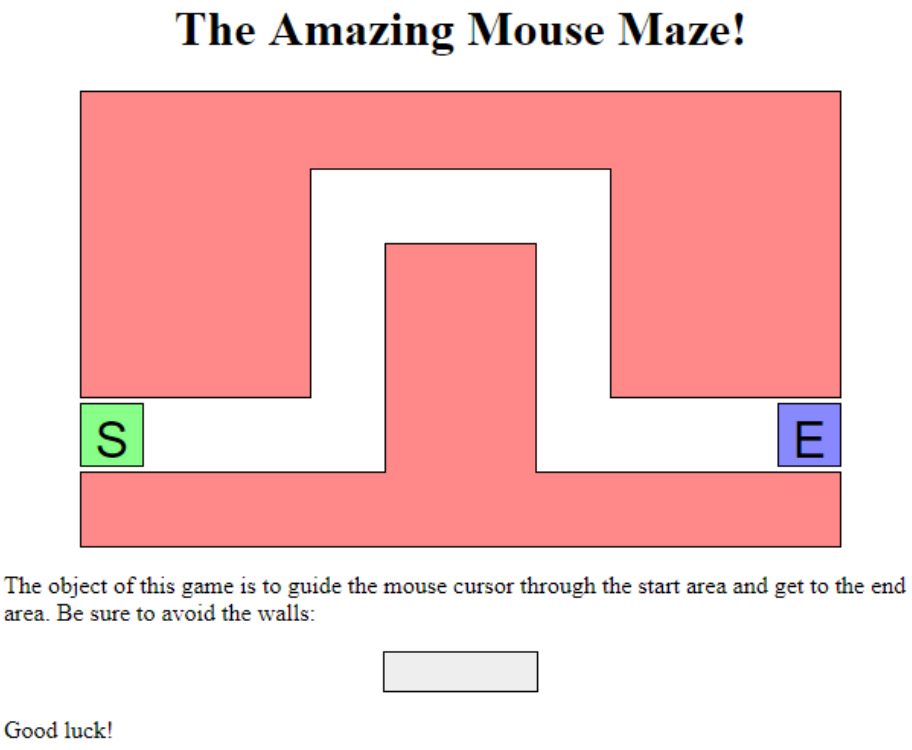
The following are more detailed suggested steps for solving this exercise:

- Write a window.onload handler that sets up any necessary event handlers on the page.
- Handle the appropriate event on the appropriate wall by making it turn red.



Exercise 2: All boundaries glow red on hover (roughly 10 minutes)

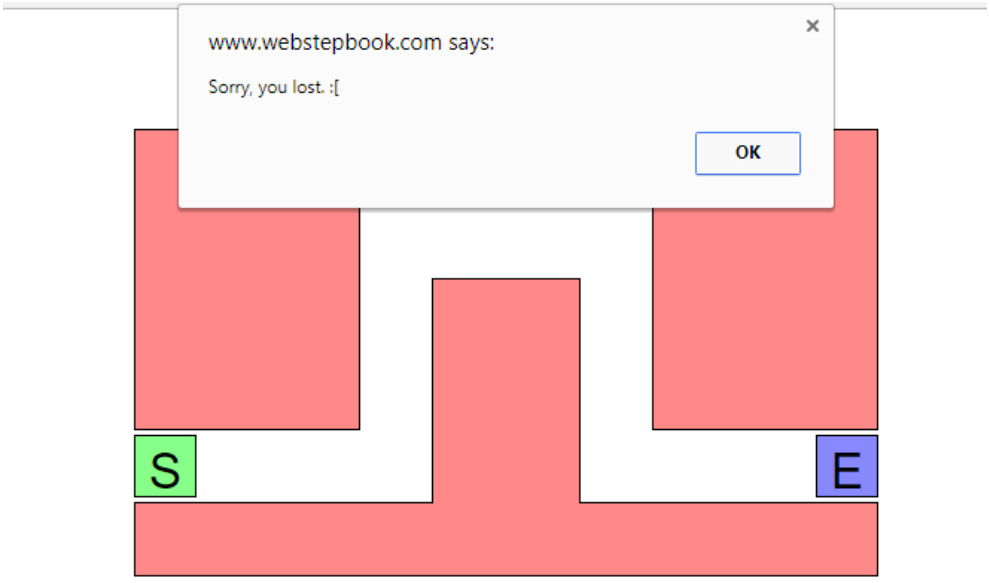
For your next task, make it so that all walls of the maze turn red when the mouse enters any one of them. To do this, you'll need to attach an event handler to each `div` representing a wall of the maze. It is a bit harder to select all of these `div`s, since they do not have `id` attributes. However, they do share a common `class` of `boundary`. You may want to look at the Lecture 11 slides about Prototype's `$$` function.



Exercise 3: Alerts on successful completion of maze (roughly 10 minutes)

Your next task is to make it so that if the user reaches the end of the maze, a congratulatory "You win!" alert message appears. The end of the maze is a `div` with an `id` of `end`.

It's not very difficult to create an event handler that pops up a message when the mouse touches the ending `div`. But note that you shouldn't pop up the "You win!" message unless the user makes it all the way to the end without touching any of the maze walls. You'll need to keep track of whether the user has hit any walls or not, and examine this information when the user touches the end square.



The object of this game is to guide the mouse cursor through the start area and get to the end area. Be sure to avoid the walls:

Good luck!

Exercise 4: Restartable maze (roughly 10 minutes)

One annoying thing you may be noticing as you test the maze so far is that it can't easily be reset to try again. So our next task will be to make it so that when the user *clicks* the mouse on the Start square (a `div` with an `id` of `start`), the maze state will reset. That is, if the maze boundary walls are red, they will all return to their normal color, so that the user can try to get through the maze again.