**Web-дизайн**

Лабораторная работа №6

**Основы JavaScript**

JavaScript – это язык программирования, который активно используется в HTML и при разработке веб-приложений.

**Запуск JavaScript**

Написать и запустить JavaScript можно двумя способами: первый заключается в том, что мы пишем код прямо на HTML странице внутри тега <script>:

<html>

<head>

<script>

Здесь код JS

</script>

</head>

<body>

Это основное содержимое страницы.

</body>

</html>

Тег **<script>** можно располагать в любом месте страницы – как в <head>, так и в <body>. Можно также указывать атрибут *type="text/javascript"*.

Второй вариант заключается в том, что JavaScript код хранится в отдельном файле (наподобие CSS) и подключается тоже с помощью тега <script> с атрибутом src, в котором указывается путь к файлу со скриптом:

<html>

<head>

<script src="путь к файлу со скриптом"></script>

</head>

<body>

Это основное содержимое страницы.

</body>

</html>

**Синтаксис JavaScript**

JavaScript – это язык, *чувствительный* к регистру символов. Рассмотрим некоторые понятия, относящиеся к синтаксису языка.

**Переменные**

Стандартный элемент языка программирования –– переменная. В JavaScript при объявлении переменной обязательно должно быть написано ключевое слово **var** (или **let**):

var a; //тут мы объявили переменную

var a, b, a1, isVar, is\_var; //тут мы объявили группу переменных

**Константы**

Константами могут выступать значения простейших типов. Также с помощью ключевого слова **const** можно определить константу, которая, как и переменная, хранит значение, однако это значение не может быть изменено.

const rate = 10; //ее нужно сразу инициализировать

**Разделитель «;»**

Как и в других языках программирования, для отделения инструкций друг от друга в языке JavaScript используется точка с запятой (;).

Обычно в JavaScript точку с запятой между инструкциями можно не ставить, если они находятся в разных строках. Точку с запятой можно также опустить в конце программы или блока кода, ограниченного фигурными скобками. Пример:

a = 3

b = 5

a1 = 3; b1 = 5;

**Комментарии**

Можно оставлять комментарии, как и в других языках программирования:

//это пример однострочного комментария.

/\* Это пример

многострочного комментария.

\*/

**Простые типы данных**

**Числовые данные (Number)**

Числа в JavaScript могут иметь две формы:

* Целые числа. Диапазон используемых чисел: от -253 до 253.
* Числа с плавающей точкой (вещественные). У них тот же диапазон: от -253 до 253.

45;

23.897;

**Строки (String)**

Тип *string* представляет строки, то есть такие данные, которые заключены в кавычки. Причем можно использовать как двойные, так и одинарные кавычки.

'строка', "строка"; //это примеры строк

**Логические значения true и false (Boolean)**

Значения true и false обозначают *истину* и *ложь* соответственно. Стандартный логический тип данных.

**Значение undefined**

Значение *undefined* обозначает неопределенность. К примеру, если мы попробуем обратиться к переменной, которой мы еще не задали значение:

var a;

alert(a); //выведет undefined

**Значение null**

Значение *null* обозначает пустоту. К примеру, мы можем присвоить переменной значение *null* в знак того, что там ничего не лежит.

На первый взгляд это значение похоже на *undefined*, однако *undefined* – это неопределенное значение, а *null* – определенное.

**Операции**

**Математические операции**

В JavaScript между числами можно совершать различные математические операции:

alert(2 + 3); //сложение - выведет 5

alert(5 - 1); //вычитание - выведет 4

alert(2 \* 3); //умножение - выведет 6

alert(6 / 2); //деление - выведет 3

alert(7 % 3); //деление по модулю (остаток от деления) - выведет 1

**Операции инкремента и декремента**

Операция **a++** или **++a** – увеличивает переменную a на единицу – инкремент.

Операция **a--** или **--a** – уменьшает переменную a на единицу – декремент.

**Операции присваивания**

Стандартный элемент языка программирования – операция присваивания. Пример:

var a = 4; //мы присвоили переменной a значение 4

Операторы математического действия и присваивания:

var a = 10;

a += 3; //этот код полностью эквивалентен коду a = a + 3;

a -= 3; //этот код полностью эквивалентен коду a = a - 3;

a \*= 3; //этот код полностью эквивалентен коду a = a \* 3;

a /= 3; //этот код полностью эквивалентен коду a = a / 3;

a %= 3; //этот код полностью эквивалентен коду a = a % 3;

**Операции сравнения**

Операции сравнения сравнивают два значения и возвращают значение *true* или *false*. Есть такой набор операций: == (равенство), === (тождественность, сравнивает значение и тип), != (неравенство), !== (не тождественность), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).

**Логические операции**

Логические операции применяются для объединения результатов двух операций сравнения. В JavaScript есть следующие логические операции: логическое И (&&), логическое ИЛИ (||), логическое НЕ (!).

**Операции со строками**

Строки могут использовать оператор + для объединения – конкатенация строк:

var name = "Том";

var fullname = name + 256;

alert(fullname); //вывод - Том256

Также в JavaScript можно получить доступ к определенному символу строки по его номеру таким образом: a[n] – n-ный символ строки (учтите, что нумерация идет с нуля):

var a, b; //объявим наши переменные

a = 'abcde'; //строка 'abcde'

b = a[0]; //в переменной b будет 'a'

c = a[4]; //в переменной c будет 'e'

**Вывод данных**

У JavaScript есть несколько различных способов «отобразить» данные:

* Запись в HTML элемент при помощи свойства **innerHTML**.
* Запись в вывод HTML при помощи метода **document.write()**.
* Запись в консоль браузера при помощи метода **console.log()**.
* Запись в окно предупреждений при помощи метода **window.alert() / alert() –** выводит текст в диалоговом окне.

**Использование innerHTML**

Чтобы получить доступ к HTML элементу, JavaScript должен воспользоваться методом *document.getElementById(id)*. Атрибут *id* определяет идентификатор HTML элемента. Свойство *innerHTML* определяет выводимый HTML контент:

<p id="demo">Этот текст будет заменен новым</p>

<script>

document.getElementById("demo").innerHTML = "Новый текст";

</script>

Изменение свойства **innerHTML** элемента HTML это обычный способ вывода данных в HTML.

**Использование document.write()**

В тестовых целях для вывода данных можно использовать метод document.write():

<script>

document.write(5 + 6);

</script>

Следует помнить, что использование метода **document.write()** после полной загрузки HTML документа удалит весь существующий HTML код. Метод **document.write()** рекомендуется использовать для тестирования.

**Использование console.log()**

Во время отладки скрипта, чтобы увидеть некие данные, вы можете вывести их в консоль браузера. Для этого используется метод **console.log()**:

<script>

console.log(5 + 6);

<script>

**Получение данных пользователя**

**Функция prompt**

Кроме функции *alert*, которая выдает диалоговое окошко, существует функция *prompt*, которая не только выдает окошко с текстом, но и позволяет получить от пользователя какой-либо текст.

Этот текст можно записать в переменную и затем выполнить над ним какие-либо операции.

var name = prompt('Ваше имя?'); //результат – строка

alert('Ваше имя: '+name);

**Функция confirm**

Если вам нужно просто спросить у пользователя 'Да' или 'Нет', не давая ему возможность ввести иной текст – используйте функцию confirm.

Эта функция вызывает окошко с вопросом, на который нужно ответить пользователю, и двумя кнопками для ответа: с кнопкой 'ОК' (*true*) и с кнопкой 'Отмена' (*false*).

var ok = confirm(Да или нет?'); //результат – булево значение

alert(ok);

**Условные конструкции**

**Конструкция if-else**

if (логическое выражение) {

Этот код выполниться,

если логическое выражение верно (то есть равно true)

} else {

Этот код выполнится,

если логическое выражение неверно (то есть равно false)

}

Блок else не обязателен.

var a = 3;

if (a > 0) {alert('Верно!');} else {alert('Неверно!');} //выведет 'Верно!'

**Логическое выражение**

Для составления логического выражения можно использовать операции сравнения и логические операции:

var a = 3;

var b = -3;

if (a > 0 && b < 0) alert('Верно!'); else alert('Неверно!'); //выведет 'Верно!'

var a1 = -3;

var b1 = -3;

if (a1 > 0 || b1 < 0) alert('Верно!'); else alert('Неверно!');

**Равенство по значению и типу**

Оператор тожественности (===) в отличии от равенства (==) сравнивает операнды не только по значению, но и по типу.

var a = '0'; //переменная a представляет собой строку, а не число 0

if (a == 0) alert('Верно!'); else alert('Неверно!');

var a = '0'; //переменная a представляет собой строку, а не число 0

if (a === 0) alert('Верно!'); else alert('Неверно!');

**Конструкция switch-case**

Данная конструкция представляет собой альтернативу if-else, ее рекомендуется использовать в случае множественного выбора.

switch (переменная) {

case 'значение1':

код;

break;

case 'значение2':

код;

break;

…

default:

код;

break;

}

**Тернарная операция**

Тернарная операция состоит из трех операндов и имеет следующее определение:

[первый операнд - условие] ? [второй операнд] : [третий операнд]

Если условие равно *true*, то возвращается второй операнд; если условие равно *false*, то третий. Например:

var a = 1, b = 2;

var result = a < b ? a + b : a - b; // выполнится a + b

alert(result); // 3

**Директива let**

Директива **let** позволяет объявить локальную переменную с областью видимости, ограниченной текущим блоком кода. В отличие от ключевого слова [*var*](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Statements/var), которое объявляет переменную глобально или локально во всей функции, независимо от области блока.

Пример с *var*:

var x = 1;

if (true) {

var x = 2; // та же переменная x

alert(x); // 2

}

alert(x); // 2

Пример с ***let***:

let x = 1;

if (true) {

let x = 2; // другая переменная

alert(x); // 2

}

alert(x); // 1

**Массивы**

Это один из сложных типов данных.

**Объявление массива**

Для работы с наборами данных предназначены массивы. Для создания массива применяется выражение **new Array()** (или просто **Array**()):

var arr = new Array();

Также можно создать массив с помощью квадратных скобок [ ]:

var arr = [];

Пока созданный нами массив не содержит никаких данных. Заполним его названиями дней недели:

var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];

//В массиве можно хранить как строки, так и числа:

var arr = ['пн', 256, 'ср', 34, 38, 'сб', 95];

//Выведем слово 'ср':

alert(arr[2]);

var arr = Array('пн', 256, 'ср', 34, 38, 'сб', 95);

Разница между способами проявляется в случае, если вы хотите сделать массив, состоящий из одного элемента, который будет целым числом, вот так:

var arr = Array(10);

В этом случае получим массив, состоящий из 10-ти пустых элементов (их значение будет *undefined*), а не массив из одного элемента 10, как в данном случае:

var arr = [10];

**Добавление новых элементов в массив**

Массив можно заполнять не только на этапе его создания. Можно добавлять новые элементы уже после. К примеру:

var arr = []

arr[0] = 1;

alert(arr) //с помощью alert выводим содержимое массива

**Многомерный массив**

Элементы массива могут быть не только строками и числами, но и массивами. То есть у нас получится  многомерный массив.

//двумерный массив студентов:

var students = [

['Коля', 'Маша', 'Петя'],

['Даша', 'Вася', 'Лена'],

];

alert(students[0][0]); //так выведем 'Коля'

**Циклы**

**Цикл while**

while ( пока выражение истинно ) {

выполняем этот код циклически;

в начале каждого цикла проверяем выражение в круглых скобках

}

Пример:

var i = 0; //счетчик цикла

while (i < 5) {

i++;

alert(i);

}

**Цикл do-while**

do {

//..блок операторов..

} while (условие);

Пример:

var a = 1;

do{

a\*=2;

alert(a);

} while(a<1000);

**Цикл for**

for ( начальные команды; условие цикла; команды после прохода цикла ) {

тело цикла

}

Пример:

for (var i = 0; i < 10; i++) {

alert(i); //выведет 0, 1, 2... 9

}

**Цикл for-in**

Для перебора массивов и объектов можно использовать цикл **for-in**.

for (key in obj) {

тело цикла

}

*obj* ­– это массив/объект, который мы перебираем, а *key* – это переменная, в которую последовательно будут помещаться индексы/ключи элемента.

Пусть у нас дан такой массив:

var obj = [10, 20, 30];

for (key in obj) {

alert(obj[key]); //выведет 10, 20, 30

}

**Инструкция break и continue**

Иногда нам необходимо прервать выполнение цикла досрочно. Такое можно сделать с помощью инструкции **break** .

Существует также инструкция **continue**, при достижении которой цикл начинает новую итерацию.

**Первая программа на JavaScript**

Создадим первую программу на javascript. Допустим мы хотим, чтобы при запуске веб-страницы возникало диалоговое окно, в которое нам необходимо было бы ввести данные. Далее возникало бы второе окно, которое спрашивало, как отобразить наши данные: в еще одном диалоговом окне или на странице.

Создадим веб-страницу под названием *index.html*. На эту страницу добавим следующий код:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>JavaScript</title>

</head>

<body>

<p id="id1"></p>

<script>

var res = prompt("Введите данные");

var b = confirm("Вывести данные в диалоговом окне?");

if(b)

alert(res);

else

document.getElementById("id1").innerHTML=res;

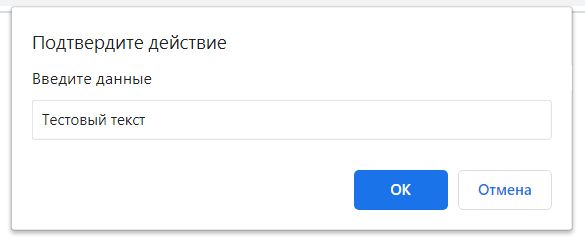
</script>

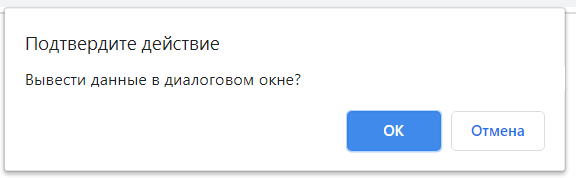
</body>

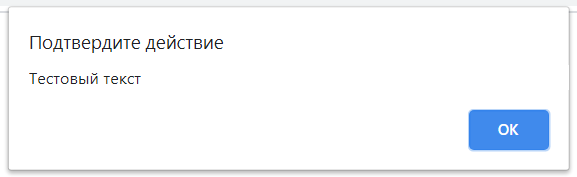
</html>

Не забудьте проследить, чтобы файл страницы был в кодировке UTF-8, иначе кириллица отобразится не правильно.

В результате работы скрипта будет, к примеру, следующей:







Знакомство с JS - <https://webref.ru/dev/learn-javascript>

Самоучитель JS - <https://learn.javascript.ru/>

Самоучитель JS - <https://metanit.com/web/javascript/>

Справочник JS - <https://javascript.ru/manual>

Справочник JS - <https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference>

**Общие задания**

Выполните следующие задания с использованием JS:

1. Считайте введенное пользователем число. Если число больше 0, то выведите квадрат этого числа на страницу в какой-нибудь элемент, если меньше – выведите в консоль браузера. Если же число равно 0, то выведите сообщение об этом через *alert*(). Если введено не число, то выведите сообщение об ошибке в консоль браузера.
2. Создайте каталог *image* рядом со страницей со скриптом. Поместите в нее несколько картинок (к примеру, img1.jpg, img2.jpg, img3.jpg и т.д.). Имена картинок храните в массиве. Отобразите на странице картинку из массива под номером, введенным пользователем. Если пользователь ввел не подходящее число, выведите сообщение об этом. Можете сделать вариант с URL-ссылками на картинки.
3. Создайте страницу, на которой создается прямоугольный блок div со случайными размерами (в некоторых пределах) и случайным цветом (можно воспользоваться функцией *rgb()* для этой цели). Для этого понадобится метод *Math.random()*. В центре div-а запишите его площадь.
4. С помощью *prompt* и цикла считайте данные пользователя в массив (изначально массив пустой). Количество элементов определите сами. Выводите этот массив на страницу.
5. Генерируется случайное число от 0 до 100. С помощью *prompt* и цикла пользователь должен угадать число. После каждой попытки ему выдается подсказка «больше» или «меньше» загаданное число, чем введенное им, а также на страницу добавляется элемент (к примеру, блок div) с введенным числом пользователя (для этого может подойти *document.write()*). Таким образом, когда пользователь угадает число, можно будет на странице увидеть все введенные им варианты.

**Индивидуальное задание**

Дополните сайт, разработанный по выбранной теме по индивидуальному заданию прошлой лабораторной работы.

Сделайте так, чтобы цвета или градиенты на элементах генерировались случайным образом.

Отображайте текущую дату в шапке страницы.

Сделайте так, чтобы на основе текущего времени менялась шапка при загрузке страницы. К примеру, если сейчас четная минута, то в шапке отображается текст темы сайта, а если нет – имя создателя сайта.

Добавьте гиперссылку, которая открывает случайную страницу сайта. Это можно организовать записью случайной ссылки при загрузке страницы. Возможные варианты храните в массиве. При загрузке страницы в эту гиперссылку записывается случайный адрес из массива.