Лабораторная работа № 3

Основы Javascript

1. Изучение основных конструкций языка.
2. Работа с DOM.

Справочник: javascript.ru

Учебник: learn.javascript.ru

Пример 1 – *Калькулятор*

http://web.kts/examples/lab3/calc.html

```
<!doctype html>
<html>
<head>
    <meta charset=utf-8>
    <title>Пример 1. Калькулятор</title>

    <style>
        div {margin: 20px; border: 1px solid #eee; padding: 10px}
        body, input, select {font-size: 1.1em}
        input { padding: 5px; }
        #result { font-size: 1.5em; font-weight: bold }
    </style>

   </head>
<body>
<div>
    <input type="text" id="operand1">
    <select id="operator">
        <option value="+" selected="selected">+</option>
        <option value="-">-</option>
        <option value="*">*</option>
        <option value="/">/</option>
    </select>
    <input type="text" id="operand2">
    <input type="button" value="=" id="doCalc"> <span id="result"></span>
</div>
 <script>
    // вспомогательная функция для быстрого доступа к getElementById(id)
    const getID = (id) => document.getElementById(id);

    // получаем элементы для использования в дальнейшем
    const operator = getID('operator');
    const operand1 = getID('operand1');
    const operand2 = getID('operand2');
    const doCalc   = getID('doCalc');
    const resultContainer = getID('result');
```

```
    // обработчик события click
    doCalc.onclick = function () {
        // переменные для результата и операндов
        let result = null;
        let a = +operand1.value;
        let b = +operand2.value;

        switch(operator.value)
        {
            case '+':   result = a + b; break;
            case '-':   result = a - b; break;
            case '*':   result = a * b; break;
            case '/':   result = a / b; break;
        };

        // результат не будет числом (NaN), если не удастся преобразовать переменные
к Number
        if (isNaN(result)) result = 'Не удалось вычислить';
        resultContainer.innerHTML = result;
    }

    </script>

</body>
</html>
```

Пример 2. Идентификация платежной карты (прототип)
http://web.kts/examples/lab3/cards.html

```
<!doctype html>
<html>
<head>
    <meta charset=utf-8>
    <title>Пример 2. Банковские карты</title>

    <style>
    body { padding: 30px; }

    body, input, select {font-size: 1.6em}
    input { padding: 10px; }
    #cards img { opacity: 0.2 }
    #cards img.hl { opacity: 1 }
    #result { color: #888; }
    </style>

   </head>
<body>
<div>
    <div id="cards">
        <img src="img/visa.png" id="visa">
```

```
        <img src="img/mc.png" id="mc">
        <img src="img/visael.png" id="visael">
        <img src="img/maestro.png" id="maestro">
        <img src="img/jcb.png" id="jcb">
        <img src="img/mir.png" id="mir">
    </div>
    <input type="text" id="cardNumberText" value = '4300111122223333'>

    <input type="button" value="Проверить" id="cardCheck"><div id="result"></div>
</div>
<script>
    // вспомогательная функция для быстрого доступа к getElementById(id)
    const getID = (id) => document.getElementById(id);

    // получаем элементы для использования в дальнейшем
    const cardNumberText = getID('cardNumberText');
    const cardCheck  = getID('cardCheck');
    const cards  = getID('cards');
    const resultContainer = getID('result');

    // определяем новый метод для String - проверка "содержит только цифры"
    //
   //
   String.prototype.isNumber = function(){ return /^\d+$/.test(this); }

       // определяем обработчик input для очистки классов изображений карт
    cardNumberText.oninput = function () {
        clearClass();
    }

    // обработчик события click
    cardCheck.onclick = function () {
        let result;
        let card;
        // очищаем класс (изображения затенены)
        clearClass();

        // если номер только из цифр и его длина = 16, формируем сообщение
        if ( !cardNumberText.value.isNumber() || cardNumberText.value.length != 16 )
{
            result = 'Неверно введен номер карты';
        }
        else {
            result = "Номер карты правильный";
            // определяем вид карты
            card = getID(checkCardByNumber(cardNumberText.value));
            // устанавливаем для карты класс для нормального отображения
            card.className = 'hl';
              }
         // выводим результат проверки
         resultContainer.innerHTML = result;
    }

    // функция для проверки номера карты и определения ее вида
    const checkCardByNumber = function (cardNumber) {
```

```
        // сначала проверяем только первый символ
        if ( +cardNumber.slice(0, 1) == 4 ) {
            // visa
            return 'visa';
        }
        else if ( +cardNumber.slice(0, 1) == 5 ) {
            // mastercard
            return 'mc';
        }

    }
    // функция для очистки класса (вызываем при изменении в поле номера карты)
    const clearClass = function () {
        const cardImage = document.querySelector('#cards img.hl');
        // есть ли такой класс у картинки? если да, выключаем
        if ( cardImage ) cardImage.classList.toggle('hl');
    }
</script>

</body>
</html>
```

Задания.

1. Доработать калькулятор из примера 1 следующим образом: добавить функции «Квадратный корень» (Math.sqrt()) и «Логарифм» по любому основанию x>0 (функция Math.log(), предусмотреть появление нового поля для ввода основания при выборе операции «Логарифм») для первого операнда. При выборе этих операций запрещать ввод во второе поле (свойство disabled = true). Добавить чекбокс «Округлить до 3-х знаков» и соответствующий функционал.
2. Доработать пример 2 следующим образом:
2.1 Добавить обработку для остальных приведенных в примере видов карт (https://www.bincodes.com/bin-list/, карта «Мир» - 220).
2.2 Добавить проверку введенного номера карты на корректность путем вычисления контрольной суммы по алгоритму Луна (Luhn algorithm, https://ru.wikipedia.org/wiki/Алгоритм_Луна)
2.3 После проверки номера отформатировать его в поле ввода, добавив пробел между блоками из четырех цифр (пример: 4300 2201 3502 1102)
3. Реализовать цифровые часы для показа текущего времени в формате чч:мм:сс.
4. Доработать задачу 3, реализовав часы для трех разных часовых поясов (Москва, Токио, Лондон).
5. Реализовать цифровой секундомер в формате чч:мм:сс:мс с кнопками Старт и Стоп.
6. Строка содержит дату и время в чч-мм-гггг чс:мин:сек (например, 11-03-2019 10:15:01). Определите содержит ли строка данные в заданном формате и корректны ли эти данные.

Контрольные примеры:

Строка	**Вывод**
11-03-2019 10:15:01	Формат корректен, данные корректны
11032019 10:15:01	Формат некорректен, данные не проверяются
11-03-2019 1015:01	Формат некорректен, данные не проверяются
29-02-2019 10:15:01	Формат корректен, данные некорректны
28-02-2019 10:99:01	Формат корректен, данные некорректны

7. Определите уровень сложности пароля (пример алгоритм определения сложности по 10 балльной шкале в таблице, баллы от 0 до 10). При проверке выводить подошедшие критерии, баллы за них и общую сумму баллов.

Номер	**Критерий**	**Балл**
1	Длина менее 5	0, отменяет остальные критерии
2	Длина 5-9	+1
3	Длина 10-12	+2
4	Длина свыше 12	+3
5	Символы (цифры, буквы, иные символы*)	+1 за каждый дополнительный вид, не более 2 баллов
6	Регистр букв	+1 за дополнительный регистр, не более 1 балла
7	Иные символы более 30% длины (не буквы и цифры)	+2
8	Все буквенные или цифровые сочетания (буквы или цифры подряд) длиной менее 3	+2

* Использовать набор: ~!#$%^&*()_-+=?/,.[]{}<>|\

Примеры расчета

Пароль	**Критерии**	**Оценка**
shdj	1	0
Sh1#	1	0
Shdj	1	0
ShdJ*	2, 5(1), 6	3
ShdJ129&yu	3, 5(2), 6	5
She*26^0p%?1(	4, 5(2), 6, 7	8
Sh*26^0p%?18(	4, 5(2), 6, 7, 8	10