Local Storage:

1: INTRODUCTION:

Quite often, the web developer might want to store information from a webpage form. There are a few ways of doing this:

- Writing the form element information into a **Database**.
  - This involves server-side programming, typically with **PHP** and **SQL** (next semester);

- Writing the form element information into a **Cookie**.
  - A cookie is a text file hosted on the user's computer. Information can be stored there to be read later. (will use cookies next year with PHP for login)
  - Cookies allow websites to snoop on your web-surfing behaviour. They also allow the webpage to be customised so that when you next access the webpage, your username might be displayed in a welcome message.
  - Cookies are limited in size to 4kB.
  - To see cookies in Chrome, have a webpage opened where you have logged in (e.g. Webmail or Moodle). Then F12 for **Developer Tools**, **Application** tab, **Storage** section → **Cookies**.
- Writing the form element information into **Local Storage**. Local Storage has the following advantages:
  - Data stored in local storage isn't sent back to a server. Data stays on the client's machine.
  - You can store up to 10MB, depending on the browser, protocol and domain.
  - Local storage means that the next time the webpage is run, it can access local storage and show the previous state of the website without having gone back to the server. HTTP is a **stateless** protocol, which means that the next time you open the webpage, its state is reset. Local storage allows us to keep information that can *customise* a webpage based on *previous times* that webpage was open.
  - Local Storage is much much easier to program than Cookies! … and it is a nice introduction to Javascript.

Local Storage

In this exercise you are going to learn:

1) **Introductory Javascript**:
    a. calling and writing a function;
    b. accessing the ID of a tag;
    c. running code based on user events (**load**ing the webpage, pressing the **submit** button)
2) How to **write** the values of form elements in a webpage **into Local Storage**;
3) How to **read Local Storage data** into tags in a webpage.

2: RESOURCES:

- Webpage 1: the **`Form_LocalStorage_BASIC.html`** webpage. Save this as **`Form_LocalStorage.html`**.
- Webpage 2: the **`thanks.html`** webpage. When all is good with the previous webpage after clicking the **`submit`** button, the browser goes to this webpage (as specified by the action property of the **`<form>`** tag.).
- There is a stylesheet, in a subfolder.
- There is a sub-folder called **`scripts`**. Inside this, there is an empty file called **`Local_storage.js`**. This is where you will write the JavaScript code.
- Can you see where the two webpages are linking to this JavaScript file?

3: WHAT IS JAVASCRIPT?

JavaScript is *the* programming language of the web. Each time you post a Twitter or Facebook feed and scroll down to see new posts, zoom in on Google Earth or Maps, or calculate a car insurance quotation, you are using JavaScript. JavaScript was the most popular programming language on GitHub between 2014 and 2021 (see graph at https://octoverse.github.com/2022/top-programming-languages)....
And TypeScript is another version of JavaScript.

Much JavaScript syntax is like Java or C#. What makes JavaScript different from those languages is that **we want to use it to do stuff to and with HTML tags and CSS styles**, depending on **EVENTS** that happen, either when the webpage is loaded, or when the user does something (e.g click on a radio button, pick an option from a selection list (change), click on a submit button, put the mouse over some text/image, go into or leave a textbox (focus / blur), type a character (keyup), etc)

4: WHAT DO WE WANT JAVASCRIPT TO DO IN THIS EXAMPLE?

There is already a small piece of JavaScript in the first webpage, in the `footer`. It gets the year from today's date and displays it on the webpage after the © HTML5 special character. This code is future-proofed as it will show 2023 next year. The reason for this is that the **Date OBJECT** has an empty set of (), so the *current* date/time is stored in the date object.

Regarding **Local Storage**, we want to:

- **Webpage 1**:
  - Use **Javascript for form validation**: to check that at least 3 (i.e. 3 or more) characters have been typed into the name textbox. If that is not true, we want to display an error message, select the faulty text and stay on the first webpage.
    - (`required` *means that HTML validation will operate so we don't need any Javascript code to check an age or country have been typed/selected*)
  - If the form data is valid, then we will write the form element values into Local Storage.

- **Webpage 2**:
  - We want to read the Local Storage values (that we wrote in webpage 1) and display them into particular tags with ids.

Essentially we have two big chunks of JavaScript to write: for webpage 1 and webpage 2.

We want to run these only when an **EVENT** happens:

- On webpage 1, when the `submit` button is clicked;
- On webpage 2, when the webpage is `load`ed.

So, two chunks and two events.

5: CHUNKS and EVENTS: FUNCTIONS and EVENT MANAGERS:

We give the chunks of JavaScript code names. These chunks are known as **functions**.

- Webpage 1 chunk: Add the following code to **`Local_storage.js`**:
  - ```
    function checkForm() {

    }
    ```
- Webpage 2 chunk: Add the following code to **`Local_storage.js`**, *after the previous } closes*:
  - ```
    function readLocalStorage() {

    }
    ```

When are we going to use these functions? When certain events happen.

- Webpage 1 **`onsubmit`** Event: Add the following code to **`Form_LocalStorage.html`**. Just the code in the red box:
  - ```
    <h2>Saving the Form Information into the Browser Memory: LOCAL STORAGE</h2>

    <form name="theForm" id="theForm" onsubmit="return checkForm()" method="post" action="thanks.html">
        <label>Your Name:</label>
        <input type="text" id="theName" name="theName" required placeholder="at least 3 characters"><br>

        <label>Your Age (years):</label>
    ```
- Webpage 2 **`onload`** Event: Add the following code to **`thanks.html`**. Just the code in the red box:
  - ```
    </head>

    <body onload="readLocalStorage()">
        <main>
            <header>
                <h1>Form with Local Storage: Page 2</h1>
                <h1><u>READING from</u> LOCAL STORAGE</h1>
            </header>
    ```

6: WEBPAGE 1: (a) COMMENTS:

We want the JavaScript to:

- //check the length of what was typed in for the Name.
- //if what was typed in was less than three characters in length:
    - //tell the user
    - //highlight the text that is in error
    - // force the browser to stay on the same webpage
- //if what was typed in has a valid number of characters
    - //write the Name to Local Storage
    - //write the Age to Local Storage
    - //write the Country to Local Storage
    - //allow the browser to go to the next webpage (whatever is in the `<form action="">` property)

Essentially, with **form validation onsubmit JavaScript functions**, the approach you should take is:

- `if` there's a problem (1): tell the user / highlight where the problem is / return **false**
- `else if` there's a problem (2): tell the user / highlight where the problem is / return **false**
- …etc (as many `else if`s {i.e. as many problems} as are necessary)
- `else` … do whatever needs to be done for the valid form (e.g. write Local Storage) / return **true**

If you are writing code, the last thing you should do is to write code!

- Understand the problem…read through the information a few times;
- Write down the instructions/pseudocode in ordinary English;
- Check you have the instructions in the right order;
- LOOPS and IF…do you need to do a line of code…
    - **ALWAYS ONCE** (neither if nor loop)
    - **SOMETIMES ONCE** (if)
    - **MORE THAN ONCE** (loop);
- Now write the code.

Copy the above comments and paste them into the **`checkForm()`** function, making sure to indent the comment lines appropriately. We will type the code shortly under the appropriate comment.

```
function checkForm() {
    //check the length of what was typed in for the Name.
    //if what was typed in was less than three characters in length:
        //tell the user
        //highlight the text that is in error
        // force the browser to stay on the same webpage
    //if what was typed in has a valid number of characters
        //write the Name to Local Storage
        //write the Age to Local Storage
        //write the Country to Local Storage
        //allow the browser to go to the next webpage (whatever is in the <form action=""> property)

}
```

7: document.getElementById("");

It is extremely common to want to access a particular tag in JavaScript. We can do this as long as that tag has an `id`. In our example, the first textbox has `id="theName"`. So, to access this tag we would type:

- `document.getElementById("theName")`

If we wanted to change the value of the **placeholder** property of this tag, we would write [**WRITE**, so *LHS* of the =]:

- `document.getElementById("theName").placeholder = "3 or more characters, if you please";`

If we wanted to get what was typed into this textbox (the **value** property) and store it in a variable [**READ**, so *RHS* of the =]:

- `var theTypedName = document.getElementById("theName").value;`

See the two (READ from / WRITE to) `document.getElementById` hosted examples…links in Moodle.

PROPERTIES of a TAG: for example `.placeholder`, `.value`. (We could also have `.step`, `.min`, `.max`, `.title`, `.size`, `.maxlength`, etc…all the form element properties can be accessed by JavaScript)

METHODS for a TAG: we can run built-in Javascript functions on a tag. These have a () after them. Examples include `.select()` or `.focus()`.

8: WEBPAGE 1: (b) CODE:

Add the following **10** lines of Javascript code to the **`checkForm()`** function, under the appropriate comment line:

```
24  function checkForm() {
25      //check the length of what was typed in for the Name.
26      //if what was typed in was less than three characters in length:
27      if(document.getElementById("theName").value.length <3) {
28
29          //tell the user
30          alert("The Name is not long enough: 3 or more characters please");
31
32          //highlight the text that is in error
33          document.getElementById("theName").select();
34
35          // force the browser to stay on the same webpage
36          return false;
37
38      //if what was typed in has a valid number of characters
39      } else {
40          //write the Name to Local Storage
41          localStorage.setItem("the_name", document.getElementById("theName").value);
42
43          //write the Age to Local Storage
44          localStorage.setItem("the_age", document.getElementById("theAge").value);
45
46          //write the Country to Local Storage
47          localStorage.setItem("the_country", document.getElementById("theCountry").value);
48
49          //allow the browser to go to the next webpage (whatever is in the <form action=""> property)
50          return true;
51      }
52  }
```

Be careful with the quotation marks inside the **`.getElementById()`**…if the quotation marks are **angled** **(“”)** the code won’t work, but they will if the quotation marks are **vertical** **("")**.

This code:

- reserves three spaces in the browser memory called **`the_name`**, **`the_age`** and **`the_country`**;
- uses the **`.setItem()`** method to WRITE to Local Storage;
- NO = in the lines of code that use **`.setItem()`**.

9: WEBPAGE 2: COMMENTS and CODE:

Simpler code, hopefully! This time we use **`.getItem()`** for Local Storage.

Add the following comments and code into **`readLocalStorage()`**, the Javascript function for Webpage 2:

```
function readLocalStorage() {
    //READ the NAME from Local Storage, and display it in an appropriate place on Webpage 2
    document.getElementById("name_LS").innerHTML = localStorage.getItem("the_name");

    //READ the AGE from Local Storage, and display it in an appropriate place on Webpage 2
    document.getElementById("age_LS").innerHTML = localStorage.getItem("the_age");

    //READ the COUNTRY from Local Storage, and display it in an appropriate place on Webpage 2
    document.getElementById("country_LS").innerHTML = localStorage.getItem("the_country");
}
```

- YES = in the lines of code that use **`.getItem()`**.

10: …IF YOUR JAVASCRIPT CODE ISN'T WORKING:

1) Make sure you have **Developer Tools / Console** open in Chrome (F12, and Console Tab);
2) If you have any **syntax** errors, they will show up immediately;
3) If you have any **run-time** errors, they will show up when the Javascript is run (after the event happens);
4) If you have any **logical** errors (for example, the if part works but the else part doesn't): add in console.log("testing 1/2/3/etc") messages into your code to see where your code logic is wrong. A logical error won't give you an error message, but the result of your code will not be what you want, **so be careful** … no errors doesn't necessarily mean there's no problem!

11: EXERCISE:

1) Add four more form elements (including a set of radio buttons and an email textbox) into the form;
2) Modify the **`checkForm()`** function so that the values from these extra form elements are written into local storage;
3) Modify the **`readLocalStorage()`** function so that the values from these extra form elements are read from local storage onto the second webpage.
4) Host this Local Storage website (both pages in a zipped file), email your lecturer the URL of the first webpage.