

Assignment 8 – WhatABook, Part 2

Setup

Each week you will be asked to create a new folder under web-330 following a naming convention of **<week>-number**. If we are on week two, the folder name should be **week-2**. All files associated with the weekly assignment will be added to the appropriate folder. All programs must be linked in the appropriate landing page. Projects will be linked under the “Projects” section of the index.html landing page.

The document title of all HTML files in this course must say “WEB 330 – Enterprise JavaScript II.” And, all HTML and CSS files must be valid HTML/CSS, tested through the WC3 validator. The links were provided during WEB 200 and 231. As part of your submission, be sure to include screenshots of the results from the validation tests (HTML and CSS validators).

User interface styling and formatting requirements are located in the Web 330 HTML, CSS, and JavaScript Requirements document.

HTML: **<yourLastName>-whatabook2.html**

CSS: **<yourLastName>-whatabook2.css**

JS: **http-client.js**

Grading Reminders

- A. (rubric) All code sources (.html, .css, .js) are measured against
 1. Code functionality: Does it work? Does it meet requirements?
 2. Adherence to standards and conventions. Are you using the appropriate data types, including proper indention, are variables named appropriate (variable x is an example of poor naming conventions), is there an appropriate use of whitespace, is the code organized, and are semicolons being used to terminate code sentences?

3. Efficiency: Use of language features. Are you practicing DRY (Don't-Repeat-Yourself?), are you leveraging built-in language features where appropriate, and are you using classes/functions to reduce code clutter?
4. Documentation: Code is maintainable by others
 - i. Code comments are present in all blocks of code, written as full sentences, free of grammatical errors, and function/class parameters and data types have been identified.
 - ii. Code attribution is present in all files and authorship is clearly annotated.
5. Error trapping/handling. Are there errors in the program? Is there evidence of coding best practices to reduce user errors?
6. Assignment Specific Compliance. Does the delivered solution follow the instructions, as they are written? Does the output match what was provided in the screenshots (including spaces, styling, etc.)?

Required Modifications

- Cite any sources in your opening programmer's comment
- Link the appropriate CSS, JavaScript, and Google fonts
- `http-client.js`

Additional JavaScript Requirements

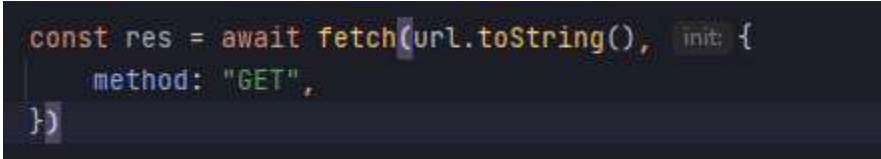
- a. Create a class named `HttpClient`
- b. In the body of the class create an async function named `get` with two parameters: `url` and `params`. Set the `params` parameter to a default empty string.

Additional JavaScript Requirements

1. Instantiate a new `URL` object, supplying it the `url` parameter and assign the results to the `url` parameter: **`url = new URL(url)`**
2. Instantiate a new `URLSearchParams` object, supplying it the `params` parameter and assigning it to the `url.search` property.
3. Create an object literal named `res` and using the `fetch()` API passing in the `url.toString()` variable and specifying the request as a GET request.

4. Return the res object literal as JSON.

Exhibit A. fetch API call



```
const res = await fetch(url.toString(), { init: {  
  method: "GET",  
}})
```

5. Export the class.

Exhibit B. User Interface (final solution)

Welcome to the WhatABook, Part 2 App!

Book Listing

ISBN	Title	Description	Pages	Authors
9780345339683	The Hobbit or There and Back Again	N/A	287	J.R.R. Tolkien
9780261103573	The Fellowship of the Ring	Being the First Part of the Lord of the Rings	416	J.R.R. Tolkien
9780593099322	Dune: Deluxe Edition	N/A	688	Frank Herbert
9780261102361	The Two Towers	Being the Second Part of The Lord of the Rings	442	J.R.R. Tolkien
9780261102378	The Return of The King	Being the third part of The Lord of the Rings	556	J.R.R. Tolkien
9780590302715	Charlotte's Web	N/A	N/A	
9780316769532	The Catcher in the Rye	N/A	277	J. D. Salinger
9780743273565	The Great Gatsby	N/A	180	F. Scott Fitzgerald
9780590405959	The Lion, the Witch and the Wardrobe	N/A	186	C. S. Lewis

[Return](#)

Selected Book

ISBN: 9780593099322
Title: Dune: Deluxe Edition
Description: N/A
Pages: 688
Authors: Frank Herbert

1. The final solution in Exhibit B is identical to the final solution in Assign_7. This includes the functionality and HTML code. The main difference is how we implement the process. In assignment seven we read the data from an XML file. In this week's assignment we will be making an API call to <https://openlibrary.org/api/books>. To this end, I will not be discussing the styling or HTML structure because you should have a working solution from assignment seven.

<yourLastName>-whatabook2.html

- a. Add an import statement for the HttpClient class.
- b. Instantiate a new HttpClient object and assign it to a new variable named http.
- c. Create a variable named isbnns and assign it an array of ISBN numbers (see below)

Exhibit C. ISBNs

```
const isbnns = [  
  '0345339681',  
  '0261103571',  
  '9780593099322',  
  '9780261102361',  
  '9780261102378',  
  '9780590302715',  
  '9780316769532',  
  '9780743273565',  
  '9780590405959'  
]
```

- d. Create an object literal named params with the following key/value pairs

Exhibit D.

```
const params = {  
  "bibkeys": `ISBN:${isbnns.join(",")}`,  
  "format": "json",  
  "jscmd": "details"  
};
```

- e. Call the `http.get` function and supply it with the open library URL and the `params` objects.

Additional JavaScript Requirements

- 1) Add a `then` clause using arrow functions with a `res` object
- 2) In the body of the call, call the `buildHtmlString()` function supplying it with the `res` object and the string value `"table"` and bind the results to the `bookListing` div.

Exhibit E.

```
http.get("https://openlibrary.org/api/books", params).then(res => {  
  console.log(res);  
  
  document.getElementById("bookListing").innerHTML = buildHtmlString(res, "table");  
  
  addIsbnClickEvents();  
})  
.catch(e => {  
  console.log(e);  
})
```

- 3) Call the `addIsbnClickEvents()` function
 - 4) Add a `catch()` statement with an arrow function for error and write the error to the console using the `console.log()` API.
- f. Create a function named `getBook()` that accepts an event object

Additional JavaScript Requirements

- 1) Create a variable named `self` and assign it the value `this`

Exhibit F.

```
function getBook(e)
{
    e.preventDefault();

    let self = this;

    let isbn = self.innerText;
```

- 2) Create a variable named isbn and assign it the innerText of the self variable.
 - 3) Build a new params object literal (see Exhibit D.)
 - 4) Call the http.get function supplying it the open library URL and params object literal.
 - 5) Add a then clause and in the body call the buildHtmlString function supplying it the res object and “ul” string value and bind the results to the selectedBook div.
 - 6) Add a catch() clause for the error object and write the error to the console using the console.log() API.
- g. Create a function named buildHtmlString with two parameters: res and format.

Additional JavaScript Requirements

- 1) Create a variable named tableString and supply it with an HTML string for a table header.
- 2) Create a variable named ulString and set it to an empty string.
- 3) Using a for...in statement, iterate over the res parameter and append the objects to the ulString and tableString variables. See Exhibit G. Please note, the code snippet in Exhibit G is a partial view of the solution. You will need to write the remaining pieces of code.

Exhibit G.

```

for (let key in res)
{
  ulString += `<ul style="list-style-type: none;">`;
  if (res.hasOwnProperty(key))
  {
    let authors = [];
    if (res[key].details.authors)
    {
      authors = res[key].details.authors.map(function(author) {
        return author.name;
      });
    }

    let book = {
      isbn: res[key].details.isbn_13 || res[key].details.isbn_10 || res[key].details.isbn_10,
      title: res[key].details.title,
      description: res[key].details.subtitle || res[key].details.subtitle || "N/A",
      pages: res[key].details.number_of_pages || res[key].details.number_of_pages || "N/A",
      authors: authors.join(", ")
    }

    ulString += `<li><b>ISBN:</b> ${book.isbn}</li><li><b>Title:</b> ${book.title}</li><li><b>Description:</b> ${book.description}</li><li><b>Pages:</b> ${book.pages}</li><li><b>Authors:</b> ${book.authors}</li>`;

    tableString += `<tr><td><a href="#" class="isbn-link">${book.isbn}</a></td><td>${book.title}</td><td>${book.description}</td><td>${book.pages}</td><td>${book.authors}</td></tr>`;
  }
}

```

- 4) Outside of the for...in loop, close the HTML tableString
- 5) Add an if statement that checks the format variable. If the format variable is a "table" return the tableString variable. Otherwise, return the ulString variable.