**Лабораторная работа №2**

**JS: работа с объектами и функциями**

**Обязательные общие требования:**

1. При разработке проекта не используются вспомогательные решения типа jQuery, bootstrap и другие фреймворки и библиотеки, готовые решения
2. Исходный код и структура проекта должна соответствовать соглашению <https://github.com/lulldev/webdev/tree/master/coding-convections>

**Оценка работы:**

* **минимальный** балл за работу (удовлетворительно, работа зачтена): **60**
* для оценки **хорошо** нужно набрать **не менее 80 баллов**
* для оценки **отлично** нужно набрать **от 90 до 100 баллов**

**Задание 1. (60 баллов)**

Разработать функционал на JS для работы с учетной записью и авторизацией. Важно: на js разрабатывается только логика, интерфейс на html+css не требуется. Доработать функции:

* register()
* signIn()
* signOut()
* resetPassword()

Скопируйте и доработайте код:

|  |
| --- |
| // переменная для хранения состояния авторизованного пользователя  // при успешной авторизации должен содержать объект с пользовательскими данными  // при signOut должен становиться null  var authUserData = null;  var userDatabase = []; // массив с зарегистрированными пользователями  function register(email, password) {  // ваш код  // проверка на валидность email, пароля (6 символов, начинается с большой буквы, должен содержать как минимум 1 цифру)  // проверка нового пользователя в userDatabase  }  function signIn(email, password) {  // ваш код  // проверка на валидность email, пароля (6 символов, начинается с большой буквы)  // проверка наличия пользователя в userDatabase  // заполнение authUserData  }  function signOut() {  // ваш код  }  function resetPassword(email, oldPassword, newPassword) {  // функция восстановления пароля  // должна изменять пароль пользователя если старый пароль введен верно и новый пароль соответствует правилам формата пароля  }  function isAuth() {  // функция возвращает true если пользователь авторизован, false если нет  } |

**Задание 2. (40 баллов)**

Разработайте валидатор данных, используя подход цепочек вызовов (<https://learn.javascript.ru/task/chain-calls>). Валидатор должен обрабатывать строки, числа и массивы. Методы для валидации:

* **min**() / **max**() для типа Number - проверяет граничные значения
* **minLength**() / **maxLength**() - проверяет длину для string и array
* **equal**() - проверяет соответствие значению переданное через аргумент
* **isString**() - проверяет входное значение на тип string
* **isArray**() - проверяет входное значение на тип array
* **isNumber**() - проверяет входное значение на тип number
* **isEmail**() - проверяет string значения на email
* **isFloat**() - проверяет number значения на соответствие типу числа с плавающей точкой
* **isDate**() - проверяет string значение на соответствие формату dd.mm.yyyy

|  |
| --- |
| validator('1').isArray(); // false  validator('1').isString(); // true  validator('1').isNumber(); // false  validator(10).isNumber().min(10); // true  validator(10).isNumber().min(4).max(9); // false  validator([]).isArray().equal([1, 2, 3]); // false  validator([1, 2, 3]).isArray().equal([1, 2, 3]); // true  validator('user@m').isString().isEmail(); // false  validator('user@mail.ru').isString().isEmail(); // true  validator('25.12.1993').isDate(); // true  // и др. кейсы |