# CSE183 Fall 2020 - Assignment 4

In this assignment you will write JavaScript classes to modify the content of HTML pages via the DOM. You will use knowledge of CSS, HTML, and JavaScript acquired in the previous two assignments and should gain new skills in the process.

**This assignment is worth 7% of your final grade.**

**Submissions are due NO LATER than 23:59, Thursday October 29 2020** (1 week)

## Installation

See instructions in Assignment 3.

## Setup

Download the starter code archive from Canvas and expand into an empty folder. I recommend, if you have not already done so, creating a folder for the class and individual folders beneath that for each assignment.

The starter code archive contains the following files that you will modify:

```
templater.html
Templater.js
picker.html
picker.css
Picker.js
picker.test.js
spinner.html
spinner.css
Spinner.js
spinner.test.js
```

And the following files that you should ***not*** modify:

```
.eslintrc.js
package.json
templater.css
templater.test.js
```

To setup the node environment, ***navigate to the folder where you extracted the starter code*** and run the following command:

```
$ npm install
```

This will take some time to execute as `npm` downloads and installs all the packages required to build and test the assignment.

To execute the tests, run the following command:

```
$ npm run test
```

To check the code quality against Google standards by running `eslint`, run the following command:

```
$ npm run lint
```

## Requirements

Basic:

- Implement a JavaScript class with the ability to set the text content of HTML table elements with values supplied as JSON in the following circumstances:

    - Where existing element text contents are `{{ }}` tagged
    - Where elements have an id attribute

    For example, in the first case, if an element is written as:

    ```
    <td>{{R11}}</td>
    ```

    And the JSON supplied is:

    ```
    {"R11":"Hello World"}
    ```

    The document should be modified such that it would have been written as:

    ```
    <td>Hello World</td>
    ```

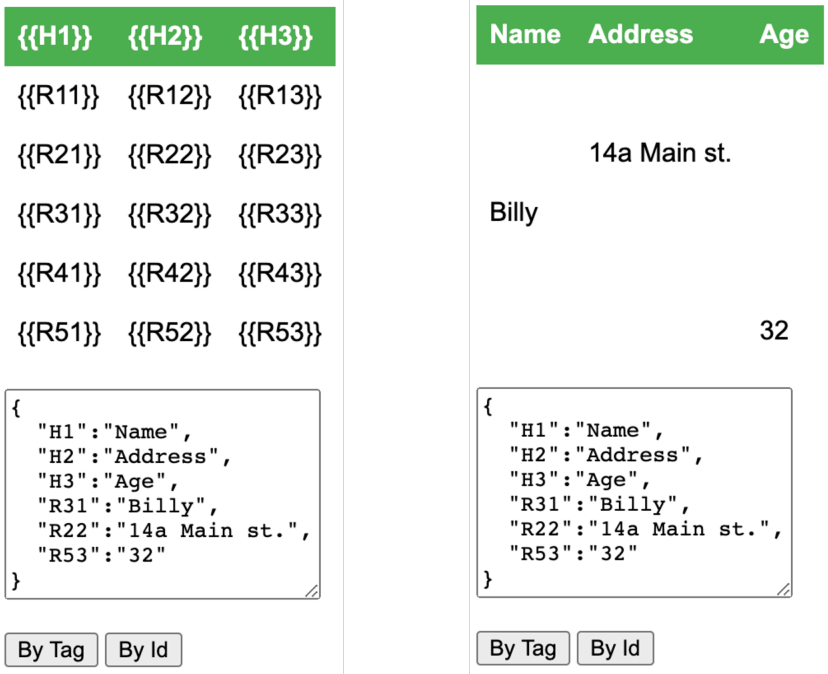
    Similarly, in the second case, if an element is written as:

    ```
    <td id="R11">Some Text</td>
    ```

    And the same JSON is supplied, the document should be modified in the same way.

- To achieve this you will need to modify the `byTag()` and `byId()` methods on the `Templater` class in `Templater.js` and modify `template.html` to make use of your implementation ***taking great care not to modify the id's of existing HTML elements***.

    Regardless of which button is clicked in `templater.html`, when your implementation is complete the result should be as shown on the right assuming the illustrated JSON is entered in the text field:



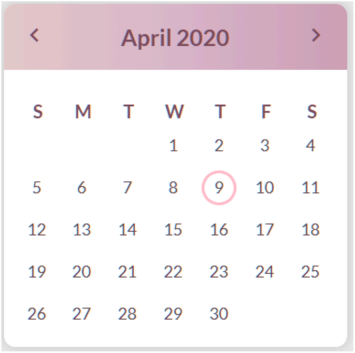

- You can test by hand using a browser, but make sure you also pass the automated tests by running:

    ```
    $ npm run test
    ```

Advanced:

- Implement a date Picker that might look something like this:




  Your Picker can look anyway you like, but the following features must be implemented:

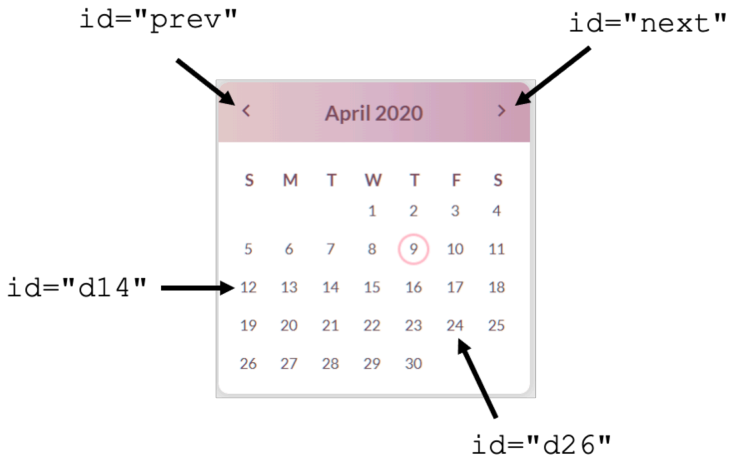
  - The current date should be shown initially
  - The month and year must be shown in text
  - The day of the month must be highlighted in some way
  - There must be “next” and “previous” navigation buttons to change the month
  - Days where the end of the previous and start of the next month would be must be blank

- Place an instance of your Picker inside a container `<div>` in `picker.html`. You will pass the id of this container to your Picker constructor.

- Regardless of how you implement your Picker, the generated document must have HTML elements with the following ids and behaviors:

  - `next` advances the date to the next month
  - `prev` retards the date to the previous month
  - 42 x `dn` where `n` is in the range 0 to 41 each representing one day

  For example:




- You can test by hand using a browser, but make sure you also pass the supplied tests by running:

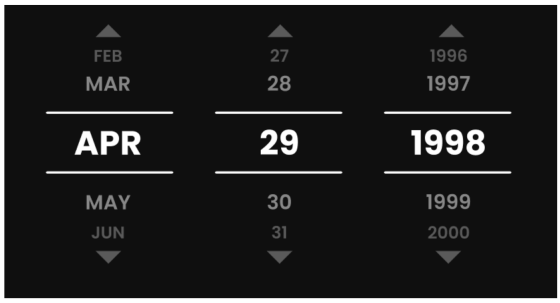
```
$ npm run test
```

- You may also want to write additional tests, in which case you should study the existing ones and research the Google Puppeteer framework which gives you the ability to manipulate a Chrmomium browser instance from your tests: https://developers.google.com/web/tools/puppeteer

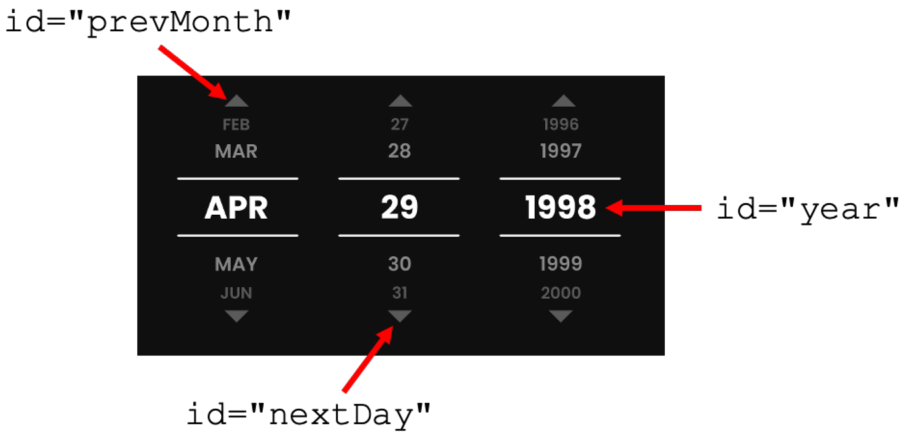
Stretch:

- Implement a date Spinner that might look something like this:




- Your Spinner can look anyway you like, but the following features must be implemented:
    - The current month, day, and year must be shown
    - There must be “next” and “previous” navigation buttons to change the month, day, and year
- Place an instance of your Spinner inside a container `<div>` in `spinner.html`. You will pass the id of this container to your Spinner constructor.
- Regardless of how you implement your Spinner, the generated document must have HTML elements with the following ids and behaviors:
    - `nextMonth` advances the date to the next month
    - `prevMonth` retards the date to the previous month
    - `nextDay` advances the date to the next day
    - `prevDay` retards the date to the previous day
    - `nextYear` advances the date to the next year
    - `prevYear` retards the date to the previous year
    - `month` shows the current month in three character upper case form ( January = JAN etc. )
    - `day` shows the current day of the month
    - `year` shows the current year in four-digit form, e.g. 1988

    For example:




- You can test by hand using a browser, but make sure you also pass the automated tests by running:

```
$ npm run test
```

- You may also want to write additional tests, see the Advanced requirement for details.

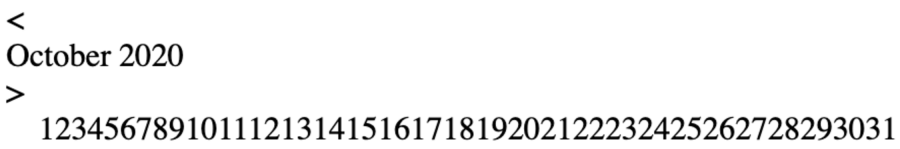
 

## What steps should I take to tackle this?

First, take a deep breath. These requirements are not as hard as they might initially appear.

The first part of Basic is similar to Assignment 3 with the addition on JSON parsing and the ability to iterate over DOM nodes of certain types - both are topics we covered in class. The second part of Basic merely requires that you get DOM nodes by their id.

For Advanced and Stretch, work on the functionality first and worry about the appearance second. Make sure you rigorously ***separate content from style*** so you can change the look and layout simply by modifying your style sheet. A simple implementation for Advanced that passes all the tests might look something like this:




And from there you can style the heck out of it to make it look nice.

## How much code will I need to write?

A model solution that satisfies all requirements adds approximately 30 lines of code to `Templater.js`, and around 150 lines each to `Picker.js` and `Spinner.js`. You will also need to modify `templater.html`, `picker.html`, `picker.css`, `spinner.html` and `spinner.css` plus related tests.

## Grading scheme

The following aspects will be assessed:

1. (72%) **Does it work?**

    a. Basic Requirement ( 3 points )
    b. Advanced Requirement ( 1 points )
    c. Stretch Requirement ( 1 point )

2. (28%) **Does it look good?**

    a. Advanced Requirement ( 1 point )
    b. Stretch Requirement ( 1 point )

3. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%). You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy.

    b. Your submission is determined to be a copy of a past or present student's submission (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 33% copied code No deduction
        - 33% to 66% copied code (-50%)
        - > 66% copied code (-100%)

## What to submit

Make a Zip Archive of the modified files:

```
templater.html
Templater.js
picker.html
picker.css
Picker.js
picker.test.js
spinner.html
spinner.css
Spinner.js
spinner.test.js
```

****** UPLOAD THIS ZIP ARCHIVE TO THE CANVAS ASSIGNMENT AND SUBMIT ******