'use strict';

const templates = {

  articleLink: Handlebars.compile(document.querySelector('#template-article-link').innerHTML),

  tagLink: Handlebars.compile(document.querySelector('#template-tag-link').innerHTML),

  authorLink: Handlebars.compile(document.querySelector('#template-author-link').innerHTML),

  tagCloudLink: Handlebars.compile(document.querySelector('#template-tagCloud-link').innerHTML),

  authorSideLink: Handlebars.compile(document.querySelector('#template-authorSide-link').innerHTML),

};

const opts = {

  articleSelector : '.post',

  titleSelector :  '.post-title',

  titleListSelector : '.titles',

  articleTagsSelector : '.post-tags .list',

  articleAuthorSelector : '.post-author',

  tagsListSelector : '.tags .list',

  cloudClassCount : 5,

  cloudClassPrefix : 'tag-size-',

  authorsListSelector : '.authors.list',

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.1 - function titleClickHandler \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function titleClickHandler(event){

  event.preventDefault();

  const clickedElement = this;

  /\* remove class 'active' from all article links \*/

  const activeLinks = document.querySelectorAll('.titles a.active');

  for(let activeLink of activeLinks) {

    activeLink.classList.remove('active');

  }

  /\* add class 'active' to the clicked link \*/

  clickedElement.classList.add('active');

  /\* remove class 'active' from all articles \*/

  const activeArticles = document.querySelectorAll('.posts .active');

  for (let activeArticle of activeArticles) {

    activeArticle.classList.remove('active');

  }

  /\* get 'href' attribute from the clicked link \*/

  const clickedLinkAttribute = clickedElement.getAttribute('href');

  /\* find the correct article using the selector (value of 'href' attribute) \*/

  const targetArticle = document.querySelector(clickedLinkAttribute);

  /\* add class 'active' to the correct article \*/

  targetArticle.classList.add('active');

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.2 - function generateTitleLinks \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function generateTitleLinks(customSelector = ''){

  /\* remove contents of titleList \*/

  const titleList = document.querySelector(opts.titleListSelector);

  titleList.innerHTML = '';

  /\* for each article \*/

  const articles = document.querySelectorAll(opts.articleSelector + customSelector);

  let html = '';

  for (let article of articles) {

    /\* get the article id \*/

    const articleId = article.getAttribute('id');

    /\* find the article title element \*/

    const titleElement = article.querySelector(opts.titleSelector);

     /\* get the title from the title article \*/

     const title = titleElement.innerHTML;

    /\* create HTML of the link \*/

    const linkHTMLData = {id: articleId, title: title};

    const linkHTML = templates.articleLink(linkHTMLData);

    /\* insert link into html variable \*/

    html = html + linkHTML;

  }

  titleList.innerHTML = html;

  const links = document.querySelectorAll('.titles a');

  for(let link of links){

    link.addEventListener('click', titleClickHandler);

  }

}

generateTitleLinks();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.9 function calculateTagsParams \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function calculateTagsParams(tags) {

  const params = {min : 99999, max : 0};

  for (let tag in tags) {

    console.log(tag + ' is used ' + tags[tag] + ' times');

    if(tags[tag] > params.max) {

      params.max = tags[tag];

    } else if (tags[tag] < params.min) {

      params.min = tags[tag];

    }

  }

return params;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.10 function calculateTagClass \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function calculateTagClass (count, params){

  const normalizedCount = count - params.min;

  const normalizedMax = params.max - params.min;

  const percentage = normalizedCount / normalizedMax;

  const classNumber = Math.floor(percentage \* (opts.cloudClassCount - 1) + 1 );

  return opts.cloudClassPrefix + classNumber;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.3 - function generateTags \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function generateTags(){

  /\* [NEW] create a new variable allTags with an empty object \*/

  let allTags = {};

  /\* find all articles \*/

  const articles = document.querySelectorAll(opts.articleSelector);

  /\* START LOOP: for every article: \*/

  for (let article of articles) {

    /\* find tags wrapper \*/

    const tagsWrapper = article.querySelector(opts.articleTagsSelector);

    /\* make html variable with empty string \*/

    let html = '';

    /\* get tags from data-tags attribute \*/

    const tags = article.getAttribute('data-tags');

    /\* split tags into array \*/

    const tagsArray = tags.split(' ');

    /\* START LOOP: for each tag \*/

    for (let tag of tagsArray) {

      /\* generate HTML of the link \*/

      const tagHTMLData = {id: tag, tagName: tag};

      const linkHTML = templates.tagLink(tagHTMLData);

      /\* add generated code to html variable \*/

      html = html + '  ' + linkHTML;

      /\* [NEW] check if this link is NOT already in allTags \*/

      if(!allTags.hasOwnProperty(tag)){

        /\* [NEW] add generated code to allTags object \*/

        allTags[tag] = 1;

      } else {

        allTags[tag]++;

      }

    /\* END LOOP: for each tag \*/

    }

    /\* insert HTML of all the links into the tags wrapper \*/

    tagsWrapper.innerHTML = html;

  /\* END LOOP: for every article: \*/

  }

  /\* [NEW] find list of tags in right column \*/

  const tagList = document.querySelector(opts.tagsListSelector);

  const tagsParams = calculateTagsParams(allTags);

  /\* [NEW] create variable for all links HTML code \*/

  const allTagsData = {tags:[]};

  /\* [NEW] START LOOP: for each tag in allTags: \*/

  for(let tag in allTags){

    allTagsData.tags.push({

      tag: tag,

      count: allTags[tag],

      className: calculateTagClass(allTags[tag], tagsParams)

    });

    /\* [NEW] END LOOP: for each tag in allTags: \*/

  }

  /\* [NEW] add html from allTagsHTML to tagList \*/

  tagList.innerHTML = templates.tagCloudLink(allTagsData);

  console.log('allTagsData:' + allTagsData)

}

generateTags();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.4 - function tagClickHandler \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function tagClickHandler(event){

  /\* prevent default action for this event \*/

  event.preventDefault();

  /\* make new constant named "clickedElement" and give it the value of "this" \*/

  const clickedElement = this;

  /\* make a new constant "href" and read the attribute "href" of the clicked element \*/

  const href = clickedElement.getAttribute('href');

  /\* make a new constant "tag" and extract tag from the "href" constant \*/

  const tag = href.replace('#tag-', '');

  /\* find all tag links with class active \*/

  const activeTags = document.querySelectorAll('a.active[href^="#tag-"]');

  /\* START LOOP: for each active tag link \*/

  for (let activeTag of activeTags) {

    /\* remove class active \*/

    activeTag.classList.remove('active');

  /\* END LOOP: for each active tag link \*/

  }

  /\* find all tag links with "href" attribute equal to the "href" constant \*/

  const tagLinks = document.querySelectorAll('a[href="' + href + '"]');

  /\* START LOOP: for each found tag link \*/

  for (let tagLink of tagLinks) {

    /\* add class active \*/

    tagLink.classList.add('active');

  /\* END LOOP: for each found tag link \*/

  }

  /\* execute function "generateTitleLinks" with tag selector as argument \*/

  generateTitleLinks('[data-tags~="' + tag + '"]');

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.5 - function addClickListenersToTags \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function addClickListenersToTags(){

  /\* find all links to tags \*/

  const allTagLinks = document.querySelectorAll('a[href^="#tag-"]');

  /\* START LOOP: for each link \*/

  for (let link of allTagLinks) {

    /\* add tagClickHandler as event listener for that link \*/

    link.addEventListener('click', tagClickHandler);

  /\* END LOOP: for each link \*/

  }

}

addClickListenersToTags();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.6 - function generateAuthors \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function generateAuthors(){

  /\* [NEW] create allAuthors object \*/

  const allAuthors = {};

  /\* find all articles \*/

  const articles = document.querySelectorAll(opts.articleSelector);

  /\* START LOOP: for every article: \*/

  for (let article of articles) {

    /\* find author wrapper \*/

    const authorWrapper = article.querySelector(opts.articleAuthorSelector);

    /\* make html variable with empty string \*/

    let html = '';

    /\* get author from data-author attribute \*/

    const author = article.getAttribute('data-author');

    /\* generate HTML of the link \*/

    const authorHTMLData = {id: author, authorName: author};

    const linkHTML = templates.authorLink(authorHTMLData);

    /\* add generated code to html variable \*/

    html = html + linkHTML;

    if(!allAuthors.hasOwnProperty(author)) {

      allAuthors[author] = 1;

    } else {

      allAuthors[author]++;

    }

    /\* insert HTML into author wrapper \*/

    authorWrapper.innerHTML = html;

    /\* END LOOP: for each article \*/

  }

  /\* [NEW] find authors list wrapper \*/

  const authorsList = document.querySelector(opts.authorsListSelector);

  /\* [NEW] create variable for all authors links HTML code \*/

  const allAuthorsData = {authors: []};

  /\* [NEW] create a link for every author with a number of his articles \*/

  for (let author in allAuthors) {

    allAuthorsData.authors.push({

      author: author,

      count: allAuthors[author],

    });

  }

  /\* generate authors list in right sidebar \*/

  authorsList.innerHTML = templates.authorSideLink(allAuthorsData);

  /\* END LOOP: for every article: \*/

}

generateAuthors();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.7 - function authorClickHandler \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function authorClickHandler(event){

  /\* prevent default action for this event \*/

  event.preventDefault();

  /\* make new constant named "clickedElement" and give it the value of "this" \*/

  const clickedElement = this;

  /\* make a new constant "href" and read the attribute "href" of the clicked element \*/

  const href = clickedElement.getAttribute('href');

  /\* make a new constant "author" and extract author from the "href" constant \*/

  const author = href.replace('#author-', '');

  /\* find all author links with class active \*/

  const activeAuthorLinks = document.querySelectorAll('a.active[href^="#author-"]');

  /\* START LOOP: for each active author link \*/

  for (let activeAuthorLink of activeAuthorLinks) {

    /\* remove class active from every activeAuthorLink\*/

    activeAuthorLink.classList.remove('active');

  /\* END LOOP: for each active tag link \*/

  }

  /\* find all author links with "href" attribute equal to the "href" constant \*/

  const authorLinks = document.querySelectorAll('a[href="' + href + '"]');

  /\* START LOOP: for each found tag link \*/

  for (let authorLink of authorsLinks) {

    /\* add class active \*/

    authorLink.classList.add('active');

  /\* END LOOP: for each found tag link \*/

  }

  /\* execute function "generateTitleLinks" with article selector as argument \*/

  generateTitleLinks('[data-author="' + author + '"]');

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* part.8 - function addClickListenersToAuthors \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function addClickListenersToAuthors(){

  /\* find all links to tags \*/

  const allAuthorsLinks = document.querySelectorAll('a[href^="#author-"]');

  /\* START LOOP: for each link \*/

  for (let authorsLink of allAuthorsLinks) {

    /\* add event listener to every authorsLink \*/

    authorsLink.addEventListener('click', authorClickHandler);

  /\* END LOOP: for each link \*/

  }

}

addClickListenersToAuthors();