# Introduction

This section of the document is the introduction. We all have been in a website which refreshes each time we search up a something or click on an element. And depending on your internet speed this will either be fast or slow. And this is where React JS comes in play because a lot of people doesn't have fast internet and updating the whole website will take time for them. React JS allows us to update components that needs to be updated instead of completely updating the whole DOM *"The DOM represents the document as nodes and objects; that way, programming languages can interact with the page."* (MDN, 2022). The reason this is possible is because React JS uses Virtual DOM *"The virtual DOM (VDOM) is a programming concept where an ideal, or "virtual", representation of a UI is kept in memory and synced with the "real" DOM by a library such as ReactDOM."* (ReactJS, n.d).

In this document I'll go through creating an movies website which will show the movies image, name and release date to show off the efficiency of React JS. It's important to note that I'll only be including important information.

# Assumptions

This section of the document is the assumptions. This document assumes that the reader has knowledge, understanding or experience of the following topics. The reason this document has assumptions is so the reader can have the best experience as possible.

- The document assumes that the reader has knowledge, understanding or experience with React JS.
- The document assumes that the reader has knowledge about API's.
- The document assumes the reader has experience with bootstrap.

These are the assumptions.

# Planning

This section of the document is the planning. Planning is one of the most important parts of a project *"Planning keeps us focused and goal-centered. Keeping our sight on our goals motivates us."* (Business, 2022). There are a lot more benefits with planning and going through them all would be unnecessary. The planning for the movie website is to implement the following features.

- Create a search bar
- Use a movie website API to obtain all the movies with that name
- Display all the movies to the end-user

- Improve the web design by using bootstrap

These are the features that I'll be implementing in our React JS website.

## Equipment's

This section of the document is the equipment's. In this document we will be using the following tools to accomplish the planning goal. *All these tools can be installed from npm.*

- Create-React-App
- Bootstrap
- Nodemon

These are the tools we will be using in this document. It's important to note that in this document I'll not go through bootstrap because this document is mainly focused on React JS.

## Programming

This section of the document is the programming. Now we have a clear overview of the planning and the equipment's that we will be using throughout the document. It's one more thing that needs to be done and that is designing our component setup because this is what makes React JS unique. The idea for the components is to split them in two.

- Movies
- Movie Box

The movie box will be the card that will be used to display the image, name and the release date of the movie. The movies will be used as our main component. Now let's start programming our application… Let's start with our movies component. The first thing we need to do is to create our useStates.

```
1.  // search will contains the data that we search.
2.  const [search, setSearch] = useState();
3.
4.  // movies contains all the data that will be returned by API.
5.  const [movies, setMovies] = useState([]);
```

The search variable will be used to store the value the end-user searched for, and movies will be used to store all the movies. Here is the function that will be used to obtain all the movies.

```
1.  async function searchMovies(title)
2.  {
3.      const response = await
    fetch(`${'http://www.omdbapi.com/?apikey=XXXXX'}&s=${title}`);
4.      const data = await response.json();
5.      setMovies(data.Search);
6.  }
```

What this function does is it obtains all the movies that the end-user searched for from omdbapi and then assigns it into the movie's variable.

```
1.  return (
2.  <div className="bg-dark mb-5">
3.      <div className="container">
4.          <div className="row">
5.              <div className="col-12 mt-5">
6.                  <h1 className="text-center text-white">Movies</h1>
7.              </div>
8.              <div className="col-8 col-lg-4 col-md-6 col-sm-8 m-auto mb-3 mt-2">
9.                  <div class="input-group mb-3">
10.                     <input type="text" className="form-control" placeholder="Search for
    a movie" value={search} onChange={ (e) => setSearch(e.target.value) }  />
11.                     <div class="input-group-append">
12.                         <button class="btn btn-primary" type="button" onClick={ () =>
    searchMovies(search) }>Search</button>
13.                     </div>
14.                 </div>
15.             </div>
16.         </div>
17.         <div className="row">
18.             { movies?.length > 0 && movies.map( (movie) => (
19.                 <MovieBox movie={ movie }/>
20.             ))}
21.         </div>
22.     </div>
23. </div>
24. );
25.
```

Pay attention to the highlighted elements. The **value={search}** holds the search variable which was declared with useState function and **onChange={ (e) => setSearch(e.target.value) }** updates our search variable if we type anything inside the input box. What does the last highlighted element do? The last highlighted element gets MovieBox the movie image, name and image. Now let's code our movie box component.

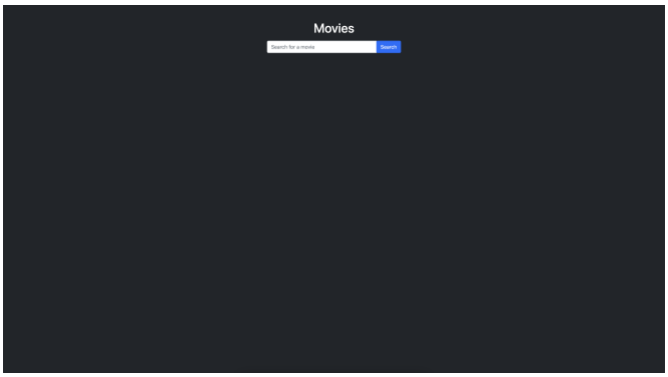
```
1.  function MovieBox({ movie })
2.  {
3.      return (
4.          <div className="col-12 col-sm-12 col-md-6 col-lg-2 m-lg-3 mb-4">
5.              <div className="card m-3 p-0 movie-card-bg m-sm-auto m-auto" style={{width:
    '13rem' }}>
6.                  <img className="card-img-top" src={movie.Poster} alt="Card image cap"/>
7.                  <div className="card-body">
8.                      <p className="card-text text-white custom-font-
    size">{movie.Year}</p>
9.                      <p className="card-text text-white">{movie.Title}</p>
10.                 </div>
11.             </div>
12.         </div>
13.     )
14. }
```

What this function does is it displays the movie image, name and release date. The code above seems complicated and it's not. The reason it looks complicated is because of Bootstrap and if I did

not include Bootstrap the code above would look much cleaner but then the website wouldn't look great.

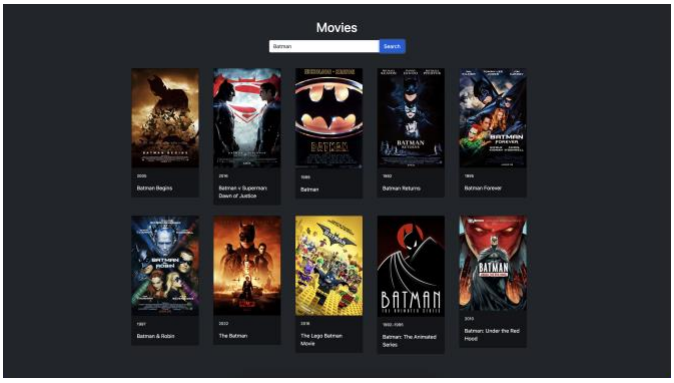
## Showcase

This section of the document is the showcase. In this section of the document, I'll showcase the movie website that was made through out this document. Here is an overview of the index page.




0.1 Website Index

Now let's search up "Batman" and see what comes up!




0.1 Website searched

The website displayed all the components without any issues and instead of updating all elements it only updates the components that needs to be updated and this is the magic of React JS. The design of the website isn't the best, but this is an small project that I choose to make.

## Features

This section of the document is the features. There are more features that could be implemented such as…

- Displaying certain genre
- Date selection to and from
- Display rating

There are some of the few elements that could be implemented…

# Conclusion

This section of the document is the conclusion. In this document we went through the introduction, planning and programming a movie website application. In introduction we went through the difference between the DOM and Virtual DOM. And in planning the document went through planning the application structure and what features would be designed. And in programming the document went through programming the whole website. At the very end of the document I you learned something new.