Technical Log

By Neil Rafferty (B00451753)

Contents

Introduction

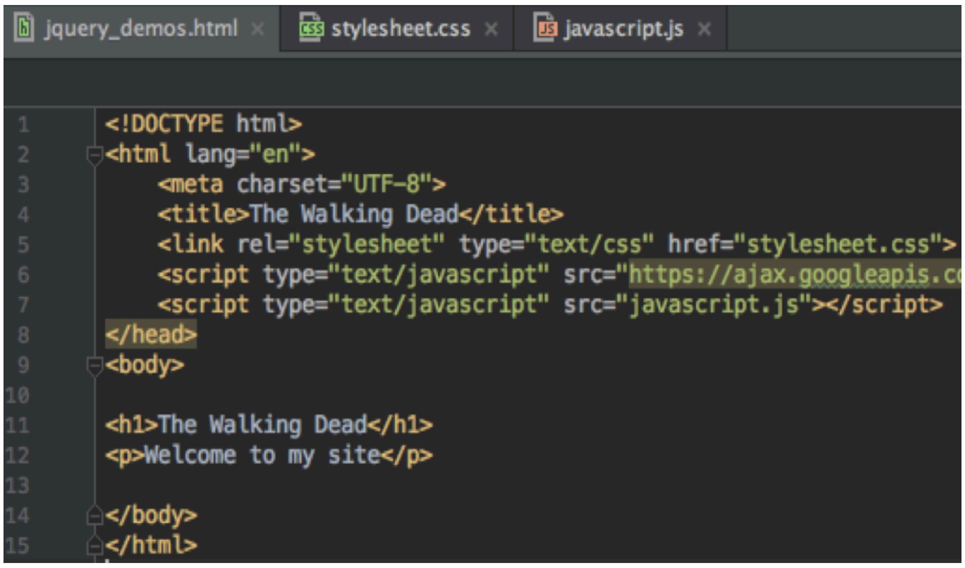
The purpose of this log is to document the progress in creating a rich internet application, to illustrate its' features and to briefly outline how they were achieved (please refer to the comments in the code for more in depth descriptions). The Walking Dead TV series was selected as the content for this project. This is a piece of coursework for interactive web computing, a final year software engineering module.

Note: This application has been made to work in google chrome.

Project Structure

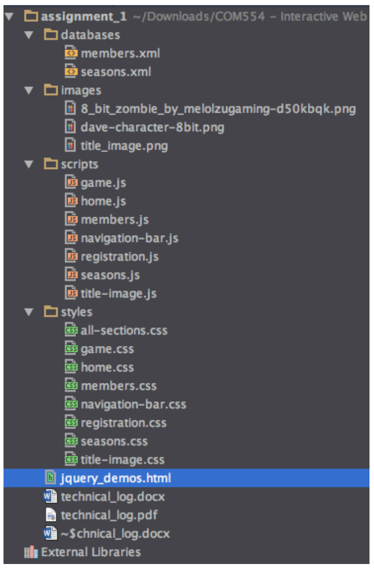
A new public GitHub repository was created to host the code and potentially provide a reference of work to future employers (https://github.com/stormrage-neilr/COM_554_Assignement_1). *- Friday the 7th of October.*

A project skeleton was then created. The skeleton consisted of three files shown at the top of figure 1. The html file connects the three files. Line 5 of the code shows where the CSS file has been linked to the html (*using the relative directory location*). Line 6 contains the import for the jQuery library *(this is from a google server online)* and line 7 contains the import for the local JavaScript file *(this file is currently empty)*. This creates a basic spike through the three main programming languages. *- Saturday the 15rd of October.*



```
1   <!DOCTYPE html>
2   <html lang="en">
3       <meta charset="UTF-8">
4       <title>The Walking Dead</title>
5       <link rel="stylesheet" type="text/css" href="stylesheet.css">
6       <script type="text/javascript" src="https://ajax.googleapis.co
7       <script type="text/javascript" src="javascript.js"></script>
8   </head>
9   <body>
10
11  <h1>The Walking Dead</h1>
12  <p>Welcome to my site</p>
13
14  </body>
15  </html>
```

Figure 1: *Initial File and HTML Structure.*



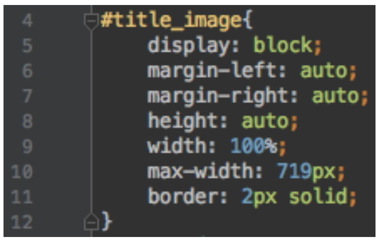
```
assignment_1 ~/Downloads/COM554 - Interactive Web
    databases
        members.xml
        seasons.xml
    images
        8_bit_zombie_by_melolzugaming-d50kbqk.png
        dave-character-8bit.png
        title_image.png
    scripts
        game.js
        home.js
        members.js
        navigation-bar.js
        registration.js
        seasons.js
        title-image.js
    styles
        all-sections.css
        game.css
        home.css
        members.css
        navigation-bar.css
        registration.css
        seasons.css
        title-image.css
    jquery_demos.html
    technical_log.docx
    technical_log.pdf
    ~$chnical_log.docx
External Libraries
```

Figure 2: *Final structure.*

Since, the file structure has developed into what is shown in figure 2. This structure is easier to navigate than when large amounts of information are stored in the same folder or file.

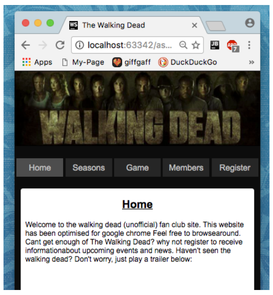
Title Image

The h1 and p tags in figure 1 were then replaced with a title image *(The Walking Dead Poster 51, Blogspot.com)* and a menu made of list items. The image was centred on the screen by setting the display to block and the horizontal margins to auto scale. The width was set to 100%, but the maximum width was set to 719 pixels. This means that the image scales with the screen until it reaches its actual image size and then it stops. The height was set to auto scale with the width so that no stretching of the image occurred. This code can be seen in figure 3 and figure 4 and 5 demonstrate its effect. - *Sunday the 16rd of October.*



```
#title_image{
    display: block;
    margin-left: auto;
    margin-right: auto;
    height: auto;
    width: 100%;
    max-width: 719px;
    border: 2px solid;
}
```

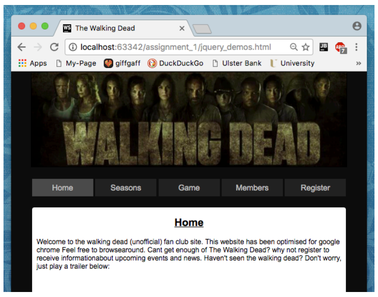
Figure 3: *Title Image CSS.*



Home

Welcome to the walking dead (unofficial) fan club site. This website has been optimised for google chrome Feel free to browsearound. Cant get enough of The Walking Dead? why not register to receive informationabout upcoming events and news. Haven't seen the walking dead? Don't worry, just play a trailer below:

Figure 4: *Small Screen.*



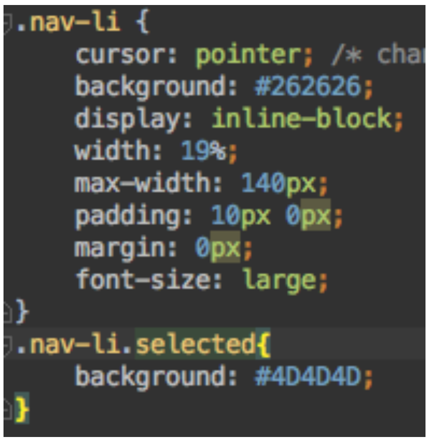
Home

Welcome to the walking dead (unofficial) fan club site. This website has been optimised for google chrome Feel free to browsearound. Cant get enough of The Walking Dead? why not register to receive informationabout upcoming events and news. Haven't seen the walking dead? Don't worry, just play a trailer below:

Figure 5: *Large Screen.*

Navigation Bar

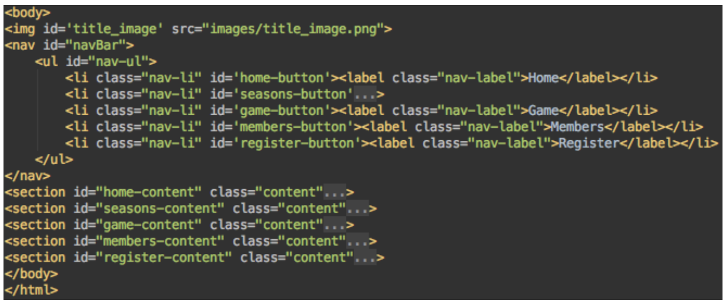
The navigation bar is made up of an unordered list inside a nav element. Each list item has been created to scale in a similar way to the title image. The cursor icon has been changed to a pointer to signify that the list item has a function (this will be consistent with other clickable elements in the project). The display of the list items has been set to inline-block. This removes their bullet points and positions them horizontally as opposed to the default vertical value. And finally a pseudo class called hover was used to highlight the item that the pointer is above. - *Sunday the 16rd of October.*



```
.nav-li {
    cursor: pointer; /* cha
    background: #262626;
    display: inline-block;
    width: 19%;
    max-width: 140px;
    padding: 10px 0px;
    margin: 0px;
    font-size: large;
}
.nav-li.selected{
    background: #4D4D4D;
}
```

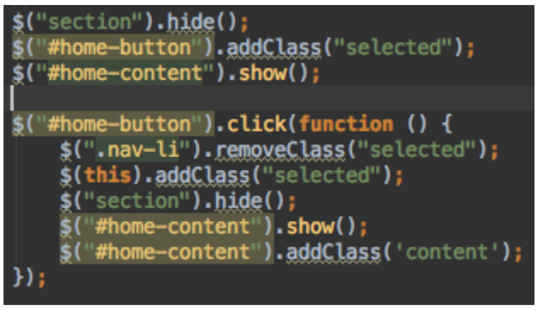
Figure 6: *List Item CSS.*

The next piece of functionality implemented was the ability to navigate between content. This is the first time that JavaScript or jQuery has been used in the project. Upon loading the page all section elements shown in figure 7 are hidden. This happens in the first line of code in figure 8. Then the home menu item has the 'selected' CSS class styles applied and the home section is shown. This is because it is the default page. A click event has been added to the menu item so that it will change itself to the selected element and show only the related content. It does this by hiding all section elements first then showing the relevant content. This code demonstrates use of the show/hide and add/remove class methods.



```
<body>
<img id='title_image' src="images/title_image.png">
<nav id="navBar">
    <ul id="nav-ul">
        <li class="nav-li" id='home-button'><label class="nav-label">Home</label></li>
        <li class="nav-li" id='seasons-button'...>
        <li class="nav-li" id='game-button'><label class="nav-label">Game</label></li>
        <li class="nav-li" id='members-button'><label class="nav-label">Members</label></li>
        <li class="nav-li" id='register-button'><label class="nav-label">Register</label></li>
    </ul>
</nav>
<section id="home-content" class="content"...>
<section id="seasons-content" class="content"...>
<section id="game-content" class="content"...>
<section id="members-content" class="content"...>
<section id="register-content" class="content"...>
</body>
</html>
```

Figure 7: *HTML navigation bar and sections structure.*



```
$("section").hide();
$("#home-button").addClass("selected");
$("#home-content").show();

$("#home-button").click(function () {
    $(".nav-li").removeClass("selected");
    $(this).addClass("selected");
    $("section").hide();
    $("#home-content").show();
    $("#home-content").addClass('content');
});
```

Figure 8: *Navigation Functionality.*

Registration Form

The registration form in figure 9 makes use of HTML5. The input types have been set to text, email, date and checkbox (as shown in figure 10). In HTML5 and in certain browsers, an input element with the type date will be presented with a date picker. However, these input types are perhaps more useful when using them for validation. By adding the word required into our input tag we can now automatically validate the user enters data of that type. This form will not allow users so submit it before they have entered a valid piece of text for a name and surname, a piece of text in email format for the email address and a date in the date

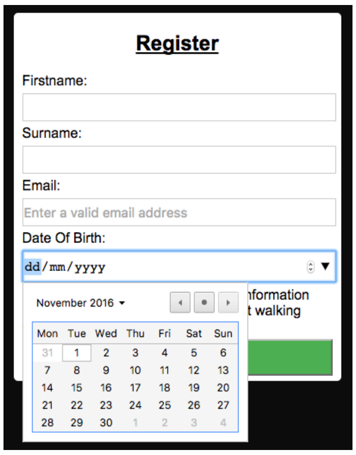
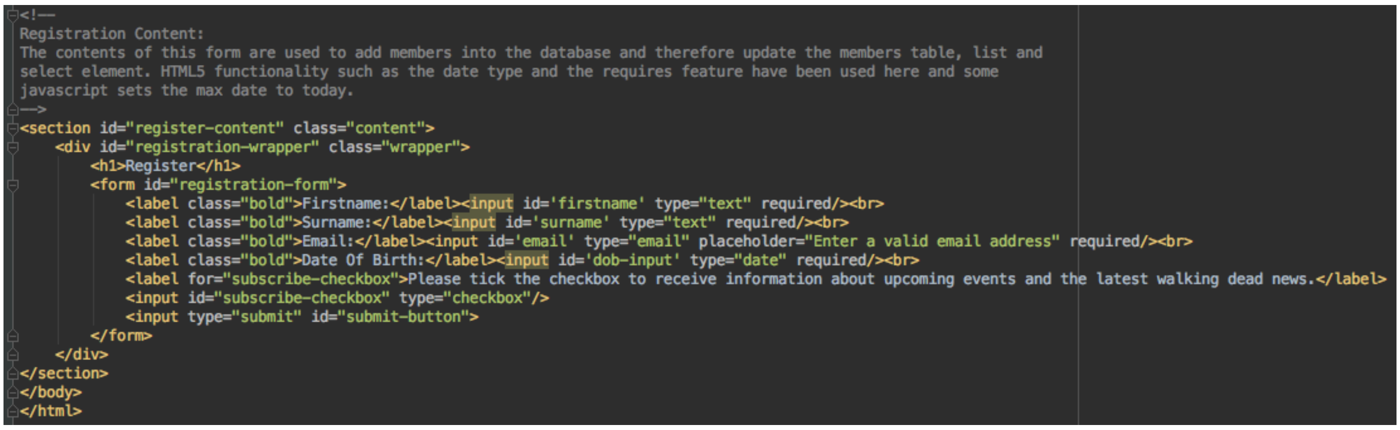


Figure 9: *Registration Form.*

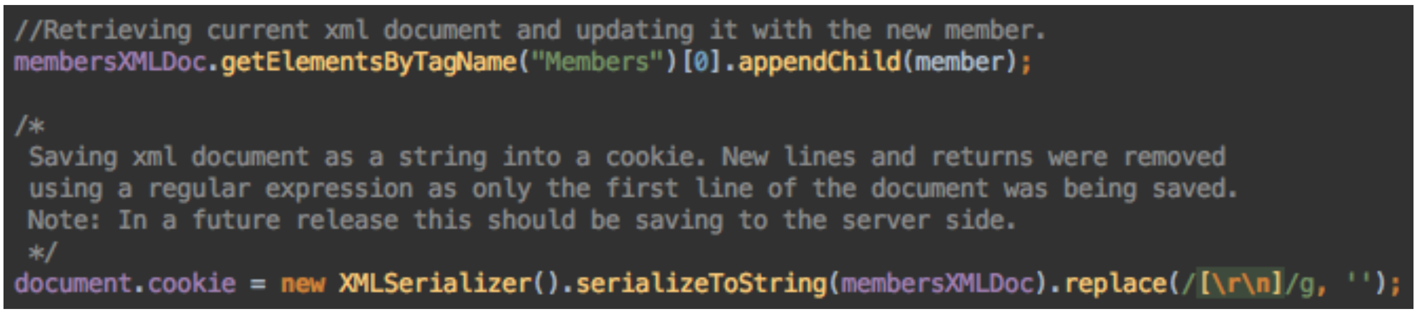
of birth field. There is also an input attribute called max. This attribute had been set in the date of birth field to todays' date using jQuery. This means that the form will not accept a future date of birth. The code for this can be found in registration.js. *– Wednesday the 26rd of October*



```
<!--
Registration Content:
The contents of this form are used to add members into the database and therefore update the members table, list and
select element. HTML5 functionality such as the date type and the requires feature have been used here and some
javascript sets the max date to today.
-->
<section id="register-content" class="content">
    <div id="registration-wrapper" class="wrapper">
        <h1>Register</h1>
        <form id="registration-form">
            <label class="bold">Firstname:</label><input id='firstname' type="text" required/><br>
            <label class="bold">Surname:</label><input id='surname' type="text" required/><br>
            <label class="bold">Email:</label><input id='email' type="email" placeholder="Enter a valid email address" required/><br>
            <label class="bold">Date Of Birth:</label><input id='dob-input' type="date" required/><br>
            <label for="subscribe-checkbox">Please tick the checkbox to receive information about upcoming events and the latest walking dead news.</label>
            <input id="subscribe-checkbox" type="checkbox"/>
            <input type="submit" id="submit-button">
        </form>
    </div>
</section>
</body>
</html>
```

Figure 10: *Registration Form.*

After submitting a valid registration form the information is processed into xml nodes that are placed in the correct structure to match a DTD XML document variable retrieved from members.xml by a XMLHttpRequest. The xml nodes, which have now constructed a new member instance, are inserted into this xml document variable which is then stored as a cookie. The method below (figure 11) shows how this new xml member is stored. *– Saturday the 29th of October.*

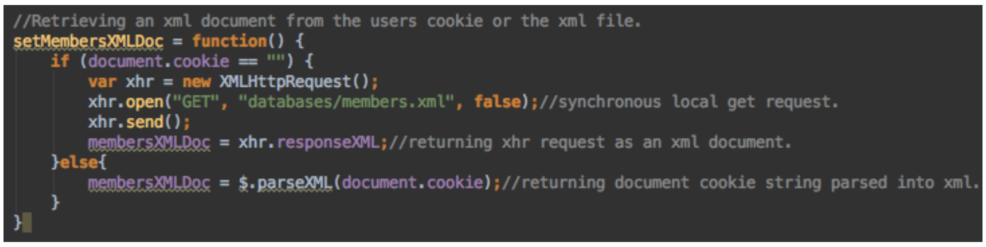


```
//Retrieving current xml document and updating it with the new member.
membersXMLDoc.getElementsByTagName("Members")[0].appendChild(member);

/*
 Saving xml document as a string into a cookie. New lines and returns were removed
 using a regular expression as only the first line of the document was being saved.
 Note: In a future release this should be saving to the server side.
 */
document.cookie = new XMLSerializer().serializeToString(membersXMLDoc).replace(/[\r\n]/g, '');
```

Figure 11: *Saving Updated Member Data to Cookie.*

Members Section

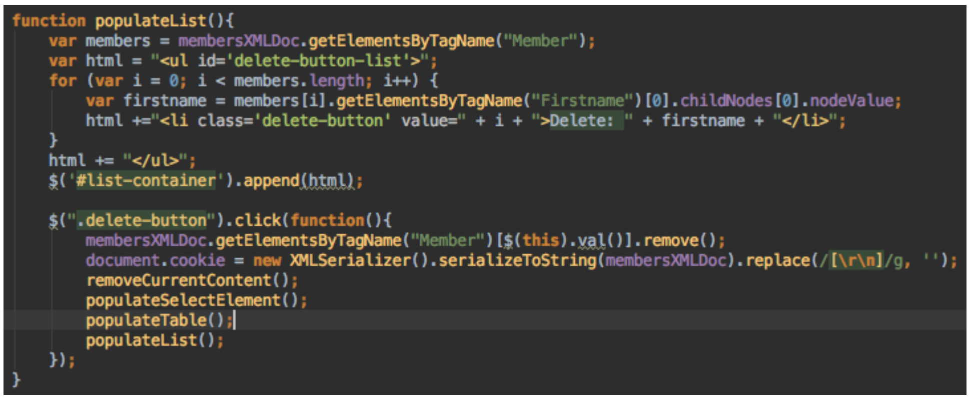
The members.xml file was created to provide some default data for populating the member table, list and select elements. However, it will only be used if there is no existing members' data stored as a cookie. Figure 12 shows how the current data is loaded, be it via XMLHttpRequest or parsing a cookie string.



```
//Retrieving an xml document from the users cookie or the xml file.
setMembersXMLDoc = function() {
    if (document.cookie == "") {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "databases/members.xml", false);//synchronous local get request.
        xhr.send();
        membersXMLDoc = xhr.responseXML;//returning xhr request as an xml document.
    }else{
        membersXMLDoc = $.parseXML(document.cookie);//returning document cookie string parsed into xml.
    }
}
```

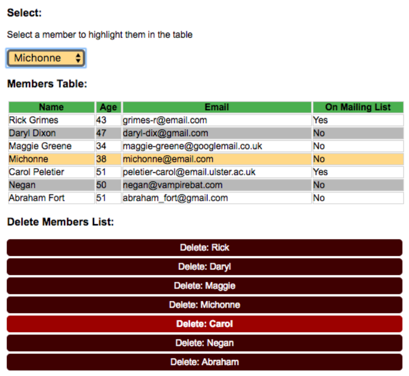
Figure 12: *Loading members' data.*

After the data has been loaded, all member instances are placed inside an array. This array is then used to concatenate a HTML string that contains a populated list, table and select element. This string is then appended to the given elements container when the members' menu item is clicked. Figure 13 shows how the list is populated in the first half of the code. In the second half of this method a click event has been



```
function populateList(){
    var members = membersXMLDoc.getElementsByTagName("Member");
    var html = "<ul id='delete-button-list'>";
    for (var i = 0; i < members.length; i++) {
        var firstname = members[i].getElementsByTagName("Firstname")[0].childNodes[0].nodeValue;
        html +="<li class='delete-button' value=" + i + ">Delete: " + firstname + "</li>";
    }
    html += "</ul>";
    $('#list-container').append(html);

    $(".delete-button").click(function(){
        membersXMLDoc.getElementsByTagName("Member")[$(this).val()].remove();
        document.cookie = new XMLSerializer().serializeToString(membersXMLDoc).replace(/[\r\n]/g, '');
        removeCurrentContent();
        populateSelectElement();
        populateTable();
        populateList();
    });
}
```

Figure 13: *Populating List Method.*



Select:

Select a member to highlight them in the table

Michonne

Members Table:

Name	Age	Email	On Mailing List
Rick Grimes	43	grimes-r@email.com	Yes
Daryl Dixon	47	daryl-dix@gmail.com	No
Maggie Greene	34	maggie-greene@googlemail.co.uk	No
Michonne	38	michonne@email.com	No
Carol Peletier	51	peletier-carol@email.ulster.ac.uk	Yes
Negan	50	negan@vampirebat.com	No
Abraham Fort	51	abraham_fort@gmail.com	No

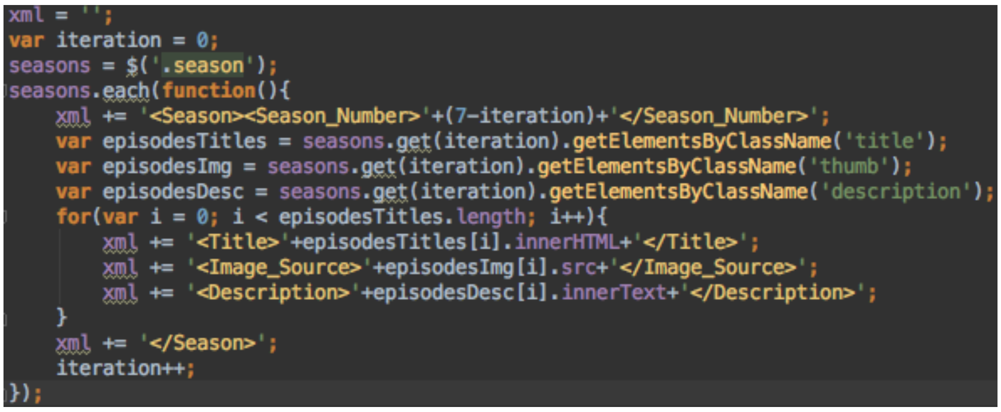
Delete Members List:

Figure 14: *List, Table & Select*

added to each new li element. This event triggers the removal of the associated member from the cookie data and repopulates the list, table and select options. In a similar way a change event has been bound to the select element. This event highlights the relative member in the table by adding and removing a selected highlighted class to the table row elements. Michonne is an example of this in figure 14. Note: The CSS pseudo class nth-child(odd) was used to colour alternative rows in the table.

Seasons Section

The seasons section is a demonstration of multiple accordions. The data used to populate these accordions is loaded from the seasons.xml file in the same way that the data used in the members' section was loaded from members.xml. As well, the season and episode containers are created using the same jQuery techniques as is demonstrated in figure 13. Functionality used to toggle different slide animations is added to these containers using jQuery. The seasons section takes advantage of the large amount of data in the seasons.xml file. This data has come from http://www.tv.com. The following code snippet (figure 15) was created and entered into the browser console when on this site. The snippet then concatenates a lengthy xml string (stored in the xml variable) which was copied and saved into the seasons.xml file.

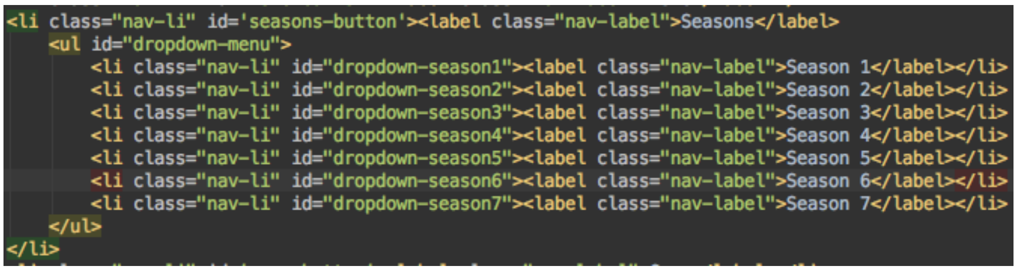


```
xml = '';
var iteration = 0;
seasons = $('.season');
seasons.each(function(){
    xml += '<Season><Season_Number>'+(7-iteration)+'</Season_Number>';
    var episodesTitles = seasons.get(iteration).getElementsByClassName('title');
    var episodesImg = seasons.get(iteration).getElementsByClassName('thumb');
    var episodesDesc = seasons.get(iteration).getElementsByClassName('description');
    for(var i = 0; i < episodesTitles.length; i++){
        xml += '<Title>'+episodesTitles[i].innerHTML+'</Title>';
        xml += '<Image_Source>'+episodesImg[i].src+'</Image_Source>';
        xml += '<Description>'+episodesDesc[i].innerText+'</Description>';
    }
    xml += '</Season>';
    iteration++;
});
```

Figure 15: *seasons.xml Generating Code Snippet.*

Seasons Dropdown Menu

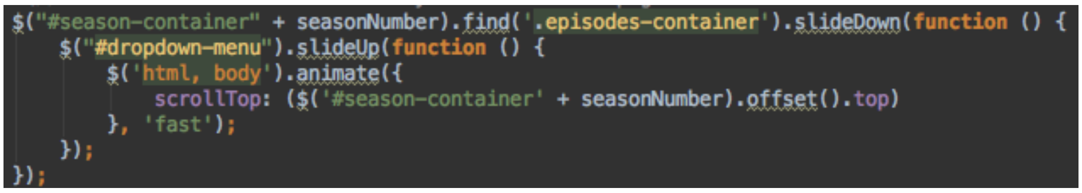
A dropdown menu was created to demonstrate more advanced menu functionality. The menu was created in the HTML as a child element of the season menu item as shown in figure 16. The functionality bound to



```
<li class="nav-li" id='seasons-button'><label class="nav-label">Seasons</label>
    <ul id="dropdown-menu">
        <li class="nav-li" id="dropdown-season1"><label class="nav-label">Season 1</label></li>
        <li class="nav-li" id="dropdown-season2"><label class="nav-label">Season 2</label></li>
        <li class="nav-li" id="dropdown-season3"><label class="nav-label">Season 3</label></li>
        <li class="nav-li" id="dropdown-season4"><label class="nav-label">Season 4</label></li>
        <li class="nav-li" id="dropdown-season5"><label class="nav-label">Season 5</label></li>
        <li class="nav-li" id="dropdown-season6"><label class="nav-label">Season 6</label></li>
        <li class="nav-li" id="dropdown-season7"><label class="nav-label">Season 7</label></li>
    </ul>
</li>
```

Figure 16: *Dropdown Menu HTML.*

these elements retrieve the last character of the elements' inner text (the season number). It executes a number of animations (including a fade animation) and by using this number it navigates the user to the selected seasons' content. Figure 17 shows how 3 sequential animations are triggered. The animation methods take a function parameter which is executed on completion of the animation. Animations were chained by placing subsequent animations in these functions. *– Saturday the 31st of October*



```
$("#season-container" + seasonNumber).find('.episodes-container').slideDown(function () {
    $("#dropdown-menu").slideUp(function () {
        $('html, body').animate({
            scrollTop: ($('#season-container' + seasonNumber).offset().top)
        }, 'fast');
    });
});
```

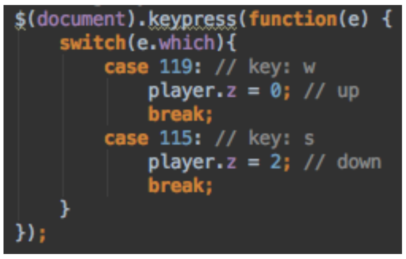
Figure 17: *Chaining Animations*

Trailer Tabs

Three iframes containing TWD trailer sources where placed in the home container. Three corresponding tab buttons were placed above them. By calling the toggle method on the iframes from the button click, in a similar way to the main menu, the iframe containers show and hide relative to the users' selection.

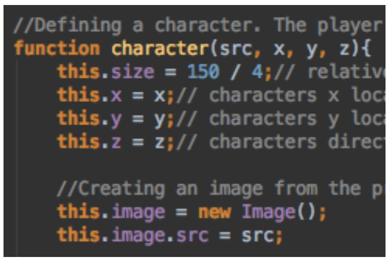
Game

The game was made using a canvas to draw on, the set interval method (for the game loop) and a key listener (figure 18). A character template was created that took an image source, x/y coordinate and a direction (figure 19). The character object was given an update method that changes its x and y coordinates depending on its direction. The set interval method continually calls the update method of all characters. It creates new enemies at a steadily decreasing interval. The canvas 2d context is used to draw the character images (taken from deviantart.net and indiedb.com) on the canvas at their x and y coordinates within this loop. If the user presses the 'w' or 's' key the direction of the player will change to up or down respectively. At the end of each iteration a collision check is performed (figure 20) on each enemy. If the player's area overlaps with an enemy the 'Game Over' (figure 21) text is displayed and a return statement is used to break out of the loop. However, the start button resets all the variable so the game can be started again. The w3schools game demo was a helpful reference point when creating this game. *- Tuesday the 1st of November*



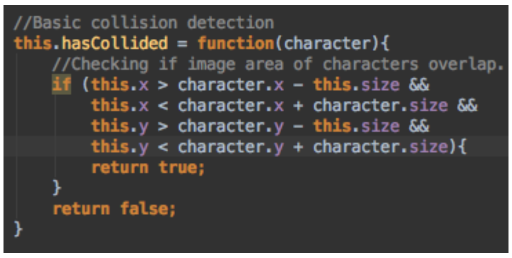
```
$(document).keypress(function(e) {
    switch(e.which){
        case 119: // key: w
            player.z = 0; // up
            break;
        case 115: // key: s
            player.z = 2; // down
            break;
    }
});
```

Figure 18: *Key Listener.*



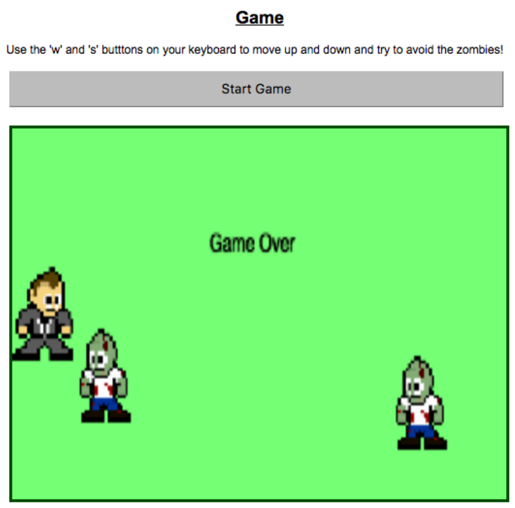
```
//Defining a character. The player
function character(src, x, y, z){
    this.size = 150 / 4;// relativ
    this.x = x;// characters x loc
    this.y = y;// characters y loc
    this.z = z;// characters direc

    //Creating an image from the p
    this.image = new Image();
    this.image.src = src;
```

Figure 19: *Character Object.*



```
//Basic collision detection
this.hasCollided = function(character){
    //Checking if image area of characters overlap.
    if (this.x > character.x - this.size &&
        this.x < character.x + character.size &&
        this.y > character.y - this.size &&
        this.y < character.y + character.size){
        return true;
    }
    return false;
}
```

Figure 20: *Collision Detection*



Game

Use the 'w' and 's' butttons on your keyboard to move up and down and try to avoid the zombies!

Figure 21: *Game Over Screen.*

References Tabs

Page Source Code, viewed 31 October 2016,
< http://www.tv.com/shows/the-walking-dead/episodes/ >.

HTML Game Example, viewed 1 November 2016,
< http://www.w3schools.com/graphics/game_intro.asp >.

Dave Character 8bit, viewed 1 November 2016,
< http://media.indiedb.com/images/games/1/45/44413/mega-man-sprite_c.1.png>.

8 Bit Zombie, Melolzugameing, 1 November 2016,
<http://orig07.deviantart.net/4f30/f/2012/141/3/6/8_bit_zombie_by_melolzugaming-d50kbqk.png>.

The walking dead poster 51, viewed 27 October 2016,
<http://1.bp.blogspot.com/-E6T6LmRhFzI/UM5Dt7Dd1wI/AAAAAAACJlU/-vTS0wIJlnI/s1600/the_walking_dead_poster51.jpg>.