

CSCI110

Spring Session 2013

Assignment 2

Javascript

8 marks

Complete “Exercise 2” before attempting this assignment.

Aim

This assignment aims to introduce you to the basics of client side scripting (Javascript).

The assignment should be completed in the laboratory using the Ubuntu (Linux) environment.

Objectives

On completion of this assignment and its associated exercise, you will be able to:

- Define your own Javascript.
- Employ Javascript for checking form data prior to submission.
- Manipulate the DOM and CSS styling using Javascript
- Implement simple animations.

Tasks

You have to create two web pages that have associated Javascript scripts that perform tasks like form input checking, and animation with user interaction.

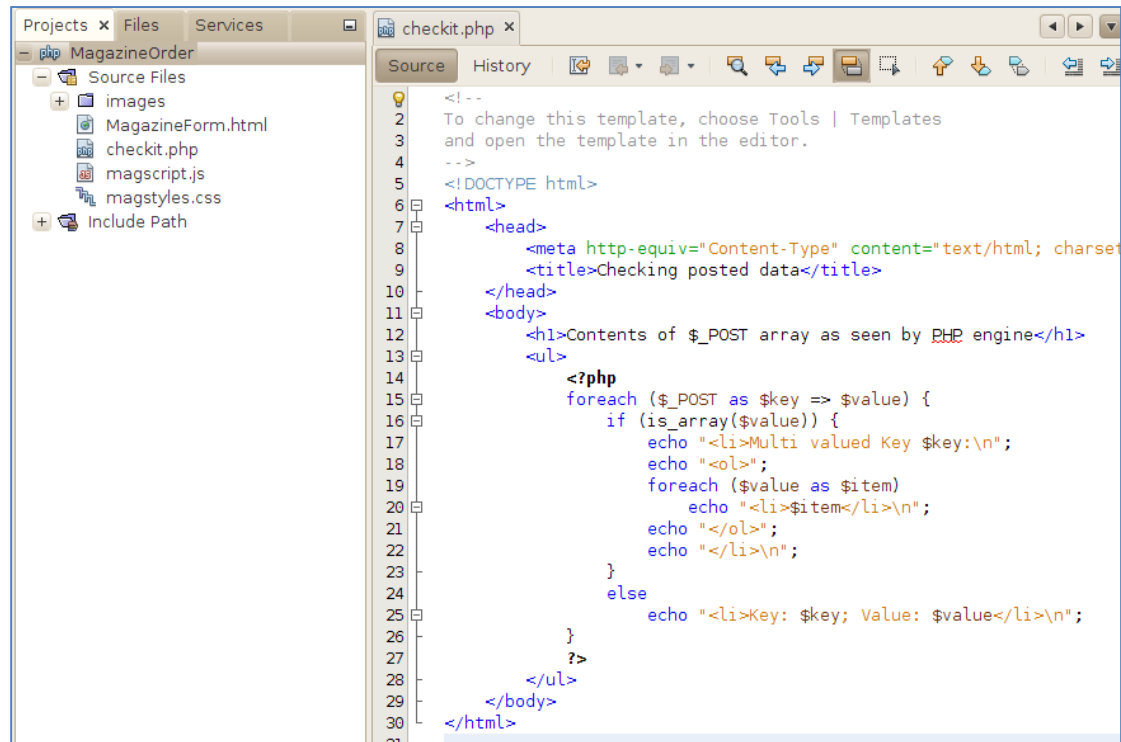
There are many Javascript libraries that could be exploited to simplify the tasks; we briefly cover jQuery later in session. These libraries contain things like animation functions. Normally, I encourage the use of libraries – *but not in this assignment*. As this is your first experience of Javascript coding, it is best if you code as much as possible yourself – rather than reduce the task to a sequence of library calls.

1. Magazine subscription form

The first web page supposedly represents a form in a web-site that allows visitors to submit orders for magazines.

This project consists of a single HTML page with several <div> sections whose visibility you control using Javascript, a CSS file, and a Javascript file. You can also include the PHP script “checkit.php” that can verify that your form would submit data.

NetBeans project for Part 1:



The HTML page is to have three <div> sections (which are all parts of a single form):































- A tabular display of the magazines that can be ordered. This is a HTML <table> with the table entries consisting of an link, some text, and a checkbox.
- A group of <fieldset> elements that contain <input> and <select> elements that allow the user to enter identification data, contact data, address, payment details.
- A final <div>, whose contents are mostly generated by your Javascript, that shows a summary of the order and has the form's submit button.

When the page is opened, only the first <div> section is displayed:

localhost/~nabg/2013-110/Part1/MagazineForm.html
Java Platform SE 7
New Tab

Magazine store

Select magazines

 Cosmopolitan; \$69.95 <input type="checkbox"/>	 Dolly; \$49.95 <input checked="" type="checkbox"/>	 Shop til you drop; \$69.95 <input checked="" type="checkbox"/>	 Vogue; \$69 <input type="checkbox"/>	 Woman's Day; \$76.70 <input type="checkbox"/>	 Women's Fitness; \$49.95 <input type="checkbox"/>	 Women's Weekly; \$69.95 <input type="checkbox"/>	 Australian Yoga Journal; \$60 <input type="checkbox"/>	 Disney Princess; \$29.95 <input type="checkbox"/>	 Girl Power; \$59.00 <input type="checkbox"/>
 Mindfood; \$65 <input checked="" type="checkbox"/>	 Recipes+; \$32 <input type="checkbox"/>	 TV Week; \$171.60 <input type="checkbox"/>	 TV Soap; \$45 <input type="checkbox"/>	 Scientific American; \$197 <input type="checkbox"/>	 Rolling Stone; \$64.95 <input type="checkbox"/>	 Camera; \$45 <input type="checkbox"/>	 The Wiggles; \$47 <input type="checkbox"/>	 Australian HiFi; \$49 <input type="checkbox"/>	 APC; \$79.95 <input type="checkbox"/>
 Bicycling Australia; \$64.95 <input type="checkbox"/>	 Inside Cricket; \$25 <input type="checkbox"/>	 Fishing World; \$90 <input type="checkbox"/>	 Golf Australia; \$89 <input type="checkbox"/>	 Rugby League Week; \$170 <input type="checkbox"/>	 Top Gear Australia; \$79.95 <input type="checkbox"/>	 Trail Rider; \$40 <input type="checkbox"/>	 Wheels; \$79.95 <input type="checkbox"/>	 House and Garden; \$69.95 <input type="checkbox"/>	 Xbox; \$69.95 <input type="checkbox"/>

Enter your details

The user may select magazines by clicking in the associated checkboxes. When he/she has made a selection, they can use the “enter detail” action button. This triggers a script. The script checks that at least one magazine has been selected (if no magazines are selected, it displays an alert suggesting that the user specify what they wish to purchase). If one or more magazines has been selected, the “choice” <div> is hidden and the “details” <div> is displayed.

(Magazine images can be “borrowed” from magshop.com.au – there are some in /share/cs-pub/110/A2stuff on banshee.)

Magazine store

Your details

Name			
Title	<input type="text" value="Miss"/>	Given name	<input type="text" value="Dizzy"/>
Middle initials	<input type="text" value="A"/>	Family name	<input type="text" value="Blonde"/>
Contacts			
Home phone	<input type="text" value="90919293"/>	Mobile	<input type="text" value="0414243444"/>
Email	<input type="text" value="dizzyb@uow.studentmail.edu.au"/>		
Address			
Address line 1	<input type="text" value="4/104 Market Street"/>		
Address line 2	<input type="text"/>		
City	<input type="text" value="Dapto"/>	Postcode	<input type="text" value="2522"/>
State	<input type="text" value="ACT"/>		
Payment			
Credit card	<input type="text" value="Mastercard"/>	Card number	<input type="text" value="5219442267829094"/>
CCV	<input type="text" value="666"/>		
Expiry Month	<input type="text" value="1 January"/>	Expiry Year	<input type="text" value="2014"/>
Go back to revise subscription choices		Go on to check then submit order	

This has fields for the following data:

- Title - <select> with options Mr, Mrs, Ms, ...
- Given name
- Initials
- Family name
- Home and mobile phone numbers
- Email
- Address – one line required, second line optional
- City
- Postcode
- State <select>
- Credit card <select>
- Card number and CCV value
- Expiry month and year (<select>)

It also has “go back” and “go on” action buttons that trigger Javascript functions. The function for “go back” simply hides the “details” <div> and re-displays the “choices” <div>. The function for “go on” triggers some data checking. If data checks fail, an alert is displayed giving details of problems in the data. If the checks are all passed, the script generates text summarising the order. This text is placed in the “innerHTML” content of <div> sections in the final part of the form.

The final section of the form page has <div> and elements that initially have no content text. Their content is filled in with an order summary and total cost value.

Magazine store

Confirm and submit order

Dolly	\$49.95
Shop til you drop	\$69.95
Mindfood	\$65

Delivery

Miss Dizzy Blonde
4/104 Market Street
Dapto
nsw 2522

Payment

Mastercard *****9094

Total cost of order \$184.9.

[Submit order](#)

[Go back to revise subscription choices](#)

[Go back to revise details](#)

The submit button will cause the form to post the data to the “checkit.php” script. This script just echoes a summary of the posted data.

Data checking code for “details” section of magazine form

The data checking code should enforce the following simple checks:

Given name and family name:

These `<input type='text' ... />` fields should each contain a text string that starts with a capital letter followed by some number of lower case letters.

Contacts

These fields for phones should contain a text string that consists solely of digits. The email field should contain a string that has a sequence of letters, digits, and underscores, an @ symbol, and another sequence of letters, digits, underscores and period characters.

Address data

The state is entered using a `<select>`. The postcode should contain 4 digits. The other address fields can contain letters, digits, spaces, hyphens, “/” symbols, or commas. Address line 1 and city must contain data; address line 2 may be left blank.

State and postcode:

The input of the state or territory is via a <selection> HTML element. Your code must check the consistency of the data in the State and postcode fields. The following rules apply to postcodes:

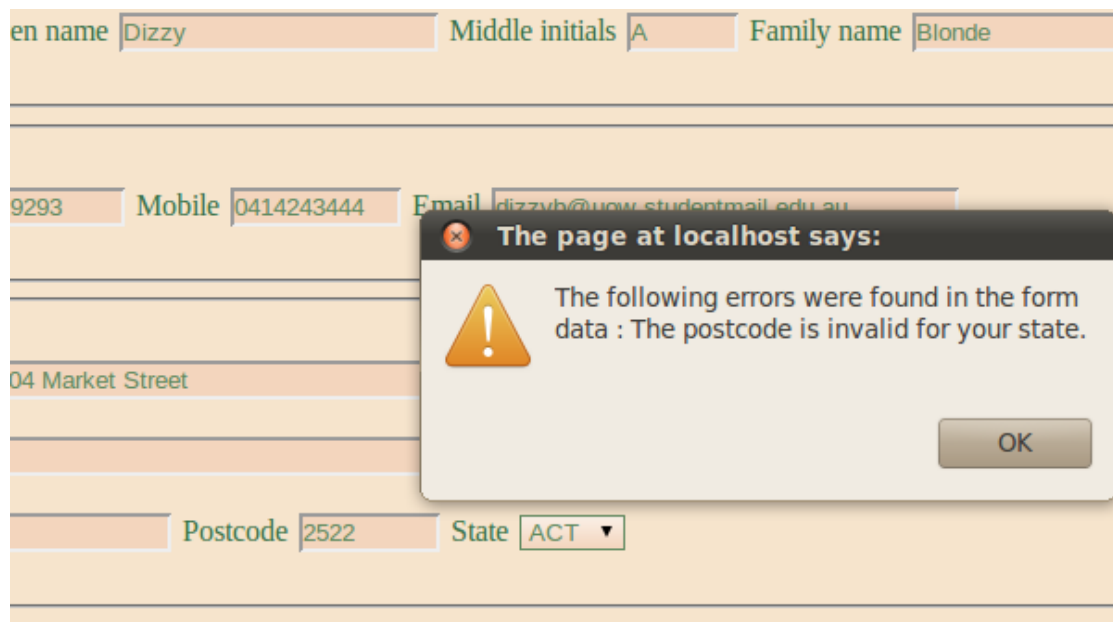
```
//Australian Capital Territory (ACT)      2600 to 2618 and 29##  
//New South Wales (NSW) 2###  
//Northern Territory (NT)      08## and 09##  
//Queensland (QLD)      4###  
//South Australia (SA) 5###  
//Tasmania (TAS) 7###  
//Victoria (VIC) 3###  
//Western Australia (WA)      6###
```

Payment

Cards from Mastercard, Visa, and American Express should be accepted. The card number should contain a sequence of 13-16 digits; the code should check that “Mastercard” numbers start with digit 5 while “Visa” numbers start with 4.

All the checks should be applied. If there are errors, an error message should be built up that contains information identifying all errors.

The checking function finishes by displaying an alert that either warns of errors and shows the error report, or switches to the submission <div> if the data appear satisfactory.



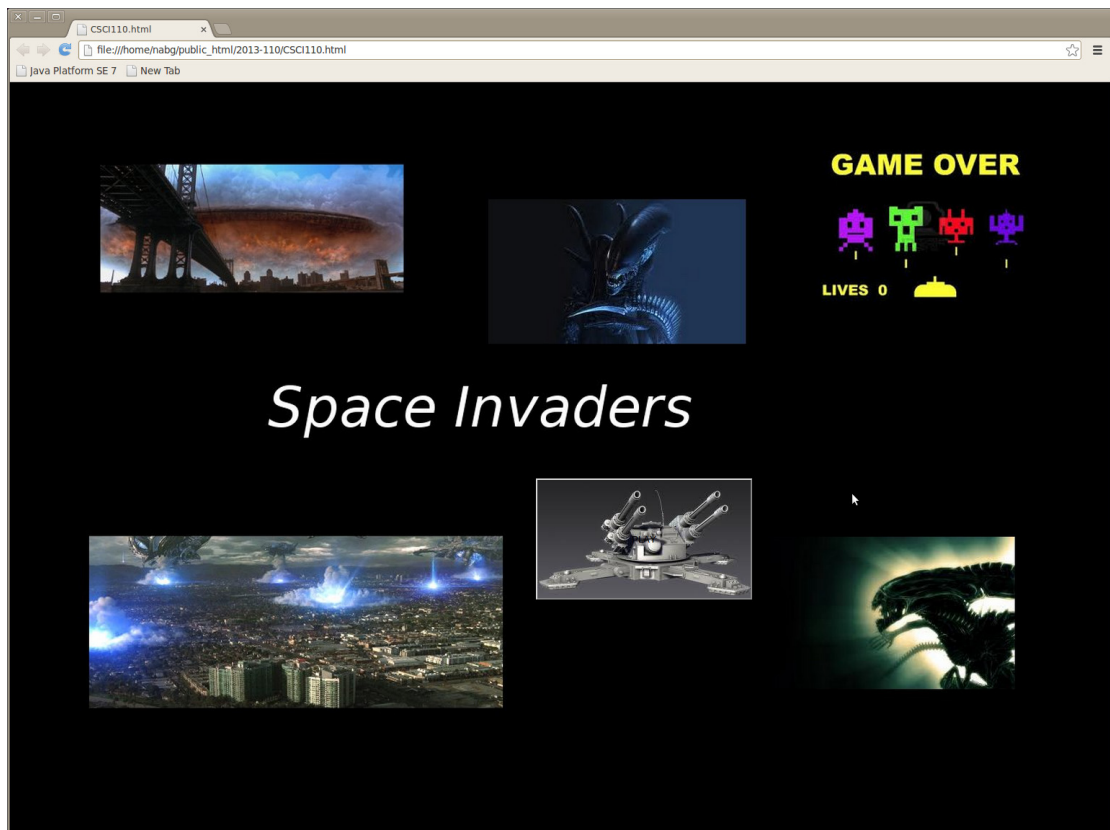
The screenshot shows a web form with several input fields. The fields are: 'First name' (value: Dizzy), 'Middle initials' (value: A), 'Family name' (value: Blonde), 'Postcode' (value: 2522), and 'State' (value: ACT). There is also a 'Mobile' field with value 0414243444 and an 'Email' field with value dizzyb@uow.studentmail.edu.au. A JavaScript alert box is displayed over the form, titled 'The page at localhost says:', with a warning icon and the message: 'The following errors were found in the form data : The postcode is invalid for your state.' The alert box has an 'OK' button.

(Don't implement the checks as one long function! Use lots of little functions called in turn from a main “checkdetails” function.)

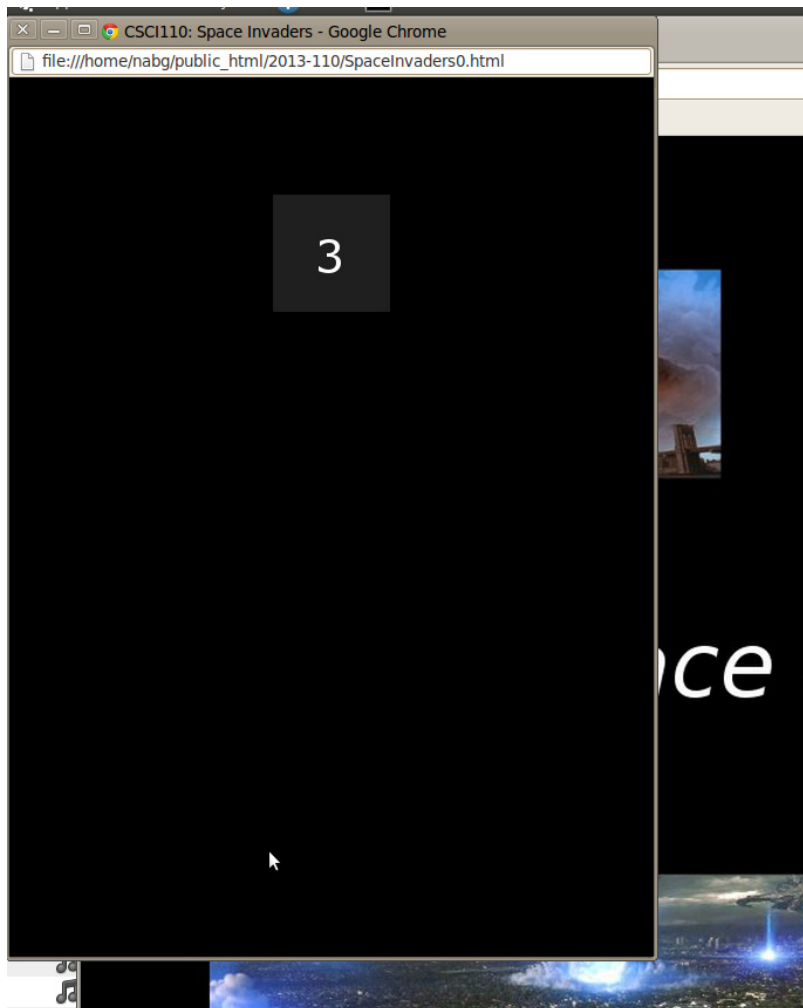
2. Space Invaders

These web pages and their associated Javascript code define a simple web based interactive game (the game is targeted at Chrome or Firefox browsers; the code may not work with other browsers).

The “Welcome” page simply has a button (the multi-barrelled cannon in view below) that when clicked opens a new window (a fixed size window without the usual toolbars etc) for actual game play:



The game window initially shows just a count-down timer (this is to allow the user to get ready to play). The count-down lasts 5 seconds.

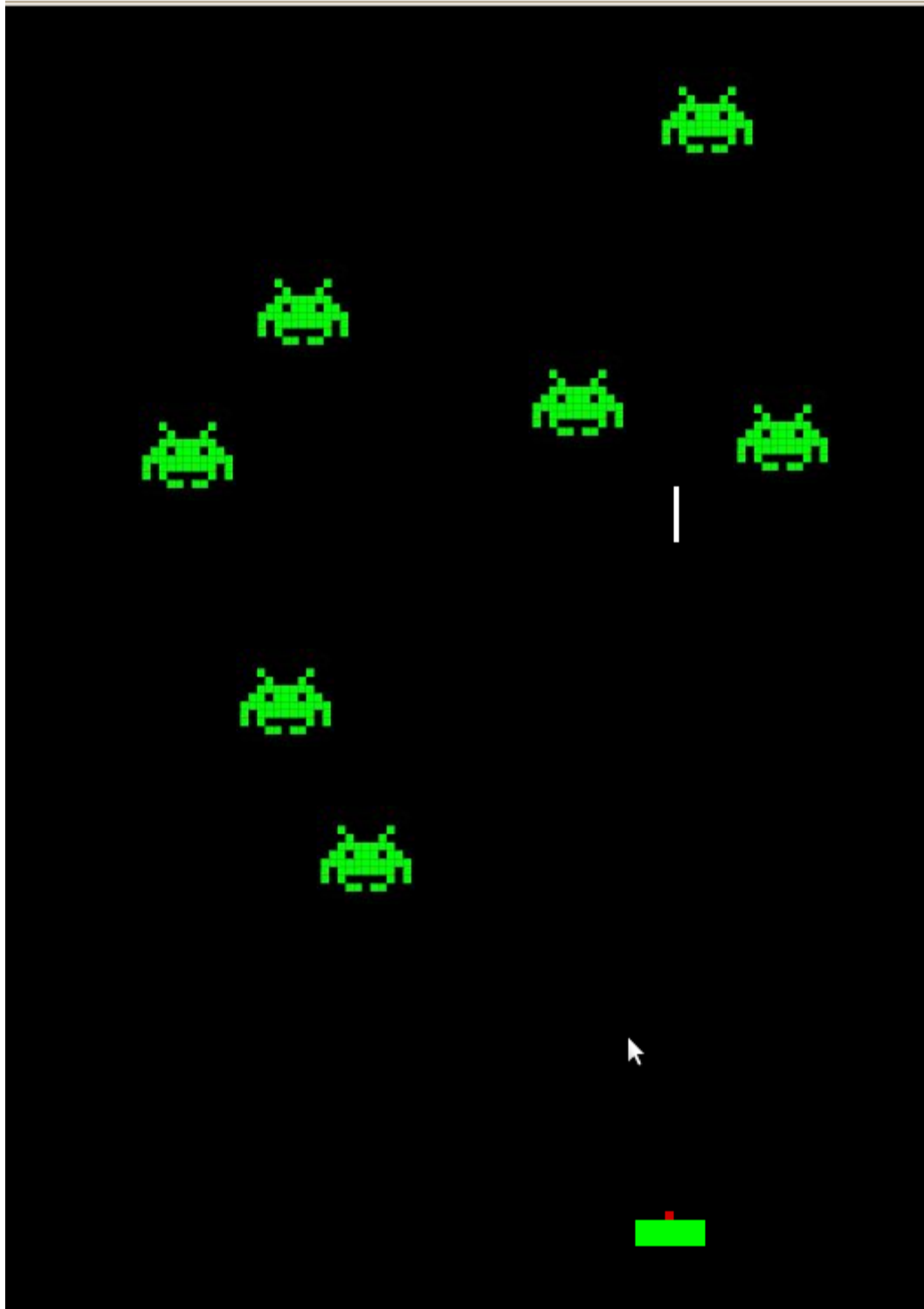


The game implements a simplified version of the classic space invaders arcade game.

The user controls a laser cannon, represented as a rectangular box near the bottom of the window. The cannon can move to left or right following the mouse cursor. Clicking the mouse button causes the cannon to fire. (Cannon fire can be represented by a brief beam of light.)

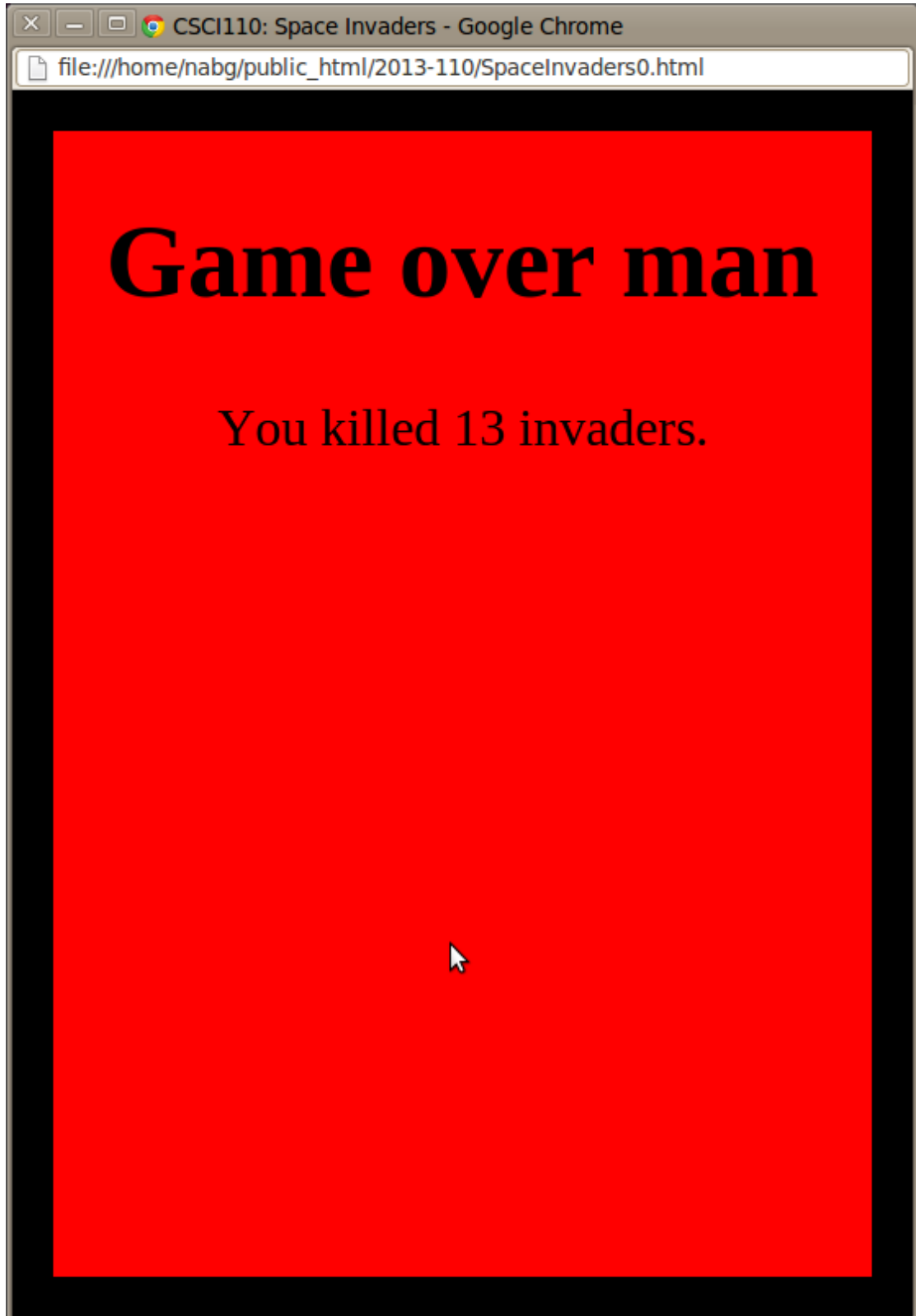
Invaders move down from the top of the screen. They move with different randomised vertical and horizontal velocities. Invaders may initially be moving either to left or right; when they approach the edge of the screen, their horizontal velocity is reversed.

file:///home/nabg/public_html/2013-110/SpaceInvaders0.html



If an invader is immediately above the laser cannon when it fires, that invader is “destroyed”. Its image is momentarily replaced by an image of an exploding invader. It is replaced by a new invader at the top of the screen. (It’s easiest to simply change its vertical position rather than destroying a structure and creating a new invader structure.)

If an invader reaches ground level – it’s “game over”:



The game over display is held for 4 seconds and then the window is closed by the Javascript code.

Only a simple implementation is required, one that is comparable to my demonstration version. (If you want to build something fancier, do the work over the

Xmas break; don't waste your time during session.) The details given below are specific to my version – you can change many of the minor details. Your game must work at least as effectively as mine.

Invaders' positions and movement:

Each invader is characterised by its (x,y) coordinates and by (x-vel,y-vel) velocity elements. These are initialized to appropriate random values. (It's easiest to model invaders as Javascript "structs" – each invader is a Javascript Object (new Object), that is given properties for x,y coordinates, velocities etc.)

Invader motion is handled using a "setInterval" function that will be called ~25 times per second. The function called will update the coordinates of each invader and check for an invader reaching ground level. If an invader would move outside the play area in next round, its horizontal velocity is reversed.

The game ends if an invader reaches ground level. Code for this eventuality removes all other timers, hides the <div> used for the game, shows a <div> with the results, and sets another timer that after 4 seconds will trigger Javascript code to close the window.

Animations:

The gun (an element inside its own <div>) is moved left or right following mouse movements (its style.left value is changed).

The gun <div> is within a <div> that represents the game. This gameboard <div> has an onclick() handler that controls gun fire, and has an event listener for mouse movements. The mouse movement event listener invokes the code to make the gun track to left or right following the mouse. (You will be moving the gun and firing – so the browser may see "*mouse movement with button down*" which it interprets as a "**drag**" action. This might result in distracting "drag feedback" visuals. You can suppress this in most browsers by adding the attribute ondragstart="return false;" to the HTML element.)

Sound effects:

Sound effects are problematic due in part to great variations amongst browsers and the need for browser plugins that support the sound format selected. Sound is therefore **not** a requirement in this assignment – but you can experiment with it if you wish.

The following code fragments use Javascript Audio objects. Not all browsers support Audio objects – but the code works well enough with Chrome and Firefox.

You create Audio objects and get them to read audio files – many formats are supported, including ".wav". Some audio files are marked as "looping" – switch this off unless you really want it. The following Javascript code fragment loads two sound files from same directory as the script (some of the invader sound files were "looping" – so for them the Audio object's loop property is explicitly set to false).

```
gunsound = new Audio("shoot.wav");
invadersound1 = new Audio("fastinvader1.wav");
invadersound1.loop = false;
```

You play a sound using the play() method (what else). (You can find the full Audio API on the web – do a Google search for Javascript Audio API.)

```
function fireGun() {
    if (gameinprogress) {
        showflash.style.left = gc + "px";
        showflash.style.display = "block";
        gunsound.play();
    }
}
```

An Audio object can generate an event when it finishes playing. This can be used in code that plays a series of different sounds. You define a function, e.g. playsound(), that will select and start the next sound, and use an event listening mechanism to arrange that this function is called when any given sound ends –

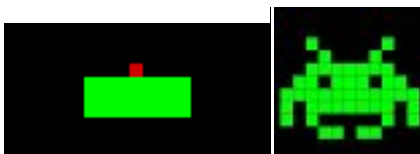
```
invadersound2 = new Audio("fastinvader2.wav");
invadersound2.loop = false;
invadersound2.addEventListener('ended', playsound);
invadersound3 = new Audio("fastinvader3.wav");
invadersound3.loop = false;
invadersound3.addEventListener('ended', playsound);
```

The playsound() function simply picks the next sound and starts it playing.

Hints:

This assignment does not utilise <svg>, <canvas>, nor any other HTML5 extension.

1. The “Welcome” window has an action button with a trivial bit of Javascript to open the game window.
The w3schools site has documentation on how to use Javascript to open a new window and select features such as fixed size, no tool bar etc.
2. The game window will have <div> sections for the game and the “game over” report. Their CSS styling attributes can be adjusted to select which is to be displayed (element.style.display = “block” | “none”).
The game script is associated with this window.
3. The count-down effect is scripted using setTimeout and a function that displays an image with a number in the centre of the game area.
4. The gun and the invaders are elements.



Your Javascript code will change style attributes (left, top).
Code setting positions must use “px”, e.g.

```
function moveaninvader(ndx) {  
    var obj = army[ndx];  
    obj.x = obj.x + obj.horiz;  
    obj.y = obj.y + obj.vert;
```

5. CSS styling for appropriate elements specifies that their positioning is “absolute”.
6. Mouse tracking – it’s somewhat browser dependent. You have to add a mousemove listener to an appropriate element in your HTML page. The W3C standard specifies an addListener function (arguments are name of function that is to be called, name of event, and a control for event propagation (an advanced feature)). IE of course differs:

```
// for IE compatability (probably unnecessary, as it won't run  
// on IE - no HTML5 Audio() object.  
if (!gameboard.addEventListener) {  
    gameboard.attachEvent("onmousemove", moveGun);  
}  
else {  
    gameboard.addEventListener("mousemove", moveGun, false);  
}
```

7. The function that handles mouse movement may again need some browser specific coding. (Browsers differ as to how they pass the actual event data to a function.) The following should allow your code to get at the mouse’s x-position in Firefox, Chrome, and IE:

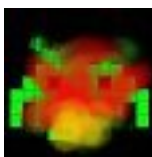
```
function moveGun(evt) {  
    if (!evt)  
        var evt = window.event;  
    if (!gameinprogress)  
        return;  
    // Try to keep centre of gun aligned with mouse pointer  
    // but do not allow paddle to move outside frame  
    var x = evt.clientX;
```

8. My project? Well, it had the following Javascript functions:

- ● checkendgame() : undefined
- ● clocktick() : unresolved
- ● createInvaders() : undefined
- ● fireGun() : undefined
- ● gameplay() : undefined
- ● goaway() : undefined
- ● makeinvader(ndx) : Object
- ● moveGun(evt) : unresolved
- ● moveaninvader(ndx) : undefined
- ● moveinvaders() : undefined
- ● playsound() : unresolved
- ● positioninvaders() : undefined
- ● showinvaders(setting) : undefined
- ● startgame() : undefined
- ● startup() : undefined

Development strategy:

1. Create the Welcome page that pops up a fixed size window.
2. Experiment with setTimeout – make the pop-up window disappear after a fixed time.
3. Create a simplified version of the game – a box with an invader. The box is simply an absolutely positioned <div> with a background colour! The single invader is an <div> containing an . Use a setInterval timer. Update the position so that the invader moves from side to side as it descends down the screen, and have it start again at the top after it reaches the bottom.
4. Create a listener for a mousemove function and code up a gun that moves from side to side in the game area (box) (make sure it cannot move out of the game area).
5. Elaborate your code so that there are several invaders.
6. Add gunfire – code that checks whether an invader is “killed”, code that provides visual feedback (show an that represents the gun’s laser bolt – hide it again at the start of the next setInterval processing; if appropriate show an exploding invader



7. Implement end game reporting and window closure.
8. Maybe experiment with sound effects and other elaborations.

Assignment report

You do not submit your web pages!

You are marked on a report that you write. This report presents your web site in the form of screen-shots, the Javascript files, the HTML markup and style sheets, along

with supplementary explanation and comment. Code listings should be included with proper formatting – use the “Print as HTML” option in NetBeans to get formatted versions that can be pasted into a report. The report should be prepared on a word processor and converted to PDF format. The Open Office word processor on Ubuntu Linux has a “print to PDF” option. There should be Word to PDF converters on the Windows OS. Ubuntu has a snapshot tool (in its accessories menu). Screenshots should be scaled down when pasted into reports.

Submission

Prepare your report and convert to PDF format as the file A2.pdf.

The report should be submitted electronically via the `turnin` system. For this assignment you submit your assignment via the command:

```
turnin -c csci110 -a 2 A2.pdf
```

Late submissions would be submitted as:

```
turnin -c csci110 -a 2late A2.pdf
```

The program `turnin` only works when you are logged in to the main banshee undergraduate server machine. As in assignment 1, you must transfer your report to the banshee machine using some form of ftp. Then from an Ubuntu workstation in the lab, you must open a terminal session on the local machine, and then login to banshee via ssh and run the `turnin` program.

Marking

The assignment is worth 8 marks total.

- Appearance of report: 1 mark
Cover, index, proper sectioning, brief explanation of task in each section, images, neatness of included code, ...
- Data entry checking: 3 marks
HTML form page with linked Javascript files; HTML for form; correct Javascript; evidence for operation from screen shots etc.
- Space Invaders: 4 marks
HTML pages with linked Javascript file; HTML and CSS listing; correct Javascript; evidence for operation from screen shots etc.