Modul Pertemuan 6 – jQuery

Pada pertemuan ini, kita akan mempelajari penggunaan jQuery dalam pengembangan web. Dokumentasi lengkap mengenai penggunaan jQuery dapat dilihat pada https://api.jquery.com. Penggunaan jQuery bertujuan untuk mempermudah pembuatan kode Javascript, misalnya untuk manipulasi DOM, *event handling*, dan pemanggilan AJAX.

Sebagai contoh, jika pada kode HTML terdapat sebuah element paragraph seperti pada contoh kode di bawah ini:

```
<p id=”title”>The Lord of the Rings</p>
```

maka dengan menggunakan Javascript, kita dapat mengubah isi dari paragraf tersebut dengan perintah seperti berikut ini:

```
document.getElementById(‘title’).innerHTML = ‘The Hobbits’;
```

Dengan menggunakan jQuery, maka perintah di atas dapat dipersingkat menjadi:

```
$(‘#title’).text(‘The Hobbits’);
```

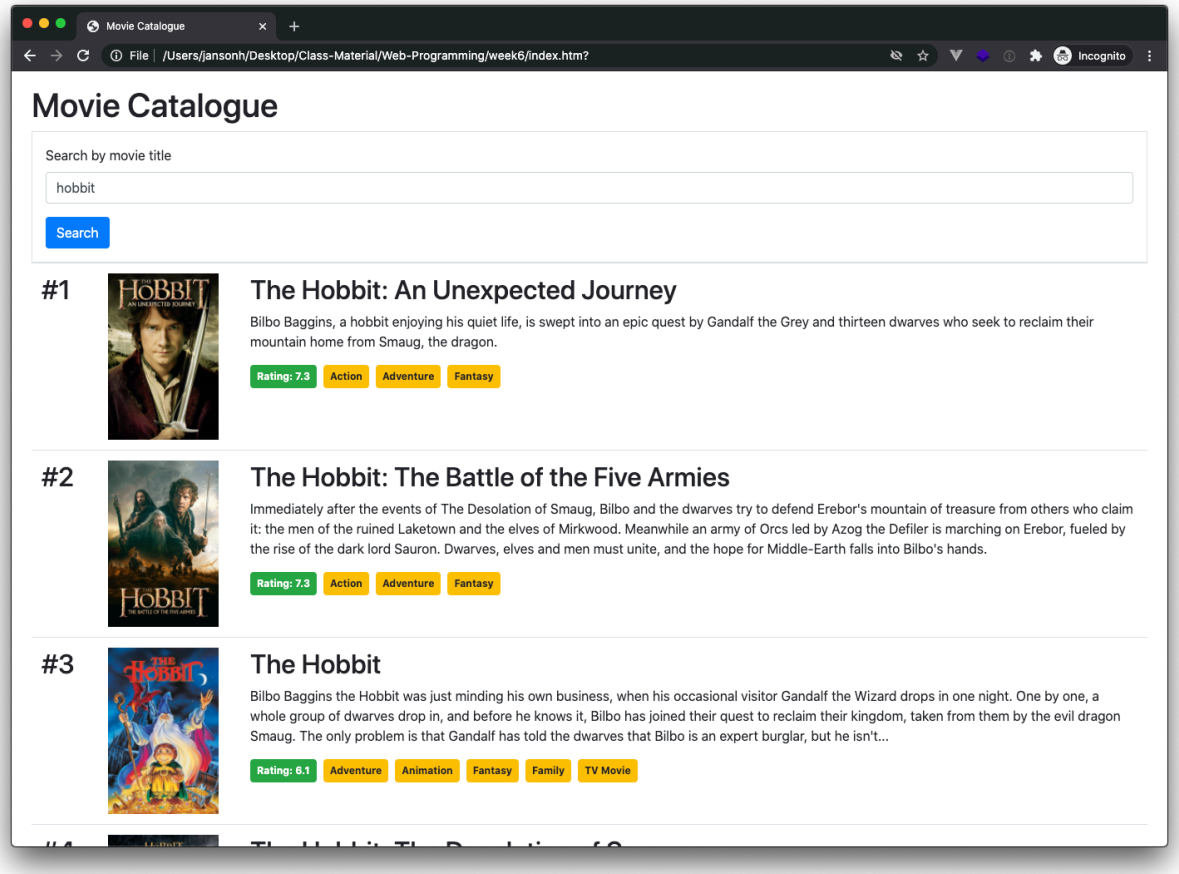
Dapat dilihat pada contoh di atas, bahwa pemanggilan `document.getElementById(id)` dipersingkat menjadi `$(‘#id’)` dengan tanda `#` menandakan bahwa element dipilih berdasarkan `id`.

Pemanggilan AJAX dengan menggunakan jQuery juga lebih sederhana dengan menggunakan fungsi `jQuery.ajax()`. Bahkan jQuery juga mempermudah lagi untuk pemanggilan AJAX dengan menggunakan fungsi `jQuery.post()`, `jQuery.get()`, dan seterusnya.

Pada contoh di modul ini akan dibuat sebuah website yang menggunakan AJAX untuk memanggil API dari TheMovieDB dan menampilkan film yang dapat dicari oleh pengguna. Kode Javascript yang dibuat pada contoh ini akan menggunakan jQuery dan juga Bootstrap. Gambar 1 menunjukkan contoh tampilan dari website contoh yang akan dibuat.

TheMovieDB.org

Mula-mula, kita perlu membuat akun pada themoviedb.org (TMDb) untuk mendapatkan API key yang akan digunakan pada proses pencarian film. Bukalah website tersebut dan buatlah akun baru dengan memilih menu “Join TMDb”. Setelah proses register dan verifikasi email berhasil, maka lakukan login dan buka “Settings” dengan memilih foto profil di sebelah kanan atas layar.



Gambar 1. Tampilan website yang akan dibuat.

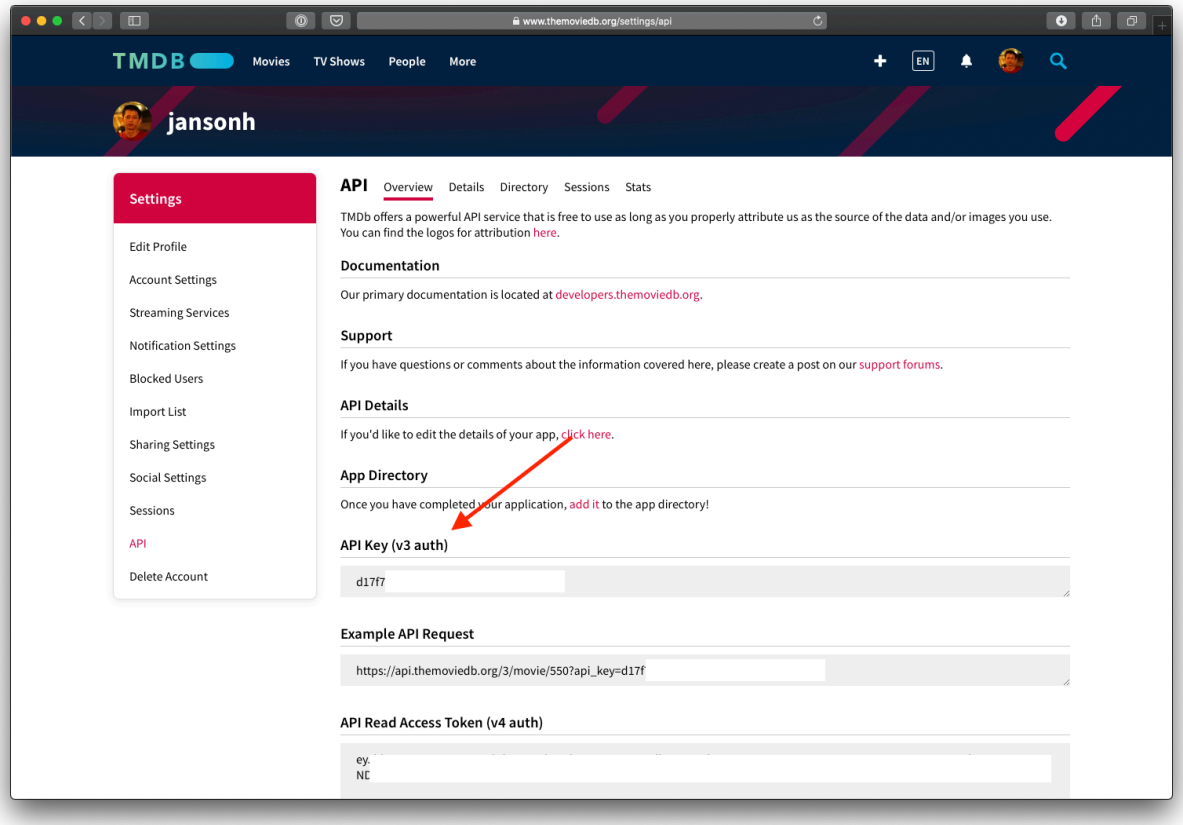
Pada “Settings”, pilihlah menu “API” pada pilihan menu di sebelah kiri layar. Jika anda belum pernah mendaftarkan API key sebelumnya, maka pilihlah menu untuk membuat API key baru. Informasi mengenai website yang mau dibuat dapat anda isi.

Setelah proses pendaftaran API key selesai, maka anda akan mendapatkan API key tersebut pada menu “Settings” -> “API”. Gunakan API v3 untuk contoh website yang akan dibuat seperti pada gambar 2.

Dokumentasi API TMDb selengkapnya dapat dipelajari di https://developers.themoviedb.org/3/. Pada contoh website yang akan dibuat pada modul ini, terdapat 2 API endpoints yang akan digunakan, yaitu:

1. Search movies: `GET /search/movie`
2. Get genres: `GET /genre/movie/list`

Dokumentasi kedua API tersebut dapat dilihat pada link berikut ini:
https://developers.themoviedb.org/3/search/search-movies
https://developers.themoviedb.org/3/genres/get-movie-list



Gambar 2. API key v3 yang digunakan pada contoh website ini.

Membuat Tampilan Website

Selanjutnya, kita akan membuat tampilan dari website ini dengan menggunakan bantuan framework Bootstrap.

Buatlah kode HTML dasar seperti di bawah ini:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Movie Catalogue</title>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css
">
</head>
<body>

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

```
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></s
cript>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"><
/script>
</body>
</html>
```

Pada kode di atas, kita menambahkan framework Bootstrap dan juga jQuery. Kemudian kita akan mulai dengan menambahkan kode berikut ini di dalam `<body>` dan sebelum `<script>`.

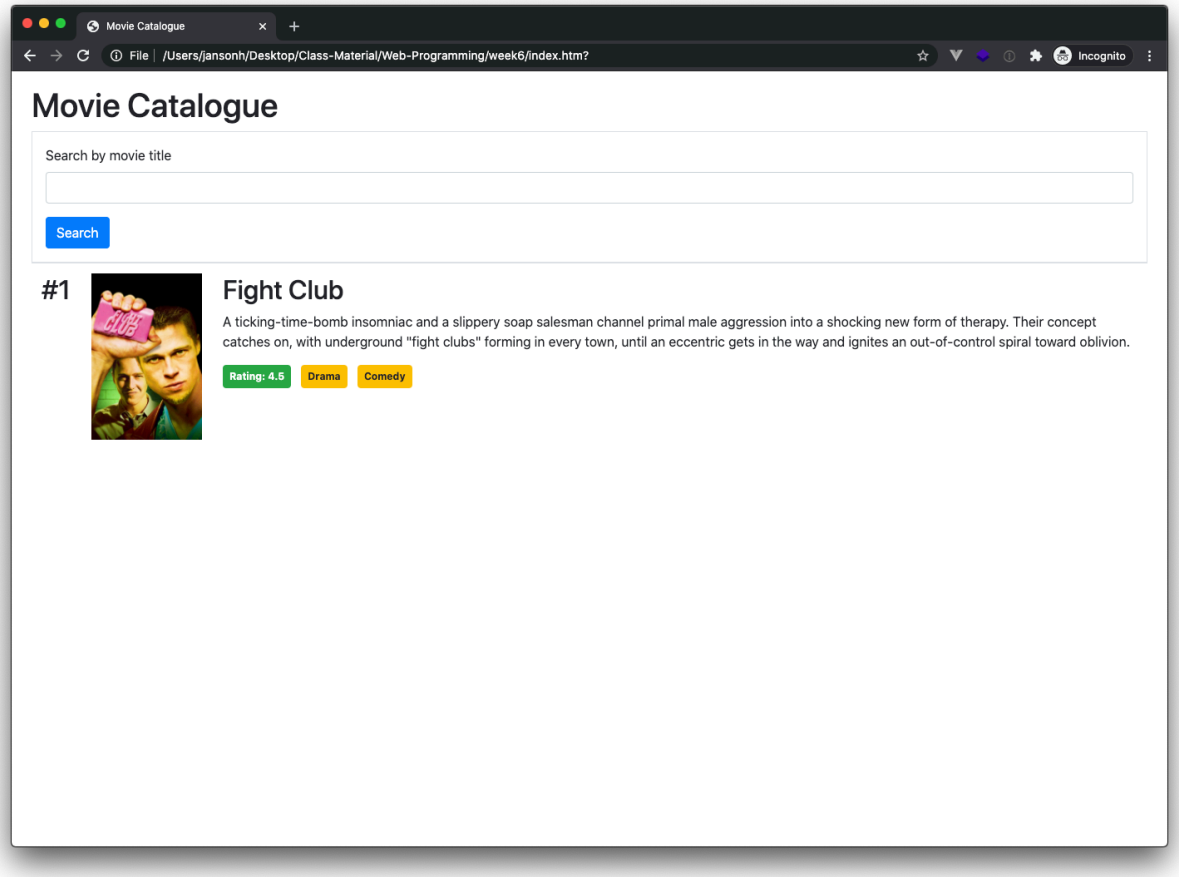
```
<div class="container-fluid my-3 px-4">
    <h1>Movie Catalogue</h1>
    <form class="form border p-3">
      <div class="form-group">
        <label for="title">Search by movie title</label>
        <input type="text" class="form-control" id="title"
               aria-describedby="error" required>
        <small class="form-text text-danger" id="error"></small>
      </div>
      <button type="submit" class="btn btn-primary" id="searchButton">
         Search
      </button>
    </form>

    <table class="table" id="result">

    </table>
  </div>
```

Kode di atas akan menghasilkan sebuah form dengan textbox untuk mengisi teks pencarian dan button untuk melakukan proses pencarian. Pada saat button search diklik, maka akan dibuat kode dalam Javascript untuk melakukan pencarian film di TMDb. Hasil pencarian tersebut akan disajikan pada table yang ada pada contoh kode program di atas.

Format dari setiap baris pada table yang merupakan hasil dari pencarian film adalah sebagai berikut yang akan menghasilkan tampilan seperti pada gambar 3.

```
<tr>
        <td width="10"><h2 class="display-5">#1</h2></td>
        <td width="100"><img
src="https://image.tmdb.org/t/p/w500/pB8BM7pdSp6B6Ih7QZ4DrQ3PmJK.jpg"
height="200"></td>
        <td>
          <h2 class="display-5">Fight Club</h2>
          <p>A ticking-time-bomb insomniac and a slippery soap salesman
channel primal male aggression into a shocking new form of therapy. Their
concept catches on, with underground "fight clubs" forming in every town,
```

```
until an eccentric gets in the way and ignites an out-of-control spiral toward
oblivion.</p>
          <span class="badge badge-success p-2">Rating: 4.5</span>
          <span class="badge badge-warning ml-2 p-2">Drama</span>
          <span class="badge badge-warning ml-2 p-2">Comedy</span>
        </td>
      </tr>
```



Gambar 3. Contoh tampilan awal.

Selanjutnya kita akan membuat script untuk melakukan pencarian film pada `script.js`. Jangan lupa tambahkan `<script src="script.js"></script>` pada kode HTML di atas. Ketikkan kode di bawah ini dan isi API key anda pada konstanta `API_KEY`.

```
[ 1] const API_KEY = '<YOUR-API-KEY>';

[ 2] // retrieve all movie genres when the document is ready
[ 3] var genres = {};
[ 4] $(() => {
[ 5] });
[ 6]
[ 7] // search movie when the button is clicked
[ 8] $('#searchButton').click((e) => {
[ 9] })
```

Terdapat dua bagian utama dari kode yang akan kita buat di atas, yaitu pada baris 4-5 dan baris 8-9. Pada baris 4 merupakan kode dari jQuery yang akan dijalankan pada saat dokumen HTML kita telah ter-*load* pada client. Pada blok kode tersebut, kita akan menambahkan kode untuk memanggil API pada TMDb dan mendapatkan daftar genre film yang tersedia (kemudian disimpan pada variabel `genres`).

Untuk mendapatkan genre film pada TMDb, API yang digunakan adalah yang ada pada URL berikut ini https://api.themoviedb.org/3/genre/movie/list dengan menambahkan juga parameter API key. Hasil dari pemanggilan API di atas adalah dalam format JSON dan dapat dicoba melalui browser seperti di bawah ini:


```
{
   "genres":[
      {
         "id":28,
         "name":"Action"
      },
      {
         "id":12,
         "name":"Adventure"
      },
      {
         "id":16,
         "name":"Animation"
      },
      {
         "id":35,
         "name":"Comedy"
      },
      {
         "id":80,
         "name":"Crime"
      },
      {
         "id":99,
         "name":"Documentary"
      },
      {
         "id":18,
         "name":"Drama"
      },
      {
         "id":10751,
         "name":"Family"
      },
      ... (truncated)
   ]
}
```

Hasil tersebut akan kita simpan dalam *dictionary* bernama `genres`, sehingga kita bisa mendapatkan nama genre dari suatu film jika diketahui ID-nya.

Tambahkan kode di bawah ini di antara baris 4 dan 5:

```
$(() => {
  $.get('https://api.themoviedb.org/3/genre/movie/list', { api_key: API_KEY })
    .done((r) => {
      r.genres.forEach((genre) => {
        genres[genre.id] = genre.name;
      })
    })
    .fail((e) => {
      alert(e.status_message);
    })
});
```

Perintah `$.get(url, data)` menggunakan fungsi pada jQuery untuk melakukan pemanggilan ke `url` dengan *method* GET dan dengan mengirimkan juga parameter `data`. *Method* pemanggilan sendiri terdiri dari berbagai macam, seperti GET, POST, PUT, DELETE, dan seterusnya, yang dapat diketahui dari dokumentasi API tersebut. *Method* GET biasanya digunakan untuk mendapatkan sebuah informasi dari server, sedangkan POST untuk mengirimkan data ke server (misalkan untuk disimpan ke database).

Jika pemanggilan API tersebut berhasil, maka kode pada `.done((r) => { ... })` akan dijalankan, yaitu untuk melakukan looping pada hasil berformat JSON di atas dan menyimpan genre film. Sementara itu, bagian `.fail((e) => { ... })` merupakan bagian kode yang dipanggil jika terdapat error pada saat pemanggilan API.

Kemudian, kita akan melanjutkan kode program di antara baris 8 dan 9 yang merupakan bagian kode program yang dipanggil pada saat tombol search ditekan. Perhatikan jika pada materi pertemuan sebelumnya mengenai DOM, kita melakukan penambahan event listener untuk action click dengan sintaks:

```
document.getElementById(“btn”).addEventListener(“click”, function (event) { ... })
```

maka pada jQuery perintah tersebut dapat dipersingkat menjadi:

```
$(‘#btn’).click(function (event) { ... })
```

atau lebih singkat lagi dengan menggunakan anonymous function:

```
$(‘#btn’).click((event) => { ... })
```

Berikut merupakan kode akhir dari bagian tersebut:

```
// search movie when the button is clicked
$('#searchButton').click((e) => {
  // clear error message if exists
  $('#error').text('');

  // disabled search button
  $('#searchButton')
    .empty()
    .attr('disabled', 'disabled')
    .append($('<span class="spinner-border spinner-border-sm" role="status"
aria-hidden="true"></span>'))
    .append(' Loading...');

  // clear all previous result from the table
  $('#result').empty();

  // the data to be sent with the GET request
  const data = {
    api_key: API_KEY,
    query: $('#title').val(),
    include_adult: false
  };

  // send GET request
  $.get('https://api.themoviedb.org/3/search/movie', data)
    .done((r) => {
      if (r.results.length === 0) {
        $('#error').text('!!! No movie with this title.')
      } else {
        r.results.forEach((movie) => {
          const tableCell = createCell(movie);
          $('#result').append(tableCell);
        })
      }
    })
    .fail((e) => {
      $('#error').text(`!!! ${e.status_message}`);
    })
    .always(() => {
      // re-enable the search button
      $('#searchButton')
      .empty()
      .removeAttr('disabled')
      .append('Search');
    });
})
```

Di bawah ini merupakan penjelasan per bagian dari kode di atas.

Pertama-tama, kita akan menghapus pesan error jika pada pemanggilan sebelumnya sudah pernah ada error.

```
// clear error message if exists
$('#error').text('');
```

Selanjutnya, pada saat button search ditekan, kita akan mengubah tampilannya sehingga button tersebut tidak dapat ditekan selama kode program sedang melakukan pencarian. Tampilan pada button tersebut akan diganti dengan tulisan loading.

```
// disabled search button
$('#searchButton')
  .empty()
  .attr('disabled', 'disabled')
  .append($('<span class="spinner-border spinner-border-sm" role="status"
aria-hidden="true"></span>'))
  .append(' Loading...');
```

Kemudian, jika sebelumnya pengguna telah melakukan pencarian, maka kita akan menghapus hasil sebelumnya pada table.

```
// clear all previous result from the table
$('#result').empty();
```

Bagian kode selanjutnya adalah untuk mendefinisikan data yang akan dikirimkan ke API pencarian film. Data tersebut adalah API key, judul yang diketik user, dan parameter-parameter lain yang mungkin saja dibutuhkan. Pilihan parameter-parameter ini dapat dilihat pada dokumentasi TMDb. Misalkan pada contoh ini kita akan menghilangkan film-film dewasa dengan parameter `include_adult` diisi dengan nilai `false`.

```
// the data to be sent with the GET request
const data = {
  api_key: API_KEY,
  query: $('#title').val(),
  include_adult: false
};
```

Kode tersebut kemudian dilanjutkan dengan pemanggilan API itu sendiri dengan bentuk kode sebagai berikut:

```
// send GET request
$.get('https://api.themoviedb.org/3/search/movie', data)
  .done((r)  => { ... })
  .fail((e)  => { ... })
  .always(() => { ... });
```

Bagian pertama merupakan URL dan data untuk pencarian film berdasarkan judul yang diketik oleh pengguna. Perintah tersebut kemudian terbagi menjadi 3 bagian, yaitu bagian kode yang dijalankan jika pemanggilan API berhasil dilaksanakan (`done`), jika gagal atau terjadi error (`fail`), dan kode yang harus dijalankan pada saat semua pemanggilan selesai (baik jika berhasil maupun error -- `always`).

Pada bagian `fail`, tambahkan kode berikut untuk menampilkan error message.

```
.fail((e) => {
  $('#error').text(`!!! ${e.status_message}`);
})
```

Sementara pada bagian always, tambahkan kode untuk mengembalikan button search seperti semula.

```
.always(() => {
  // re-enable the search button
  $('#searchButton')
    .empty()
    .removeAttr('disabled')
    .append('Search');
});
```

Bagian kode utama dari program ini adalah yang ada pada bagian done. Setelah proses pemanggilan selesai dan berjalan sukses, maka pertama-tama kita perlu mengecek apakah hasil pencarian mengembalikan hasil (atau apakah film tersebut ditemukan). Jika tidak ada, maka kita perlu memunculkan pesan error.

```
.done((r) => {
  if (r.results.length === 0) {
    $('#error').text('!!! No movie with this title.')
  } else {
    ...
  }
})
```

Variabel `r` merupakan hasil dari pemanggilan API. Untuk mengecek apakah ada hasil, maka kita melihat panjang dari array `results` pada variabel `r` tersebut. Perhatikan contoh JSON hasil pencarian film di bawah ini:


```
{
   "page":1,
   "total_results":1674,
   "total_pages":84,
   "results":[
      {
         "popularity":76.642,
```

```
         "vote_count":25,
         "video":false,
         "poster_path":"\/cDbOrc2RtIA37nLm0CzVpFLrdaG.jpg",
         "id":513584,
         "adult":false,
         "backdrop_path":"\/qEo36RiT8w4hMFlf9KohOSuX3HZ.jpg",
         "original_language":"en",
         "original_title":"Think Like a Dog",
         "genre_ids":[
            35,
            18,
            10751
         ],
         "title":"Think Like a Dog",
         "vote_average":6.3,
         "overview":"A 12-year-old tech prodigy whose science experiment goes
awry and he forges a telepathic connection with his best friend, his dog. The
duo join forces and use their unique perspectives on life to comically
overcome complications of family and school.",
         "release_date":"2020-08-06"
      },
      ... (truncated)
   ]
}
```

Hasil pencarian film berada pada key “results” yang merupakan array dari informasi-informasi film tersebut. Maka jika “results” kosong, maka pencarian film tidak ditemukan. Jika ada hasil, maka isilah kode berikut ini pada bagian else.

```
...
} else {
  r.results.forEach((movie) => {
    const tableCell = createCell(movie);
    $('#result').append(tableCell);
  })
}
...
```

Pada potongan kode di atas, kita melakukan looping pada array `results` untuk mendapatkan informasi film yang ada pada array tersebut. Di setiap iterasi, kita akan memanggil fungsi `createCell(movie)` yang menghasilkan elemen `<tr>` yang ditambahkan ke table `result` pada perintah selanjutnya.

Dengan melihat contoh tampilan setiap baris pada table di halaman 4-5, maka tambahkanlah kode di bawah ini:

```
function createCell(movie) {
  var row = $('<tr></tr>');
```

```
  // row number
  const movieNo = $('#result > tr').length + 1;
  var colNo = $('<td width="10"></td>');
  colNo.append($('<h2 class="display-5"></h2').text(`#${movieNo}`));
  row.append(colNo);

  // poster
  const posterUrl = (movie.poster_path !== null) ?
    `https://image.tmdb.org/t/p/w500${movie.poster_path}` :
    '';
  var colPoster = $('<td width="100"></td>');
  colPoster.append($(`<img src="${posterUrl}" height="200">`));
  row.append(colPoster);

  // movie information
  const td = $('<td></td>');
  row.append(td);

  // title
  const title = $('<h2 class="display-5"></h2>').text(movie.title);
  td.append(title);

  // overview
  const overview = $('<p></p>').text(movie.overview);
  td.append(overview);

  // rating
  const rating = $('<span class="badge badge-success p-
2"></span>').text(`Rating: ${movie.vote_average}`);
  td.append(rating);

  // genres
  movie.genre_ids.forEach((id) => {
    const genre = $('<span class="badge badge-warning ml-2 p-
2"></span>').text(genres[id]);
    td.append(genre);
  });

  return row;
}
```

Pada potongan kode di atas, mula-mula kita membuat element baru dengan perintah:

```
var row = $(‘<tr></tr>’);
```

Padanan dari baris kode di atas pada Javascript tanpa jQuery kira-kira adalah:

```
var row = document.createElement(‘tr’);
```

Kemudian pada barisan kode selanjutnya adalah proses membuat elemen HTML untuk menampilkan informasi judul, rating, genre, dan lain-lain dari film yang didapat. Beberapa bagian kode yang mungkin perlu diperhatikan adalah pada pemunculan gambar poster film dan genre.

```
const posterUrl = (movie.poster_path !== null) ?
    `https://image.tmdb.org/t/p/w500${movie.poster_path}` :
    '';
```

Pada perintah di atas, yang kita lakukan sebenarnya adalah mengisi variabel `posterUrl` berdasarkan suatu kondisi apakah `movie.poster_path` bernilai null atau tidak. Jika null, maka tidak ada gambar poster yang dapat ditampilkan. Sedangkan jika tidak null, maka tampilkan gambar yang ada pada url di atas.

Sementara pada bagian kode untuk menampilkan genre, ingat-ingat kembali bagian kode yang dimulai di baris 4. Mula-mula kita telah mendapatkan daftar genre film yang ada pada TMDb dengan bentuk struktur data *dictionary*. Jika melihat hasil JSON pencarian film, yang didapatkan hanyalah id dari genre film.

Misalkan pada film “Think Like a Dog” didapatkan hasil pemanggilan API untuk genre adalah sebagai berikut:

```
"genre_ids":[
    35,
    18,
    10751
]
```

Maka kita perlu melakukan looping untuk `movie.genre_ids` dan mendapatkan nama genre dari `id` tersebut dari *dictionary* `genres` dengan perintah `genres[id]`.