**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ**

**КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Поволжский государственный университет телекоммуникаций и информатики»**

**КОЛЛЕДЖ СВЯЗИ**

|  | УТВЕРЖДАЮ:  Зам директора по ИПО  \_\_\_\_\_\_\_\_\_\_\_Ходотова Е.А  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г. |
| --- | --- |

**СБОРНИК**

**практических занятий**

по МДК.В.01.03

**Веб - программирование**

**Часть 1**

для специальности: 09.02.03 – Программирование в компьютерных системах

Номера занятий: **№ 1 – 16**

Сборник рассчитан на 36 часов

Составлен преподавателем Шомас Е.А.

| Рассмотрено на заседании П(Ц)К  «Информационные системы и технологии»  Председатель \_\_\_\_\_\_\_\_\_\_\_\_ Шомас Е.А.  Протокол №\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_2022г. |  |
| --- | --- |

Самара

2022г.

**ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ**

| **Общие компетенции** | |
| --- | --- |
| **ОК 1.** | Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес; |
| **ОК 2.** | Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество; |
| **ОК 3.** | Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность; |
| **ОК 4.** | Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития; |
| **ОК 5.** | Использовать информационно – коммуникационные технологии в профессиональной деятельности; |
| **ОК 6.** | Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями; |
| **ОК 7.** | Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий; |
| **ОК 8.** | Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации; |
| **ОК 9.** | Ориентироваться в условиях частой смены технологий в профессиональной деятельности. |
| **Профессиональные компетенции** | |
| **ПК 1.1** | Выполнять разработку спецификаций отдельных компонент |
| **ПК 1.2** | Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля |
| **ПК 1.3** | Выполнять отладку программных модулей с использованием специализированных программных средств |
| **ПК 1.4** | Выполнять тестирование программных модулей |
| **ПК 1.5** | Осуществлять оптимизацию программного кода модуля |
| **ПК 1.6** | Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций |

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1**

**НАИМЕНОВАНИЕ**: Создание простого сценария JavaScript

1. **ЦЕЛЬ**: Составление простых сценариев JavaScript и внедрение их в HTML – код, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6
2. **ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Назначение языка JavaScript». Основные понятия и определения» и ответить на следующие вопросы:

2.1 Назначение языка JavaScript?

2.2 Где можно разместить сценарии JavaScript в HTML – документе?

2.3 Какие ключевые слова можно использовать для определения переменных?

1. **ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

1. **ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

1. **ЗАДАНИЕ**:

5.1 Используя метод prompt(), узнайте у пользователя его имя, а далее с помощью метода alert() поприветствуйте его.

5.2 Измените сценарий таким образом, чтобы приветствие выводилось в окне браузера.

5.3 Дана переменная со значением '15.5px' и переменная со значением '9.25px'. Найдите сумму пикселей из этих переменных и выведите ее на экран. Модифицируйте результат таким образом, чтобы сумма выводилась в окно браузера с единицей измерения – px. Узнайте тип данных у результата суммирования.

5.5 Создание калькуляторов фигур:

**Вариант** 1. Создайте калькулятор, который будет находить площадь и периметр прямоугольника по введенным пользователем данным. (Используем метод prompt()). Результат выводим в окне браузера.

**Вариант** 2. Создайте калькулятор, который будет находить площадь круга и длину окружности по введенным пользователем данным. (Используем метод prompt()). Результат выводим в окне браузера.

**Вариант** 3. Создайте калькулятор, который будет находить площадь треугольника по трем сторонам. (Используем метод prompt()). Результат выводим в окне браузера.

1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Сначала необходимо объявить переменную и присвоить ей значение, вводимое с помощью метода prompt():

переменная = prompt();

Для вывода результата в диалоговом окне используйте метод alert(), для вывода результата в окне браузера конструкция document.write("Привет, " +***ваша переменная*** + ". Спасибо, что зашли.")

6.2 Тип данных можно узнать с помощью функции typeof ().

6.3 Для преобразования строки в число используются функции:

* parseInt() – функция принимает строку и преобразовывает ее в целое число;
* parseFloat() – функция возвращает числа с плавающей точкой.

1. **СОДЕРЖАНИЕ ОТЧЕТА**:
   1. Наименование и цель работы
   2. Задание, под каждым поместить код программы и полученные результаты в виде скринов.
   3. Ответы на контрольные вопросы
   4. Вывод о проделанной работе
2. **КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Какие методы ввода/вывода информации используются в JavaScript?

8.2 Как сценарий JavaScript внедряется в HTML – документ?

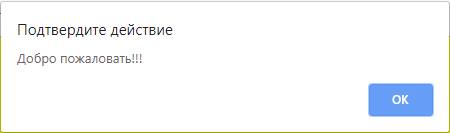
8.3 К каким языкам программирования относится JavaScript?

**ПРИЛОЖЕНИЕ**:

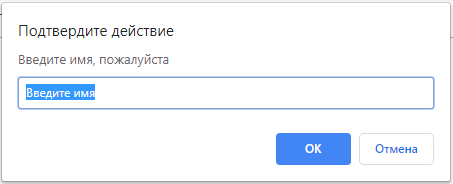
**JavaScript** — это объектно-ориентированный язык программирования, предназначенный для написания сценариев, работающих как на стороне клиента, так и на стороне сервера.

Рассмотрим основные методы ввода/вывода информации в сценариях JavaScript:

Метод **alert** () применяется для вывода на экран простейшей диалоговой панели, отображающей какое – либо сообщение. После вызова этого метода выполнение сценария задерживается до тех пор, пока пользователь не нажмет кнопку ОК в окне с сообщением. Синтаксис: alert (“Сообщение”)



Если вам необходимо получить от пользователя одну текстовую строку, то можете применить метод **prompt** (). Он отображает на экране диалоговую панель, в которой есть одно текстовое поле ввода и кнопка ОК. Когда пользователь нажимает эту кнопку, метод возвращает введенную строку.



Код на JavaScript встраивается в веб – страницу вместе с кодом HTML. Его присутствие обозначается тегами <script> </script>. Как и HTML – код, сценарий на JavaScript выполняется с верхней строки вниз и обрабатывается выражение за выражением (пошагово).

Кроме размещения текста сценария в теле HTML – документа его можно поместить в отдельный файл. Это должен быть отдельный текстовый файл, с расширением **\*.js**. В этом файле не должно быть ничего кроме операторов JavaScript. В тексте HTML - документа содержимое контейнера <script> приведите к следующему виду: <script src=”\*.js”></script >

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2**

**НАИМЕНОВАНИЕ**: Создание электронной визитной карточки

1. **ЦЕЛЬ**: Использование функции prompt и возможностей динамического построения тела HTML – документа при помощи конструкции document.write, формирование ОК 2, ОК 4, ОК 5, ОК 8, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6
2. **ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Назначение языка JavaScript. Основные понятия и определения» и ответить на следующие вопросы:

2.1 Какие группы операторов существуют в языке JavaScript?

2.2 Каким образом можно осуществить взаимодействие пользователей с веб ресурсом средствами языка JavaScript?

1. **ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

1. **ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

1. **ЗАДАНИЕ**:

5.1 Создать электронную визитную карточку с указанием своих

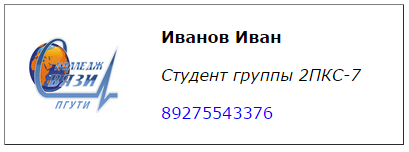
- имени и фамилии;

- статуса;

- номера телефона;

- логотип КС ПГУТИ (можно использовать другой логотип)

по следующему образцу:

******

5.2 Добавьте свой e-mail;

5.3 Самостоятельно произвести оформление визитной карточки средствами CSS

1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**
   1. Сначала создается статический HTML – макет в текстовом редакторе. Создавать макет можно через таблицу или блок div. Ширина таблицы (блока) будет 400px.
   2. Для управления рамками таблицы и оформления текста воспользуйтесь средствами CSS.
   3. Далее происходит динамическая верстка тела HTML – документа при помощи конструкции document.write (пример использования конструкции можно посмотреть в приложении)
   4. Вводим переменные для обозначения имени и фамилии, статуса, телефона и e-mail. Присваиваем значения переменных к функции prompt()
   5. Тестируем свою программу
2. **СОДЕРЖАНИЕ ОТЧЕТА:** 
   1. Наименование и цель работы
   2. Задание, под каждым поместить код программы и полученные результаты в виде скринов.
   3. Ответы на контрольные вопросы
   4. Вывод о проделанной работе
3. **КОНТРОЛЬНЫЕ ВОПРОСЫ:**
   1. Каким образом можно осуществить динамическое формирование содержимого HTML – документа?
   2. Каким образом осуществляется отладка сценария JavaScript?

**ПРИЛОЖЕНИЕ**:

Пример динамической верстки тела HTML – документа при помощи конструкции document.write приведен в следующем листинге:

<script>

document.write('<table width="400" border="1">');

document.write('<tr><td width="150">');

document.write('<img src="logo.jpg"></td>');

document.write('<td>');

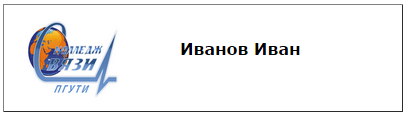
document.write('<b>' + name + '</b><br><br>');

document.write('</td></tr>');

document.write('</table>');

</script>

В данном примере формирование визитной карточки осуществляется через таблицу. Результат:



Чтобы осуществлять запрос на вывод фамилии и имени введите в свою программу функцию prompt().

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3**

**НАИМЕНОВАНИЕ**: Использование условных операторов в JavaScript

1. **ЦЕЛЬ**: Создание сценариев с использованием условных операторов, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6
2. **ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Условные конструкции в JavaScript» и ответить на следующие вопросы:
   1. Какие условные операторы вам известны?
   2. В каких случаях каждый из условных операторов применяется?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

5.1*.* Создайте сценарий JavaScript который находил бы значение функции в определенной точке:

| **№ варианта** | **Задание** |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

5.2 С помощью оператора **switch** создайте сценарий JavaScript, в котором определенный цвет на русском языке соответствовал этому же цвету на английском языке.

5.3 Измените предыдущее задание таким образом, чтобы в соответствии с заданным цветом изменялся фон документа.

5.4 В зависимости от введенного пользователям числа (четного или нечетного), цвет фона должен окрашиваться в разные цвета (всего два цвета).

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 ***Порядок выполнения первого задания:***

6.1.1 Сначала объявляются переменные x и y.

6.1.2 С помощью метода **prompt ()** запросите значение х.

6.1.3 Преобразуйте введенную строку в число.

6.1.4 Воспользовавшись оператором **if** или **if…else** определите значение функции;

6.1.5 С помощью метода **alert()** выведите значение функции

6.2 ***При решении второго задания определяйте не менее шести цветов;***

6.2.1 Сначала объявляется переменная\_1;

6.2.2 Объявляется переменная\_2, в которой с помощью метода **prompt ()** запрашивается название цвета

6.2.3 Прописываем соответствие цветов (переменная\_2 (название цвета на английском языке): переменная\_1 (название цвета на русском языке))

6.2.3 Изменение фона документа осуществляется через свойство **document.body.style.background = цвет;**

6.3 Для определения четности/нечетности числа используйте оператор деления по модулю - %.

**7. СОДЕРЖАНИЕ ОТЧЕТА**:

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

7.3 Ответы на контрольные вопросы

* 1. Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Синтаксис оператора if…else

8.2 Синтаксис оператора switch

8.3Назначение и принципиальное отличие операторов break и continue

**ПРИЛОЖЕНИЕ**:

**Условные операторы**

В процессе создания сценариев обычно требуется изменить порядок выполнения кода. Оператор **if** управляет последовательностью выполнения команд. Синтаксис:

if (условие) {

блок команд;

}

Сначала вычисляется логическое выражение, затем, если оно, верно, выполняются операторы, если оно не верно, то операторы пропускаются, и продолжается выполнение сценария.

Оператор **if…else** используется, когда необходимо задать действия, которые выполняются, если логическое выражение не верно. Синтаксис:

if (условие) {

операторы 1;

}

else {

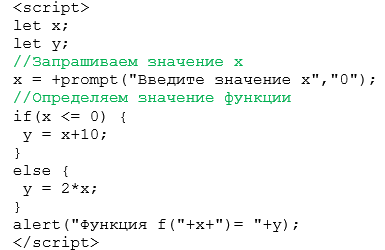
операторы 2;

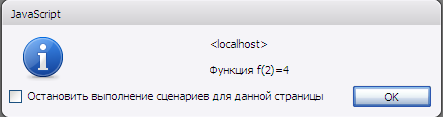
}

**Пример**: Функция f(x) определяется следующим образом:



**Сценарий**





Оператор **switch** позволяет сравнивать значение с множеством других. Этот оператор дает возможность предусмотреть операторы по умолчанию, которые выполняются, если не найдено ни одного соответствия.

switch (выражение)

{

case значение1:операторы 1;

break;

case значение2:операторы 2;

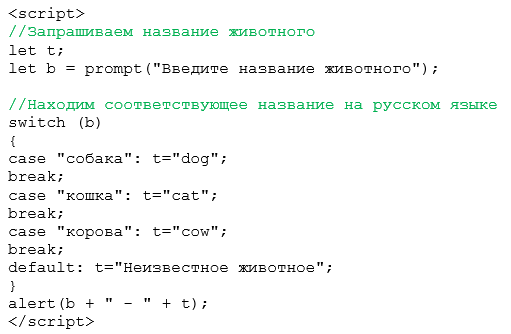
break;

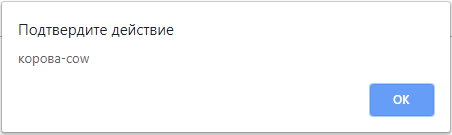
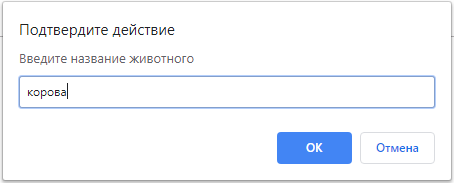
…

default:операторы;

}

Рассмотрим пример сценария, который будет переводить названия животных с английского языка на русский:





Определить четность/нечетность числа можно с использованием оператора деления по модулю:



**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4**

**НАИМЕНОВАНИЕ**: Использование циклов в JavaScript

**1. ЦЕЛЬ**: Применение операторов цикла в сценариях JavaScript, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Операторы цикла JavaScript» и ответить на следующие вопросы:

* 1. Какие операторы цикла вам известны?
  2. В каких случаях каждый из операторов применяется?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

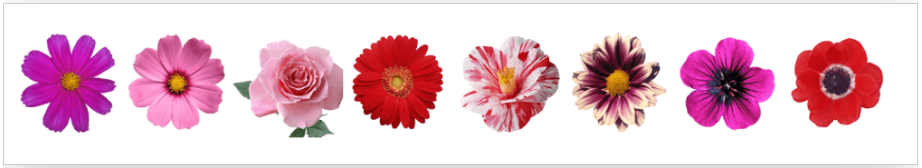
**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

* 1. Создайте документ HTML с заголовком «Сюрприз». Фон документа должен быть белым. Потом с помощью JavaScript досчитайте до 5 (можно использовать метод alert() для подсчета, либо использовать таймер для задержки), на этом этапе цвет фона меняется на желтый и появляется текст: «Скоро будет еще один цветной сюрприз…». Снова досчитайте до 5, и измените цвет фона.
  2. Запросите у пользователя запрос: «Сколько раз пожелать вам доброго дня?». Ограничимся предельным значением – 25. Пожелайте доброго дня столько количество раз, сколько запросил пользователь. В случае числа больше 25, выводится сообщение об ошибке.
  3. Вывести с помощью цикла 8-10 подобных изображений (цветы, автомобили, игрушки, чашки и т.д.), например, так:



1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**
   1. Для выполнения первого задания воспользуйтесь оператором **for**.
   2. Для выполнения второго задания воспользуйтесь оператором **while**.
2. **СОДЕРЖАНИЕ ОТЧЕТА:**
   1. Наименование и цель работы
   2. Задание, под каждым поместить код программы и полученные результаты в виде скринов.
   3. Ответы на контрольные вопросы
   4. Вывод о проделанной работе
3. **КОНТРОЛЬНЫЕ ВОПРОСЫ:**
   1. Для каких целей в сценариях используют счетчики?
   2. Чем отличаются операторы while от do…while?

**ПРИЛОЖЕНИЕ**:

Для многократного выполнения кода используют операторы цикла. Кроме того, циклы предоставляют удобные средства для манипулирования массивами.

Оператор **for** служит для создания цикла. Синтаксис:

for (выражение инициализации; выражение условия; выражение цикла)

{

операторы

}

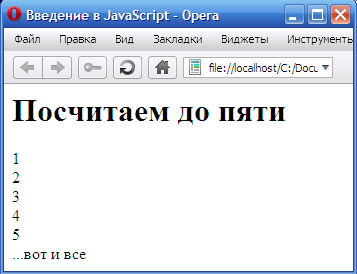
**Выражение инициализации** служит для задания начального значения счетчика цикла. **Выражение условия** позволяет прекратить выполнять цикл, когда условие перестанет выполняться. **Выражение цикла** обычно осуществляет инкремент или декремент счетчика цикла. Любое из этих выражений может быть пропущено, но соответствующая точка с запятой должна ставиться.

**Пример**: Напишем программу, выводящую на страницу числа от 1 до 5.

for (i=1;i<=5;i++) {

document.write (i+'<br />');

}



Оператор **while** сходен с оператором for, но он не производит инициализацию и инкремент счетчика в своем объявлении. Синтаксис:

while (выражение условия) {

операторы;

}

Если **выражение условия** в цикле while сразу ложно, то операторы не выполнятся ни разу.

Оператор **do…while** практически идентичен оператору while, но, поскольку в нем проверка условия осуществляется в конце, он гарантирует выполнение операторов, по крайней мере, один раз. Синтаксис:

do {

операторы;

} while (выражение условия)

**Пример:** Пожелаем Доброго вечера три раза

<script>

let m=3;

let n=1;

while (n <= m){

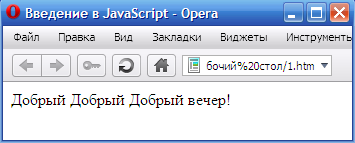
document.write("Добрый ");

n=n+1;

}

document.write("вечер!")

</script>



**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 5**

**НАИМЕНОВАНИЕ**: Работа с простыми объектами JavaScript

**1. ЦЕЛЬ**: Использование встроенных объектов (дата, время, строка) JavaScript на веб - страницах, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Встроенные объекты языка JavaScript» и ответить на следующие вопросы:

2.1 Какой объект применяется для работы с датой и временем?

2.1 Перечислите известные вам методы для получения данных о дате и времени

1. **ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

1. **ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

1. **ЗАДАНИЕ**:
   1. Напишите сценарий, который будет определять текущую дату и время и выводить ее в формате: **В Самаре** 2 февраля 2020, среда

**Время** 11:45:31

5.2 Поиск и замена: дана строка **aaa@bbb@ccc**. Необходимо заменить все @ на ! (используйте метод replace)

5.3 Преобразование формата даты: в переменной date лежит дата в формате **2020-02-18**. Преобразуйте эту дату в формат **2020:02:18** (используйте метод split)

5.4 Работа с регистром символов: дана строка – **Я изучаю JavaScript!**. Преобразуйте строку в верхний регистр, а затем в нижний регистр.

5.5 Во фразе – **Я изучаю JavaScript!** найдите количество символов.

5.6 Извлеките слово **изучаю** из предыдущей фразы с помощью методов **substr,** **substring** и **slice** и выведите ее на страницу

5.7 Раскодируйте значение строки, используя метод **fromCharCode()**  и выведите ее на страницу:

(1052)(1086)(1089)(1082)(1074)(1072)

5.8 После того как строка будет раскодирована закодируйте ее обратно, используя метод **charCodeAt()** и сравните свой результат с данными задачи.

* 1. Закодируйте слово САМАРА

1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1.1 При выполнении задания необходимо подобрать соответствующие методы для объекта Date. Методы представлены в таблице Приложения.

6.1.2 Для представления в нужном нам формате названия месяца и дня недели необходимо применить оператор switch.

….

switch (mon){

case 0: s="января"; break;

case 1: s="февраля"; break;

….

1. **СОДЕРЖАНИЕ ОТЧЕТА:**
   1. Наименование и цель работы
   2. Задание, под каждым поместить код программы и полученные результаты в виде скринов.
   3. Ответы на контрольные вопросы
   4. Вывод о проделанной работе
2. **КОНТРОЛЬНЫЕ ВОПРОСЫ:**
   1. В чем отличие метода getDate от setDate?
   2. Какое значение будет выведено в окне браузера при использовании следующей строки:

let myDate=new Date();

**ПРИЛОЖЕНИЕ**:

В JavaScript для работы с датой и временем применяется встроенный объект Date. Этот объект не имеет свойств, но обладает несколькими методами, позволяющими устанавливать и изменять дату и время.

Объект Date создается с помощью оператора new и конструктора – Date. Например,

**let date = new Date();**

значение переменной **date** будет текущая дата и время. Методами объекта Date можно получить отдельно значения месяца, дня недели, часов, минут и секунд:

| **Метод** | **Действие** |
| --- | --- |
| getDate | Возвращает число в диапазоне от 1 до 31, представляющее число месяца |
| getHours | Возвращает час суток в диапазоне от 0 (полночь) до 23 |
| getMinutes | Возвращает минуты в диапазоне от 0 до 59 |
| getSeconds | Возвращает секунды в диапазоне от 0 до 59 |
| getMillieconds() | Возвращает миллисекунды в диапазоне от 0 до 59 |
| getDay | Возвращает день недели, как целое число от 0 (воскресенье) до 6 (суббота) |
| getMonth | Возвращает номер месяца в году, как целое число от 0 (январь), до 11 (декабрь) |
| getFullYear | Возвращает год в виде четырех цифр |

Предположим мы хотим написать сценарий, который будет определять текущее время и выводить его в формате «чч:мм:сс»

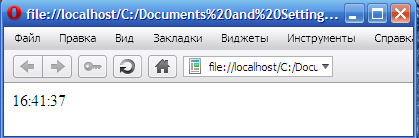
let t = new Date();

let h = t.getHours();

let m = t.getMinutes();

let s = t.getSeconds();

document.write(h+ ":" +m+ ":" +s);



**Объект String –** предназначендля создания строк символов и выполнения операций над ними. Он выступает в качествеконтейнера для всех функций, обрабатывающих текстовые строки. Объект String имеет всего одно свойство **length**, содержит длину строки в символах.

Чтобы создать строковый объект можно воспользоваться конструктором newString.

Например, let s = newString(“Итого:”);

Если применить свойство объекта: s.length, результатом будет число 6.

| **Метод** | **Действие** |
| --- | --- |
| charAt () | Возвращает символ находящийся на указанной позиции в строке |
| charCodeAt () | Возвращает цифровой код (Юникод) символа на указанной позиции в строке |
| concat () | Объединяет две или более строки |
| fromCharCode() | Преобразует Юникод в символ |
| slice () | Извлекает часть существующей строки и возвращает новую строку |
| indexOf () | Возвращает позицию первого найденного совпадения выражения в методе со строкой текста |
| lastIndexOf () | Возвращает позицию последнего найденного совпадения выражения в методе со строкой текста |
| split () | Разделяет строку на массив подстрок |
| toUpperCase () | Преобразует строку символов в верхний регистр |
| toLowerCase () | Преобразует строку символов в нижний регистр |
| replase() | Предназначен для поиска подстроки в строке и замены найденных фрагментов другой подстрокой |

Рассмотрим на примерах некоторые методы объекта **String**.

Синтаксис:

строка.**substr**(начало, количество)

let str = 'Европа — одна из шести частей света.';

document.write(str.substr(9,4) + '<br>');// одна

Синтаксис: строка.**substring**(начало, конец)

let str = 'Европа — одна из шести частей света.';

document.write(str.substring(8,22) + '<br>'); // одна из шести

С помощью метода **fromCharCode** Вы можете преобразовывать произвольные значения Юникода в соответствующие им символы. Синтаксис:

String.fromCharCode(Юникод1, Юникод2, … ЮникодN)

document.write(String.fromCharCode(1070,1085,1080,1082,1086,1076));//

В браузере отобразиться – Юникод

Метод **charCodeAt** возвращает Юникод символа на указанной позиции в строке. Первый символ в строке имеет позицию 0, второй 1 и т.д. Синтаксис: строка.charCodeAt(позиция)

let str1 = new String('Я короткая строка!');

document.write(str1.charCodeAt(4)+'<br>');// 1088

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 6**

**НАИМЕНОВАНИЕ**: Работа с массивами JavaScript

**1. ЦЕЛЬ**: Составление сценариев JavaScript, позволяющих работать с элементами массива. Ознакомление с многомерными массивами, массивом images. Формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Встроенные объекты JavaScript» и ответить на следующие вопросы:

2.1 Какие способы объявления массива вы знаете?

2.2 Каков индекс первого элемента массива?

2.3 Как узнать длину массива?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

**5.1 Работа с простыми массивами:**

5.1.1 Создайте два массива:

*mass1 = ['Банан', 'Яблоко', 'Груша', 'Апельсин', 'Манго', 'Свекла', 'Капуста', 'Морковь', 'Огурец', 'Томат']*

*mass2 = ['Тигр', 'Волк', 'Лев', 'Медведь', 'Рысь', 'Синица', 'Филин', 'Ворона', 'Воробей', 'Сорока']*

Выведите их на экран браузера.

5.1.2 Из первого массива удалите все овощи и сохраните новый массив с названием **fruits**. Из второго массива удалите всех животных, не являющимися птицами, сохраните получившийся массив под названием **bird**.

5.1.3 Соедините полученные массивы (**mass3**) в один и выведите их на экран.

5.1.4 Узнайте длину массивов

5.1.5 Добавьте в начало полученного массива - **Мандарин**, а в конец – **Соловей**.

5.1.6 Узнайте новую длину массива

5.1.7 Измените порядок следования элемента массива на противоположный и выведите результат на экран

5.1.8 Отсортируйте массив и выведите результат на экран.

5.1.9 В массив **fruits** после каждого фрукта добавьте слово - **съел**

5.1.10 В массиве **bird** замените Ворону и Воробья на Галку и Голубя. Выведите полученный массив.

* 1. **Работа с многомерными массивами**
     1. Сформируйте многомерный массив – Времена года, по следующему образцу:



Примерный код программы можно посмотреть в приложении.

* 1. **Работа с массивом images**
     1. Создайте массив с изображениями (шесть различных изображений). С помощью метода prompt() запросите у пользователя, какой рисунок загрузить в браузере. Место для загрузки изображения - 400х250. Примерный код программы представлен в приложении.
  2. **Работа с объектами (ассоциативными массивами)**
     1. Создайте пустой объект **worker**.
     2. Добавьте свойство **name** с любым значением имени работника
     3. Добавьте свойство **surname** с любым значением фамилии работника
     4. Добавьте возраст (**age**) и размер заработной платы (**salary**) с внесением данных.
     5. Выведите в окно браузера все данные о работнике, каждое с новой строки.
     6. Измените имя сотрудника
     7. Удалите возраст сотрудника
     8. Создайте метод, который будет возвращать заработную плату сотрудника в окне браузера.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия;

* 1. Дать ответы на контрольные вопросы;

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

7.3 Ответы на контрольные вопросы

* 1. Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Определение и назначение массива?

8.2 Какие способы сортировки массива вы знаете?

8.2 Каков формат обращения к n-му элементу массива?

**ПРИЛОЖЕНИЕ**:

**Массив** – это переменная, в которой в упорядоченном виде можно хранить целый набор значений.

***Создание массива:***

let arr **=** ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ', 'ВС'];

***Вывод отдельного элемента массива***

let arr **=** ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ', 'ВС'];

alert(arr[2]);// СР

**Нумерация элементов массива начинается с 0!**

***Замена или добавление элемента в массив*** осуществляется номеру (индексу или ключу):

arr[2] = 'ПТ'; // вместо среды будет пятница

arr[7] = 'ВС'; // после воскресенья добавится еще одно воскресенье

***Общее число элементов массива*** можно узнать через свойство **length**:

let arr = ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ', 'ВС'];

alert(arr.length); // 7

***Вывести массив целиком*** можно через **alert**:

let arr = ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ', 'вс'];

alert (arr);

**Методы для работы с массивами**

1. **Метод** **push** – добавляет неограниченное количество элементов в конец массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Пример**: добавим два новых элемента в конец массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ "];

arr.push("СБ ", "ВС ");

document.write(arr); // ПН ,ВТ ,СР ,ЧТ ,ПТ ,СБ ,ВС

Результатом выполнения кода будет новый массив из 7 элементов.

С помощью метода **push** можно заполнить числовой массив.

**Пример**: заполнить массив числами от 1 до 9:

let arr = [];

for (let i = 1; i <= 9; i++){

**arr.push(i);**

}

document.write(arr); // 1,2,3,4,5,6,7,8,9

1. **Метод unshift** – добавляет неограниченное количество новых элементов в начало массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Пример**: добавим два новых элемента в начало массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ "];

arr.unshift("СБ ", "ВС ");

document.write(arr); // СБ ,ВС ,ПН ,ВТ ,СР ,ЧТ ,ПТ

1. **Метод** **shift** – удаляет первый элемент из массива.

**Пример**: удалим первый элемент из массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

arr.shift();

document.write(arr); // ВТ ,СР ,ЧТ ,ПТ ,СБ ,ВС

1. **Метод pop** – удаляет последний элемент из массива.

**Пример**: удалим последний элемент из массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

arr.pop();

document.write(arr); // ПН ,ВТ ,СР ,ЧТ ,ПТ ,СБ

1. **Метод splice** – позволяет **удалять, добавлять и производить замену** элементов в массиве.

***Удаление элементов***: в методе указывается два числовых параметра (1 – с какого элемента начать удаление, 2 – количество удаляемых элементов.

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

arr.splice(2, 2);

document.write(arr); // ПН ,ВТ ,ПТ ,СБ ,ВС

***Добавление элементов***: чтобы добавить элементы в массив с помощью метода splice также необходимо использовать параметры (1 – число перед каким элементом массива следует начинать добавление, 2 – число 0. После этого указываем элементы, которые необходимо вставить в указанную позицию).

let arr = ["ПН ", "ВТ ", "ПТ ", "СБ ", "ВС "];

arr.splice(2, 0, "СР ", "ЧТ ");

document.write(arr); // ПН ,ВТ ,СР ,ЧТ ,ПТ ,СБ ,ВС

***Замена элементов*** осуществляется путем удаления элемента, а затем добавления новых.

let arr = ["ПН ", "ВТ ", "ПТ ", "СБ ", "ВС "];

arr.splice(2, 2, "Среда ", "Четверг ");

document.write(arr); // ПН ,ВТ ,Среда ,Четверг ,ВС

1. **Метод reverse** – меняет порядок элементов в массиве

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

arr.reverse();

document.write(arr); // ВС ,СБ ,ПТ ,ЧТ ,СР ,ВТ ,ПН

1. **Метод sort** – сортирует элементы массива (не работает с числами)

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

arr.sort();

document.write(arr); // ВС ,ВТ ,ПН ,ПТ ,СБ ,СР ,ЧТ

1. **Метод join** – устанавливает разделители между элементами массива

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

document.write(arr.join("- ")); // ПН - ВТ - СР - ЧТ - ПТ - СБ – ВС

1. **Метод concat** – позволяет объединить массивы

let arr1 = ["ПН ", "ВТ ", "СР "];

let arr2 = ["ЧТ ", "ПТ ", "СБ ", "ВС "];

document.write(arr1.concat(arr2)); // ПН ,ВТ ,СР ,ЧТ ,ПТ ,СБ ,ВС

1. **Метод slice** – копирует часть элементов массива и возвращает новый массив, содержащий скопированные элементы. Исходный массив при этом не меняется

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ ", "СБ ", "ВС "];

document.write("Будни - " +arr.slice(0, 5)); // Будни - ПН ,ВТ ,СР ,ЧТ ,ПТ

**Перебор элементов**

Старый способ перебора элементов массива – это через цикл for:

let arr = ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ'];

for (let i = 0; i < arr.length; i++){

document.write(arr[i] + " "); // ПН ВТ СР ЧТ ПТ СБ

}

Для массива возможен другой (более современный) вариант цикла, **for…of**:

let arr = ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ'];

for (let ar of arr){

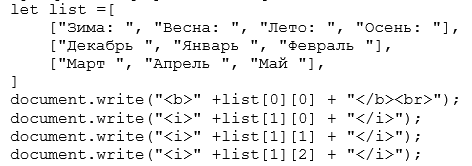
document.write(ar+ " "); // ПН ВТ СР ЧТ ПТ СБ

}

Для второго случая перебор элементов в массиве будет зависеть от названия массива (**ar of arr**, **если массив называется** items, то **item of items**, если это массив с изображениями, то **image of images).**

**Многомерные массивы:**

Многомерные массивы в JavaScript — это массивы, содержащие внутри себя другие массивы. Примерный код программы многомерного массива:



**Вид в браузере:**



**Массив images**

Каждый объект img на странице хранится в массиве images[]. Этот массив – свойство объекта Document. Первый элемент на странице находится в элементе document.images[0], второе – в document.images[1] и т.д.

При желании можно сделать так, чтобы переменная ссылалась на объект img в массиве images [] – так будет легче читать код. Например, если вы напишите:

let myImage = document.images[1];

Переменная **myImage** будет содержать ссылку на объект img внутри массива images[] с индексом 1. Теперь вы можете писать myImage вместо document.images[1] в вашем коде – результат будет тот же.

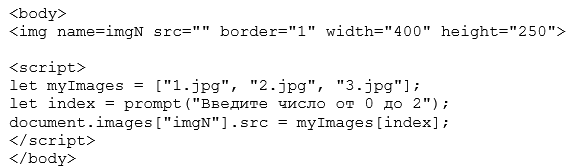
Вы можете также получить доступ к объектам img в массив images по имени. Например, к объекту img, созданному тегом <img>, название которого myImage, можно получить доступ через свойство images объекта Document, которое является массивом, следующим образом:

document.images["myImage"]

Поскольку свойство document.images – это массив, у него, в свою очередь, есть свойства встроенного объекта JavaScript Array – такие как length. Например, если вы хотите узнать, сколько изображений находится на странице, об этом сообщит код document.images.length.

Объект img имеет множество полезных свойств. Самое важное из них – src. Изменяя его, вы можете изменить загруженное изображение.

**Пример:**



**Объекты** используются для хранения коллекций различных значений и более сложных сущностей. Объекты создаются с помощью {…} с необязательным списком свойств. Можно сказать, что объекты – это ассоциативные массивы.

**Свойства** – это пара ***«ключ: значение»***, где ***ключ*** – это строка (имя свойства), а ***значение*** может быть, чем угодно.

**Создание объекта:**

let user = {

name: "John",

age: 30

};

У каждого свойства есть ключ (имя или идентификатор). После него ставится двоеточие, а затем указывается значение свойства. Если в объекте несколько свойств, то они перечисляются через запятую.

Для обращения к свойствам используют запись «через точку»:

document.write(user.name);

document.write(user.age);

**Операции с объектами**

Основные операции с объектами – это добавление новых свойств, изменение уже существующих, удаление свойств и обращение к свойствам.

Добавить новое свойство в объект можно присвоив свойству значение. Чтобы присвоить свойству значение, к нему нужно получить доступ. Для доступа используется оператор «точка» и квадратные скобки [ ].

user.surname = "Ivanov"; // Добавили новое свойство

user["patronymic"] = "Ivanovich"; // Добавили новое свойство

***Обращение к свойству и изменение значения*** осуществляется также через точку или [].

document.write(user.name +" "); // Обращение к свойству

document.write(user["patronymic"]); // Обращение к свойству

user.age = 25; // Изменение значения

***Для удаления свойства*** используют оператор delete:

delete user.age;

***Для перебора свойств объекта*** используется цикл for in:

for (let prop in user) {

alert(prop);

}

**Методы объекта**

Свойство, значением которого является функция, называется ***методом***. Вызов метода осуществляется точно также, как и вызов обычной функции – с помощью оператора ().

let user = {

name: "Ivan",

age: 30,

sayHi: function(){

alert("Hello!");

}

};

user.sayHi(); // Hello!

***Для доступа к свойствам объекта внутри метода*** используется ключевое слово **this**. Оно содержит ссылку на объект, с помощью которого был вызван метод:

let user = {

name: "Ivan",

sayName: function(){

alert(this.name);

}

};

user.sayName(); // Ivan

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 7**

**НАИМЕНОВАНИЕ**: Создание пользовательских функций

1. **ЦЕЛЬ**: Создание собственных функций и внедрение их в сценарий, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Функции JavaScript» и ответить на следующие вопросы:

1. Какие функции существуют и используются в сценариях JavaScript?

2. Какие функции JavaScript вы использовали на практических занятиях?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Создайте функцию, которая возвращает квадрат числа. Число передается параметром и вводится через окно Запроса (метод prompt)

5.2 Сделайте функцию, которая возвращает сумму двух чисел. Числа вводить через Запрос пользователя

5.3 Сделайте функцию, которая отнимает от первого числа второе и делит на третье. Числа вводить через Запрос пользователя

5.4 Сделайте функцию, которая принимает параметром число от 1 до 7, а возвращает день недели на русском языке

5.5 Напишите функцию, которая принимает цвет фона, цвет текста, строку с текстом и выводит в окно браузера абзац с соответствующими стилями.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Изучите теоретический материал по теме практической работы перед выполнением задания.

6.2 Примеры выполнения создания простейших функций можно найти в Приложении.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

* 1. Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Что такое функция? Синтаксис функции.

8.2 Для чего предназначен оператор **return**?

8.3 Привести пример использования события на веб - странице

8.4 На какие группы можно разделить события?

**ПРИЛОЖЕНИЕ**:

**Функции** – это набор инструкций, выполняющих определенное действие или вычисляющих определенное значение. Синтаксис:

function имя\_функции([параметр [, ...]]){

    // Инструкции

}

Определение функции начинается с ключевого слова **function**, после которого следует имя функции. К имени функции применяются те же правила, что и для имени переменной. После имени функции в скобках идет перечисление параметров. Если параметров у функции нет, то скобки остаются пустыми. В фигурных скобках идет тело функции, содержащее набор инструкций. Пример простейшей функции:

function hello(){

document.write("Привет!");

}

hello();// вызываем функцию

**Параметры функции**

Мы можем передавать функции информацию, используя параметры (аргументы функции).

Рассмотрим передачу параметров функции:

function display(x){ // определение функции

let z = x \* x;

document.write("Квадрат " + x + " равен " + z);

}

display(5); // вызов функции

Функция display принимает параметр – х. Поэтому при вызове функции мы можем передавать для него значение. В данном примере, передавалось число 5.

## **Возврат значения**

## Функцияможет возвращать результат. Для этого используется оператор **return**:

function sum(**a, b**) {

return **a + b**; }

let **result** = sum(2, 3);

alert(**result**);// выведет 5

Оператор return может находиться в любом месте тела функции. Как только выполнение доходит до этого места, функция останавливается, и значение возвращается в вызвавший ее код (присваивается переменной result).

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 8**

**НАИМЕНОВАНИЕ**: Работа с DOM в JavaScript

**1. ЦЕЛЬ**: Обращение, создание и обработка элементов веб – страницы средствами DOM, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Основы работы с DOM в JavaScript» и ответить на следующие вопросы:

2.1. Что такое DOM?

2.2. Каким образом происходит обращение к конкретному элементу в DOM?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Изменение текста в инпуте: Пусть в текстовом поле записан любой текст (допустим - Привет!), при нажатии кнопки текст должен поменяться (например, Пока!)

5.2 В предыдущей задаче изменить цвет текста при нажатии кнопки (допустим, фраза Пока! должна изменить цвет на зеленый)

5.3 Создайте окно с уведомлением следующего вида средствами JavaScript. Стили прописать самостоятельно внутренний отступ 15px;

рамка – 1px, сплошная, цвет - #d6e9c6;

скругление углов – 4px;

цвет текста - #3c763d;

цвет фона - #dff0d8;



5.4 Модернизируйте предыдущий скрипт, чтобы уведомление исчезало через 3 секунды (используйте таймер).

5.5 Модернизируйте предыдущий скрипт, чтобы уведомление появлялось при клике по кнопке и исчезало через 3 секунды. Кнопку создать средствами HTML (не через JavaScript).

5.6 Создайте нумерованный список из трех позиций:

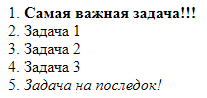


Через JavaScript первым пунктом вставьте сообщение – «Самая важная задача!!!»

последним пунктом – «Задача на последок!»

При решении задачи используйте методы различных вариантов вставки.

Должно получиться:



**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Изучите теоретический материал по теме практической работы перед выполнением задания.

6.2 Примеры выполнения создания простейших функций можно найти в Приложении.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

* 1. Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Как получить ссылку на элемент HTML?

8.2 Для чего используется свойство innerHTML?

8.3 Как можно создать новый элемент (тег) в HTML средствами JavaScript?

8.4 Как можно отобразить новый, созданный средствами JavaScript, элемент на веб – странице?

**ПРИЛОЖЕНИЕ**:

**DOM** – это объектная модель документа, которая представляет все содержимое страницы в виде объектов, которые можно менять.

Объект **document** – основная входная точка. Пример:

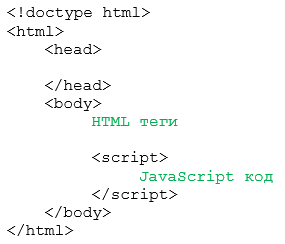
// заменим цвет фона на красный,

document.body.style.background = "red";

// а через секунду вернём как было

setTimeout(() => document.body.style.background = "", 1000);

Для того, чтобы начать работу с DOM необходимо в шаблоне HTML – документа ***скрипт размещать перед закрывающим тегом <body>***, чтобы не возникало проблем.



***Получить ссылку на элемент*** html и поместить его в переменную можно с помощью метода **querySelector** объекта document. Этот метод параметром принимает CSS селектор и возвращает ссылку на первый найденный по этому селектору элемент.

let elem = document.querySelector('#elem');

***Альтернативные методы*:**

***querySelectorAll*** - позволяет получить все элементы страницы по произвольному селектору CSS.

**getElementById** – позволяет получить один элемент страницы по его id.

***getElementsByName*** – позволяет получить элементы страницы по атрибуту name тега.

***getElementsByClassName –*** позволяет получить элементы страницы по их классу.

**Привязывание обработчиков событий к элементам**

Действия пользователя, которые можно отследить через JavaScript, называют **событиями**.

***Популярные события:***

**click** – клик по элементу;

**dblclick** – двойной клик по элементу;

**mouseover** – наведение курсора на элемент;

**mouseout** – уход курсора с элемента;

**Пример**: создадим кнопку, по нажатию на которую на экран выведется какой-то текст.

1. Создадим кнопку:

<input type="submit" id = "button">

2. Получим ссылку на кнопку в переменную:

let button = document.querySelector('#button');

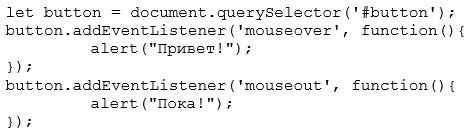
3. Задаем реакцию кнопки при клике по ней. Для этого в JavaScript существует метод **addEventListener**, первым параметром принимающий название события, а вторым параметром – функцию, выполняющуюся при возникновении события.

button.addEventListener('click', function(){

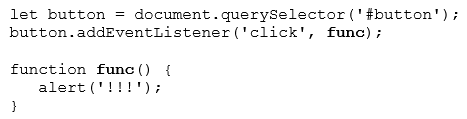
alert("!!!");

});

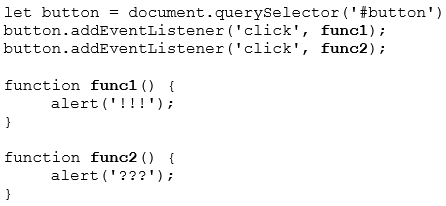
***К одному элементу можно привязать обработчики различных типов событий***. Например, привяжем к одному элементу реакцию на наведение курсора и реакцию на уход:



***Можно использовать не только анонимные функции, но и обычные.***



***К одному элементу можно привязать сразу несколько функций.***



***Привязываемые функции также могут быть анонимными.***

**Работа с текстом тега на JavaScript**

Свойство **innerHTML** – позволяет прочитывать/записывать текст DOM элементов.

**Пример**: Прочитаем текст с помощью свойства **innerHTML** из элемента div по клику. Заменим после этого текст на другой.

1. У нас есть абзац и кнопка

<div **id="elem"**> Текст из тега </div>

<input type="submit" **id = "button"**>

1. Получим ссылки на абзац и кнопку в соответствующие переменные:

let button = document.querySelector('#button');

let elem = document.querySelector('#elem');

1. Выведем по нажатию кнопки текст абзаца на экран:

button.addEventListener('click', **func**);

function **func**() {

alert(**elem.innerHTML**);

}

**Запись текста**

Свойство innerHTML позволяет не только прочитывать, но и записывать текст.

**Пример**: по нажатию на кнопку запишем в абзац какой-нибудь текст

function func() {

elem.innerHTML = 'Новая запись в блоке';

}

**Запись тегов**

Можно записать не только обычный текст, но и текст с тегами.

function func() {

elem.innerHTML = '<b>Новая запись в блоке</b>';

}

**Работа с текстовыми полями**

Научимся получать текст от пользователей сайта. Текстовое поле организуется с помощью тега <input>.

<input id="elem" type="text" **value = "Текст">**

Получим ссылку на инпут в переменную:

let elem = document.querySelector('#elem');

***Выведем на экран*** текущий текст инпута:

alert(elem.value); // выводим текст инпута

***Поменяем текст*** инпута на другой:

elem.value = 'Новый текст!'; // изменяем текст инпута

**Стилизация элементов через атрибут style**

Изменение атрибута style в CSS происходит путем выстраивания цепочки. Например, чтобы поменять цвет у элемента, нужно составить следующую цепочку: **elem.style.color** и присвоить нужное значение цвета.

Если свойство записывается через дефис, например, font – size, преобразуется в fontSize:

**elem.style.fontSize = '20px';**

**Создание нового элемента (тега)**

Создать новый элемент можно через метод **createElement(tag)**

let div = document.createElement('div');

Запишем сообщение в элемент div с классом alert:

let div = document.createElement('div');

div.className = "alert";

div.innerHTML = "<strong>Важно!</strong> Идет важное сообщение.";

Создан элемент с сообщением, но пока он только в переменной. Чтобы он отобразился на странице используем метод **append**:

document.body.append(div)

**Методы для различных вариантов вставки:**

**node.append() –** добавляет узлы или строки в конец node;

**node.prepend() –** вставляет узлы или строки в начало node;

**node.before() –** вставляет узлы или строки до node;

**node.after() –** вставляет узлы или строки после node;

**node.replaceWidth()-** заменяет node заданными узлами или строками.

Пример использование методов для добавления элементов в список и текст до/после него:

<ol id = "ol">

<li>0</li>

<li>1</li>

<li>2</li>

</ol>

<script>

ol.before('before'); // вставляем строку перед <ol>

ol.after('after'); // вставляем строку после <ol>

let liFirst = document.createElement('li');

liFirst.innerHTML = 'начало';

ol.prepend(liFirst); // вставляем liFirst в начало <ol>

let liLast = document.createElement('li');

liLast.innerHTML = 'конец';

ol.append(liLast); // вставляем liLast в конец <ol>

</script>

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 9**

**НАИМЕНОВАНИЕ**: Обработка событий в JavaScript

**1. ЦЕЛЬ**: Создание событий пользователя, воздействие на элементы веб – страницы через пользовательские события, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «События в JavaScript» и ответить на следующие вопросы:

2.1. Что такое событие? Когда их применяют на веб – страницах?

2.2. Какие обработчики событий вам известны?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

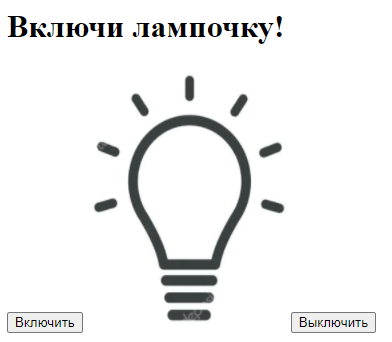
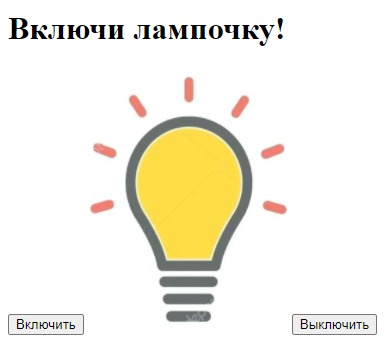
**5. ЗАДАНИЕ**:

5.1 Создайте кнопку, которая будет подсчитывать количество кликов по ней. Количество кликов должно отображаться на самой кнопке.

5.2 Поместите небольшую картинку на страницу, под ней кнопку. При нажатии кнопки картинка должна поменяться на другую.

5.3 Пусть изменение картинки будет происходить при наведении курсора мыши на нее, когда убираем курсор мыши с картинки, она меняется на первоначальную.

5.4 Реализуйте следующую задачу: есть лампочка и две кнопки – одна включает, другая выключает. Можно использовать другой пример с окрашиванием предмета по клику (двойному клику). Примените самостоятельное оформление примера с использованием CSS стилей.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Изучите теоретический материал по теме практической работы перед выполнением задания.

6.2 Примеры выполнения создания простейших функций можно найти в Приложении.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

* 1. Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Привести пример использования события на веб - странице

8.2 На какие группы можно разделить события?

**ПРИЛОЖЕНИЕ**:

**Событие** – это сигнал от браузера о том, что что-то произошло. Наиболее популярные события:

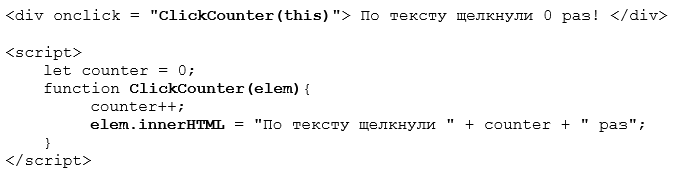
**click** – клик по элементу;

**dblclick** – двойной клик по элементу,

**mouseover** – наведение курсора на элемент,

**mouseout** – уход курсора с элемента.

Рассмотрим пример обработки функции по подсчету кликов по тексту:



Каждый раз, когда кликаем по тексту срабатывает функция ClickCounter, текст изменяется (через свойство innerHTML) и происходит увеличение счетчика на 1.

Рассмотрим пример смены картинки по щелчку мыши по кнопке.

<img id = "myImage" src = "on.png"><br>

<button onclick = "document.querySelector('#myImage').src = 'off.png'">Сменить картинку</button>



**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 10**

**НАИМЕНОВАНИЕ**: Работа с элементами управления в формах

**1. ЦЕЛЬ**: Воздействие на элементы управления в формах HTML, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Работа с формами» и ответить на следующие вопросы:

2.1. Что такое форма?

2.2. Какие элементы управления вам известны?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 https://learn.javascript.ru/ - Современный учебник по JavaScript

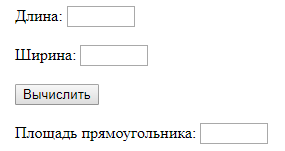
**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Создайте форму для ввода длины и ширины прямоугольника. При нажатии кнопки должна выводится площадь (периметр) прямоугольника в соответствующем текстовом поле.



5.2 Создайте текстовое поле (textarea) и разместите под ним список дисциплин. При наведении курсора мыши на дисциплину, в текстовом поле появляется ФИО преподавателя, кто ведет данную дисциплину. Когда курсор уводится от элемента списка – текстовое поле очищается. Пример подобной задачи можно посмотреть в Приложении.

5.3 Дан инпут и две кнопки. Пусть нажатие на первую кнопку блокирует инпут, а нажатие на вторую - отблокирует.

5.4 Дан чекбокс, кнопка и абзац. По нажатию на кнопку, если чекбокс отмечен, выведите в абзац слово 'Привет', а если чекбокс не отмечен, то слово 'Пока'.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Изучите теоретический материал по теме практической работы перед выполнением задания.

6.2 Примеры выполнения создания простейших функций можно найти в Приложении.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

* 1. Задание, под каждым поместить код программы и полученные результаты в виде скринов.

7.3 Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Привести пример использования события на веб - странице

8.2 На какие группы можно разделить события?

**ПРИЛОЖЕНИЕ**:

Формы в документе входят в специальную коллекцию document.forms. Это «именованная» коллекция: мы можем использовать для получения формы ее имя или порядковый номер в документе:

**document.forms.myform** – форма с name="myform"

**document.forms[0]** – первая форма в документе

Когда мы уже получили форму, любой элемент формы становится доступен через **form.elements**.

<form name = "myform">

<input name = "one" value = "1">

<input name = "two" value = "2">

</form>

<script>

let form = document.forms.myform; // получаем форму

let elem = form.elements.one; // получаем элемент

alert(elem.value); // 1

</script>

Может быть несколько элементов с одним и тем же именем – кнопки переключатели – radio.

В этом случае поступаем следующим образом:

<form name = "myform">

<input type = "radio" name = "age" value = "1">

<input type = "radio" name = "age" value = "2">

</form>

<script>

let form = document.forms.myform; // получаем форму

let elem = form.elements.age; // получаем элемент

alert(elem[0]); // [object HTMLInputElement]

</script>

**Элементы формы**

**input и textarea**

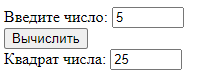
К их значению можно получить доступ через свойство **input**.**value** (строка) или **input.checked** (булево значение) для чекбоксов**.**

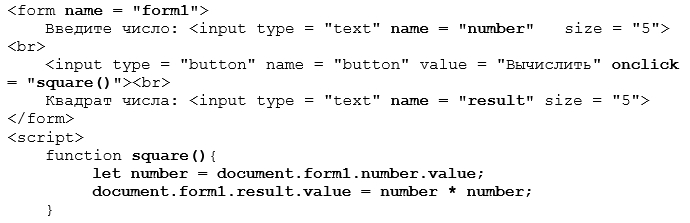
input.value = "Новое значение";

textarea.value = "Новый текст"

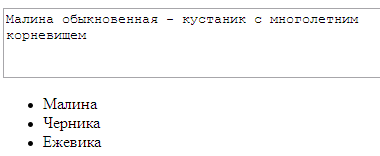
input.checked = true; // для чекбоксов и переключателей

Пример реализации формы для вычисления квадрата числа:



Рассмотрим еще один пример. Пусть у нас будет список ягод, а при наведении мышкой на название ягоды, ее описание появится в текстовом поле.



Нам потребуется два параметра: один с именем объекта (формы), второй с описанием ягод. Наша функция будет выглядеть так:

function **ИмяФункции**(obj, n) {

obj.*desc*.value=n;

}

где *desc* – имя текстового поля для ввода описания. Эта функция будет срабатывать, когда на название ягоды наведен курсор.

Теперь напишем код html – страницы. Обработчик события, когда указатель мыши помещается над элементом, называется **onmouseover**.

<form name="**ИмяФормы**">

<textarea name="*desc*" cols= rows= ></textarea>

</form>

<ul>

<li onmouseover="**ИмяФункции**(**ИмяФормы**, 'Малина обыкновенная - кустарник с многолетним корневищем');">Малина</li>

</ul>

**Атрибут disabled**

***Атрибут disabled используется для блокировки элемента***. Для того, чтобы установить такой атрибут соответствующему свойству требуется присвоить значение true, а, чтобы убрать – значение false

Пусть у нас есть заблокированный инпут:

<input id = "elem" disabled>

Выведем значение атрибута disabled:

let elem = document.querySelector('#elem');

console.log(elem.disabled); // выведет true

Теперь разблокируем его:

elem.disabled = false;

**Работа с checkbox**

Чекбокс представляет собой специальную галочку, которая может находиться в двух состояниях: отмечено и нет.

Чтобы сделать чекбокс отмеченным – ему нужно написать атрибут **checked**:

<input type = "checkbox" id = "elem" checked>

**Чередование атрибутов без значений**

Пусть дан заблокированный инпут и кнопка:

<input type = "checkbox" id = "elem" disabled>

<input id = "button" type = "submit">

Сделаем так, чтобы каждое нажатие на кнопку привело к смене состояния кнопки – с заблокированного на разблокированное и наоборот:

let elem = document.querySelector('#elem');

let button = document.querySelector('#button');

button.addEventListener('click', function(){

if(elem.disabled){

elem.disabled = false;// разблокировка

} else{

elem.disabled = true; // блокировка

}

});

Задачу можно решить короче:

button.addEventListener('click', function(){

**elem.disabled = !elem.disabled;**

});

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №11**

**НАИМЕНОВАНИЕ**: Работа с таймерами

1. **ЦЕЛЬ**: Создание и запуск таймеров в сценариях JavaScript, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Таймеры в JavaScript» и ответить на следующие вопросы:

1. Какие таймеры в JavaScript существуют?

2. Назначение таймеров в веб - приложениях?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 <http://old.code.mu/books/javascript/dom/rabota-s-tajmerami-v-javascript.html>- Работа с таймерами

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

4.1 ПЭВМ, подключенные к сети Интернет

4.2 Текстовый редактор

**5. ЗАДАНИЕ**:

5.1Напишите код, который будет выводить в браузер количество секунд, прошедших с момента запуска программы в формате:

Прошло: 1 сек.

Прошло: 2 сек. // и так далее.

При достижении 5 секунд таймер останавливался и надписи больше не выводились.

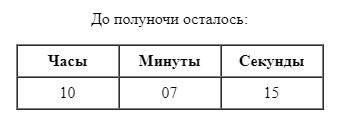
5.2Создайте электронные часы, которые самостоятельно идут.

5.3Создайте таймер обратного отсчета (от 10 секунд)

5.4Создайте простой слайдер, который каждую секунду меняет изображение на другое. Использовать минимум 5 изображений.

5.5Напишите сценарий, в котором при нажатии на кнопку с соответствующим фруктом, открывалось новое окно с изображением данного фрукта. Фруктов и изображений должно быть не менее трех.

5.6Создайте отсчет до полуночи в следующем формате:



Время должно изменяться автоматически!

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 Изучите теоретический материал по теме практической работы перед выполнением задания.

6.2 Примеры выполнения создания простейших функций можно найти в Приложении.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Задание, под каждым поместить код программы и полученные результаты в виде скринов.

* 1. Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Назначение таймеров

8.2 Синтаксис запуска таймера и его остановки?

**ПРИЛОЖЕНИЕ**:

Можно создать два типа таймеров – **одноразовые** – срабатывают один раз после определенного промежутка времени и **многоразовые** – срабатывают непрерывно через определенный интервал времени.

На веб – странице может присутствовать несколько таймеров и запускать их можно в любом месте кода, например, когда произойдет событие **onload** окна или будет нажата кнопка. Область применение таймеров – рекламные баннеры, которые чередуются через определенные интервалы времени или отображают изменяющееся время на веб – странице; все виды анимации также нуждаются в таймерах.

**Одноразовый таймер**

Для его установки используется метод **setTimeout()** объекта Window.

setTimeout(“ваш JavaScript- код” , задержка в миллисекундах)

Метод setTimeout() принимает два параметра – первый – это Javascript -код, который вы хотите выполнить, второй – задержка, после которой будет выполняться код.

Метод возвращает значение (целое число), которое является уникальным идентификатором (ID) таймера. Если позже вы решите, что хотите остановить срабатывание таймера, вы будете использовать этот идентификатор, чтобы указать JavaScript, к какому таймеру вы обращаетесь.

Например, чтобы установить таймер, который сработает через три секунды после загрузки страницы, можете использовать следующий код:

<script>

let timerID;

function window\_onload(){

timerID = setTimeout("alert('Время вышло')", 3000);

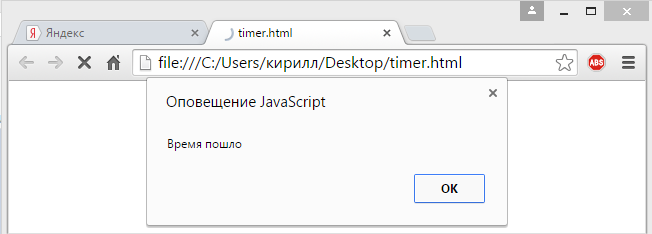
alert("Время пошло");

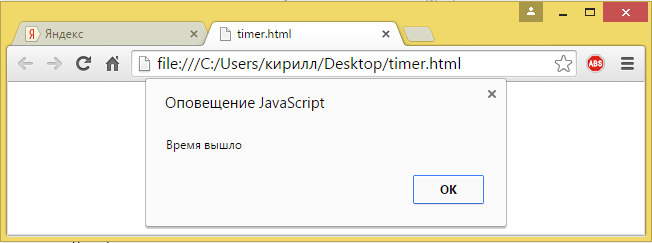
}

</script>

</head>

<body onload="window\_onload()">





Установка таймера не приостанавливает выполнение сценариев. Таймер выполняется в фоновом режиме и срабатывает, когда заканчивается время его задержки. Страница при этом выполняется как обычно, и после того, как вы запускаете обратный отсчет таймера, любые сценарии будут немедленно выполнены. В данном примере окно с сообщением, говорящим об установке таймера, появится сразу же после выполнения кода для этой установки.

Чтобы сбросить таймер используется метод **clearTimeout**(). Он принимает только один параметр – ID таймера, который возвращает метод **setTimeot**(). Изменим пример и создадим кнопку, которую вы можете нажать, чтобы остановить таймер.

<html>

<head>

<script>

let timerID;

function window\_onload(){

timerID = setTimeout("alert('Время вышло')", 3000);

alert("Время пошло");

}

**function butStopTimer\_onclick(){**

**clearTimeout(timerID);**

**alert("Таймер сброшен");**

**}**

</script>

</head>

<body onload="window\_onload()">

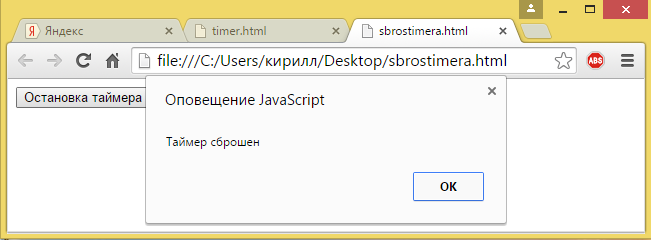
**<form name=form1>**

**<input type="button" value="Остановка таймера" name=butStopTimer onclick="return butStopTimer\_onclick()">**

**</form>**

</body>

</html>



**Многоразовый таймер**

Для установки многоразового таймера используют метод – **setInterval**(), для его сброса **clearInterval**(). Метод **setInterval**() принимает те же параметры, что и **setTimeout**(), только второй параметр теперь определяет интервал в миллисекундах между каждым сбрасыванием таймера. Например, чтобы установить таймер, который запускает функцию **myFunction**() каждые пять секунд, код должен быть следующим:

**var myTimerID = setInterval(“myFunction()”, 5000);**

Как и **setTimeout**(), метод **setInterval**() возвращает уникальный идентификатор таймера, который понадобиться, если вы хотите очистить таймер с помощью clearInterval():

**clearInterval(myTimerID);**

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №12**

**НАИМЕНОВАНИЕ**: Создание приложения с блоком тестирования

1. **ЦЕЛЬ**: Создание тестирующей программы средствами HTML и JavaScript, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Использование HTML – форм» и ответить на следующие вопросы:

2.1 Назначение элементов формы на веб - странице?

2.2 Какие элементы формы вам известны?

2.3 Обязательные теги для создания формы?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 <https://learn.javascript.ru/form-elements> - Работа с формами в JavaScript

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет
  2. Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Разработать веб – приложение по определенной тематике (тема выдается преподавателем), где должен быть размещен материал и вставлен блок тестирования. Тест состоит минимум из 10 вопросов, которые рассматриваются в представленном теоретическом материале.

5.2 Подключите таймер, чтобы на обдумывания над ответом отводилось секунд 30;

5.3. Результат выводился в виде оценки и надписи, количества неверных ответов.

61% правильных ответов – оценка «3»;

81% - оценка «4»;

100% - оценка «5»;

1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия и проверить соответствие с критериями оценивания, которые представлены в Приложении;

* 1. Дать ответы на контрольные вопросы.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

* 1. Наименование и цель работы
  2. Готовое веб - приложение

7.3 Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Обязательные атрибуты элементов формы, чтобы осуществить связь с кодом JavaScript?

8.2 Каким образом подключается таймер к скрипту?

**ПРИЛОЖЕНИЕ**:

**Критерии оценки:**

**Дизайн:** (максимально 15 баллов)

| **Аспект** | **Вес аспекта, балл** | **Начисляемый балл** |
| --- | --- | --- |
| Страницы приложения имеют одинаковый графический интерфейс (шапка, меню, контент, подвал) | 2 |  |
| *Общее впечатление о дизайне приложения* | *до 5* |  |
| *Оптимальное соотношение яркости (контраст) между текстом и фоном* | *до 4* |  |
| Однородность текста и равномерность распределения пустых мест на всем сайте | 2 |  |
| Материал соответствует теме задания | 2 |  |

**Верстка:** (максимально 15 баллов)

| **Аспект** | **Вес аспекта, балл** | **Начисляемый балл** |
| --- | --- | --- |
| Применяется блочная верстка | 4 |  |
| Подключение CSS – файла, посредством которого осуществляется оформление сайта | 4 |  |
| Весь основной материал представлен локально (не в виде ссылок на внешние ресурсы) | 3 |  |
| Наличие примеров с кодом не менее трех | 4 |  |

**Программирование:** (максимально 50 баллов)

| **Аспект** | **Вес аспекта, балл** | **Начисляемый балл** |
| --- | --- | --- |
| Наличие блока тестирования | 5 |  |
| Количество вопросов – не менее 10 (штраф за каждый не выполненный вопрос – 0, 25 баллов) | 10 |  |
| Подключение таймера | 5 |  |
| Вывод результата тестирования (статистика) | 3 |  |
| Наличие в коде пояснений (комментариев) | 2 |  |
| *Все вопросы выводятся сразу* | *10* |  |
| *Вопросы выводятся друг за другом, последовательно* | *15* |  |
| Отмечаются правильные/неправильные ответы | 10 |  |

**Тестирование:** (максимально 20 баллов)

| **Аспект** | **Вес аспекта, балл** | **Начисляемый балл** |
| --- | --- | --- |
| *Корректность (понятность) вопроса* | *До 5* |  |
| Ответ на вопрос можно найти, опираясь на представленный материал | 5 |  |
| Выбор одного варианта ответа (кнопки – переключатели) | 5 |  |
| Наличие множественного варианта ответа или поля для ввода варианта ответа (другой вариант) | 5 |  |

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №13**

**НАИМЕНОВАНИЕ**: Работа с классами в JavaScript

1. **ЦЕЛЬ**: Изучить основные возможности работы с ООП средствами JavaScript, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Введение в ООП» и ответить на следующие вопросы:

2.1 Что такое класс объектов? С чем его можно сравнить?

2.2 Что относится к свойствам объекта, а что к его методам? Привести примеры.

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет
  2. Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Реализуйте класс **Worker** (Работник), который будет иметь следующие свойства: **name** (имя), **surname** (фамилию), **rate** (ставка за день работы), **days** (количество отработанных дней). Также класс должен иметь метод **getSalary**(), который выводит зарплату работника .

5.1.1 С помощью класса создайте двух рабочих, выведите на экран их данные.

5.1.2 Найдите сумму зарплат двух рабочих и выведите их на экран.

1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия

6.3 Дать ответы на контрольные вопросы.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

* 1. Наименование и цель работы
  2. Выполненное задание

7.3 Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 В чем разница между методами начинающимися с get и set?

8.2 Когда вызывается метод constructor()?

**ПРИЛОЖЕНИЕ**:

В ООП все действия происходят с **объектами**. Например, каждая отдельная машина – это объект и будет принадлежать она к определенной группе объектов (Автомобили). Группа будет называться **классом**. **Класс** – это чертеж, по которому сделана каждая отдельная машина. Таким образом **объект** – это представитель конкретного класса.

Создается класс следующим образом:



Пока это только чертёж, самого объекта еще нет. Создается объект с помощью команды new, написанный перед названием класса:

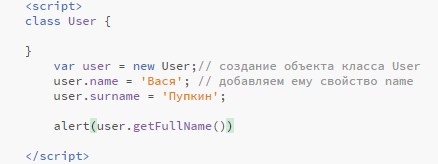


Машина имеет характеристики – цвет, количество колес, вид кузова и т.д. Эти характеристики называются **свойствами** объекта.

Каждая машина может выполнять определенные действия – ехать, стоять, поворачивать и т.д. Такие действия называют **методами**.

**Создание объекта**

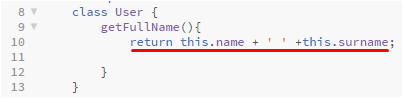
Пусть у нас есть класс User



Создаем объект класса User, добавляем ему свойства **name** и **surname**, прописываем значения этих свойств.

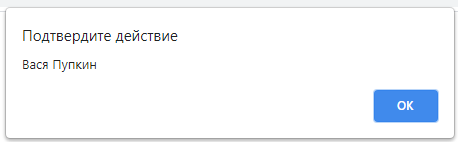
Сделаем так, чтобы наш объект с помощью метода **getFullName** смог вывести значения свойств на экран.

Запишем реализацию метода **getFullName**:



Обращение внутри метода происходит через **this** и через точку пишется свойство, которое хотим вернуть.

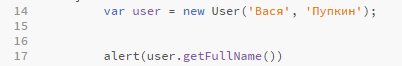
Результат работы примера:



**Конструктор**

Свойства нашего объекта создаются после создания самого объекта. Это не всегда удобно. Сделаем так, чтобы в момент создания объекта мы обязаны были указать имя и фамилию нашего пользователя в качестве параметров.

Для этого, при создании объекта **User**, в качестве параметров передаем ему имя и фамилию. Свойства, записанные ниже – удаляем:

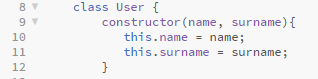


Эти два параметра ('Вася', 'Пупкин') попадут в метод с название **constructor**(). Создадим этот метод и разберем что это такое:

Метод **constructor()** вызывается в момент создания объекта. Имя и фамилия попадут в параметры данного метода.



Но эти две переменные будут **локальными**. Нам нужно, чтобы имя и фамилия записались в свойства нашего объекта. Для этого в свойство нашего объекта **name** запишем его содержимое. Тоже самое делаем и для **surname**.

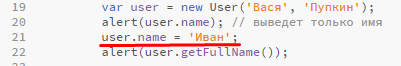


Запустим скрипт и убедимся, что все работает.

Теперь мы можем после создания объекта **User** вывести попеременно имя или фамилию нашего пользователя:



Выполним следующее действие: поменяем имя нашего пользователя:



При запуске, сначала у нас выведется имя Вася, во втором алерте оно заменится на Иван:

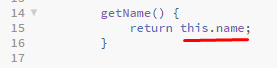


**Геттеры**

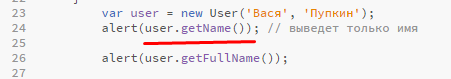
Изменять свойства снаружи у объектов не рекомендуется, так как в итоге можно потерять данные пользователя. В JavaScript нельзя жестко проставить запрет на изменение свойств. Поступают следующим способом:

Чтобы получить возможность прочитать (посмотреть) свойства используются **геттеры**:

Для этого создается метод **getName()**:



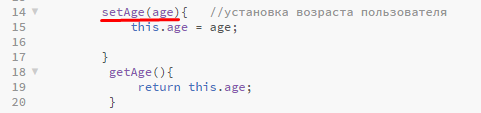
И теперь снаружи мы можем прочитать метод:



Это более правильное обращение для чтения свойств объекта.

**Сеттеры**

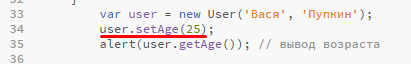
Определим для нашего пользователя возраст. Сделаем это следующим образом:



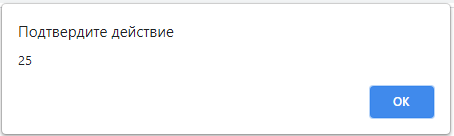
Метод **setAge()** – устанавливает (записывает) возраст пользователя;

Метод **getAge()** – считывает возраст пользователя.

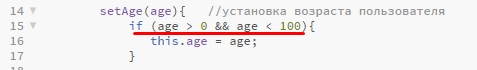
Зададим возраст следующим образом:



И сразу выведем его на экран:

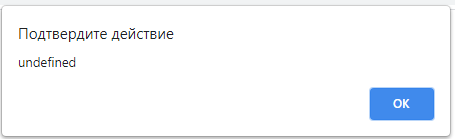


При использовании сеттеров можно прописать условия его работы (проверку корректности его значения). Когда определяем значение свойства напрямую, проверку сделать невозможно.



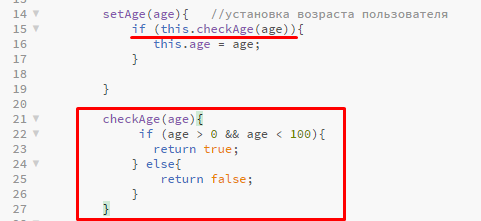
Передадим некорректный возраст, например, 225 и убедимся, что браузер выдаст ошибку:





**Вспомогательные методы**

Проверку правильности ввода возраста можно осуществить через вспомогательный метод. Создадим такой метод – **checkAge(),** который в случае корректного ввода значения возраста выдаст значение **true**, иначе **false**.



**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №14**

**НАИМЕНОВАНИЕ**: Работа с Canvas

1. **ЦЕЛЬ**: Рисование графических объектов средствами JavaScript, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Canvas. Работа с графикой» и ответить на следующие вопросы:

2.1 Как организовать рабочую область для рисования?

2.2 Какие методы для рисования простейших объектов вам известны?

2.3 Какие свойства для работы с объектами вам известны?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 <http://old.code.mu/books/javascript/canvas/osnovy-raboty-s-canvas-v-javascript.html> - Основы работа с canvas в JavaScript

3.3 <https://joompress.biz/animatsiya-sprajtov-v-html5-canvas/> - Анимация спрайтов

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет
  2. Текстовый редактор

**5. ЗАДАНИЕ**:

5.1 Организуйте холст для рисования, на котором разместите следующие нарисованные объекты:

- решетка (рисуется через объект линия);

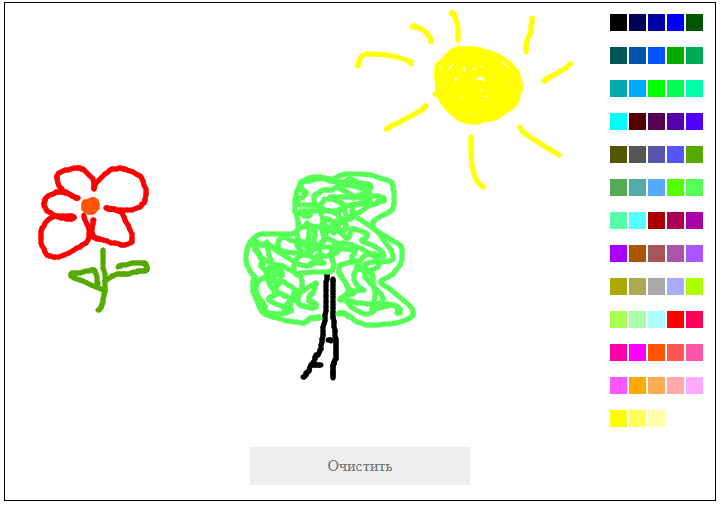
- елка (рисуем через 2 – 3 треугольника);

- домик с окошком, который необходимо раскрасить;

- cмайл, также необходимо раскрасить;

5.2 Создать анимированный спрайт. Пояснения даны в пункте Приложение

5.3 Реализуем проект Paint. Пояснения даны в пункте Приложение.



1. **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия

6.3 Дать ответы на контрольные вопросы.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

* 1. Наименование и цель работы

7.2 Выполненное задание

7.3 Ответы на контрольные вопросы

* 1. Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Синтаксис организации холста для работы с canvas?

8.2 Что такое спрайт, где они применяются?

**ПРИЛОЖЕНИЕ**:

Элемент **canvas** позволяет рисовать графические объекты и манипулировать ими. Определяется элемент на веб-странице следующим образом:

<body>

**<canvas id="myCanvas" width="300" height="200">**

**…**

**</canvas>**

</body>

Для элемента canvas задаются ширина и высота, если не указывать эти атрибуты, то по умолчанию ширина составляет – 300px, а высота – 150px.

Все рисование на **canvas** производится с помощью кода JavaScript. Чтобы начать рисовать на canvas, надо получить его контекст:

**let canvas = document.getElementById("myCanvas"),**

**context = canvas.getContext("2d");**

Для получения контекста используется функция **getContext("2d")**. В данном случае получаем двухмерный контекст для создания двухмерной графики.

**Рисование прямоугольников**

Для рисования простейших фигур – прямоугольников, нам понадобятся три метода:

**clearRect(x, y, w, h)** – очищает определенную прямоугольную область, верхний левый угол которой имеет координаты x и y, ширина равна w, а высота h;

**fillRect(x, y, w, h)** – заливает цветом прямоугольник, верхний левый угол которой имеет координаты x и y, ширина равна w, а высота h;

**strokeRect(x, y, w, h)** – рисует контур прямоугольника без заливки его каким-то определенным цветом.

**Настройка рисования**

Контекст элемента canvas предоставляет ряд свойств, с помощью которых можно настроить отрисовку на canvas. К таким свойствам относятся:

**strokeStyle** – устанавливает цвет линии или цвет контура, по умолчанию черный;

**fillStyle** – устанавливает цвет заполнения фигур, по умолчанию черный;

В качестве значений этих двух свойств указывают название цвета.

**linewidth** – устанавливает толщину линий, по умолчанию1.0;

**lineJoin** – устанавливает стиль соединения линий. Может принимать следующие значения:

* **miter** – прямые соединения;
* **round** – закругленные соединения;
* **bevel** – конические соединения.

**globalAlpha** – устанавливает прозрачность отрисовки на canvas. Принимает в качестве значения число от 0 (полностью прозрачный) до 1.0 (не прозрачный);

**setLineDash** – создает линию из коротких черточек. В качестве параметра принимает массив чисел, которые устанавливают расстояние между линиями.

**Рисование фигур**

Элемент canvas позволяет рисовать и более сложные фигуры. Для этого используется концепция геометрических путей, которые представляют набор линий, окружностей, прямоугольников и более мелких деталей, которые необходимы для построения сложной фигуры.

Для создания нового пути надо вызвать метод **beginPath**(), а после завершения пути – метод **closePath**():

let canvas = document.getElementById("myCanvas"),

    context = canvas.getContext("2d");

**context.beginPath();**

// здесь инструкции по созданию фигур

**context.closePath();**

Между вызовами данных методов находятся методы, непосредственно создающие различные участки пути.

Для начала рисования пути, надо зафиксировать начальную точку этого пути. Это делается с помощью метода **moveTo**(), который имеет следующее определение:

**moveTo(x, y)**

Метод перемещает нас на точку с координатами х и y.

Метод **lineTo**() рисует линию, определение у нее такое же как и у метода **moveTo**(). Метод рисует линию от текущей позиции до точки с координатами x и y.

**Анимированный спрайт:**

В качестве движущихся персонажей в веб – играх используют анимированные спрайты.

**Спрайт** – это графический файл с последовательностью изображений, которые выполнены в виде отдельных кадров. Каждый кадр, отличается от другого. Если кадры последовательно заменять, то получается движущийся объект.



Анимирование спрайта происходит в элементе Canvas. Пример JavaScript кода, который описывает процесс анимации представлен ниже:

<script>

let width = 100, *// ширина элемента canvas*

height = 100, *// высота элемента canvas*

frames = 4, *// количество фреймов в спрайте (считается с 0)*

currentFrame = 0, *// текущей фрейм в спрайте*

canvas = document.getElementById("myCanvas"); *// переменная для доступа к контексту*

let ctx = canvas.getContext("2d"); *// доступ контекста для формата 2D*

let image = new Image(); *// переменная, содержащая файл со спрайтом*

image.src = 'sprite.png';

let draw = function() {

ctx.clearRect(0, 0, width, height);

***// анимация вертикального спрайта***

**ctx.drawImage(image, 0, height \* currentFrame, width, height, 0, 0, width, height);**

if (currentFrame == frames) {

currentFrame = 0;

} else {

currentFrame++;

}

}

setInterval(draw, 100); *// интервал смены кадров в спрайте*

</script>

***Если спрайт нарисован в горизонтальном виде***, то выделенная строка изменится на следующую:

**ctx.drawImage(image, width \* currentFrame, 0, width, height, 0, 0, width, height);**

**Создаем простой графический редактор**

1. ***Создаем объекты:***

<div class ="conteiner">

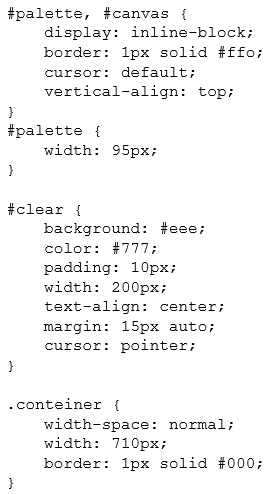
<canvas id = "canvas" width="600" height="400"></canvas> // холст для рисования

<div id = "palette"> </div> // палитра

<div id = "clear"> Очистить </div> // кнопка очистки контекста

</div>

1. ***Стили CSS***



Можно применить свои стили для оформления.

1. ***Код JavaScript***

// завершение загрузки страницы

window.addEventListener('load', function onWindowLoad() {

generatePalette(document.querySelector('#palette'));// генерируем палитру

let canvas = document.querySelector('#canvas');

let context = canvas.getContext('2d');

// инициализируем переменные для рисования

context.lineCap = "round";

context.lineWidth = 5;

// устанавливаем обработчик на кнопку очистки изображения

document.querySelector('#clear').onclick = function clear() {

context.clearRect(0, 0, canvas.width, canvas.height);

};

// на любое движение мыши по canvas будет выполняться данная функция

canvas.onmousemove = function drawIfPressed(e) {

let x = e.offsetX; // в “e” попадает экземпляр MouseEvent

let y = e.offsetY;

let dx = e.movementX;

let dy = e.movementY;

// проверка нажатия кнопки мыши, если да – то рисуем

if(e.buttons > 0) {

context.beginPath();

context.moveTo(x, y);

context.lineTo(x - dx, y - dy);

context.stroke();

context.closePath();

}

};

// функция генерации палитры

function generatePalette(palette) {

for (let r = 0, max = 3; r <= max; r++){

for (let g = 0; g <= max; g++) {

for (let b = 0; b <= max; b++) {

let paletteBlock = document.createElement('div');

paletteBlock.className = 'button';

paletteBlock.addEventListener('click', function changeColor(e) {

context.strokeStyle = e.target.style.backgroundColor;

});

paletteBlock.style.backgroundColor = (

'rgb(' + Math.round(r \* 255 / max) + ", "

+ Math.round(g \* 255 / max) + ", "

+ Math.round(b \* 255 / max) + ")"

);

palette.appendChild(paletteBlock);

}

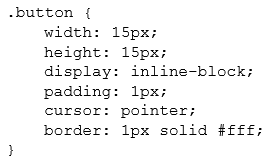
}

}

}

});

1. Дописываем CSS код для палитры, которая была сгенерирована JavaScript – кодом (был создан тег <div class = "button" > 



Можно сгенерировать палитру на большее количество цветов.

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №15**

**НАИМЕНОВАНИЕ**: Работа с библиотекой jQuery

1. **ЦЕЛЬ**: Изучение возможностей библиотеки jQuery, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6

**2. ПОДГОТОВКА К ЗАНЯТИЮ**: По предложенной литературе повторить тему «Библиотека jQuery» и ответить на следующие вопросы:

2.1 Как можно подключить библиотеку jQuery?

2.2 Назначение библиотеки?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 <http://old.code.mu/books/javascript/jquery/osnovy-raboty-s-jquery.html> - Основы работа с jQuery

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет
  2. Текстовый редактор

**5. ЗАДАНИЕ**:

1. На веб странице поместите текст, состоящий из нескольких заголовков второго уровня и абзацев. Также на странице должен присутствовать список из некоторого числа позиций. Необходимо:

5.1.1 Во всех h2 поменять текст на *Изучаем jQuery!* Цвет текста изменить, а также сменить размер шрифта;

5.1.2 Через alert вывести поочередно все пункты списка

5.2 ***Эффекты и анимация в jQuery***

5.2.1 Пусть у вас есть блок, в котором содержится список (10 – 15 пунктов). Необходимо над блоком поместить кнопку или ссылку на которой написано «Свернуть». При нажатии на ссылку, блок со списком начинает сворачиваться. Когда полностью свернется, запись на кнопке (ссылке) изменяется на «Развернуть». При нажатии на нее, блок со списком снова появляется (разворачивается). Данная задача решается с использованием метода ***slideToggle***.

5.2.2 Продублируйте блок со списком из предыдущего задания. Примените к нему эффект исчезновения и появления. Задача на отработку методов ***hide*** и ***show***.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия

6.3 Дать ответы на контрольные вопросы.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

* 1. Выполненное задание

7.3 Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 С чего начинается работа с библиотекой jQuery?

8.2 Какие селекторы jQuery вам известны, перечислить?

**ПРИЛОЖЕНИЕ**:

jQuery представляет собой библиотеку JavaScript – набор готовых функций, которые помогают делать некоторые вещи проще и удобнее, чем это делается на числом JavaScript.

Для сравнения, первый код на чистом JavaScript, второй на jQuery:

document.getElementById('elem').innerHTML **=** '!'; // на JS

$('#elem').html('!'); // на jQuery

Код jquery выглядит намного компактнее и проще для написания.

**Подключение jQuery**

Для начала нужно скачать jQuery с официального сайта - <https://jquery.com/download/> , положить файл с вашим сайтом и подключить его в разделе head:

<script src="путь к файлу с jQuery"></script>

**Начало работы с jQuery**

Работа с библиотекой jQuery осуществляется с помощью универсальной функции **$**. После нее следует писать круглые скобки, в которых указываются параметры: **$(параметр).** Чаще всего с помощью $ получают группу HTML элементов по CSS селектору. Например, получить все элемента с классом .www:

let elems = $('.www');

если нужны только абзацы с классом .www:

let elems = $('p.www');

Если работаем с id, то обращение происходит через #.

**Набора jQuery**

В переменную **elems** записывается группа выбранных элементов, которые называются *набором элементов* jQuery. Затем одновременно для всего набора элементов можно изменить свойства. Переменная **elems** чаще всего не нужна, т.к. jQuery позволяет применять полезные функции прямо к набору, в виде методов. Например, получим все элементы с классом .**www** и назначим им красный цвет:

$('.www').css('color', 'red');

Почти все методы jQuery возвращают набор, к которому этот метод был применен, что позволяет выстраивать *цепочки методов любой длины*.

Пример: получим все элементы с классом .www, поставим и красный цвет и сменим их текст на новый:

$('.www').css('color', 'red').html('новый текст');

Если для работы с jQuery понадобиться переменная, которая в себе содержит обернутый набор jQuery, то такую переменную принято начинать с $ (т.е. вместо **elems**, писать $**elems**).

Пример: Изменим текст во всех абзацах с классом .www:

let $elems = $('.www');

$elems.html('новый текст');

**Вывод текста элемента через jQuery**

Метод .**html**() можно использовать не только для изменения текста элементов, но и для вывода его на экран. Например, давайте получим элемент и выведем алертом его текст:

<p id="test">Абзац.</p>

alert($('#text').html());

Если попытаться получить текст не с одного элемента, а многих, получите только содержимое первого элемента.

**Работа с CSS через jQuery**

Для работы с CSS предусмотрен метод .**css**(). Первым параметром он принимает имя CSS свойства, вторым – его новое значение. Выше рассматривался пример с изменением цвета. Можно использовать свойства, которые пишутся через дефис:

$('.www').css('font-size', '20px');

Если хотите задать несколько свойств CSS, то для этого можно передать параметром в метод .**css**() **объект**, ключами которого будут CSS свойства, а значениями объекта – значения CSS свойств:

$('.www').css({color: 'red', font: '12px Arial'});

***Внимание***! Свойства, которые пишутся через дефис, в этом случае изменяются по принципу **fontSize**.

**Селекторы jQuery**

jQuery поддерживает все стандартные селекторы CSS. В дополнение к ним идут и нестандартные селекторы, которые позволят расширить селекторы CSS.

Например, псевдокласс :**first** позволяет выбрать первый элемент в наборе jQuery:

$('li:first');// выбран первый пункт списка

Псевдокласс :**last** находит последний элемент набора.

С помощью :**eq** можно выбрать элемент с любым номером в наборе, допустим третий элемент - :**eq**(3).

С помощью :**even** и :**odd** можно выбирать четные и нечетные элементы в наборе.

Псевдокласс :**header** выбирает одновременно все заголовки от h1 до h6.

Псевдокласс :**has** выбирает элементы по их содержимому. Например, можно выбрать все абзацы, внутри которых есть теги b: $('p:has(b)');

Псевдокласс :**contains** выбирает элементы по наличию определенного текста в них: $('p:contains("текст")');

Псевдоклассы :**empty** и :**parent** позволяют выбирать все пустые элементы (в которых нет текста), и наоборот, все непустые элементы.

Существует еще и **метод .parent()**, который позволяет выбрать родителя элемента.

**Метод each**

Метод .**each**() позволяет применить какую – либо функцию для всех элементов набора jQuery. При этом внутри функции можно разделить элементы и поступить по – разному с каждым из них.

По сути .**each**() является циклом, с помощью которого можно перебрать все найденные элементы. Ссылка на тот элемент, по которому сейчас проходит цикл будет лежать в **this**.

Пример: получим все элементы с классом .**www** и выведем на экран их содержимое.

Разбор кода: с помощью $('.www')получаем нужные нам элементы, затем с помощью .**each**() к каждому полученному элементу применяем функцию. Сначала она применится к первому абзацу, потом ко второму и т.д. Внутри функции **this** будет указывать на тот элемент, к которому применяется функция.

<p class="www">Текст абзаца 1.</p>

<p class="www">Текст абзаца 2.</p>

<p class="www">Текст абзаца 3.</p>

<p>Абзац.</p>

$('.www').each(function() {

alert($(this).html());

});

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №16**

**НАИМЕНОВАНИЕ**: Работа с фреймворком Vue.js

1. **ЦЕЛЬ**: Изучение возможностей фреймворка Vue.js, формирование ОК 2, ОК 4, ОК 5, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6
2. **ПОДГОТОВКА К ЗАНЯТИЮ**: по предложенной литературе повторить тему «Фреймворк Vie.js» и ответить на следующие вопросы:
3. Назначение фреймворка? На разработку каких приложений он ориентирован?
4. Как можно подключить фреймворк?

**3. ЛИТЕРАТУРА**:

3.1 Шомас Е.А. Технологии стороны клиента. Язык JavaScript., Учебное пособие., Самара КС ПГУТИ, 2019г. – 4,9 п. л.

3.2 <https://ru.vuejs.org/> - Документация по Vue.js

**4. ПЕРЕЧЕНЬ ОБОРУДОВАНИЯ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**:

* 1. ПЭВМ, подключенные к сети Интернет
  2. Текстовый редактор

**5. ЗАДАНИЕ**:

* 1. Вывести сообщение в абзаце. Прикрепить к нему обработчик события, который позволит изменить текст сообщения на новый при наведении на него курсора мыши, когда убираешь курсор мыши, текст меняется на прежний. Оформить все через методы.
  2. Изменить стиль CSS, отвечающий за цвет текста. Цвет текста вводится самостоятельно, в текстовое поле. Использовать не менее 5-ти цветов
  3. Создайте появляющийся и исчезающий рисунок. Появление и исчезновение пусть происходит под управлением кнопки. Можно реализовать процесс включения и выключения устройства.
  4. Организуйте список (массив) из списка дисциплин и ФИО преподавателя, кто ведет данную дисциплину. Выведите через цикл сначала просто список дисциплин, затем список дисциплин с ФИО преподавателя.

**6. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:**

6.1 По предложенной литературе изучить необходимый материал и ответить на вопросы для допуска к практическому занятию;

6.2 Выполнить задание практического занятия

6.3 Дать ответы на контрольные вопросы.

**7. СОДЕРЖАНИЕ ОТЧЕТА:**

7.1 Наименование и цель работы

7.2 Выполненное задание

7.3 Ответы на контрольные вопросы

7.4 Вывод о проделанной работе

**8. КОНТРОЛЬНЫЕ ВОПРОСЫ:**

8.1 Какие настройки объекта класса Vue вам известны?

8.2 Как можно назначить событие? Отметить две разные формы.

**ПРИЛОЖЕНИЕ**:

**Vue.js** – один из самых популярных JavaScript фреймворков. Используется он для написания SPA – одностраничных приложений.

**Подключение**

Для подключения фреймворка необходимо перейти на официальный сайт - <https://ru.vuejs.org/>. Скачиваем JS – файл и подключить тегом <script> на странице проекта (файл index.html). Собственный файл – main.js подключаем ниже.



**!!!** Подключение собственного файла происходит только после подключения фреймворка, иначе работать не будет.

Разделим область в IDE (редакторе) на два окна для удобства разработки:



**Запуск Vue**

Для запуска Vue необходимо создать новый объект класса Vue. Этот класс доступен после подключения Vue. Создадим новый объект этого класса:

**new Vue();**

Далее объекту передаются настройки. Настройки передаются внутри {}:

Самой главной настройкой является настройка **el** (элемент) – в нее передаем селектор того элемента, в котором будет работать наш Vue:

new Vue({

el: 'селектор элемента',

});

Пусть у нас есть элемент с id = demo **(#demo**), передадим его в качестве настройки параметра **e**l:

new Vue({

el: '#demo',

});

Также создадим элемент в файле index.html - <div id="demo"></div>.

Если запустить код, то он ничего не выдаст. Чтобы код был полезным надо добавить еще настроек.

Следующая настройка – **data** – хранит в себе данные (переменные), которые мы хотим выводить на нашей веб – странице. Данные хранятся в виде объекта ***ключ: значение***.

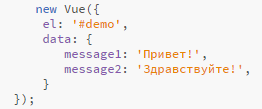
Пусть в data в ключе message храниться сообщение с текстом “Привет!”:



Это сообщение мы можем вывести в любом месте нашей веб – страницы следующим образом: {{ message }}, вот так:



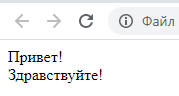
Пусть в data хранится не одно, а несколько сообщений:



Выводятся они на экран следующим образом:



Результат работы скрипта:



**События, методы и атрибуты**

**Назначение события**

Пусть у нас есть <div>, в котором мы помещаем абзац с выводом сообщения «Привет». Добавим кнопку, которая изменит сообщение на «Здравствуйте».

Назначается событие с помощью директивы **v-on.**



В настройке **data** прописываем сообщение, которое будет выведено первоначально.



Протестируем пример, и убедимся, что при нажатии кнопки сообщение поменялось.

Можно использовать сокращённую форму назначения события - **@click**:



Если нам нужно использовать другое событие, то просто прописываем его: **@mouseover** (наведение кнопки мыши)

**Работа с методами**

Следующая настройка Vue – **methods** – хранит в себе набор функций, которые можно навешивать на события. Во Vue эти функции принято называть **методами**.

Изменим предыдущий пример. Пусть у нас при наведении мыши на текст измениться сообщение. Изменения сообщения будет происходить с применением метода. Назовем метод – **changeText**.

В HTML пишем:



А в скрипт JS добавляем метод, следующим образом:

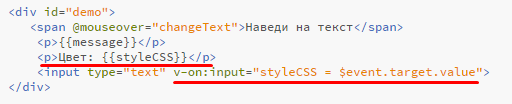


Метод добавляется через настройку. К свойству, записанному в настройке **date** можно обратиться в методе через **this**.

Протестируем пример и убедимся, что метод сработал.

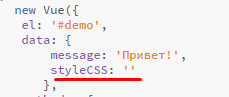
**Работа с HTML – атрибутами**

Для работы с атрибутами в коде HTML добавим абзацы и новые переменные в HTML:

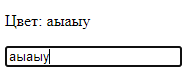


В переменную styleCSS будет попадать текст, который будет записываться в текстовое поле. Для того, чтобы это происходило, необходимо к текстовому полю прописать обработчик события со значением **input="styleCSS = $event.target.value"**. Значение это стандартное и позволяет обратиться именно к значению данного текстового поля.

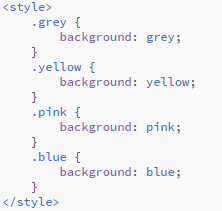
В скрипте в настройке data добавим новую переменную styleCSS:



Значение переменной оставим пустым, т.к. оно будет заполняться в зависимости от того, что вводится в поле input.



Далее к нашему документу подключим стили (можно отдельным файлом), где пропишем следующее:



Далее подключаем к нашему абзацу в HTML CSS – класс через директиву **v-bind**:



Теперь у нас при введении в текстовое поле значения класса будет срабатывать стиль CSS и изменяться цвет фона по текстом:



Можно использовать сокращенную форму записи директивы:



**Условия, списки и циклы**

Рассмотрим, как во Vue определяются условия на следующем примере. Пусть у нас в HTML будет помещен абзац с текстом. В теге <p> пропишем следующую инструкцию: <p **v-if = "show">**, где show - это переменная.



Создадим ее в data и первоначально она будет иметь значение true:



Если мы протестируем пример, то мы увидим надпись, помещенную внутри абзаца. Если в переменной мы заменим значение переменной show на false, то надпись исчезает.

Модернизируем пример. Добавим кнопку с надписью: «Скрыть/Показать» и добавим к кнопке обработчик события, который будет изменять значение true на противоположное.



Протестируем пример и убедимся, что при нажатии кнопки текст то появляется, то исчезает.

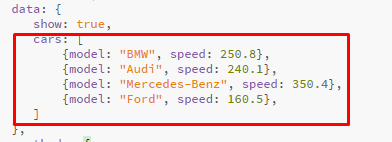
Кроме условной конструкции **v-if**, существует условная конструкция **v-else**.

Рассмотрим на примере. Добавим еще один абзац, куда поместим другой текст:



Тестируем пример и видим, что у нас тексты заменяются в зависимости от условия.

**Списки (массивы)** создаются во Vue,js следующим образом:



Выведем определенную модель автомобиля в виде элемента списка. Для этого в HTML – коде организуем список и обращаемся в нужному нам элементу массива:



Будет выведена модель Mercedes-Benz, т.к. именно она занимает элемент в массиве под индексом 2.

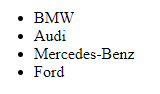


Такой способ вывода элементов из массива не всегда эффективен. Чтобы вывести сразу все элементы массива используются **циклы**.

Для организации цикла используют конструкцию **v-for**. Далее мы перебираем массив cars и заносим его элементы в переменную, в нашем случае это car. Для вывода только модели автомобиля оставляем {{car.model}}:

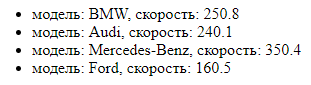


Убедимся, что все работает:



Чтобы вывести еще и скорость, можно дописать строку:





Элементы из массива мы можем вызвать с ключами (индексами). Делается это следующим образом:

