Noroff
School of technology and digital media

Technical Report

Semester Project 2: Board Game

Jørgen Årnes

Word count

Summary: 250 | Main text:500

Table of Contents

1. Summary

General ideas,

Collecting information of the topic I was designing helped me a lot while figuring out which elements to include. By recreating graphic works from others you can save a lot of time.

The Game of Thrones topography is a Gothic one and I used it in the logo and on some lower level headings. Other fonts used in this project are "Nanum Gothic" and "Roboto".

Letter spacing and space between sections is important for good legibility, visual consistency, hierarchy and have therefore also been used wisely.

The logo I created is a recreation taken from an image. It consists of the Game of Thrones fonts combined a sword shaped as a T letter.

A graphic image of the Iron Throne chair was created for the last page with the purpose of letting the winner token be placed on it.

A dark blue colour is used as the dominant colour to create an association of honour and power like The Game of Thrones environment. The colours are also used to either highlight elements for attention or just create good visual contrast for legibility on text.

The design sketches are almost identical to the coded ones but the coded ones are improved a lot when it comes to visual hierarchy in terms of layouts and consistency.

In order to create the cards and the dynamical functionality for the character select page to work as required in this assignment, I benefited the use of array, variables, for loop, forEach method, literal templates, addeventListener, conditional statement, functions and local storage. The array is used to provide content to the cards and the active ID's and classes. The for loop is used to create the cards with literal templates and the forEach method is used to attach the click event listener to each card. Boolean values of null on the variables and if else statements checks to see if the variable is true or false. All of the DOM manipulations and local storage placements occurs inside each if else block. The this keyword is frequently used to get the instance click event.

JS file used for the game page has almost identical JS functionality except that instead of array I used object literals with properties and class methods. The properties in the object literal works like variables and fetches the local storage content that are used for the tokens in the board game. The method is used to manipulate the DOM. Like the first JS file, the for loop is used to create HTML content

2. Body

2.1. Introduction

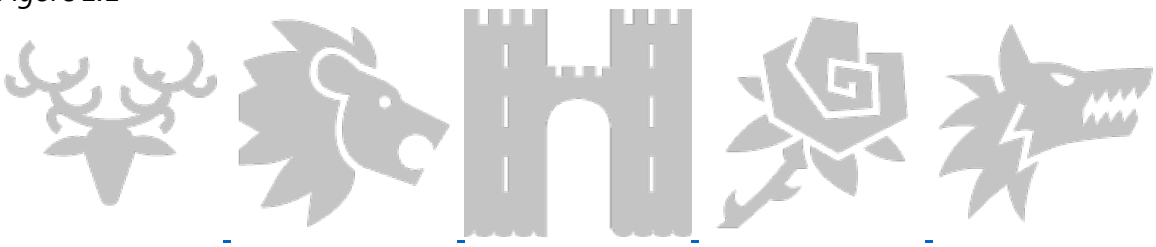
2.2. Main section of report

GitHub repository:
https://github.com/jorgenaa/Semester_project_2

Icons

Since the task is to make a board game, it was natural for me to begin looking at Game of Thrones board games to quickly get some ideas on what kind of content I should include in the design for the website. After doing some research I came up with the idea of making 10 icons for each of the cards that I'm supposed to create for the index page. I found valuable information about each of the characters on an entertainment media brand site called Fandom. They have a encyclopedia called "Wiki", that provides a lot specific information about the different characters personality, skils, culture etc. At first, my plan was to create icons with same ratio and style from scratch, but it would probably have taken me longer time to achieve icons with similarity if I had to do the whole proccess in designing the graphics. Fortunately I found 10 free icons for the Game of Thrones on a online repository called Game-icon.net. All of the icons I found have the CC BY attribution and since I'm supposed to create at least five by my own I hence, redraw four of the icons created by author "George R. R. Martin", and used five of he's remaining icons on the list for the rest of my character cards selection. The last Icon I made that's not included on the list was an icon of a tower representing the House Frey symbol. Even though I recreated four of the icons it may be some difference on the curves and shapes compared to the original. Below on figure 1.1, you can see which Icons I redraw for the game character page with Adobe Illustrator including the tower icon.

Figure 1.1



Typography

After a quick search on the web I found out that the typography that's been used as headings in many occasions in Game of Thrones is a Gothic one with serif.
I found a Game of Thrones font designed by "Charlie Samways" on this web page "https://www.fonts4free.net/game-of-thrones-font.html", and used it in the logo and lower level headings, "*see figure 1.1*". I tried to find other gothic fonts that I could pair with one another and found a font called "Nanum Gothic" that is designed by "Sandoll Communication", and paired it with Roboto Regular font. Although Nanum Gothic was a design for the purpose of writing Korean script, I think it works well for this theme as well. I

used the Nanum Gothic bold weight for the main heading and some sub headings. For the body text and buttons, I used Roboto Regular.

Figure **1.2**

GAME OF THRONES

To achieve better legibility, I adjusted the letter-spacing to 1 px. According to the recommendation given on Material Design "Understanding typography"
it is advised to apply tracking to the body text but maintain the headlines tight together, I therefor only applied spacing to the body-text. Apart from that I distinguish content and create hierarchy and kept the distance between headings and body text with 24 px spacing. For better visual consistency I applied 16px padding around each module.

Logo

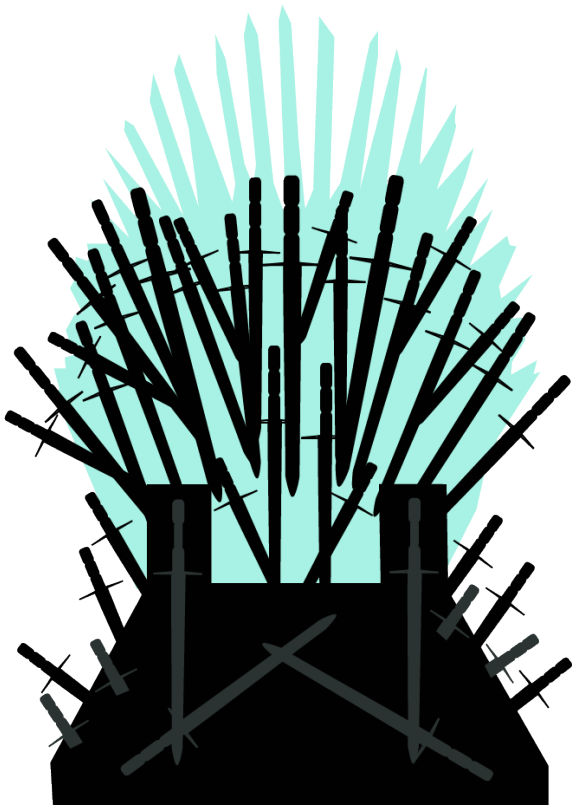
I was inspired by an image I found on pinterest.com that lead me to create something similar in Illustrator. The image is uploaded by someone with a nick name "krunal". I draw a similar sword and added some linear gradient to it. Then I combined the Game of Thrones fonts to the sword that is shaped like a T letter, "*see figure 1.3*".

Figure 1.3

Graphichs

I got the idea to create a graphic image of the Iron Throne chair for the last page. The image consists of many layers of swords with different sizes placed one the foreground and background that illustrates an ice shape of sharp edges like frozen swords. The winner will have their token placed on the throne while it rotating horizontal 360 degrees. On *Figure 1.4*, you can see the Iron Throne graphic I created.

Figure 1.4



Colours

Since it's a lot of pride and honour in the Game of Thrones I thought it would be a good idea to let blue be among the dominant colours in my colour palette. According to the article, "Color Psychology in Web Design – Big Websites Case Studies ", MAY 25, 2011, by blogger, design enthusiast and author Adriana Marinica, it is also mentioned that the blue colour is also is associated with honour and power. The colour scheme I put together is meant to express a mood of coolness like winter but also includes some warm colours for buttons, content with active states and animation.

Figure 1.5

#074a59	#1D7373	#57face	#b30000	#ff0000

#010101	#ffffff	#d35400	#ffa502

To improve the visual hierarchy, I used a dark blue teal colour with 70% opacity on the background of the main content and set the background image to fixed. In this way only the body text gets scrolled and you are still able to see the background image since the rest is transparent. The cards on the index page have a blur filter to make it stand out from the rest of the content and enhance the legibility of the text nested inside of the cards.

I used a black colour on the header, footer, the tiles with traps and some boxes that provides status information. The black background enables the warm and cool colours on elements such as the logo, the active navigation buttons and tokens to pop up and attract the user's attention. I applied two different colours with high vibrancy (#57face, #ff0000) for the active text to distinguish player 1 from player 2 and vice versa. For the rest of the body text and headlines I used white colour. Unlike the active text elements, I used two colours with darker value and more chroma as active background colours on the selected cards on the character select page "*see figure 1.4 (#238c8c, #b30000)*". The reason why I did so is not only to differentiate which cards belongs to which players after being selected, but to create more contrast for the text content. I used a tool to ensure that the contrast was optimal according WCAG standards. The green elm kind of colour type combined with white scored 5,6 in contrast ratio while the red one combined with white scored 7,20. For consistency reasons, I also used the same background colour that I used on the selected cards on the tokens. I did this with the intention of creating a relationship between the selected card and the tokens that were given to the player.

Design Sketches

https://xd.adobe.com/view/bed34cb9-b448-4000-7f62-e72406e0e5c53-c8e2/

I don't think I ever have made a design sketch that is identical to the coded version, however the sketches indeed helps speeding up the efficiency while coding.
The design sketches are almost identical to the coded version except that the coded version has better hierarchy in terms of layout. The coded version is also more consistent. For instance, In the main section, I created a distinction between elements that belongs together and not by nesting headings and updated status content in a separate header container while the content in the main body beneath the header section. The other two pages have the same pattern except that the interface game page also has a sidebar container that holds the dice for the game.

Javascript

The order I chose during the process of this project may not be exactly what I've learned so far, but since I don't have that much experience in making games yet, I still chose to code a bit on the first page while I was sketching. One of the challenges of creating the first page was to know how to create a function that only allows you to select a maximum of 2 cards and that differentiates between the first and the second player. The first thing I did was creating an array of objects with properties containing information about the characters I already had collected. I created a section tag with a class of row as a container for the cards. Then I created 10 cards with template literals in a for loop that loops through the character array. In each card I used tables, image tags and injected the content from the array with the dot notation. I assigned a variable to a "querySelectorAll()" method and attached it to a "forEach()" function and added a parameter to an event listener. I named the parameter "card" since I'm selecting each "card" in the cards container and then I referred to an external named function called "activateCard" to the event listener.

For the function I created to variable "player1" and "player2" and assigned both of them to the value of null. With some support from my tutor "Connor O Brien", I learned how to create an if statement inside of the function that checks if both player 1 and player 2 are selected and if neither one of them are selected then the "return" statement will exit the function. If both players are selected then one of them must be unselected. In the same statement I also checked to see if it not contained two CSS class of background colours. In order to check that the click event worked I console logged out the this "this" keyword since it refers to the object function and the function name is as mentioned referred to in the event listener that reads all of the cards from its parent that is a "forEach" method. In order to make the background colour classes work active when clicked, the second if statement checks if first player is equal to null and that it's not contains the background colour class to the second player. If the statement is true then it will only add the background colour class to the first player. Else if the first player is already selected then the first player is set to be equal to null and its class is removed. To make this behaviour also apply for the second player, I copied the if else statements for player 1 and renamed it to player 2. Although the active background colours are a good signifier to tell the user that she or he has selected a card, I also wanted to explicit express which of the cards that represented player one or player two with text. I therefore assigned a variable named "selectPlayer" and appointed it to a class name ".game__card--active-text". To ensure this class respond to the click event I also used the "this" keyword and set the "innerHTML" to be equal to a string literal. I changed the content of the string based on the instance, for instance if player 1 is selected "selectedPlayer = 'This is player 1' " and if the first player is selected the string is equal to an empty string.

In addition to the active text on the cards, I used two data attributes to retrieve the name and icon value from the objects. The two data attributes are then assigned to both players including the "this" keyword and in relation to the order of the conditional statement in the "activateCard" function. In this way whenever a card is selected the players name will appear in the inner HTML element that holds the ID attached to the variable. To make the page more dynamic I added CT button that only displays when both of the players are selected and when clicked direct to the next page. To create the button, I simply wrapped an "a" tag to the button tag and an ID that is assigned to variable and attached to an event listener. I then created a function with a reference to the event listener like I did with the "activateCard" function. Inside the function I used an "if else" statement that checks if both players are not equal to null. If the statement is true then button is disabled by setting it to equal to the Boolean value of false else it's equal to true. To make the button toggle I used the "classList" property and set the class in CSS to display none;

The next challenge was to move both of the selected players to the next html page.
I used local storage for this task and I also applied the same data attributes to the local storage value. Since the tokens representing both of the players were frequently used on the game page, I found it useful to use an object literal rather than a function. Inside the object literal, I literally created 4 properties and set the values of each property to retrieve the local storage items. Inside the same object literal, I created a method that appends the characters names and icons to their HTML containers. With the object literal I could easily access the icons and names and apply them inside of other function by just referring to the object name and dot notation.

I created tiles for the board game with a for loop and appended 30 div tags with the template literals method. I found a simple method on how to create a dice on a blog postet on medium.com and customized the code with 6 graphic dice icons I created in Illustrator and an even listener. I attached the event listener to an image tag with an ID and used a class to get the image sources. Each dice icon has a unique number that matches the "Math.random()" function, so every time a click event occurs on the dice image, it will change according to the random dice total. "*See figure 1.5*".

Figure 1.6

```
document.querySelector("#dice").addEventListener("click", rollDice);

function rollDice() {
            event.preventDefault();
            var diceResult = Math.floor(Math.random() * 6) + 1;
            var dice = document.querySelector('.game__dice');
            dice.style.display = 'block';
            dice.src = 'graphics/icons/dice-' + diceResult + '.png';
            dice.innerHTML = diceResult;
}
```

To get both players moving on the board in tur, I created a variable called playerTurn and set it to be equal to 1. Then I created two variables and set both of their values to be equal to zero. The two last variables will be the running total that get added to the dice score each time the dice is rolled. In the rollDice function I used an if else statement that checks to see if the variable playerTurn is equal to 1. If this is true then playerTurn is equal to 2. In the same block, player 1's total gets added with the dice result. Since it's only 30 tiles, I needed to ensure that the maximum total score of the player not exceeded higher than 30. I hence, used a second if statement inside the first block statement that checks to see the player1Total is greater than 30, if so then player1Total is equal to 30. To move a token to a tile, I created an undefined variable and named it tile. Inside the first playerTurn condition block, I assigned it to be equal to an ID that matches the total score. In this case the id is the player1Total variable. Then I appended the token as a child element to the tile. The token is retrieved from the object literal localStorageItems that collect the local storage content from the index page. Since the first condition only work for the the first player, I copied the first if statement and pasted it in the else if statement and changed the playerTurn variable to the opposite value of player 1. Then I changed the following variables (player1Total, currentScorePlayer1, token1Wrapper) to (player2Total, currentScorePlayer2, token2Wrapper).

The next step was to create 5 traps for the board game. The first conditional statement I created for the traps looked too complex and was too deep. So, when I reached a total of 54 lines of code, I understood I was doing something wrong. The first mistake was that I repeated the same if statement for every trap on the board and when you multiply the code with 2 it suddenly becomes overwhelming. So instead of writing like shown on *figure 1.6.*

Figure 1.7

```
if (player1Total === totalTiles[4]) {
    if (player1Total === 5) {
        status.innerHTML = traps[0].enemyTrap1;
        player1Total = player1Total - 1;
}
```

I manage to shorten the code to only 12 lines like shown below on *figure 1.7.*

figure 1.8

```
      if (trap) {
                        player1Status = traps[player1Total];
                        modalPenalty.style.display = 'block';
                        player1Total = player1Total + trap.penalty;
                        modalStatus.innerHTML = playerStatus.description;
                    }
```

The trap condition is nested inside of the if statement "if (playerTurn === 1)", and with the [] notation, both of the variables "player1Totalt and player2Total" is defined as keys in the object "traps". Since the variable named trap work as a property accessors that provides access to the traps object's properties, every time "player1Total" and "player2Total" match the same number as the property number, I don't need to write a condition for each of the properties in "traps" object literal that check every time a player land on a tile. The traps object consists of 5 properties with objects as value and each of the 5 properties has a specific number that is equal to where the traps are positioned on the board. Inside of each object block there is 2 properties "desription" and "penalty". The description is a string that is used to describe the trap while the "penalty" propertie has a value of a negative integer that is used to add to player 1 or player 2 when they land on the trap. I used a modal to warn the player whenever their token land on a trap. The modal is placed in both trap conditions and has an ID that access the traps description value. I also used a class that is set to "display: none" as default and displayed as block inside the trap condition.

The last condition statement I created for the game page was an if else if statement that checks to see if one of the players has landed on the last tile or passed it. If player1Total or player1Total is greater than or equal to 30 then a modal will display with the player characters name shown on the modal. Both of the if blocks also get the winner token and winner name from the object "localStorageItems" and set them as items in local storage to be reused on the last page.

On the last page I only used a function for all of the variables that are used to manipulate the retrieved data from the local storage. The functionality is similar to the class I created inside the object literal "localStorageItems" except that the variables are only accessible inside the functions scope. I also cleared all of the local storage items after retrieving them by placing the clear function underneath the variables.

For the animation task, I created an animation of fireflies with CSS and JS. I only used JS to create the elements and place them randomly in the HTML container. I started with returning the HTML container with an ID to a variable named "animationContainer" and applied a for loop with an innerHTML and an addition assignment operator that adds 19 template literals with backticks. Each element is like the other previous one's I created styled from an external CSS stylesheet. To position all of the elements, I literally just created to variables, "width" and "height", and assigned them to the animationContainer's offsetWidth and offsethHeight. Then I used forEach method and styled them with top and left position and the Math.round(Math.random()) function to place each item randomly inside the container. *See figure 1.8.*

Figure 1.9

```
fireflies.forEach(function(firefly){
    firefly.style.top = Math.round(Math.random(300) * height) +  'px';
    firefly.style.left = Math.round(Math.random(500) * width) + 'px';
});
```

As a junior developer it is not unlikely that my code can end up with too many statements, look too complex and too deep. To eliminate some of the code smells, I therefor tried refactored and streamlined the code.

HTML/CSS

While working with the HTML and CSS structure, I benefited what I've acquired about CSS frameworks and CSS pre-processors. I tried to combine both Bootstrap and SASS along with the BEM naming conventions.

Although the BEM methodology can cut down on the number of lines of CSS codes and offers better overview of classes and is easier to scale, I found it difficult to name all of the top-level blocks. The reason for this is that I not only want to create disposable block elements, but also reusable block elements that work like templates. If the block names don't provide any logical meaning when some of their elements are reused for a different purpose another place, then perhaps I need to rewrite an existing element with a different name and suddenly BEM becomes a disadvantage on the long term. Another issue I run into was how to avoid nesting too many elements in one block since HTML tags like main as top level block component can exceptionally grow into long chains of children elements. The first BEM structure I created looked just like that but even worse since I used the same template on every HTML page and it was also necessary to add additional block elements. This resulted in a very long block with elements. My solution to the problems was to create smaller blocks and tried to name them so that they mattered to the content. *Figure 1.9* shows an example of my reusable blocks. Every page gets the same background style but each page has their own modifier that provides different background images.

Figure 2.1

```
.background-img {
    background-position: inherit;
    background-repeat: no-repeat;
    background-size: cover;
    background-attachment: fixed;

    &--characterPage {
        background-image: url('../img/castle.jpg');
    }

    &--gamePage {
        background-image: url('../img/map.jpg');
    }

    &--finalePage {
        background-image: url('../img/trees.jpg');
    }
}
```

I used Bootstrap mainly with the purpose to quickly set up responsive layouts without having to write too much CSS. I also applied some of the Bootstrap classes for adjusting the paddings and margins to some of the boxes. The only place I used CSS grids was to create tiles for the board game. I simply used the "grid-template-columns: repeat(5, 170px);" rule on the parent container that creates 5 equal columns for the tiles. Then I used "grid-auto-flow: row;" to make the tiles place in order from left to right.

Keyframes

- A discussion of your work process including the user testing
- Your reflections on the project
- A list of references used on the project

https://www.youtube.com/watch?v=JVlfj7mQZPo

https://developers.google.com/web/fundamentals/design-and-ux/animations/css-vs-javascript

2.3. Conclusion

3. References

Website: *https://gameofthrones.fandom.com/wiki/House_Tully*
Website: *https://game-icons.net/tags/game-of-thrones.html*
Website: *https://www.fonts4free.net/game-of-thrones-font.html*
Website: *https://material.io/design/typography/understanding-typography.html#readability*
Website: *https://no.pinterest.com/pin/539798705333445408/*
Website: *https://pixel77.com/color-psychology-web-design-color-schemes-big-websites/*

4. Acknowledgements

Start writing here

5. Appendices

Start writing here