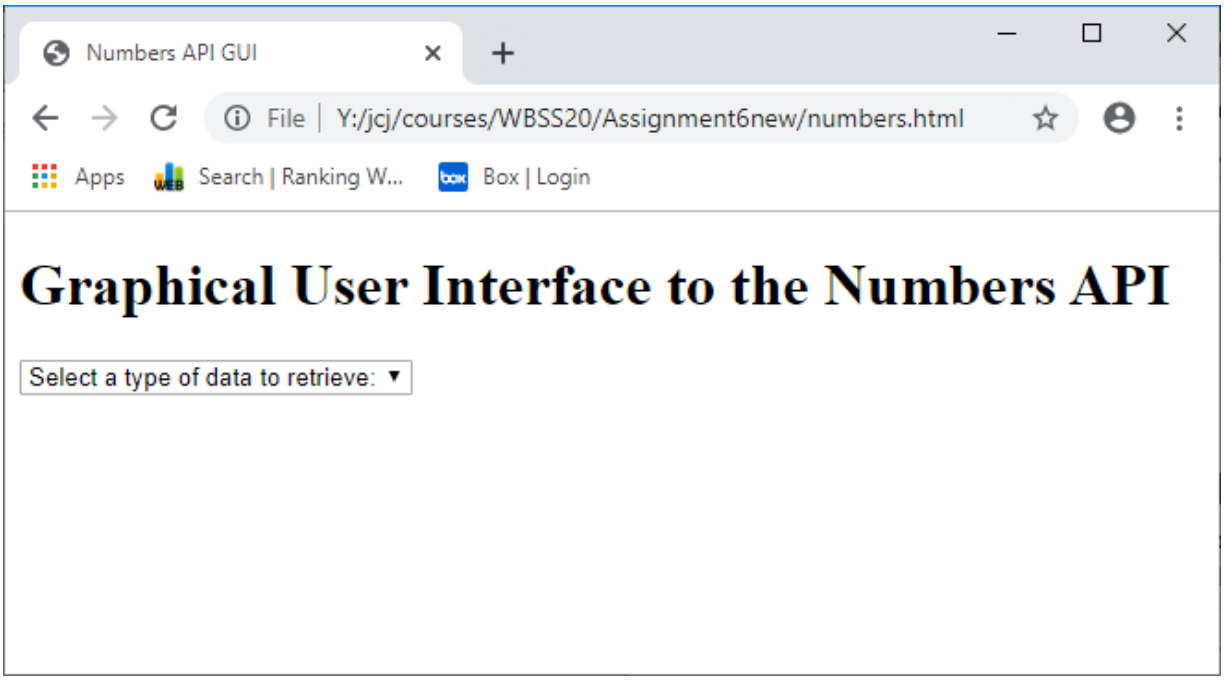
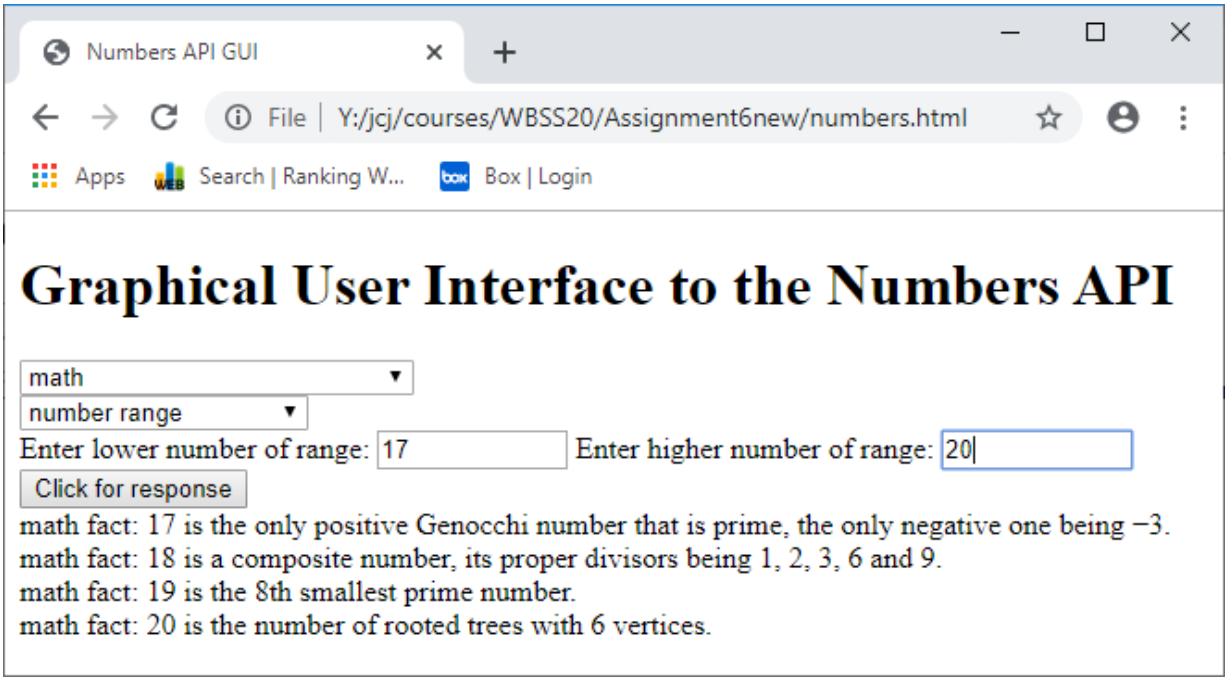
Client-side Programming Assignment

Overview

David Hu and Mack Duan have created a publicly available *Numbers API*. Visit http://numbersapi.com for details. Your assignment is to write a Graphical User Interface (GUI), using HTML and JavaScript, to make it easy for users to access this API. The screenshots below show an example of initially visiting the app followed by the results after entering some information (the results can vary randomly).





The Numbers API supports four types of data about numbers:

- trivia, e.g., "105 is the atomic number of hahnium, also known as dubnium."
- math: see the second screen shot above.
- date, e.g., "February 25th is the day in 1951 that the first Pan American Games are held in Buenos Aires, Argentina."
- year, e.g., "2020 is the year that the Large Hadron Collider will be substantially upgraded."

As shown, your application will begin with a selector that will allow the user to select one of these four types. For the trivia, math, and year types, the application should next display a second selector that allows the user to choose an input format:

- random (no input)
- number (user inputs single positive integer)
- number range (user inputs two positive integers, the second greater than the first, as shown in the example).

If the random format is chosen, the application should immediately request a random fact of the specified type from the API. For instance, to obtain a random math fact, the application should fetch from the following URL:

http://numbersapi.com/random/math?json

To obtain a random trivia or year fact, substitute "trivia" or "year" for "math" in the URL.

If the number input format is chosen, the application should display a text box and a submit button. A pattern attribute should be used on the text box to ensure that the user enters only digits in the box. Once valid data is entered and the button is clicked, the application should fetch from a URL such as the following, which assumes that the user has selected the math data type and has entered 105 in the text box:

http://numbersapi.com/105/math?json

The range input format is illustrated in the earlier screenshot. The URL fetched for this example was:

http://numbersapi.com/17..20/math?json

Finally, if the user selects the date data type, they should see two text boxes, one for month and one for day of the month, along with a submit button. Your application should use pattern attributes to ensure that each of the user entries in these boxes consists of either 1 or 2 digits. If so, when the submit button is clicked, the user information should be sent to the API using a URL such as (this assumes month 2 and day 25):

http://numbersapi.com/2/25/date?json

In response to fetching these URLs, the Numbers API will return a JSON string. The exact string returned by the API depends on the data type and input format. For the date data type and for the random and

number formats of the other data types, the string returned represents a single object having the following properties (there might also be others, but we will only use the following):

- `found`: true if the API has data to return for the specified input, false otherwise.
- `type`: the type of fact returned (math, for this URL)
- `text`: the text of the fact
- `number`: normally the number that the fact concerns (although it means something different for the date data type)

If the data type is not date and the format is number range, the API will return an object with one numeric property for each number in the range; the value of each property will be an object as described above. For instance, the range URL above might return the following:

```
{
 "17": {
  "text": "17 is a repunit prime in hexadecimal (11).",
  "number": 17,
  "found": true,
  "type": "math"
 },
 "18": {
  "text": "18 is a heptagonal number, and as the sum of the first three
pentagonal numbers, it is a pentagonal pyramidal number.",
  "number": 18,
  "found": true,
  "type": "math"
 },
 "19": {
  "text": "19 is a centered triangular number, centered hexagonal number and
a Heegner number.",
  "number": 19,
  "found": true,
  "type": "math"
 },
 "20": {
  "text": "20 is the basis for vigesimal number systems.",
  "number": 20,
  "found": true,
  "type": "math"
 }
}
```

For an individual return object, your application should display the data type of the fact and the text of the fact, as shown in the second screenshot above. Or, if `found` is false, an appropriate message should be displayed, such as the number for which the fact was requested followed by a string such as "no fact found". For an object-of-objects as returned by a range, each individual object (or not-found message) should be displayed on a separate line, as shown in the earlier example.

Grade Levels

I am supplying an HTML file that you should use without modification. You should write JavaScript code in a file numbers.js. Your program will be graded using levels.

Level 1: Display a random fact (12 point maximum)

For this level, your application needs to implement only the following features:

1. Add a "change" event listener to the provided HTML select element having id attribute value "dataType".
    a. Use the DOM `querySelector()` method to obtain a reference to the select element.
2. Write the event listener itself, which will be called when the user selects an option in the select element.
    a. This function should do nothing if the "date" option is selected (this option is not implemented at this grading level).
    b. If any of the other three options is selected, this function should create a new select element as a child of the div having id "follow_up". This new select element should display options "random", "number", and "number range".
        i. You can use DOM methods such as createElement() and appendChild() to create this element, or you are allowed to set the innerHTML property of DOM elements. Note that quoting a JavaScript string with the backtick (`) character allows strings to extend over multiple lines, which is handy for use with innerHTML.
    c. The function should also add an event listener for "change" events to the new select element.
        i. If "random" is selected by the user, the listener should initiate a fetch as explained below for the appropriate type of data (trivial, math, or year) and specifying random data.
        ii. The listener can ignore other selections ("number" or "number range").
3. Write the function that will be called in order to fetch JSON data from a specified URL of the Numbers API.
    a. This function should call fetch(), parse the returned data into a JavaScript object, then display the data in the JavaScript object in the div having id "display" as described earlier.
        i. Use the textContent DOM property to add this data to the div.
    b. The fetch() call should include an appropriate error handler (which can simply log any exception received to the console).
    c. Also, immediately after fetch() is called, the "display" div should display a message such as "fact requested…". This message will be replaced after the JSON data has been received and parsed (in my testing, there is often a noticeable delay between when the API is called and when the data is returned).

Level 2: Add the date data type (14 points maximum)

1. Modify the event listener on the dataType select element so that it also responds to the "date" option.
    a. If "date" is selected, a form containing two input boxes and a submit button should be displayed in the "follow_up" div.
    b. Use pattern attributes to ensure that each box contains either one or two digits. Note: Backslash (\) is a special character in JavaScript, just as it is in Java. So, if your JavaScript

code contains a string representing a regular expression that contains a backslash, you will need to escape this character by preceding it with a backslash (use two consecutive backslashes to represent a single backslash character), just as in Java.

   c. Add a "submit" event listener to the form.

2. Write the "submit" event listener.
   a. Using the technique discussed in class, the listener should prevent the browser from actually attempting to submit the form, the browser's default behavior when a submit button is pressed. (We are only going through the initial portion of submit processing so that the browser will test the user input against the specified patterns.)
   b. Instead of submitting, the listener should:
      i. Extract data from the text boxes on the form
      ii. Use the data to create an appropriate Number API URL, and
      iii. Fetch that URL and display the response using the fetching function written for Level 1 (possibly slightly modified).
   c. It is now possible that a date will have no API data. Don't forget to display an appropriate message if data is not found.

Level 3: Write the complete program (15 points maximum)

1. Implement all data types and all input formats (number and number range as well as random).
   a. Number and number range boxes should have pattern attributes to ensure that they consist only of digits.
   b. Default submit processing should again be prevented.
2. The application should behave well if the user
   a. Enters new number or numbers in existing text boxes and clicks submit.
   b. Chooses a new input format from the drop-down (if this drop-down is showing).
   c. Chooses a new data type.

   In all of these cases, portions of the page that are no longer relevant (if any) should "disappear" from the page.