**React & React Native Course Spring 2019**

**Завданням було:**

Реалізувати **SPA** на **чистому JavaScript** (без використання будь-яких сторонніх бібліотек), використовуючи The Movie Database API -<https://www.themoviedb.org/documentation/api>

Для отримання access token потрібно буде зареєструватись.

Головні вимоги:

1. При заході на головну сторінку потрібно вивести tranding фільми/серіали.

2. По кнопці пошуку потрібно виводити список фільмів, які задовільняють дані, які ввели в інпуті

пошуку.

3. По кліку на конкретний елемент списку, потрібно показати заголовок, про фільм, постер.

4. Для кожного фільму потрібно показати його рекомендації."

**Рекомендації:**

1. Завжди потрібно перевіряти чи 100% код працює (були випадки, коли це було не так)
2. Стиль написання коду — не менш важливий ніж самий код. Можливість швидко розібратись в коді дозволяє вашим колегам швидше його перевірити та допомогти, якщо є якісь проблеми.  
   Дуже сильно раджу звернути увагу на [airbnb code style](https://github.com/airbnb/javascript).
3. Часто люди не знають про нові можливості JavaScript, такі як template string literal, які сильно спростили б генерування HTML. Раджу глянути пояснення за [цим посиланням](https://understandinges6.denysdovhan.com/manuscript/02-Strings-and-Regular-Expressions.html), розділ “Шаблонні літерали”
4. В переважній більшості використовується var, а не let/const. Детальніше можна почитати за [цим посиланням](https://understandinges6.denysdovhan.com/manuscript/01-Block-Bindings.html).
5. Змінні (функції) оголошуються в глобальній області видимості та використовуються напряму в html, наприклад *onlick=”search()*”
6. Практично ніхто не використовує *fetch*, який прийшов на зміну *XMLHttpRequest*. Детальніше можна почитати [за цим посиланням](https://codeguida.com/post/756).
7. Використання for для ітерації масивів, замість forEach/map методів масивів. Детальніше про [Array#map](https://developer.mozilla.org/uk/docs/Web/JavaScript/Reference/Global_Objects/Array/map) та [Array#forEach](https://developer.mozilla.org/uk/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach).
8. Для того щоб дізнатись чи відбувся клік на якусь з нод (наприклад, елемент LI списку UL, де відображаються результати пошуку та рекомендації), більшість просто вішають на кожну ноду event listener, або взагалі прокидають *onclick=”loadMovie(id)”*, проте найбільш оптимальним варіантом буде використання *data-id=”movieId”* на пару із [closest](https://developer.mozilla.org/en-US/docs/Web/API/Element/closest), приклад можна глянути нижче.
9. Деякі люди вирішили використати замокані (скопійовані з API) дані, таким чином зробивши вигляд, що вони працювали з АРІ.
10. Багато робіт використовували HTML темплейт (готову верстку), яка оновлялась за допомогою JS, а не генерувалась нова, відповідно їм доводилось вписувати якісь дані для того, щоб показати їх до того, як щось прийшло з сервера, або ж відображали фіксованової кількості список рекомендацій чи результатів пошуку.
11. Практично ніхто не використав ‘use strict’. Тому якщо його використати, зразу вилазять помилки *Uncaught ReferenceError: variable is not defined*.

**Прикад використання template string literals.**

|  |
| --- |
| function renderMovie(element, movie) {  element.innerHTML = `  <img src="http://image.tmdb.org/t/p/w300${movie.poster\_path}" /></br>  <h1>${movie.title}</h1><br/>  <p>${movie.overview}</p><br/>  <h3>Recommendations:</h3><br/>  <ul id="recommendations"></ul>  `; }  // чи так краще? function renderMovie(movie) {  var html = '';   html += '<img src="http://image.tmdb.org/t/p/w300' + movie.poster\_path + ' /></br>';  html += '<h1>' + movie.title + '</h1><br/>';  html += '<p>' + movie.overview + '</p><br/>';  html += '<h3>Recommendations:</h3><br/>';  html += '<ul id="recommendations"></ul>';   return html; } |

**Приклад використання** [**dataset**](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/dataset) **та** [**closest**](https://developer.mozilla.org/en-US/docs/Web/API/Element/closest)

|  |
| --- |
| function renderList(list, container) {  const listHtml = list.map(item => {  *// each item is a movie object*  return `<li data-id="${item.id}>${item.title}</li>`;  }).join('');   *// inster generated html into container*  container.innerHTML = listHtml;  container.onclick = (evt) => {  *// check if the target element is a `li` or its children*  *// and return li element if it's true*  const node = evt.target.closest('li');   if (!node) {  return;  }   *// handle single element click*  *// for example render movie details page*  }; } |