<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<style>

table {

border-collapse: collapse;

border: 3px solid black;

color: #006400;

}

td {

border-bottom: 2px solid lightgray;

border-left: 2px solid lightgray;

}

</style>

</head>

<body>

<table border="1">

<tr>

<td><b id="dam">День, месяц и год</b></td><td id="day\_month\_year"></td>

</tr>

<tr>

<td><b id="dow">День недели</b></td><td id="week"></td>

</tr>

</table>

<hr>

<p id= "sesstart">Сессия начинается 16 июня</p>

<p id="days\_left">До начала сессии <span id="sessbef"></span> дней<p>

<hr>

<button onclick="important\_date\_func();" id="imp">Важная дата</button>

<p id="data">Дней прошло : <input id="amount\_of\_days"/></p>

<hr>

<p id="cel">Последовательность: 1+(-2)^n</p>

<p id="first">Введите значение первого члена n <input type="text" id="n"/></p>

<p id="cnt">Введите кол-во слагаемых k <input type="text" id="k"/></p>

<button id="zad" onclick="members\_of\_the\_sequence();">Cчитать</button><br>

<p id="your">Ваша последовательность: <input type="text" disabled="" id="sequence"/></p>

<p id="amount">Сумма заданного количества членов числовой последовательности: <input type="text" disabled="" id="sum"/></p>

<hr>

<div style="position:absolute;">

<textarea placeholder="enter text to search" id="toSearch" wrap="off" cols="30" rows="5"></textarea>

<input id="ad" type="button" value="Абдуллаев" onclick="searchFunc()"></input>

<div id="found"></div>

<input id="sw" type="button" value="Switch language" onclick="new function(){ui.lang.switchLanguage(ui.lang.current=='ru'?'eng':'ru');}"></input>

<script type="text/javascript">

function print(msg) {document.write(msg + "<br>");}

</script>

<script type="text/javascript" src="Module.js"></script>

<script type="text/javascript" src="script.js"></script>

</body>

</html>

Js

var f\_1 = function(){

Module.createNamespace("ui.lang.ru", 1.0);

Module.createNamespace("ui.lang.eng", 1.0);

Module.require("ui.lang.ru", 1.0);

Module.require("ui.lang.eng", 1.0);

Module.createNamespace("about",1.0);

Module.require("about",1.0);

about.info = "This program is made as a 7'th laboratory work";

ui.lang.objects = ["days\_left","amount","sw","sesstart","dow","dam","imp","data","cel","your","zad","first","cnt"]

ui.lang.ru.strings = {

"dow":"День недели",

"dam":"День, месяц и год",

"DOW":["Воскресенье","Понедельник","Вторник","Среда","Четверг","Пятница","Суббота"],

"Mon":["Января ","Февраля ","Марта ","Апреля ","Мая ","Июня ","Июля ","Августа ","Сентября ","Октября ","Ноября ","Декабря "],

"days\_left":"До начала сессии <span id='sessbef'></span> дней",

"sesstart":"Сессия начинается 16 июня",

"amount":"Сумма заданного количества членов числовой последовательности: <input type='text' disabled='' id='sum'/>",

"imp":"ВАЖНАЯ ДАТА",

"data":"Дней прошло : <input id='amount\_of\_days'/>",

"cel":"Последовательность: 1+(-2)^n",

"your":"Ваша последовательность: <input type='text' disabled='' id='sequence'/>",

"zad":"Считать",

"first":"Введите значение первого члена n <input type='text' id='n'/>",

"enter":"Введите памятную для вас дату (ДД.ММ.ГГГГ): ",

"about":"С чем это связано?",

"cnt":"Введите кол-во слагаемых k <input type='text' id='k'/>",

"toSearch":"Введите текст для поиска",

"err":"Значения введены некорректно!",

"ad":"ГГГГ.ММ.ДД",

"sw":"сменить язык",

"prompt":"Введите номер первого члена и кол-во слагаемых последовательности (через пробел): "}

ui.lang.eng.strings = {

"dow":"Day of the week",

"dam":"Day, month and year",

"DOW":["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"],

"Mon":["January ","February ","March ","April ","May ","June ","Jule ","August ","September ","October ","November ","December "],

"days\_left":"Days left <span id='sessbef'></span> days",

"sesstart":"The session begins on June 16",

"amount":"The sum of a given number of members of a number sequence: <input type='text' disabled='' id='sum'/>",

"imp":"IMPORTANT DATE",

"data":"Days passed : <input id='amount\_of\_days'/> ",

"cel":"Sequence: 1+(-2)^n",

"your":"Your sequence: <input type='text' disabled='' id='sequence'/>",

"zad":"Count",

"enter":"Enter a memorable date for you (DD.MM.YYYY): ",

"about":"What is the reason for this?",

"toSearch":"enter text to search",

"first":"Set the initial value of n <input type='text' id='n'/>",

"cnt":"Set the count k of sequence numbers <input type='text' id='k'/>",

"err":"Values entered incorrectly!",

"ad":"YYYY.MM.DD",

"sw":"switch language",

"prompt":"Enter position of first member and count of members (divide them by spacebar)"}

ui.lang.switchLanguage = function(l){

switch(l){

case "eng":ui.lang.strings = ui.lang.eng.strings; ui.lang.current = "eng";break;

case "ru":ui.lang.strings = ui.lang.ru.strings; ui.lang.current = "ru";break;

default:ui.lang.strings = ui.lang.eng.strings; ui.lang.current = "eng";break;

}

ui.lang.initLanguage();

}

ui.lang.initLanguage = function(){

var now = new Date();

ui.lang.objects.forEach(function(el){

switch(document.getElementById(el).localName){

case "button":document.getElementById(el).innerHTML = ui.lang.strings[el];break;

case "tr":document.getElementById(el).innerHTML = ui.lang.strings[el];break;

case "p":document.getElementById(el).innerHTML = ui.lang.strings[el];break;

case "b":document.getElementById(el).innerHTML = ui.lang.strings[el];break;

default:document.getElementById(el).value = ui.lang.strings[el];break;}

})

week.innerHTML=ui.lang.strings["DOW"][new Date().getDay()];

day\_month\_year.innerHTML= now.getDate() + ' ' + ui.lang.strings["Mon"][now.getMonth()] + date.getFullYear();

sessbef.innerHTML = Math.floor((new Date(2019,5,16) - new Date()) / 86400000)

toSearch.placeholder = ui.lang.strings["toSearch"]

}

ui.lang.switchLanguage("ru");

alert("Модуль администрирования модулей загружен!");

//функция print затрёт то, что было загружено

}

Module.registerInitializationFunction(f\_1);

var date = new Date();

function important\_date\_func() {

// document.getElementById("important\_date").innerHTML = "";

// document.getElementById("amount\_of\_days").innerHTML = "";

var impday = prompt(ui.lang.strings["enter"]);

// document.getElementById("important\_date").value = impday;

var date\_array = impday.split(".");

var memorable\_date = new Date(Number(date\_array[2]),Number(date\_array[1]-1),Number(date\_array[0]));

document.getElementById("amount\_of\_days").value = Math.floor((date - memorable\_date) / 86400000);

}

function members\_of\_the\_sequence() {

// document.getElementById("sequence").innerHTML = "";

// document.getElementById("sum").innerHTML = "";

var n = document.getElementById("n").value;

var quantity = document.getElementById("k").value;

var sum = 0;

var element;

function isInteger(num) {

return num - (Math.floor(num)) === 0 && num > 0;

}

if (isInteger(n) && isInteger(quantity)){

for (var i=0; i<quantity; i++) {

element = 1 + Math.pow(-2,n);

document.getElementById("sequence").value = document.getElementById("sequence").value + String(element) + " ";

sum = sum + element;

n++;

}

document.getElementById("sum").value = sum;

}

else {

alert(ui.lang.strings["err"]);

}

}

function searchFunc(){

let str = toSearch.value;

let resstr = str.match( /Абдуллаев/i );

found.innerHTML="<span class='highlight'>"+resstr[0]+"</span>";

}

Модуль(Module.js)

// Пример. Модуль с функциями для обслуживания модулей

/\*\*

\* Module.js: Функции для работы с модулями и пространствами имен

\*

\* Этот модуль содержит функции для работы с модулями, которые

\* совместимы с модулями из архива JSAN.

\* Данный модуль определяет пространство имен Module.

\*/

// Убедиться, что данный модуль еще не загружен

var Module;

if (Module && (typeof Module != "object" || Module.NAME))

throw new Error("Пространство имен 'Module' уже существует");

// Создать собственное пространство имен

Module = {};

// Далее располагается метаинформация об этом пространстве имен

Module.NAME = "Module";

// Название этого пространства имен

Module.VERSION = 0.1;

// Версия этого пространства имен

// Далее следует список общедоступных символов, которые будут

// экспортироваться этим пространством имен.

// Эта информация интересна тем, кто будет пользоваться модулем

Module.EXPORT = ["require", "importSymbols"];

// Далее следует перечень символов, которые также будут экспортироваться.

// Но они, как правило, используются только авторами модулей

// и обычно не импортируются.

Module.EXPORT\_OK = ["createNamespace", "isDefined",

"registerInitializationFunction",

"runInitializationFunctions",

"modules", "globalNamespace"];

// Начинается добавление символов в пространство имен

Module.globalNamespace = this;

// Так мы всегда ссылаемся на глобальную область видимости

Module.modules = { "Module": Module };

// Соответствие Module [name] -> namespace.

/\*\*

\* Данная функция создает и возвращает объект пространства имен с заданным

\* именем и выполняет проверку на наличие конфликта между этим именем

\* и именами из любых ранее загруженных модулей.

\* Если какой либо компонент пространства имен уже существует

\* и не является объектом, генерируется исключение.

\*

\* В качестве значения свойства NAME устанавливается имя этого пространства имен.

\* Если был задан аргумент version, устанавливает свойство пространство имен VERSION.

\*

\* Отображение нового пространства имен добавляется в объект Module.modules

\*/

Module.createNamespace = function(name, version) {

// Проверить корректность имени. Оно должно существовать и не должно

// начинаться или заканчиваться символом точки или содержать в строке

// два символа точки подряд.

if (!name) throw new Error("Module.createNamespace(): не указано имя");

if (name.charAt(0) == '.' ||

name.charAt(name.length - 1) == '.' ||

name.indexOf("..") != -1)

throw new Error("Module.createNamespace(): неверное имя: " + name);

// Разбить имя по символам точки и создать иерархию объектов

var parts = name.split('.');

// Для каждого компонента пространства имен следует создать объект

// либо убедиться, что объект с таким именем уже существует.

var container = Module.globalNamespace;

for(var i = 0; i < parts.length; i++) {

var part = parts[i];

// Если свойства или контейнера с таким именем не существует,

// создать пустой объект.

if (!container[part]) container[part] = {};

else if (typeof container[part] != "object") {

// Если свойство уже существует, убедиться, что это объект

var n = parts.slice(0,i).join('.');

throw new Error(n + " уже существует, но не является объектом");

}

container = container[part];

}

// Последний контейнер, который был рассмотрен последним, – это то, что нам нужно.

var namespace = container;

//Было бы ошибкой определить одно и то же пространство имен дважды,

//но нет никакого криминала, если объект уже существует и у него

//не определено свойство NAME.

if(namespace.NAME) throw new Error("Модуль "+name+" уже определен ");

// Инициализировать поля с именем и версией пространства имен

namespace.NAME = name;

if(version) namespace.VERSION = version;

// Зарегистрировать это пространство имен в списке модулей

Module.modules[name] = namespace;

// Вернуть объект пространства имен вызывающей программе

return namespace;

}

/\*\*

\* Проверить, был ли определен модуль с заданным именем.

\* Вернуть true, если определен, и false – в противном случае.

\*/

Module.isDefined = function(name) {

return name in Module.modules;

};

/\*\*

\* Эта функция возбуждает исключение, если модуль с таким именем не определен

\* или его версия меньше указанной. Если пространство имен существует

\* и имеет допустимый номер версии, эта функция просто возвращает управление,

\* не предпринимая никаких действий. Эта функция может стать источником

\* фатальной ошибки, если модуль, требуемый вашему программному коду, отсутствует.

\*/

Module.require = function(name, version) {

if (!(name in Module.modules)) {

throw new Error("Модуль " + name + " не определен");

}

// Если версия не указана, проверка не выполняется

if (!version) return;

var n = Module.modules[name];

// Если номер версии модуля ниже, чем требуется, или пространство

// имен не объявляет версию, генерируется исключение.

if (!n.VERSION || n.VERSION < version)

throw new Error("Модуль " + name + " имеет версию " +

n.VERSION + " требуется версия " + version + " или выше.");

};

/\*\*

\* Данная функция импортирует символы из заданного модуля. По умолчанию

\* импорт производится в глобальное пространство имен, однако с помощью

\* второго аргумента можно определить иное место назначения.

\*

\* Если никакие символы явно не указываются, будут импортированы символы

\* из массива EXPORT модуля. Если этот массив, а также массив EXPORT\_OK

\* не определен, будут импортированы все символы модуля from.

\*

\* Чтобы импортировать явно указанный набор символов, их имена должны

\* передаваться в виде аргументов вслед за именем модуля и именем

\* пространства имен, куда производится импорт. Если модуль содержит

\* определение массива EXPORT или EXPORT\_OK, импортированы будут только

\* те символы, которые перечислены в одном из этих массивов.

\*/

Module.importSymbols = function(from) {

// Убедиться, что модуль корректно задан. Функция ожидает получить объект

// пространства имен модуля, но также допускается строка с именем модуля

if (typeof from == "string") from = Module.modules[from];

if (!from || typeof from != "object")

throw new Error("Module.importSymbols(): требуется указать объект пространства имен");

// Вслед за аргументом с источником импортируемых символов может

// следовать объект пространства имен, куда производится импорт,

// а также имена импортируемых символов.

var to = Module.globalNamespace; // Место назначения по умолчанию

var symbols = [];

// По умолчанию нет символов

var firstsymbol = 1;

// Индекс первого аргумента с именем символа

// Проверить, задано ли пространство имен места назначения

if (arguments.length > 1 && typeof arguments[1] == "object") {

if (arguments[1] != null) to = arguments[1];

firstsymbol = 2;

}

// Получить список явно указанных символов

for(var a = firstsymbol; a < arguments.length; a++)

symbols.push(arguments[a]);

// Если ни одного символа не было передано, импортировать набор символов,

// определяемых модулем умолчанию, или просто импортировать все символы.

if (symbols.length == 0) {

// Если в модуле определен массив EXPORT, импортировать символы из этого массива.

if (from.EXPORT) {

for(var i = 0; i < from.EXPORT.length; i++) {

var s = from.EXPORT[i];

to[s] = from[s];

}

return;

}

// Иначе, если массив EXPORT\_OK в модуле не определен,

// импортировать все символы из пространства имен модуля

else if (!from.EXPORT\_OK) {

for(s in from) to[s] = from[s];

return;

}

}

// В этой точке имеется массив импортируемых символов, определенных явно.

// Если пространство имен определяет массивы EXPORT и/или EXPORT\_OK,

// прежде чем импортировать символы, убедиться, что каждый из них

// присутствует в этих массивах. Генерировать исключение, если запрошенный

// символ не определен или не предназначен для экспорта.

var allowed;

if (from.EXPORT || from.EXPORT\_OK) {

allowed = {};

// Скопировать допустимые символы из массивов в свойства объекта.

// Это даст возможность более эффективно проверить допустимость импорта символа.

if (from.EXPORT)

for(var i = 0; i < from.EXPORT.length; i++)

allowed[from.EXPORT[i]] = true;

if (from.EXPORT\_OK)

for(var i = 0; i < from.EXPORT\_OK.length; i++)

allowed[from.EXPORT\_OK[i]] = true;

}

// Импортировать символы

for(var i = 0; i < symbols.length; i++) {

var s = symbols[i];

// Имя импортируемого символа

if (!(s in from))

// Проверить его наличие

throw new Error("Module.importSymbols(): символ " + s + " не определен ");

if (allowed && !(s in allowed)) // Убедиться, что это общедоступный символ

throw new Error("Module.importSymbols(): символ " + s +

" не является общедоступным " + "и не может быть импортирован.");

to[s] = from[s];

// Импортировать символ

}

};

// Эта функция используется модулями для регистрации одной

// или более функций инициализации.

Module.registerInitializationFunction = function(f) {

// Сохранить функцию в массиве функций инициализации

Module.\_initfuncs.push(f);

// Если обработчик события onload еще не зарегистрирован, сделать это сейчас.

Module.\_registerEventHandler();

}

// Данная функция вызывает зарегистрированные функции инициализации.

// В клиентском JavaScript она автоматически вызывается по окончании загрузки документа.

// В других контекстах исполнения может потребоваться вызвать эту функцию явно.

Module.runInitializationFunctions = function() {

// Запустить каждую из функций, перехватывая и игнорируя исключения,

// чтобы ошибка в одном модуле не помешала инициализироваться другим модулям.

for(var i = 0; i < Module.\_initfuncs.length; i++) {

try { Module.\_initfuncs[i](); }

catch(e) { /\* игнорировать исключения \*/}

}

// Уничтожить массив, т.к. такие функции вызываются всего один раз.

Module.\_initfuncs.length = 0;

}

// Частный массив, где хранятся функции инициализации для последующего вызова

Module.\_initfuncs = [];

// Если модуль был загружен веб броузером, эта частная функция регистрируется

// как обработчик события onload, чтобы иметь возможность запустить все функции

// инициализации по окончании загрузки всех модулей.

// Она не допускает обращение к себе более одного раза.

Module.\_registerEventHandler = function() {

var clientside = // Проверить хорошо известные клиентские свойства

"window" in Module.globalNamespace && "navigator" in window;

if (clientside) {

if (window.addEventListener) { // Регистрация по стандарту W3C DOM

window.addEventListener("load",Module.runInitializationFunctions,false);

}

else if (window.attachEvent) { // См. регистрацию в других браузерах

window.attachEvent("onload", Module.runInitializationFunctions); // IE5+

} else {

// IE4 и более старые броузеры, если тег <body> определяет атрибут onload,

// этот обработчиксобытия будет перекрыт и никогда не будет вызван.

window.onload = Module.runInitializationFunctions;

}

}

// Функция перекрывает сама себя пустой функцией,

// чтобы предотвратить возможность повторного вызова.

Module.\_registerEventHandler = function() {};

}



