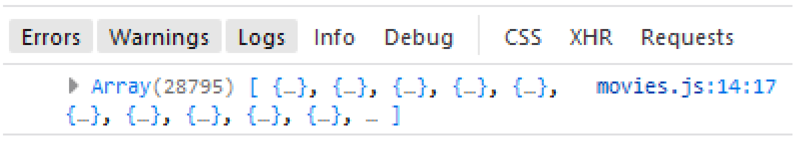
Assignment 4 Report

1. General Information
For coding and testing, I used Firefox Version 81.0 (64-bit).

2. Introduction
For this practical, I decided to be restrictive while assigning my variables. So, I prevented to use "var" and made my code work with implementing solely "let" for declaration purposes. Besides, I introduced a constant which is called "emptyCell" and comes to work when a movie has incomplete JSON-data. To ensure convenient testing, I modified "movies.html" and added a CSS file named "style.css". Due to this stylesheet, I guaranteed that every table cell has a border and added a margin between the table and the selects.

3. Read JSON-file and populate table (required task 1 and task 2)
For reading the data and fetching it to the "movies.html"-file, there is nothing special to mention. The code which does the fetching is standardized and does what it should. To ensure functional fetching, I logged the receive data to the console and could gaze at 28795 movies. Figure 1 shows this enormous number.

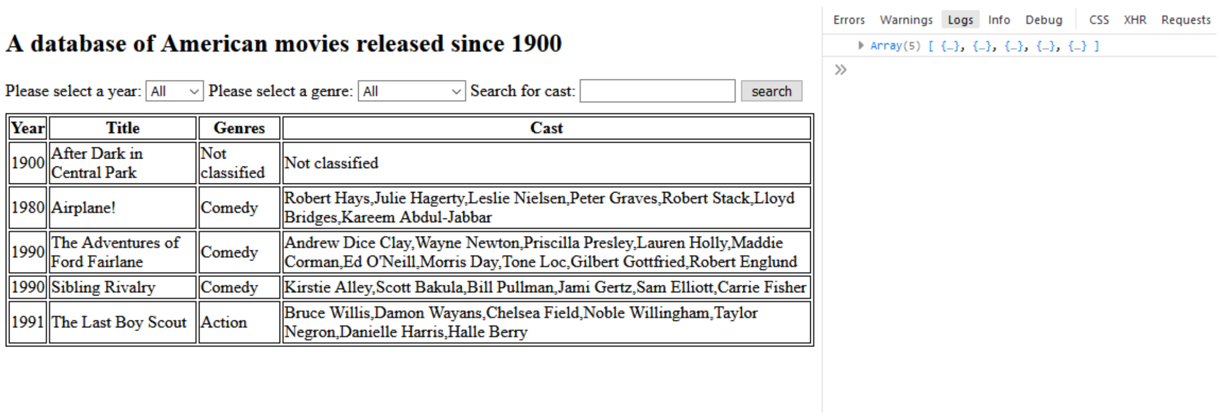


```
▶ Array(28795) [ {…}, {…}, {…}, {…}, {…},    movies.js:14:17
{…}, {…}, {…}, {…}, {…}, … ]
```

Figure 1: Logging "movies.json" to console after fetching

After I received the data given in figure 1, I created two functions which help me populating the table. The function *populateTableWith(data, tableID, tableBodyTag)* is responsible to insert rows with filled cells. I had to ensure that the rows are appended in the right order accordingly to the labelling of the table. Moreover, it was important to control if a movie had incomplete information for genre or cast. For this, I implemented the function *checkContentAndAppend(textValue, cell)* which monitors the empty genres/casts and then assigns the constant "Not classified" to the corresponding cell. My design decision to fill incomplete movies with a placeholder is based on the fact that I wanted to have an additional sorting option. Furthermore, it is more user-friendly to display a placeholder and tell that some genres or casts are missing instead of showing nothing. Otherwise, showing nothing could provide room for speculation.

To test the populate table functionality, I minimized the JSON-file "movies-small.json" to 5 movies and appended them to the table. Figure 2 shows that the populate table functionality is granted and that empty genres or cast are assigned the placeholder text "Not classified".



A database of American movies released since 1900

Year	Title	Genres	Cast
1900	After Dark in Central Park	Not classified	Not classified
1980	Airplane!	Comedy	Robert Hays,Julie Hagerty,Leslie Nielsen,Peter Graves,Robert Stack,Lloyd Bridges,Kareem Abdul-Jabbar
1990	The Adventures of Ford Fairlane	Comedy	Andrew Dice Clay,Wayne Newton,Priscilla Presley,Lauren Holly,Maddie Corman,Ed O'Neill,Morris Day,Tone Loc,Gilbert Gottfried,Robert Englund
1990	Sibling Rivalry	Comedy	Kirstie Alley,Scott Bakula,Bill Pullman,Jami Gertz,Sam Elliott,Carrie Fisher
1991	The Last Boy Scout	Action	Bruce Willis,Damon Wayans,Chelsea Field,Noble Willingham,Taylor Negron,Danielle Harris,Halle Berry

```
▶ Array(5) [ {…}, {…}, {…}, {…}, {…} ]
```

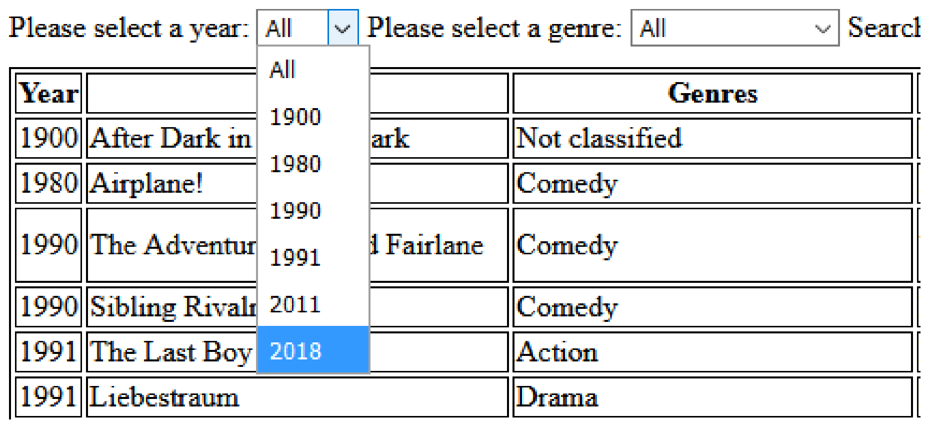
Figure 2: Populating table with modified "movies-small.json"

4. Populate selects for year and genre and show results (required task 3 and task 4)

To populate the selects for year and genre, I implemented three functions which dynamically load the different years or genres to the corresponding selector. My general approach to ensure that both selects contain solely unique option was conducted with a helper array. The functions *populateYearSelect(data)* and *populateGenresSelect(data)* contain a helper array which gets assigned "All" to grant that the user can sort the movies by all years or all genres. In *populateYearSelect(data),* I decided to loop through all data and collect all available years. To ensure that every year is pushed once to the helper array, I checked it with the "indexOf()"-function which returns the index position if the value is already in the array. If this function confirms that the year is not already within the helper array by returning "-1", I allowed pushing the corresponding year to the helper array.

Afterwards, the helper array was given with the corresponding select id to the function *populateSelect(valuesYear, 'selectYear').* This function is used as a helper function to fill both selects accordingly to the passed helper array and select. In detail, the *function populateGenresSelect(data)* works similarly to *populateYearSelect(data)* but is slightly more complicated. Since a movie can have multiple genres or none, I had to implement a nested for-loop to catch all genres of a movie. Furthermore, I had to check if a movie does not have a genre. Therefore, I decided to check this with an if-clause which only once assigns the placeholder for empty movie cells to the helper array. Again, to fill the genre select the corresponding parameter are given to the reusable function *populateSelect(valuesGenres, 'selectGenre').*

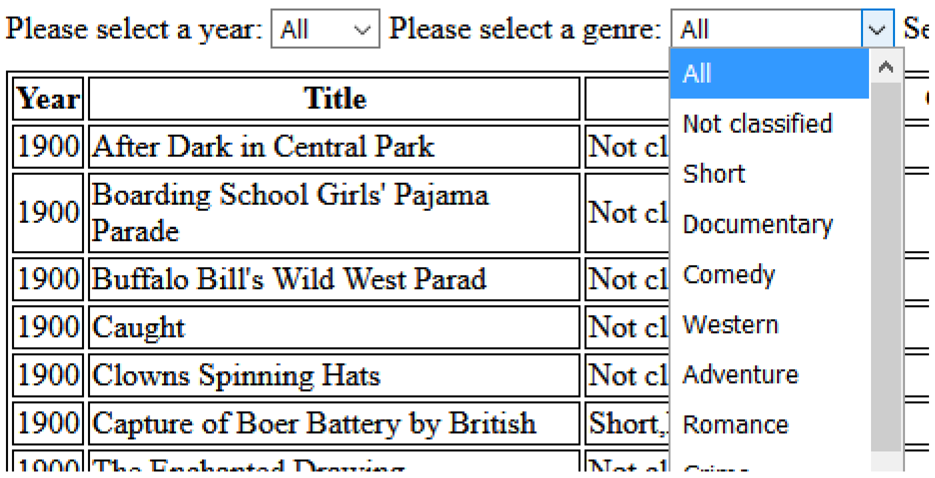
To test the functionality of these three functions, I used "movies-small.json" and "movies.json". First, to grant that both selects contain the correct amount of options, I let my programme display the different years in "movies-small.json". Besides, the option "All" is displayed in both selects. To ensure that all years were displayed correctly, I noted all different years on a piece of paper. Afterwards, I compared my notes with the provided years in "movies-small.json" and could confirm the completeness. The circumstance that all years are sorted ascendingly in the JSON files resulted in a displayed order which facilitated my uncommon testing procedure. Figure 3 shows the outcome for the select year.



Year		Genres
1900	After Dark in ark	Not classified
1980	Airplane!	Comedy
1990	The Adventur d Fairlane	Comedy
1990	Sibling Rivalr	Comedy
1991	The Last Boy	Action
1991	Liebestraum	Drama

Figure 3: Functional select year - "movie-small.json"

The file "movies.json" was used to test the functionality of the select genre. Moreover, it was very important to choose the "movies.json" for final select testing because it contains more years and genres than "movies-small.json". During development, I ran into the bug that the placeholder "Not classified" was as often displayed in the selector as movies do not have genre classifications. In the first stance, I wrongly assumed that my code works for both JSON files. When I tested my code with the bigger JSON file, I could see that "Not classified" was more than once displayed in the genre select. As a result, I recognized that the smaller JSON is not adequate for testing because one movie without a genre classification led to one genre select option. Whereas, "movies.json" had dozens of movies without a genre that caused dozens of "Not classified" options in the genre select. Nevertheless, I found this bug and fixed it. Figure 5 shows the genre select based on "movies.json".


Please select a year: All Please select a genre: All Se

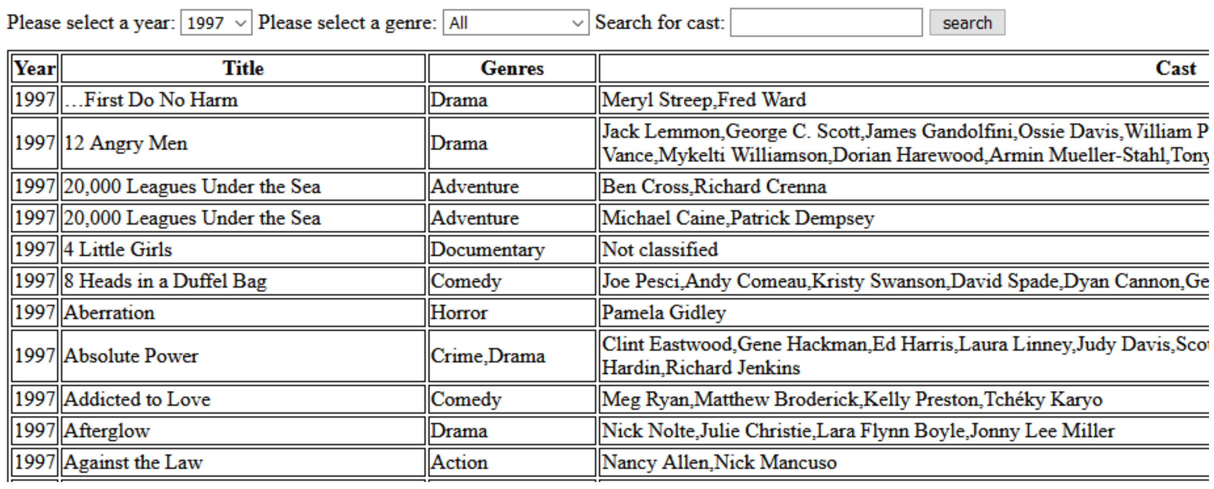
Year	Title	
1900	After Dark in Central Park	Not cl
1900	Boarding School Girls' Pajama Parade	Not cl
1900	Buffalo Bill's Wild West Parad	Not cl
1900	Caught	Not cl
1900	Clowns Spinning Hats	Not cl
1900	Capture of Boer Battery by British	Short,
1900	The Enchanted Drawing	Not cl

Figure 4: Functional select genre - "movies.json"

To make the selects working, I registered onchange event listeners and assigned the function *filterYearAndGenre()* to react on option changes. For filtering purposes, I decided to loop through the whole table and let the matched movies stay displayed which means not matched movies by genre or year are set to none display. To cover all possible changes by one of the selects or both, I implemented different if-clauses which work with the following user selections:

- year is “All” & genre is “All”
- year is “All” & one specified genre option, not “All”
- one specified year option, not “All” & genre is “All”
- one specified genre and year option, both not “All”

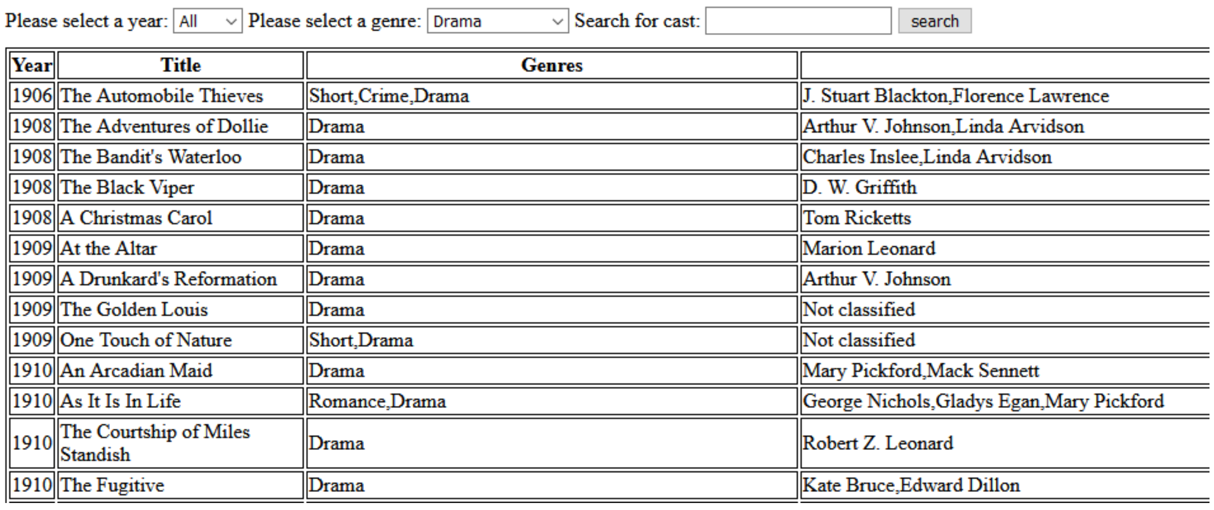
To test the functionality of both selectors, I tested them first individually and then linked together. The following test was conducted with "movies.json". When the user selects solely a year, then the outcome shows all movies from this specific year. Figure 5 is an example of this.


Please select a year: 1997 Please select a genre: All Search for cast: search

Year	Title	Genres	Cast
1997	...First Do No Harm	Drama	Meryl Streep,Fred Ward
1997	12 Angry Men	Drama	Jack Lemmon,George C. Scott,James Gandolfini,Ossie Davis,William P Vance,Mykelti Williamson,Dorian Harewood,Armin Mueller-Stahl,Tony
1997	20,000 Leagues Under the Sea	Adventure	Ben Cross,Richard Crenna
1997	20,000 Leagues Under the Sea	Adventure	Michael Caine,Patrick Dempsey
1997	4 Little Girls	Documentary	Not classified
1997	8 Heads in a Duffel Bag	Comedy	Joe Pesci,Andy Comeau,Kristy Swanson,David Spade,Dyan Cannon,Ge
1997	Aberration	Horror	Pamela Gidley
1997	Absolute Power	Crime,Drama	Clint Eastwood,Gene Hackman,Ed Harris,Laura Linney,Judy Davis,Sco Hardin,Richard Jenkins
1997	Addicted to Love	Comedy	Meg Ryan,Matthew Broderick,Kelly Preston,Tchéky Karyo
1997	Afterglow	Drama	Nick Nolte,Julie Christie,Lara Flynn Boyle,Jonny Lee Miller
1997	Against the Law	Action	Nancy Allen,Nick Mancuso

Figure 5: Year "1997" selected - genre "All" selected - "movies.json"

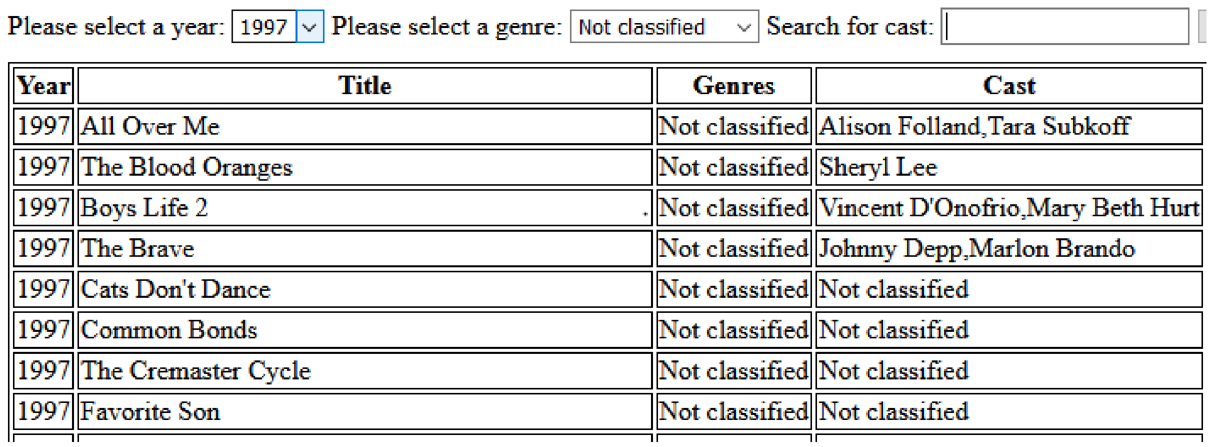
When the user selects solely a genre, then only movies which are assigned to this specific genre will be displayed. Figure 6 is an example of this.


Please select a year: All Please select a genre: Drama Search for cast: search

Year	Title	Genres	
1906	The Automobile Thieves	Short,Crime,Drama	J. Stuart Blackton,Florence Lawrence
1908	The Adventures of Dollie	Drama	Arthur V. Johnson,Linda Arvidson
1908	The Bandit's Waterloo	Drama	Charles Inslee,Linda Arvidson
1908	The Black Viper	Drama	D. W. Griffith
1908	A Christmas Carol	Drama	Tom Ricketts
1909	At the Altar	Drama	Marion Leonard
1909	A Drunkard's Reformation	Drama	Arthur V. Johnson
1909	The Golden Louis	Drama	Not classified
1909	One Touch of Nature	Short,Drama	Not classified
1910	An Arcadian Maid	Drama	Mary Pickford,Mack Sennett
1910	As It Is In Life	Romance,Drama	George Nichols,Gladys Egan,Mary Pickford
1910	The Courtship of Miles Standish	Drama	Robert Z. Leonard
1910	The Fugitive	Drama	Kate Bruce,Edward Dillon

Figure 6: Year "All" selected - genre "Drama" selected - "movies.json"

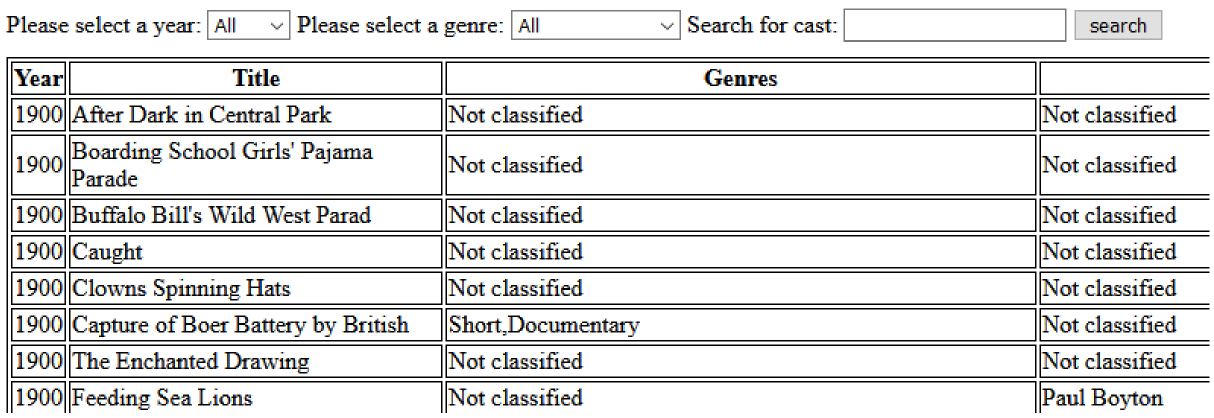
Moreover, Figure 7 shows that the search for the genre placeholder value "Not classified" is functional and how both filters are appropriately combined.


Please select a year: 1997 Please select a genre: Not classified Search for cast:

Year	Title	Genres	Cast
1997	All Over Me	Not classified	Alison Folland,Tara Subkoff
1997	The Blood Oranges	Not classified	Sheryl Lee
1997	Boys Life 2	Not classified	Vincent D'Onofrio,Mary Beth Hurt
1997	The Brave	Not classified	Johnny Depp,Marlon Brando
1997	Cats Don't Dance	Not classified	Not classified
1997	Common Bonds	Not classified	Not classified
1997	The Cremaster Cycle	Not classified	Not classified
1997	Favorite Son	Not classified	Not classified

Figure 7: Year "1997" selected, genre "Not classified" selected - "movies.json"

Finally, Figure 8 shows that the programme works appropriately with displaying everything by reselecting "All" for both filters.

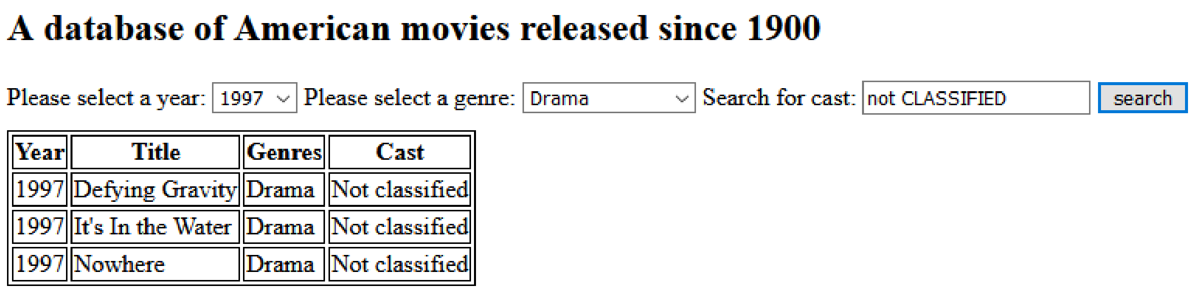

Please select a year: All Please select a genre: All Search for cast: search

Year	Title	Genres	
1900	After Dark in Central Park	Not classified	Not classified
1900	Boarding School Girls' Pajama Parade	Not classified	Not classified
1900	Buffalo Bill's Wild West Parad	Not classified	Not classified
1900	Caught	Not classified	Not classified
1900	Clowns Spinning Hats	Not classified	Not classified
1900	Capture of Boer Battery by British	Short,Documentary	Not classified
1900	The Enchanted Drawing	Not classified	Not classified
1900	Feeding Sea Lions	Not classified	Paul Boyton

Figure 8: reselecting "All" for both selects - "movies.json"

5. Extension task 1 search cast

For extension task 1, I assumed that the search should refine the current displayed movies selection. So, I designed the function that it can only hide movies which do not match the input field. In detail, this means that the function works independently from the selects and is not combined directly with them. When the user searches for individual cast data, the filter creates the corresponding outcome, but the result will be overwritten if one of the selects will be changed. Nevertheless, the search cast works appropriately from a usability view. For testing, when the user first selects year or/and genre and then refines the results with the cast function, it creates the outcome shown in Figure 9. Furthermore, the cast search is not case-sensitive which means that movie casts and the search input are set to lower case for comparison purposes.


A database of American movies released since 1900

Please select a year: 1997 Please select a genre: Drama Search for cast: not CLASSIFIED

Year	Title	Genres	Cast
1997	Defying Gravity	Drama	Not classified
1997	It's In the Water	Drama	Not classified
1997	Nowhere	Drama	Not classified

Figure 9: Functionality of search cast – "movies.json"

6. Reflexion

From my point of view, I learned a lot in this practical. I tried an analogous testing procedure by noting years on paper. For this, I have to say that this only makes sense with small test sets and human errors like wrong note-taking could occur. In general, it is more convenient to test with the machine.

Furthermore, I enjoyed the progress of conceiving and implementing the extra feature with empty genres/casts. Moreover, I think this additional piece of functionality enhanced the code complexity. Compared to the third practical, I was more restrictive with variables and added more comments to my code to improve the readability. If I had to start again, I would use the for/in loops instead of the "classical" to enhance the readability further.

References

- *Mozilla Foundation, research, https://developer.mozilla.org/en-US/docs/Web/API/HTMLTableCellElement, last accessed 2020-11-20*
- *Mozilla Foundation, research, https://developer.mozilla.org/en-US/docs/Web/API/HTMLTableRowElement/insertCell, last accessed 2020-11-20*
- *Mozilla Foundation, research, https://developer.mozilla.org/en-US/docs/Web/API/Document/createTextNode, last accessed 2020-11-20*
- *Mozilla Foundation, research, https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild, last accessed 2020-11-22*

- *Mozilla Foundation, research, https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML, last accessed 2020-11-22*