основы Javascript

необходимые инструменты

Установка Node.js

Установите последнюю версию "Long Term Support (LTS)" Node.js (текущая 6.10.3). Она может быть установлена отсюда - https://nodejs.org/en/download. Чтобы удостовериться что Node.js и Node package manager (npm) верно установлено выполните команды в консоли:

*node -v*

И:

*npm -v*

которые должны запуститься и вывести версию node и npm.

Установка

После установки Node.js и npm, создайте каталог project1 и распакуйте таду содержимое приложенного zip файла. Архив содержит test-project1.html and test-project1.js, где вы должны выполнять персональные задания

Чтобы установить инструмент контроля качества кода, JSHint, выполнив следующую команду в каталоге project1:

npm install

Это позволит получить JSHint в подкаталоге node\_modules. Вы сможете запустить его для всех файлов JavaScript в каталоге project1, выполнив команду:

npm run jshint

Код, который вы отправляете, должен начинаться с «use strict»; в каждом файле JavaScript и при запуске JShint не должен выводить никаких предупреждений. Наличие предупреждний снижают общий балл за лабораторную

Чтобы запустить лабораторную работу, откройте файл test-project1.html в своем браузере. Веб-страница загрузит JavaScript и проведет несколько тестов. Так как Node.js также содержит виртуальную машину JavaScript, вы также можете запускать тесты без браузера с помощью команды:

npm test

В браузере среда разработки JavaScript намного лучше, поэтому мы предлагаем вам использовать ее для отладки кода.

В рамках этого задания, необходимо написать или модифицировать некоторые js функции.

Задачи в этом задании носят практический характер, и функциональность, которую вы разрабатываете, будет полезна для завершения следующих лабораторных работ.

Существует множество различных фреймворков, с помощью которых можно решить поставленные задачи двумя строчками кода, но задача работы состоит в том, чтобы узнать ядро языка, поэтому **использование фреймворков запрещено**.

Допустимо использовать встроенные функции работы с массивами и датами.

Задание 1. разработка функции – фильтра (10 баллов)

В каталоге project1, создайте новый файл make-multi-filter.js, в котором вы будете разрабатывать требуюмую функцию.

Определите глобальную функцию MakeMultiFilter, которая принимает на вход массив originalArray как параметр и возвращает функцию, которая может быть использована в качестве фильтра для элементов массива. Первоначально currentArray установлен как originalArray. Функция arrayFilterer принимает две функции в качестве параметров:

* *filterCriteria* – Функция принимает на вход элемент массива и возвращает boolean. Эта функция будет вызываться для каждого элемента currentArray и currentArray отражает результат применения функции filterCriteria. Если filterCriteria возвращает false для элемента, то этот элемент должен быть удален из currentArray. Иначе, элемент остается в currentArray. Если filterCriteria не функция, то возвращаемая функция (arrayFilterer) должна возвращать currentArray неизмененным.
* *callback* – функция которая вызывается, когда фильтрация закончена. callback принимает currentArray как аргумент. Объект *this* внутри callback функции должен ссылаться на значение originalArray. Если параметр callback – не функция, он должен игнорироваться. callback не имеет возвращаемого значения.

Функция arrayFilterer должна возвращаться сама, если параметр filterCriteria не указан, и в этом случае он должен возвращать currentArray. Должно быть возможно одновременное использование нескольких функций arrayFilterer.

Нижеследующий код показывает как могут использоваться разработанные функции:

// вызов MakeMultiFilter() with originalArray = [1,2,3] возвращает функцию, сохраненную в переменную arrayFilterer1,

//that can be used to repeatedly filter the input array

var arrayFilterer1 = MakeMultiFilter([1,2,3]);

// вызов arrayFilterer1 (с callback функцией) чтобы отфильтровать все числа не равные 2

arrayFilterer1(function (elem) {

return elem !== 2; // check if element is not equal to 2

}, function (currentArray) {

console.log(this); // 'this' должно выводить originalArray который [1,2,3]

console.log(currentArray); // prints [1, 3]

});

// вызов arrayFilterer1 (без callback функции) чтобы отфильтровать все числа не равные 3

arrayFilterer1(function (elem) {

return elem !== 3; // check if element is not equal to 3

});

// вызов arrayFilterer1 без filterCriteria должен вернуть currentArray

var currentArray = arrayFilterer1();

console.log('currentArray', currentArray); // должен вывести только [1], поскольку 2 и 3 уже отфильтрованы

// Поскольку arrayFilterer возвращает себя, вызовы могут быть выстроены в цепочку

function filterTwos(elem) { return elem !== 2; }

function filterThrees(elem) { return elem !== 3; }

var arrayFilterer2 = MakeMultiFilter([1,2,3]);

var currentArray2 = arrayFilterer2(filterTwos)(filterThrees)();

console.log('currentArray2', currentArray2); // должен вывести только [1], поскольку 2 и 3 отфильтрованы

// Несколько активных фильтров в одно время

var arrayFilterer3 = MakeMultiFilter([1,2,3]);

var arrayFilterer4 = MakeMultiFilter([4,5,6]);

console.log(arrayFilterer3(filterTwos)()); // prints [1,3]

console.log(arrayFilterer4(filterThrees)()); // prints [4,5,6]

Задание 2. Процессор шаблонов (5 баллов)

В каталоге project, создайте новый файл template-processor.js.

Создайте класс шаблонизатора (TemplateProcessor) который имеет свойство *template*  и метод *fillIn*.

Аргумент метода – словарь пар ключ\значение. В результате выполнения метод возвращает template в котором заменены ключи из массива аргумента на значения. TemplateProcessor должен быть написан используя стандартный конструктор JavaScript и свойство prototype.

Шаблон template содержит идентфикаторы {{property}} которые должны методом *fillIn* подменяться на значения из переданного массива.

Если шаблон содержит свойство, которого нет в исходном массиве, то оно должно замениться на пустую строку. Пример: "This {{undefinedProperty}} is cool" -> "This is cool".

Допускается обработка только правильно оформленных шаблонов. Необходимо возвращать undefined в случаях:

Вложенных свойств - {{foo {{bar}}}} or {{{{bar}}}} or {{{bar}}}

Несбалансированных скобок - {{bar}}}

Нижеследующий код показывает как могут использоваться разработанные функции:

var template = 'My favorite month is {{month}} but not the day {{day}} or the year {{year}}';

var dateTemplate = new TemplateProcessor(template);

var dictionary = {month: 'July', day: '1', year: '2016'};

var str = dateTemplate.fillIn(dictionary);

assert(str === 'My favorite month is July but not the day 1 or the year 2016');

//Case: property doesn't exist in dictionary

var dateTemplate2 = new TemplateProcessor(template);

var dictionary2 = {day: '1', year: '2016'};

var str = dateTemplate.fillIn(dictionary2);

assert(str === 'My favorite month is but not the day 1 or the year 2016');

Задание 3. Изменить test-project1.js чтобы он не засорял глобальную область видимости (5 баллов)

Тестовый js файл который вам дан (test-project1.js) определяет некоторые переменный в глобальной области видимости. Например, после скрипта переменная p1Message определяется глобальной и другой скрипт может получить доступ к ней. Измените test-project1.js так, чтобы он использовал javascript паттерн «модуль» (используя анонимную функцию) для того, чтобы скрыть глобальные переменные, но сохранить функциональность.

бонус. стиль (5 баллов)

Начисляется, если js код чистый, комментированный и JSHint  не возвращает предупреждения