MOOC_html_mod6-strings_iteradores_regex_entrega

Objetivos

- Practicar con Strings, arrays e iteradores
- Utilizar expresiones regulares para identificar patrones en textos

Descripción de la práctica

Esta práctica consiste en desarrollar una aplicación web para contar y buscar letras, palabras y frases en textos utilizando Strings, iteradores y expresiones regulares. Se proporciona una versión inicial de la aplicación con las funcionalidades de contar y buscar para letras y frases. El alumno deberá lograr la misma funcionalidad para palabras.

Descargar el código del proyecto

El proyecto debe clonarse en el ordenador desde el que se está trabajando:

```
$ git clone
  https://github.com/ging-moocs/MOOC_html_mod6-strings_iteradores_regex_entrega
```

A continuación se debe acceder al directorio de trabajo y abrir el fichero index.html con el editor de la elección del alumno.

```
$ cd MOOC_html_mod6-strings_iteradores_regex_entrega
```

El código de la aplicación web, disponible en el fichero `index.html` , es el siguiente:

```
<!DOCTYPE html><html>
<head><meta charset="UTF-8">
<script type="text/javascript" source="stopwords-es.json"></script>
<script type="text/javascript">
  const getById = id => document.getElementById(id);
  const getAll = sel => document.querySelectorAll(sel);
```

```
const getIO = () => ({ text: getAll("#text")[0]
                                .value
                                .normalize(),
                        search: getAll("#search")[0]
                                .value
                                .normalize(),
                        view: getAll("#view")[0]
});

const clean_string = (text) =>
  text
    .replace(/[\n\r\t]+/igm, " ")
    .replace(/[^a-zñáéíóú0-9 \.,;:()¿?¡!“”“”«»‘’'’’\-_]+/igm, "")
    .replace(/[ ]+/gm, " ")

const char_array = (text) =>
  clean_string(text)
    .replace(/[^a-zñáéíóú]+/igm, "")
    .split("")
    .filter((w) => (w!==""))

// TODO
const word_array = (text) => []

const sentence_array = (text) =>
    clean_string(text)
    .replace(/([\.:;?!\n]+)/gm, "$1+")
    .split("+")
    .filter((w) => (w!==""))
    .map((s) => (s.replace(/^[ 0-9]+(.*$)/, "$1")))

const repetitions = (ordered_array) =>
  ordered_array
    .reduce(
      (acc, el, i, a) => {
        if (i===0)            acc.push({s: el, n: 1});
        else if (el===a[i-1]) acc[acc.length-1].n++;
        else                  acc.push({s: el, n: 1});
        return acc;
      },
    []
  );

const count = () => {
  let {text, view} = getIO();

  let result =  `Caracteres: ${char_array(text).length}\n`;
      result += `Palabras: ${"No implementado"/*TO DO*/}\n`;
      result += `Frases: ${sentence_array(text).length}\n`;
      result += `Lineas: ${text.split("\n").length}\n`;

  view.innerHTML = result;
};

const letter_index = () => {
```

```
  let {text, view} = getIO();

  let ordered_letters =
    char_array(text)
      .map(el => el.toLowerCase())
      .sort();

  let result =
    repetitions(ordered_letters)
    .map(el => `${el.s}: ${el.n}`)
    .join("\n");

  view.innerHTML = result;
};

// TODO
const word_index = () => {
};

const sentence_index = () => {
  let {text, view} = getIO();

  let ordered_sentences =
    sentence_array(text)
    .map(el => el.toLowerCase())
    .sort();
  let result =
    repetitions(ordered_sentences)
    .map(el => `${el.s}: ${el.n}`)
    .join("\n");

  view.innerHTML = result;
};

const search_letters = () => {
  let {text, view, search} = getIO();

  let ordered_letters =
    char_array(text)
      .map(el => el.toLowerCase())
      .filter(el => el.includes(search.toLowerCase()))
      .sort();

  let result = `Hay ${ordered_letters.length} ocurrencias de
                la letra '${search}'.\n\n`

  result +=
    repetitions(ordered_letters)
      .map(el => `${el.n} repeticiones de:  ${el.s}`)
      .join("\n");

  view.innerHTML = result;
};

// TODO
```

```
  const search_words = () => {
  };

  const search_sentences = () => {
    let {text, view, search} = getIO();

    let searched_sentences =
      sentence_array(text)
        .filter(el => el.includes(search))
        .sort()

    let result = `Hay ${searched_sentences.length} frases
                    que contienen '${search}'.\n\n`

    result +=
      repetitions(searched_sentences)
      .map(el => `${el.n} repeticiones de:   ${el.s}`)
      .join("\n");

    view.innerHTML = result;
  };

    // ROUTER de eventos
  document.addEventListener('click', ev => {
      if      (ev.target.matches('.count'))        count();
      else if (ev.target.matches('.letter_index'))  letter_index();
      else if (ev.target.matches('.word_index'))    word_index();
      else if (ev.target.matches('.sentence_index'))    sentence_index();
      else if (ev.target.matches('.search_letters'))  search_letters();
      else if (ev.target.matches('.search_words'))  search_words();
      else if (ev.target.matches('.search_sentences'))  search_sentences();
  });
</script>
</head>
<body>
<h1>Análisis de texto</h1>

Introduzca un texto para analizarlo:<br>
<textarea rows=10 cols=50 id="text" name="text"></textarea>
<p>
  <button class="count" >Contar</button><!-- TODO-->
  <button class="letter_index" >Repeticiones de letras</button>
  <button class="word_index" >Repeticiones de palabras</button>
  <button class="sentence_index" >Repeticiones de frases</button>
</p>
<p>
  Buscar repeticiones que contienen:
  <input type="text" id="search" name="search" >
  <button class="search_letters" >Buscar letras</button>
  <button class="search_words" >Buscar palabras</button>
  <button class="search_sentences" >Buscar frases</button>
</p>

<pre id="view"></pre>
```

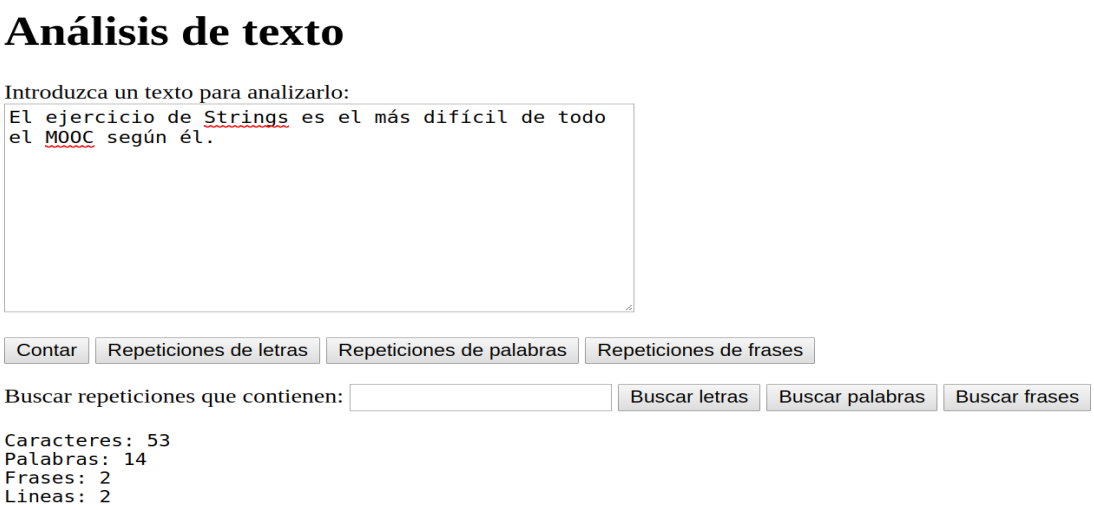
```
</body>
</html>
```

Tareas

El alumno debe desarrollar las funcionalidades que se indican con la palabra clave `TODO` en el código, las cuales se detallan a continuación:

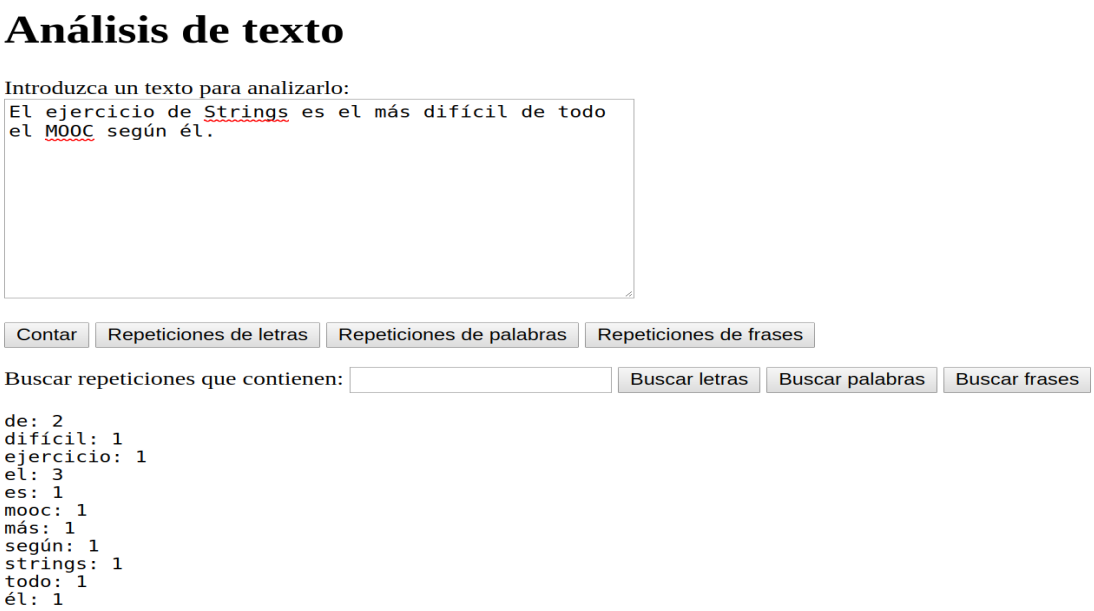
Contar palabras

Al pulsar el botón "Contar" debe mostrase el número de palabras que hay en el texto introducido por el usuario en la caja superior. Para ello, primero completa la función `word_array` para que convierta el texto en un array de palabras. A continuación, haz uso de la función `word_array` desde la función `count` para mostrar el número total de palabras en la interfaz web, tal y como se indica en la siguiente captura:



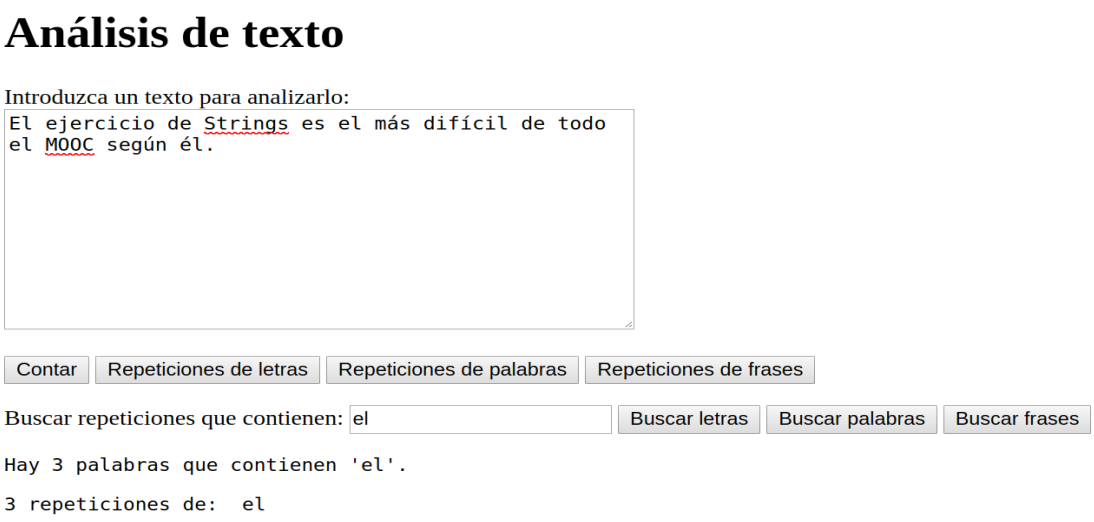
Repeticiones de palabras

Al pulsar el botón "Repeticiones de palabras", deben aparecer listadas todas las palabras del texto por orden alfabético junto al número de veces que aparece cada palabra. Si una palabra aparece algunas veces con mayúscula y otras con minúscula deben contarse como una misma palabra. Por el contrario, si una palabra aparece a veces acentuada y otras veces no, debe contarse como palabras distintas. Deben ignorarse los números y signos de puntuación. En la siguiente captura se muestra un ejemplo de esta funcionalidad:



Buscar palabras

Al pulsar el botón "Buscar palabras", deben aparecer listadas todas las palabras del texto que contienen las letras introducidas en el campo de texto, así como el número de veces que aparece cada palabra, tal y como se muestra en la siguiente captura. De nuevo deben ignorarse los signos de puntuación y la diferencia entre mayúsculas y minúsculas pero tenerse en cuenta las diferencias de acentuación.



Prueba de la práctica

Para ayudar al desarrollo, se provee una herramienta de autocorrección que prueba las distintas funcionalidades que se piden en el enunciado. Para utilizar esta herramienta debes tener node.js (y npm) (https://nodejs.org/es/) y Git instalados.

Para instalar y hacer uso de la herramienta de autocorrección en el ordenador local, ejecuta los siguientes comandos en el directorio del proyecto:

```
$ npm install -g autocorector     ## Instala el programa de test
$ autocorector                    ## Pasa los tests al fichero a entregar
.........................         ## en el directorio de trabajo
... (resultado de los tests)
```

También se puede instalar como paquete local, en el caso de que no se dispongas de permisos en el ordenador desde el que estás trabajando:

```
$ npm install autocorector     ## Instala el programa de test
$ npx autocorector             ## Pasa los tests al fichero a entregar
.........................      ## en el directorio de trabajo
... (resultado de los tests)
```

Se puede pasar la herramienta de autocorrección tantas veces como se desee sin ninguna repercusión en la calificación.