

# Studio Ghibli Part 1

---

Introduction to 3rd Party APIs

# What is an API?

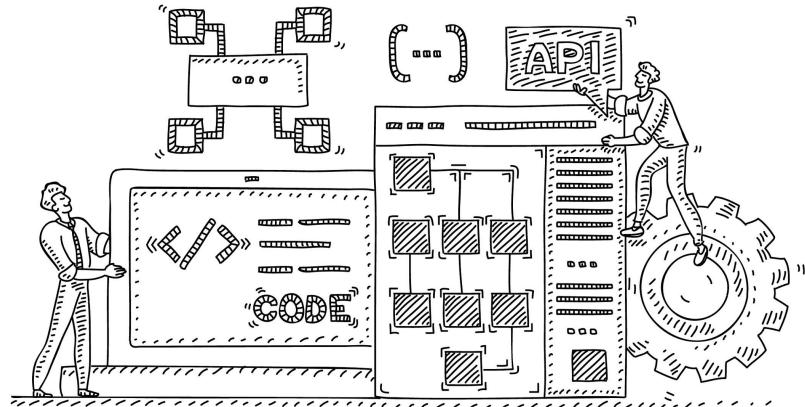
- An application interface provides access to features of a program or function.
- Just like how the accelerator in a car performs functions under the hood that you don't need to know about.



# You're Already Working with APIs

We have been using APIs all throughout our lessons, without specifically calling attention to them.

Fetch and setTimeOut are browser APIs. So is accessing the DOM through methods like getElementById.



APPLICATION PROGRAMMING INTERFACE

# 3rd Party APIs

3rd party APIs frequently provide data you can interact with, usually in a JSON format, which is easily consumed by JavaScript.



# Studio Ghibli Project

For this project, we will use the Studio Ghibli API which provides information on movies created by this fantastic Japanese animation studio.

Studio Ghibli

File | /Users/wmmead/Desktop/studioghibli4/index.html

スタジオジブリ作品  
STUDIO GHIBLI

FILMS    PEOPLE    LOCATIONS    SPECIES    VEHICLES

Release Date

**Castle in the Sky**  
Director: Hayao Miyazaki  
Released: 1986  
The orphan Sheeta inherited a mysterious crystal that links her to the mythical sky-kingdom of Laputa. With the help of resourceful Pazu and a rollicking band of sky pirates, she makes her way to the ruins of the once-great civilization. Sheeta and Pazu must outwit the evil Muska, who plans to use Laputa's science to make himself ruler of the world.  
Rotten Tomatoes Score: 95

**My Neighbor Totoro**  
Director: Hayao Miyazaki  
Released: 1988  
Two sisters move to the country with their father in order to be closer to their hospitalized mother, and discover the surrounding trees are inhabited by Totoros, magical spirits of the forest. When the youngest runs away from home, the older sister seeks help from the spirits to find her.  
Rotten Tomatoes Score: 93

**Grave of the Fireflies**  
Director: Isao Takahata  
Released: 1988  
In the latter part of World War II, a boy and his sister, orphaned when their mother is killed in the firebombing of Tokyo, are left to survive on their own in what remains of civilian life in Japan. The plot follows this boy and his sister as they do their best to survive in the Japanese countryside, battling hunger, prejudice, and pride in their own quiet, personal battle.  
Rotten Tomatoes Score: 97

# Studio Ghibli API Documentation

The place to start with any API is with the documentation, so you know what it can do and how to use it.

The screenshot shows the Studio Ghibli API documentation for the 'Films' endpoint. The left sidebar lists endpoints: FILMS (selected), PEOPLE, LOCATIONS, SPECIES, and VEHICLES. The main content area has a title 'Return all films' and a description: 'The Films endpoint returns information about all of the Studio Ghibli films.' It shows 'PARAMETERS' for 'fields' (string, comma-separated list of fields to include in the response) and 'limit' (integer <int64>, amount of results (default 50) (maximum 250)). Below is a 'Responses' section with three items: '200 An array of films' (green), '400 Bad request' (red), and '404 Not found' (pink). To the right, there's a 'REQUEST SAMPLES' section with a 'curl' button and code samples for Ruby and Python, and a 'RESPONSE SAMPLES' section showing a JSON array of film objects. One object is highlighted with a green border.

```
curl -X GET .H "Content-Type: application/json" https://ghibliapi.herokuapp.com/films
```

```
[{"id": "2ba70d1-42bb-4437-b551-e5fed", "title": "Castle in the Sky", "description": "The orphan Sheeta and Pazu must stop the evil Laputa from taking over the world.", "director": "Hayao Miyazaki", "producer": "Isao Takahata", "release_date": "1986", "rt_score": "95"}, {"id": "12cfb892-ac00-4c5b-94af-52185", "title": "Grave of the Fireflies", "description": "In the latter part of World War II, two young orphans, Seita and Setsuko, struggle to survive after their parents are killed in a bombing raid.", "director": "Isao Takahata", "producer": "Toru Hara", "release_date": "1988", "rt_score": "97"}]
```

# Accessing the Films

According to the API, you can go to this endpoint to access data on all the films:

<https://ghibliapi.herokuapp.com/films>



The screenshot shows a browser window with the title "Studio Ghibli API" and the URL "https://ghibliapi.herokuapp.com/films". The page displays a JSON array of film objects. The first object is for "Castle in the Sky", which has an id of "2bafe70d1-42bb-4437-b551-e5fed5a87abe". It includes details like the director (Hayao Miyazaki), producer (Isao Takahata), release date (1986), RT score (95), and a plot summary about Sheeta and Pazu. Other fields include people, species, locations, vehicles, and a URL for the film itself. The second object is for "Grave of the Fireflies", with an id of "12cf892-aac0-4c5b-94af-521852e46d6a". It follows a similar structure, detailing the plot of the movie about a boy and his sister surviving after their mother's death.

```
[{"id": "2bafe70d1-42bb-4437-b551-e5fed5a87abe", "title": "Castle in the Sky", "description": "The orphan Sheeta inherited a mysterious crystal that links her to the mythical sky-kingdom of Laputa. With the help of resourceful Pazu and a rollicking band of sky pirates, she makes her way to the ruins of the once-great civilization. Sheeta and Pazu must outwit the evil Muska, who plans to use Laputa's science to make himself ruler of the world.", "director": "Hayao Miyazaki", "producer": "Isao Takahata", "release_date": "1986", "rt_score": "95", "people": [{"url": "https://ghibliapi.herokuapp.com/people/"}], "species": [{"url": "https://ghibliapi.herokuapp.com/species/af3910a6-429f-4c74-9ad5-dfe1c4aa04f2"}], "locations": [{"url": "https://ghibliapi.herokuapp.com/locations/"}], "vehicles": [{"url": "https://ghibliapi.herokuapp.com/vehicles/"}], "url": "https://ghibliapi.herokuapp.com/films/2bafe70d1-42bb-4437-b551-e5fed5a87abe"}, {"id": "12cf892-aac0-4c5b-94af-521852e46d6a", "title": "Grave of the Fireflies", "description": "In the latter part of World War II, a boy and his sister, orphaned when their mother is killed in the firebombing of Tokyo, are left to survive on their own in what remains of civilian life in Japan. The plot follows this boy and his sister as they do their best to survive in the Japanese countryside, battling hunger, prejudice, and pride in their own quiet, personal battle.", "director": "Isao Takahata", "producer": "Toru Hara", "release_date": "1988", "rt_score": "97", "people": [{"url": "https://ghibliapi.herokuapp.com/people/"}], "species": [{"url": "https://ghibliapi.herokuapp.com/species/af3910a6-429f-4c74-9ad5-dfe1c4aa04f2"}], "locations": [{"url": "https://ghibliapi.herokuapp.com/locations/"}], "vehicles": [{"url": "https://ghibliapi.herokuapp.com/vehicles/"}]}
```

# Try This To Start

```
const mainElement = document.querySelector('main');

async function getFilms() {
  const filmsPromise = await fetch('https://ghibliapi.herokuapp.com/films');
  const films = await filmsPromise.json();
  console.log(films);
}

getFilms();
```

You are not putting anything on the page yet with this, but start by getting the <main> element on the document and putting it into a variable. That is where this will go, shortly.

# Creating the Card

The task is to create a card for each movie. If you look at the Studio Ghibli API for films, you will see the type of data you can get for each movie.

```
[           Copy   Expand all   Collapse all
- {
    "id": "2baf70d1-42bb-4437-b551-
    "title": "Castle in the Sky",
    "description": "The orphan Shee
    "director": "Hayao Miyazaki",
    "producer": "Isao Takahata",
    "release_date": "1986",
    "rt_score": "95"
},
- {
    "id": "12cfb892-aac0-4c5b-94af-
    "title": "Grave of the Firefly
    "description": "In the latter p
    "director": "Isao Takahata",
    "producer": "Toru Hara",
    "release_date": "1988",
    "rt_score": "97"
}
]
```

## Howl's Moving Castle

Director: Hayao Miyazaki

Released: 2004

When Sophie, a shy young woman, is cursed with an old body by a spiteful witch, her only chance of breaking the spell lies with a self-indulgent yet insecure young wizard and his companions in his legged, walking home.

Rotten Tomatoes Score: 87

# Creating The Card

Javascript will create the card, which is just an article element with the h2 set to be the heading. Create each element, create each text node and append it to the element, then append all the elements to the card, then append the card to the page.

```
function createCard(data) {  
    const card = document.createElement('article');  
    const movieTitle = document.createElement('h2');  
    const movieTitleTxt = document.createTextNode(data.title);  
    movieTitle.appendChild(movieTitleTxt);  
  
    const director = document.createElement('p');  
    const directorTxt = document.createTextNode(`Director: ${data.director}`);  
    director.appendChild(directorTxt);  
  
    //write similar code for year, description and score  
  
    card.appendChild(movieTitle);  
    card.appendChild(director);  
    // append the year, description and score  
  
    mainElement.appendChild(card);  
}
```

# The Whole Function

Here is a picture of the whole function you should have been able to finish from the previous slide.

```
function createCard(data) {  
    const card = document.createElement('article');  
    const movieTitle = document.createElement('h2');  
    const movieTitleTxt = document.createTextNode(data.title);  
    movieTitle.appendChild(movieTitleTxt);  
  
    const director = document.createElement('p');  
    const directorTxt = document.createTextNode(`Director: ${data.director}`);  
    director.appendChild(directorTxt);  
  
    const year = document.createElement('p');  
    const yearTxt = document.createTextNode(`Released: ${data.release_date}`);  
    year.appendChild(yearTxt);  
  
    const description = document.createElement('p');  
    const descriptionTxt = document.createTextNode(data.description);  
    description.appendChild(descriptionTxt);  
  
    const rating = document.createElement('p');  
    const ratingTxt = document.createTextNode(`Rotten Tomatoes Score: ${data.rt_score}`);  
    rating.appendChild(ratingTxt);  
  
    card.appendChild(movieTitle);  
    card.appendChild(director);  
    card.appendChild(year);  
    card.appendChild(description);  
    card.appendChild(rating);  
  
    mainElement.appendChild(card);  
}
```

# Updated getFilms() Function

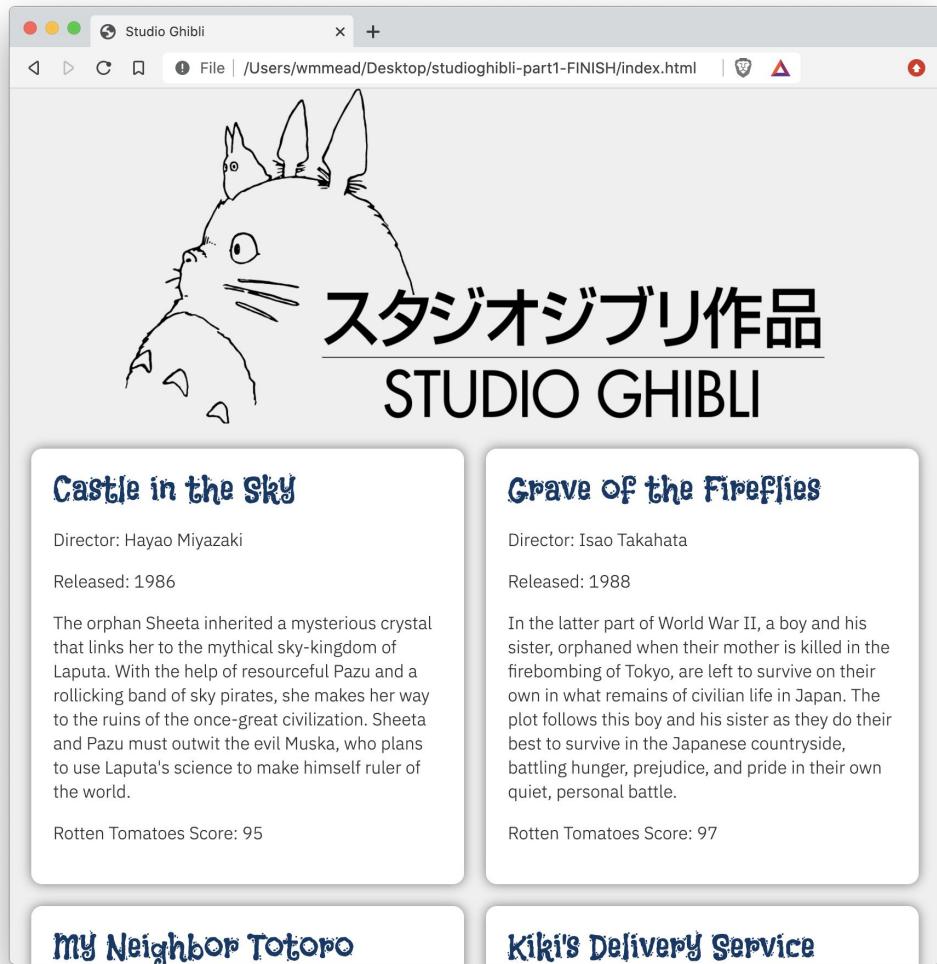
```
async function getFilms() {  
  const filmsPromise = await fetch('https://ghibliapi.herokuapp.com/films');  
  const films = await filmsPromise.json();  
  
  films.forEach(eachFilm => {  
    createCard(eachFilm);  
  });  
}
```

Did you get something like this?

The variable **eachFilm** represents each element in the array, which is, in fact a film. So a card is created and added to the <main> element for each film.

# Results So Far

Since the CSS is already written, it should just display rather nicely.



The screenshot shows a web browser window titled "Studio Ghibli". The address bar indicates the file is located at "/Users/wmmead/Desktop/studioghibli-part1-FINISH/index.html". The page features a large illustration of a young girl with pigtails, looking over her shoulder. To the right of the illustration, the text "スタジオジブリ作品" (Studio Ghibli Works) is displayed in large, stylized Japanese characters above the word "STUDIO GHIBLI" in English. Below this, there are two movie cards: "Castle in the Sky" and "Grave of the Fireflies". Each card includes the movie title, director, release year, a brief plot summary, and the Rotten Tomatoes score. At the bottom of the page, there are links for "my Neighbor Totoro" and "Kiki's Delivery Service".

Movie	Director	Released	Rotten Tomatoes Score
Castle in the Sky	Hayao Miyazaki	1986	95
Grave of the Fireflies	Isao Takahata	1988	Not explicitly shown, but implied to be high based on context.
my Neighbor Totoro	Hayao Miyazaki	Not explicitly shown, but implied to be high based on context.	Not explicitly shown, but implied to be high based on context.
Kiki's Delivery Service	Not explicitly shown, but implied to be high based on context.	Not explicitly shown, but implied to be high based on context.	Not explicitly shown, but implied to be high based on context.

# Sorting the Films

```
async function getFilms() {  
  const filmsPromise = await fetch('https://ghibliapi.herokuapp.com/films');  
  const films = await filmsPromise.json();  
  
  films.sort((a, b) => (a.title > b.title) ? 1 : -1);  
  
  films.forEach(eachFilm => {  
    createCard(eachFilm);  
  });  
}
```

Check out the additional line of code highlighted in the script.

# Alphabetically Sorted

You can see the films are sorted alphabetically. However, you might want some choices in how they are sorted. For example you might want to sort by release year or by rating.



The screenshot shows a web browser window titled "Studio Ghibli". The page displays four movie cards in a grid:

- Arrietty**  
Director: Hiromasa Yonebayashi  
Released: 2010  
14-year-old Arrietty and the rest of the Clock family live in peaceful anonymity as they make their own home from items 'borrowed' from the house's human inhabitants. However, life changes for the Clocks when a human boy discovers Arrietty.  
Rotten Tomatoes Score: 95
- Castle in the Sky**  
Director: Hayao Miyazaki  
Released: 1986  
The orphan Sheeta inherited a mysterious crystal that links her to the mythical sky-kingdom of Laputa. With the help of resourceful Pazu and a rollicking band of sky pirates, she makes her way to the ruins of the once-great civilization. Sheeta and Pazu must outwit the evil Muska, who plans to use Laputa's science to make himself ruler of the world.  
Rotten Tomatoes Score: 95
- From Up on Poppy Hill**  
Director: Gorō Miyazaki
- Grave of the Fireflies**  
Director: Isao Takahata

# Add This Form

Add this nav element after the <header> on the index.html file.

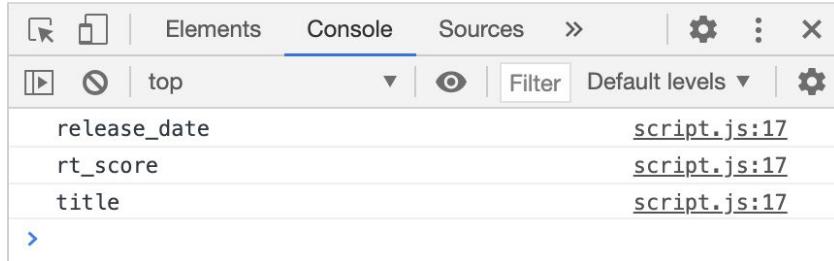
```
<nav>
  <form>
    <select id="sortorder">
      <option value="title">Title</option>
      <option value="release_date">Release Date</option>
      <option value="rt_score">Rotten Tomatoes Score</option>
    </select>
  </form>
</nav>

nav {
  padding: 20px;
  text-align: center;
}
```

# Listening for Change

```
document.getElementById('sortorder').addEventListener('change', function () {
  console.log(document.getElementById('sortorder').value);
});
```

You need an event handler that captures when the user changes the value of the select list.



# Sorting Function

```
function setSort(array) {  
  const sortOrder = document.getElementById('sortorder').value;  
  switch (sortOrder) {  
    case 'title': array.sort((a, b) => (a.title > b.title) ? 1 : -1); break;  
    case 'release_date': array.sort((a, b) => (a.release_date > b.release_date) ? 1 : -1); break;  
    case 'rt_score': array.sort((a, b) => (a.rt_score > b.rt_score) ? 1 : -1); break;  
  }  
}
```

Next, you need a function that will take the value from the select list, and run a different sort based on that value.

This function is using a switch statement, but could just as easily use if/else if statements.

# Part 1 - Running the Sort

Run the setSort() function inside the getFilms function. Pass in the films array.

When the page loads, the select list will be on the title option by default and the films will sort by title.

```
async function getFilms() {
  const filmsPromise = await
fetch('https://ghibliapi.herokuapp.com/films');
  const films = await filmsPromise.json();

  setSort(films);

  films.forEach(eachFilm => {
    createCard(eachFilm);
  });
}
```

# Part 2 - Changing the Sort

```
document.getElementById('sortorder').addEventListener('change', function () {
  mainElement.innerHTML = '';
  getFilms();
});
```

Clear out all the HTML in the main element, then run getFilms(). Since the select list value has changed, a new sort will be passed in and the getFilms() function will get the films and apply the new sort to them.

# Rotten Tomatoes Sort Fixed

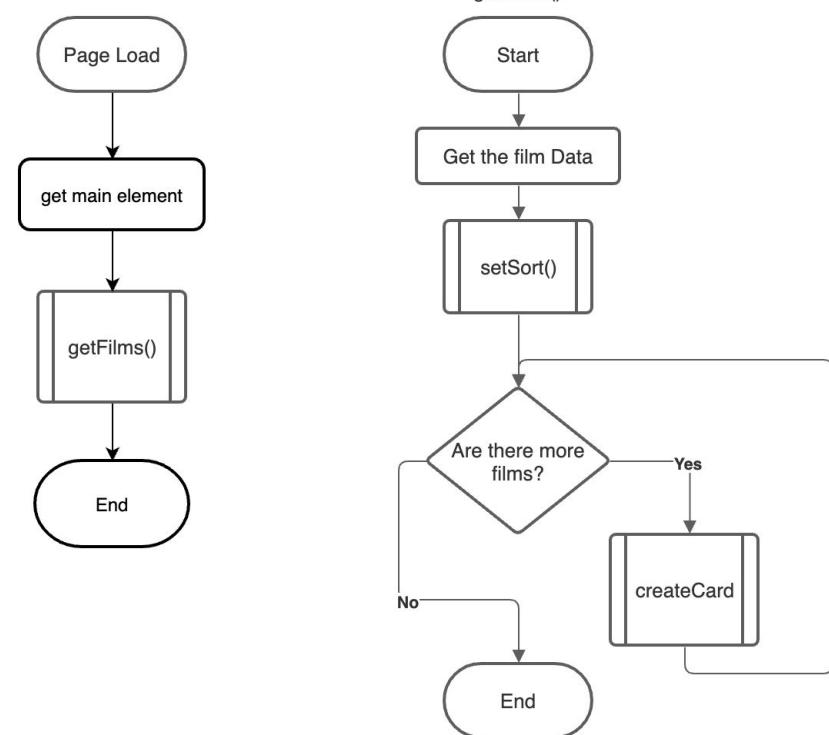
```
case 'rt_score': array.sort((a, b) => (parseInt(a.rt_score) > parseInt(b.rt_score)) ? -1 : 1); break;
```

Did you solve it?

If you wrap the a.rt\_score and b.rt\_score with parseInt(),  
it will convert those two strings to numbers.

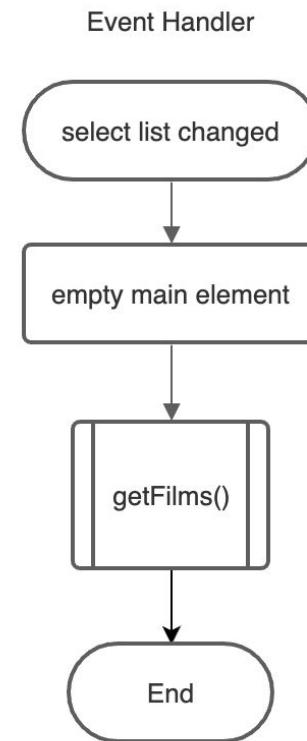
# The Current Flow

In the current flow, the page loads, we get the main element from the DOM and then run the `getFilms()` function, which gets the data, sets the sort order, and loops through the data, creating the cards and adding them to the page.



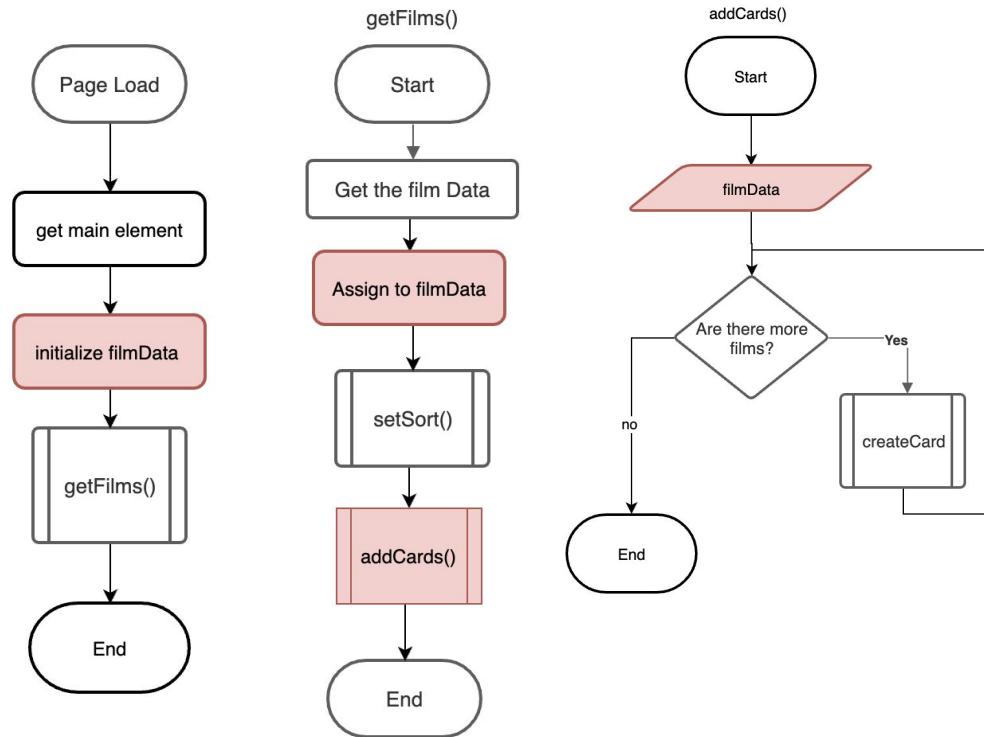
# The Problem

The problem is that when the event handler detects a change in the select list, it runs `getFilms()` again (with a new sort), and that forces a reload of all the data from the API.



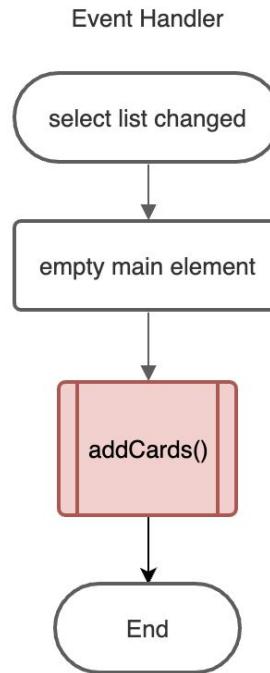
# The Solution Strategy

- Initialize the variable, filmData
- Pull the loop out and create a function called addCards()



# The Event Handler

Now the event handler can run addCards() instead of getFilms(). Since addCards() is pulling data from the filmData variable, there is no reason to go back to the API to get the data.



# Add the filmData Variable

```
const mainElement = document.querySelector('main');

let filmData;

async function getFilms() {
```

Add the variable filmData, just after the variable for the mainElement. It has no value to start with, but you want it in the global scope of the script.

# AddCards Function

Next add the addCards() function, which takes an array as an input. This is just a very simple function for looping through the array.

```
function addCards(array) {  
    array.forEach(eachItem => {  
        createCard(eachItem);  
    });  
}
```

# Update the getFilms() Function

```
async function getFilms() {  
  const filmsPromise = await fetch('https://ghibliapi.herokuapp.com/films');  
  const films = await filmsPromise.json();  
  
  setSort(films);  
  addCards(films);  
  filmData = films;  
}
```

Update the getFilms() function with the addCards() function, instead of the forEach loop that adds all the cards. You pass in the films array. Then you assign that same films array to the filmData variable created in the global scope.

# Update the Click Handler

```
document.getElementById('sortorder').addEventListener('change', function () {  
    mainElement.innerHTML = '';  
    setSort(filmData);  
    addCards(filmData);  
});
```

Update the click handler so that it is calling addCards() instead of getFilms().

Pass addCards() the variable with all the data in it and that will keep the script from going back to the API to get the data again.

# One More Thing

It is possible, although unlikely, that a user could try to use the sort order select list *before* the data has loaded from the API. This would run the addCard() function, which passes in the filmData array, which would be empty at this time, therefore throwing an error.

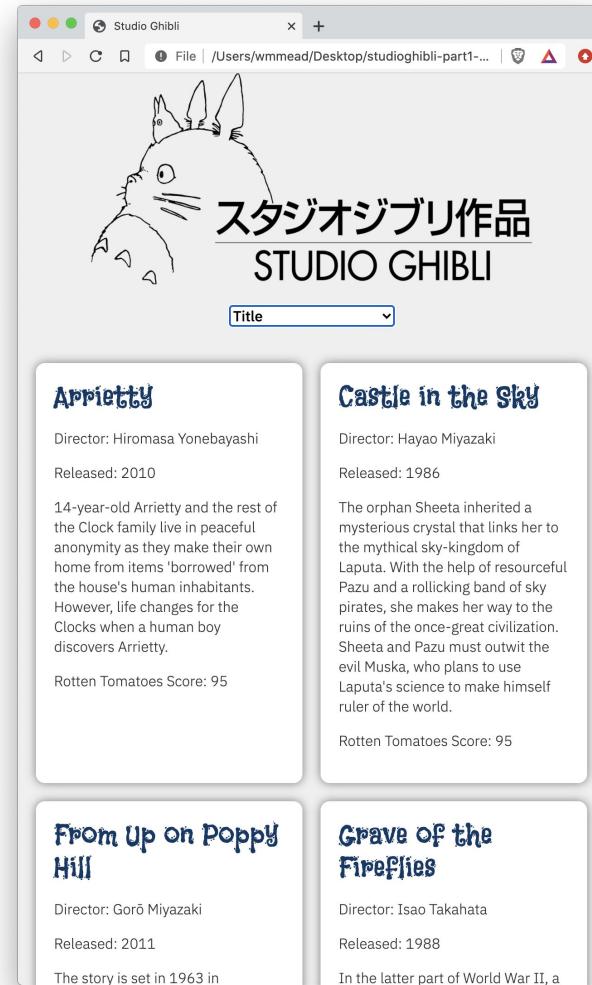
In the HTML just add a disabled attribute to the select list, as shown here. Then, in the getFilms() function, at the very bottom, remove that attribute.

```
<form>
  <select id="sortorder" disabled>
    <option value="title">Title</option>
    <option value="release_date">Release Date</option>
    <option value="rt_score">Rotten Tomatos Score</option>
  </select>
</form>
```

```
document.getElementById('sortorder').removeAttribute('disabled');
```

# Summary

In this project you have taken data available through a free API and consumed it with JavaScript, then used asynchronous functions to display the data on a web page.



The screenshot shows a web browser window titled "Studio Ghibli". The page features a large illustration of a catbus from My Neighbor Totoro. Below the illustration, the text "スタジオジブリ作品" and "STUDIO GHIBLI" is displayed. A dropdown menu is open, showing the word "Title". The main content area displays four movie cards:

- Arrietty**  
Director: Hiromasa Yonebayashi  
Released: 2010  
14-year-old Arrietty and the rest of the Clock family live in peaceful anonymity as they make their own home from items 'borrowed' from the house's human inhabitants. However, life changes for the Clocks when a human boy discovers Arrietty.  
Rotten Tomatoes Score: 95
- Castle in the Sky**  
Director: Hayao Miyazaki  
Released: 1986  
The orphan Sheeta inherited a mysterious crystal that links her to the mythical sky-kingdom of Laputa. With the help of resourceful Pazu and a rollicking band of sky pirates, she makes her way to the ruins of the once-great civilization. Sheeta and Pazu must outwit the evil Muska, who plans to use Laputa's science to make himself ruler of the world.  
Rotten Tomatoes Score: 95
- From Up on Poppy Hill**  
Director: Gorō Miyazaki  
Released: 2011  
The story is set in 1963 in
- Grave of the Fireflies**  
Director: Isao Takahata  
Released: 1988  
In the latter part of World War II, a