Министерство образования и науки Российской Федерации

Уральский Государственный Экономический Университет

Отчёт по основам JavaScript

Институт менеджмента и информационных технологий

Исполнитель: Архипова М.М.

Направление (специальность) Информатика и вычислительная техника

Группа ИВТ-14-1

Кафедра статистики, эконометрики и информатики

Преподаватель Коршунов М.К.

(должность, звание)

Екатеринбург

2017

СОДЕРЖАНИЕ

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №1: ДИАЛОГОВЫЕ ОКНА ВВОДА-ВЫВОДА, ДАННЫЕ, ПЕРЕМЕННЫЕ

В любое место html-документа можно вставить контейнер script, в теле которого написать сценарий на JavaScript. Например:

```
<script>
alert("Hello world!");
</script>
```

При этом опреление функции может быть в одном script-контейнере, а вызов в другом; т.е. код доступен из любого места html-документа.

Для подключения внешнего файла с кодом на JavaScript используется следующий синтаксис:

```
<script src="/путь/имя.js"></script>
```

Для гарантированного правильного отображения символов в браузере используется следующий тег, вставляемый в контейнер head:

```
<meta charset="utf-8">
```

Перед тегом html указывается doctype. В настоящее время достаточно написать так:

```
<!DOCTYPE HTML>
```

Методы ввода-вывода данных

В JavaScript существует три стандартных метода для ввода и вывода данных: alert, prompt и confirm.

Метод alert позволяет выводить диалоговое окно с заданным сообщением и кнопкой ОК. Синтаксис: alert (сообщение).

Метод confirm позволяет вывести диалоговое окно с сообщением и двумя кнопками - ОК и Отмена (Cancel). В отличие от метода alert этот метод возвращает логическую величину, значение которой зависит от того, на какой из двух кнопок щелкнул пользователь. Если он щелкнул на кнопке ОК, то возвращается значение true (истина, да); если он щелкнул на кнопке Отмена, то возвращается значение false (ложь, нет). Синтаксис: confirm (сообщение).

Метод prompt позволяет вывести на экран диалоговое окно с сообщением, а также с текстовым полем, в которое пользователь может ввести данные. Кроме того, в этом окне предусмотрены две кнопки: ОК и Отмена (Cancel). В отличие от методов alert и confirm данный метод принимает два параметра: сообщение и значение, которое должно появляться в текстовом поле ввода данных по умолчанию. Синтаксис: prompt (сообщение, значение_поля_ввода_данных).

Типы данных

Данные которые хранятся в памяти компьютера и подвергаются обработке, можно отнести к различным типам. В зависимости от их типа над данными можно выполнять те или иные операции. Например, операция деления применяется к данным числового типа, а вот разделить строку на строку или строку на число нельзя.

Числа как данные числового типа всегда записываются без кавычек. Строковые данные заключаются в кавычки. Поэтому если число заключить в кавычки, то оно уже будет являться строкой.

Переменные и оператор присвоения

Чтобы сохранять данные в памяти и в то же время оставлять их доступными для дальнейшего использования, в программах используют переменные.

Переменную можно считать контейнером для хранения данных. Данные, сохраняемые в переменной, называются значениями этой переменной.

Переменная имеет имя - последовательность букв, цифр и символа подчеркивания без пробелов и знаков препинания, начинающуюся обязательно с буквы или символа подчеркивания.

Оператор присвоения значения переменной обозначается символом равенства "=". Не следует путать этот оператор с отношением равенства и соответствующей операцией сравнения.

Выражение с оператором "=" интерпретатор вычисляет следующим образом: сначала вычисляется значение справа от знака равно, после чего результат присваивается переменной слева.

Если x и y две переменные, то выражение x = y интерпретируется так: переменной x присваивается значение переменной y.

ЛИСТИНГИ ПО УРОКУ №1

Листинг 1. Методы ввода-вывода данных

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            alert("Hello World!");
            qu=confirm("Can close the window?");
            document.writeln(qu+';');
            uname=prompt("Enter       your       name",
"Anonymous");
            document.writeln("Your name is "+uname);
        </script>
    </body>
</html>
```

Листинг 2. Типы данных

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
```

```
        <ul>
            <li> Сложение для числовых типов данных:
                <script>
                    x=5
                    y=x+3
                    document.writeln(y)
                </script>
            </li>
            <li> Конкатенация для строкового типа
данных:
                <script>
                    a="5"
                    b=a+"3"
                    document.writeln(b)
                    c="Большая "+"книга"
                    document.writeln(c)
                </script>
            </li>
        </ul>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №1

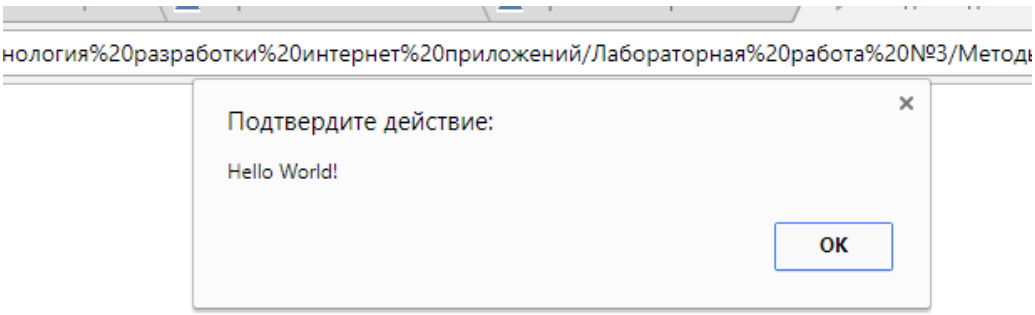


Рисунок 1 – Метод alert

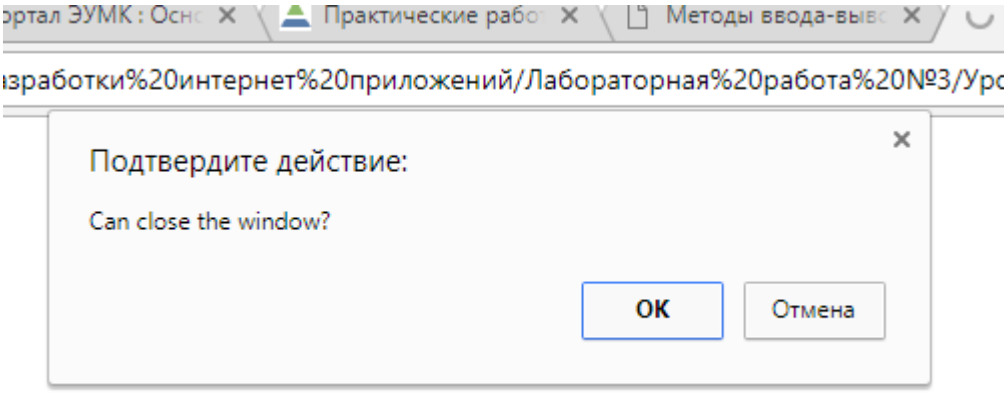


Рисунок 2 – Метод confirm

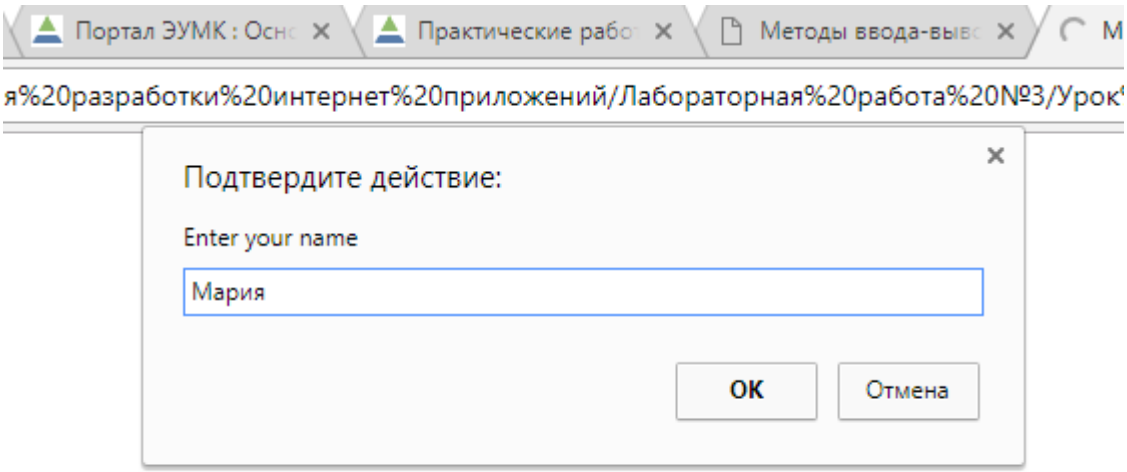


Рисунок 3 – Метод prompt

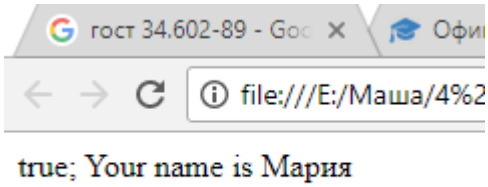


Рисунок 4 – Считывание значения методом prompt

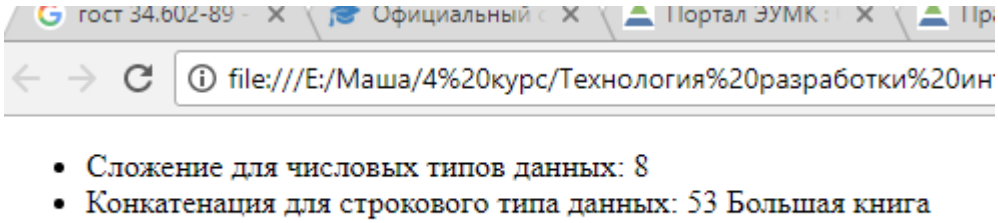


Рисунок 5 – Работа с различными типами данных

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №2: ОПЕРАТОРЫ УСЛОВНОГО ПЕРЕХОДА

Вычислительный процесс можно направить по тому или иному пути в зависимости от того, выполняется ли некоторое условие или нет.

Оператор if (если)

Синтаксис:

if (условие)

{ код, который выполняется, если условие выполнено }

else

{ код, который выполняется, если условие не выполнено }

Часть этой конструкции, определяемая ключевым словом else (иначе), необязательна.

Оператор switch (переключатель)

Синтаксис:

```
switch (выражение) {
    case значение1:
        код
        [break]
    case значение2:
        код
        [break]
        ...
    [default:
        код]
```

```
}
```

Сначала вычисляется выражение, указанное в круглых скобках сразу за ключевым словом switch. Полученное значение сравнивается с тем, которое указано в первом варианте. Если они не совпадают, то код этого варианта не выполняется и происходит переход к следующему варианту. Если же значения совпали, то выполняется код, соответствующий этому варианту. При этом, если не указан оператор break, то выполняются блоки кода всех остальных (расположенных ниже) веток case.

ЛИСТИНГИ ПО УРОКУ №2

Листинг 3. Оператор if

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            num=10
            if(num<50)
                alert("Мало!")
            else
                alert("Много!")
        </script>
    </body>
</html>
```

Листинг 4. Оператор switch

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
```

```
    </head>
    <body>
        <script>
            x=2
            switch(x)
            {
                case 1: alert("Мало!")
                case 2: alert("Средне!")
                case 3: alert("Много!")
            }
        </script>
    </body>
</html>
```

Листинг 5. Оператор switch с использованием оператора break

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            x=2
            switch(x)
            {
                case 1: alert("Мало!")
                    break
                case 2: alert("Средне!")
                    break
                case 3: alert("Много!")
```

```
                    break
                }
            </script>
        </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №2

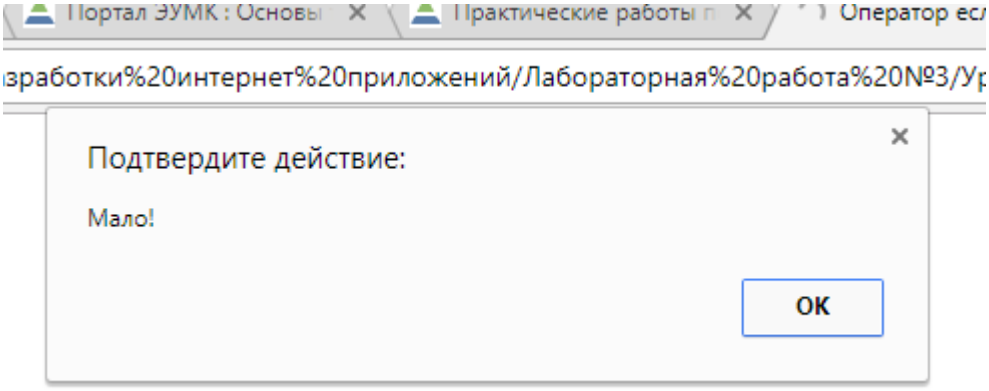


Рисунок 6 – Работа оператора if

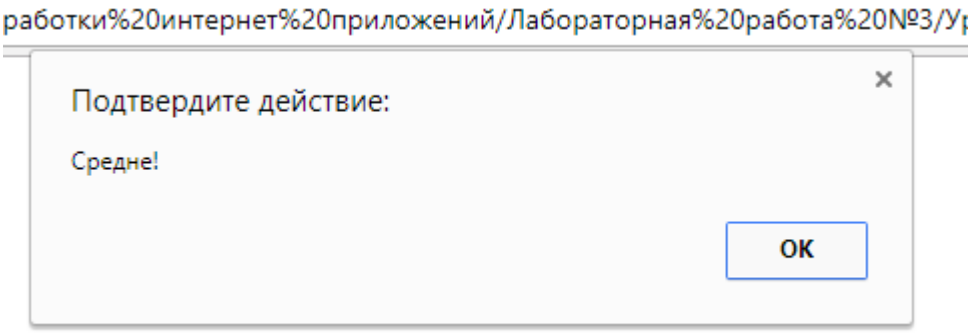


Рисунок 7 – Работа оператора switch

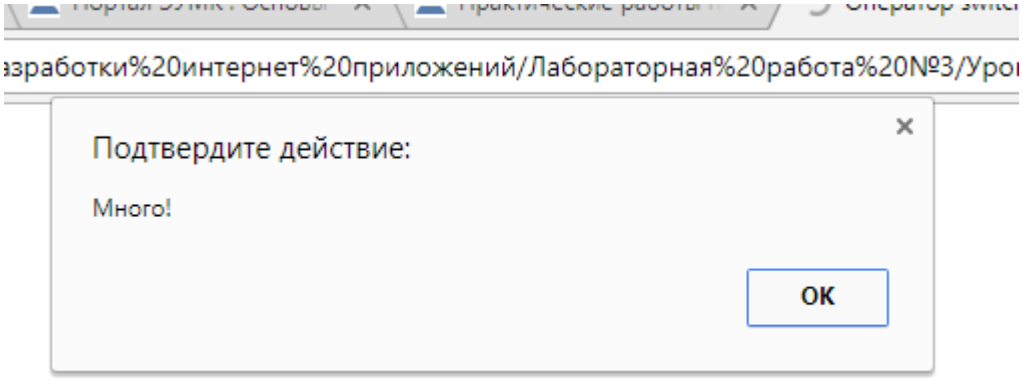


Рисунок 8 – Работа оператора switch

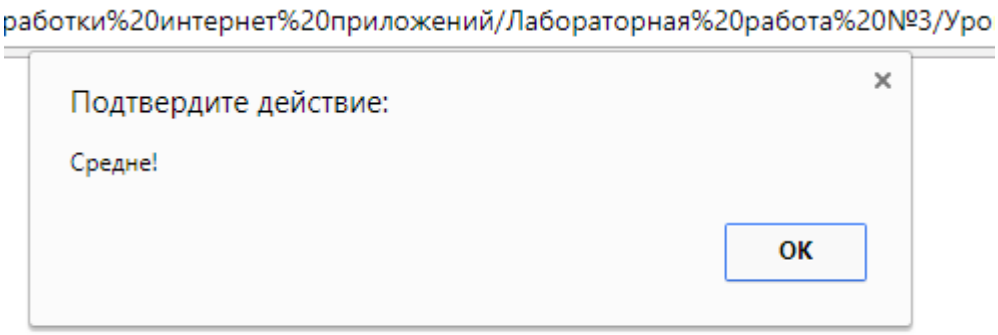


Рисунок 9 – Работа оператора switch с оператором break

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №3: ОПЕРАТОРЫ ЦИКЛА

Операторы цикла обеспечивают многократное выполнение блока программного кода до тех пор, пока выполняется заданное условие.

Оператор for (для)

Синтаксис:

for ([счетчик] ; [условие] ; [изменение_счетчика]) {

код

}

Оператор while (пока)

Синтаксис:

while (условие) {

код

изменение_переменной

}

Оператор do-while (делай до тех пор, пока)

Синтаксис:

do {

код

изменение_переменной

} while (условие)

ЛИСТИНГИ ПО УРОКУ №3

Листинг 6. Оператор for

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            var s=0
            for(i=1; i<=4; i++)
            {
                s=s+i
            }
            document.writeln(s)
        </script>
    </body>
</html>
```

Листинг 7. Оператор while

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            var s=0
            i=1
            while(i<=4)
            {
```

```
                s=s+i
                i++
            }
            document.writeln(s)
        </script>
    </body>
</html>
```

Листинг 8. Оператор do-while

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            var s=0
            i=1
            do
            {
                s=s+i
                i++
            }while(i<=4)
            document.writeln(s)
        </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №3

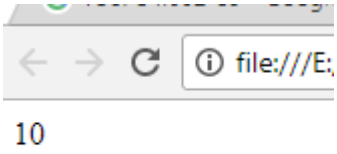


Рисунок 10 – Результат работы оператора for

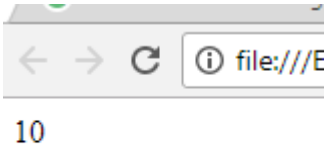


Рисунок 11 – Результат работы оператора while

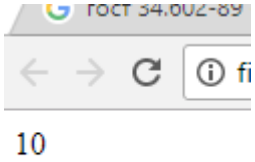


Рисунок 12 – Результат работы оператора do-while

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №4: НЕКОТОРЫЕ ВСТРОЕННЫЕ ФУНКЦИИ

Функция представляет собой подпрограмму, которую можно вызвать для выполнения, обратившись к ней по имени.

В JavaScript есть встроенные функции, которые можно использовать в программах.

parseInt(строка) - преобразует указанную строку в целое число.

parseFloat(строка) - преобразует указанную строку в число с плавающей разделительной (десятичной) точкой.

isNaN (значение) - возвращает true, если указанное в параметре значение не является числом, иначе - false.

eval (строка) - вычисляет выражение в указанной строке.

ЛИСТИНГИ ПО УРОКУ №4

Листинг 9. Функция parseInt

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            a=parseInt("2.5")
            b=parseInt("10 apples")
            c=parseInt("apples 10")
            d=parseInt("-15.87")
            document.writeln(a)
            document.writeln(b)
            document.writeln(c)
            document.writeln(d)
        </script>
    </body>
</html>
```

Листинг 10. Функция parseFloat

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            a=parseFloat("2.5")
```

```
            b=parseFloat("10 apples")
            d=parseFloat("-15.87")
            document.writeln(a)
            document.writeln(b)
            document.writeln(d)
        </script>
    </body>
</html>
```

Листинг 11. Функция isNaN

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            a=isNaN("2.5")
            b=isNaN("10 apples")
            d=isNaN("-15.87")
            document.writeln(a)
            document.writeln(b)
            document.writeln(d)
        </script>
    </body>
</html>
```

Листинг 13. Функция eval

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
```

```
    </head>
    <body>
        <script>
            a=5
            x="if(a<10){a+2}"
            document.writeln(x)
            x=eval("if(a<10){a+2}")
            document.writeln(x)
        </script>
    </body>
</html>
```

Листинг 14. Функция eval второй пример

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h3> Редактор кодов JavaScript</h3>
        <p>
            Код: <br>
            <textarea id="user_code" rows="10"
cols="60"></textarea>
        </p>
        <p>
            Результат: <br>
            <textarea id="result" rows="3"
cols="60"></textarea>
        </p>
        <p>
```

```
            <button
onclick="document.all.result.value=eval(user_code.value
)"> Выполнить </button>
            <button
onclick="document.all.user_code.value='';
document.all.result.value=''"> Очистить </button>
        </p>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №4

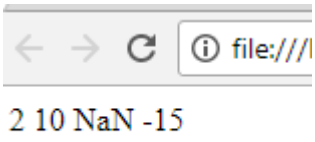


Рисунок 13 – Применение функции parseInt

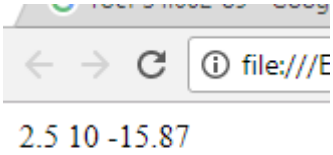


Рисунок 14 – Применение функции parseFloat

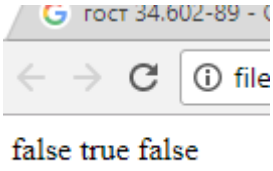


Рисунок 15 – Применение функции isNaN

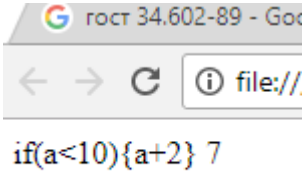


Рисунок 16 – Применение функции eval

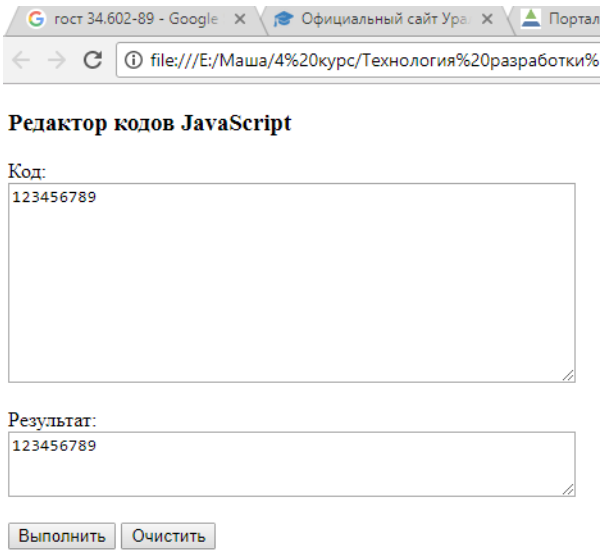


Рисунок 17 – Второй пример применения функции eval

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №5: ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Пользовательские функции - это функции, которые вы можете создавать сами, по своему усмотрению, для решения своих задач.

Определение функции имеет следующий синтаксис:

function имя_функции (параметры) {

код

}

Если требуется, чтобы функция возвращала некоторое значение, то в ее теле используется оператор возврата return с указанием после него того, что следует возвратить. В качестве возвращаемой величины может выступать любое выражение: простое значение, имя переменной или вычисляемое выражение. Оператор return может встречаться в коде функции несколько раз. Впрочем, возвращаемую величину, а также сам оператор return можно не указывать. В этом случае функция ничего не будет возвращать.

Функция не выполняется до тех пор, пока не будет вызвана. Выражение вызова имеет следующий синтаксис:

имя_функции (аргументы)

ЛИСТИНГИ ПО УРОКУ №5

Листинг 15. Функция вычисления площади прямоугольника

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function sq_rect(width, height)
            {
                s=width*height
                return s
            }
            x=sq_rect(5,8)
            document.writeln(x)
        </script>
    </body>
</html>
```

Листинг 16. Функция вычисления факториала

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function factorial(n)
            {
                if(n<=1) return 1
```

```
                result=2
                for(i=3; i<=n; i++)
                {
                    result=result*i
                }
                return result
            }
            x=factorial(7)
            document.writeln(x)
        </script>
    </body>
</html>
```

Листинг 17. Вызов функции при нажатии на кнопку

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <title>Пример</title>
        <script>
            function calculate()
            {
                a=prompt("Первое число")
                b=prompt("Второе число")
                a=parseInt(a)
                b=parseInt(b)
                alert(a+b)
            }
        </script>
    </head>
    <body>
```

```
        <form>
            <input    type="button"    value="Посчитать"
onclick="calculate()">
        </form>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №5

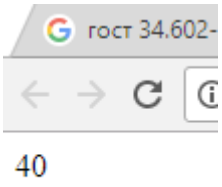


Рисунок 18 – Результат функции, вычисляющей площадь прямоугольника

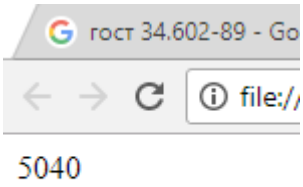


Рисунок 19 – Результат функции, вычисляющей факториал

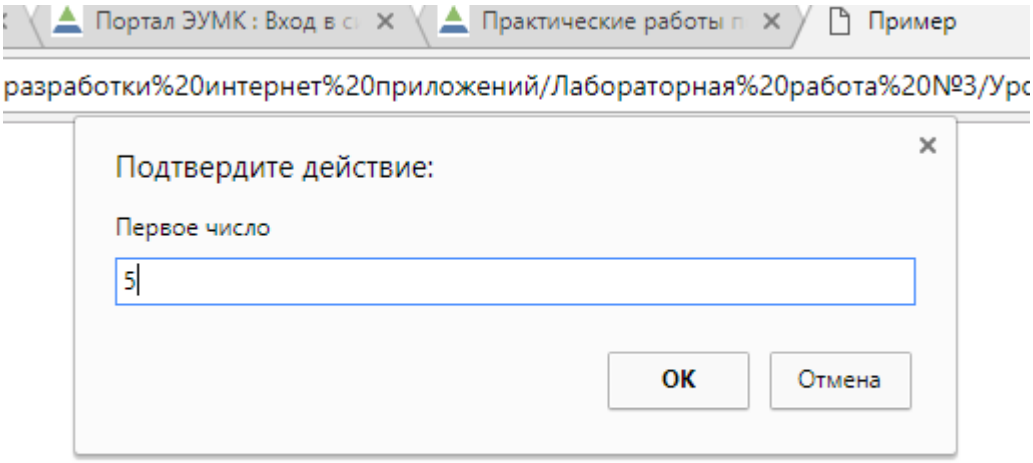


Рисунок 20 – Вызов функции при нажатии на кнопку

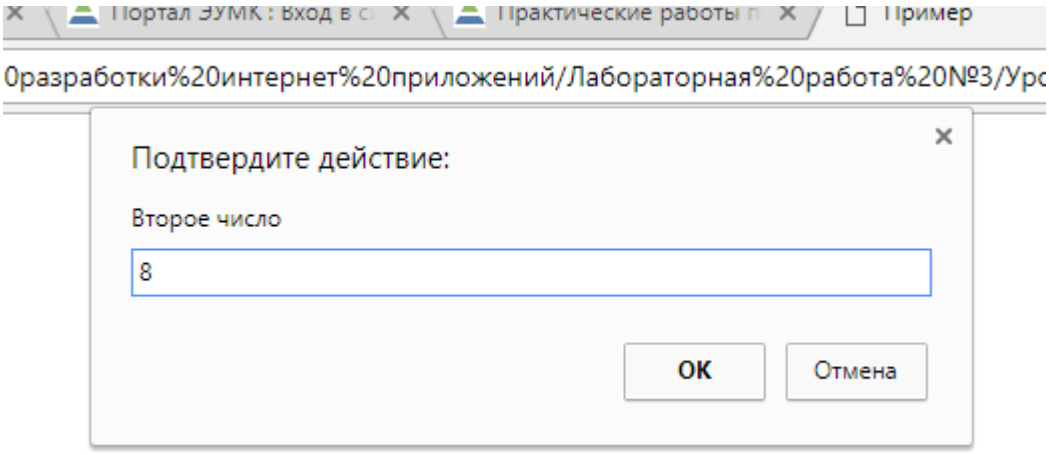


Рисунок 21 – Вызов функции при нажатии на кнопку

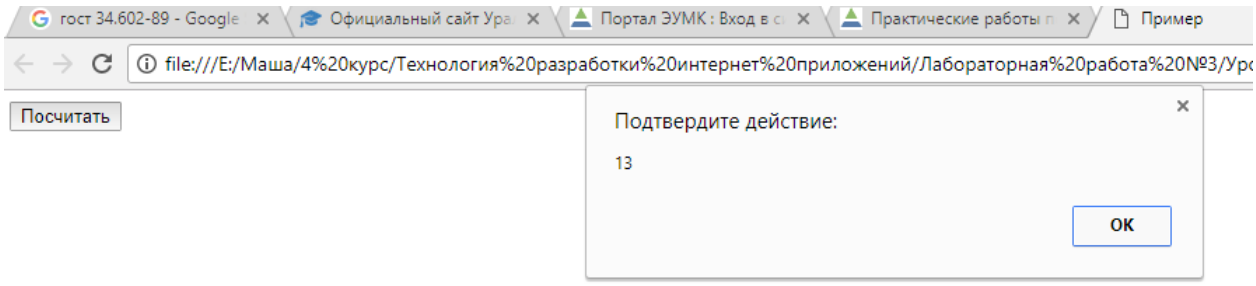


Рисунок 22 – Результат работы функции, вызывающейся по кнопке

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №6: ВСТРОЕННЫЕ ОБЪЕКТЫ. STRING

Объекты представляют собой программные единицы, обладающие некоторыми свойствами. Об объекте мы можем судить по значениям его свойств и описанию того, как он функционирует.

Встроенные объекты имеют фиксированные названия, свойства и методы. Свойства имеют имена и значения. Значение большинства свойств можно менять. Методы аналогичны функциям. Они могут принимать аргументы или нет.

Чтобы узнать значение свойства объекта, необходимо указать имя этого объекта и имя свойства, отделив их друг от друга точкой: имя_объекта.свойство.

Мы можем заставить объект выполнить тот или иной присущий ему метод. В этом случае также говорят о применении метода к объекту.

Синтаксис соответствующего выражения такой: имя_объекта.метод(параметры).

Объект String (строка)

Создание строкового объекта

Для создания строкового объекта используют выражение следующего вида:

имя_переменной = new String ("строковое_значение")

Однако можно создавать строковый объект и с помощью обычного оператора присваивания:

имя_переменной = "строковое_значение"

Методы обработки строк

charAt (индекс) - возвращает символ, занимающий в строке указанную позицию.

Синтаксис: строка.charAt(индекс)

Индекс является числом, индекс первого символа равен 0.

concat (строка) - конкатенация (склейка) строк.

Синтаксис: строка1.concat(строка2)

Этот метод действует так же, как и оператор "+" сложения для строк: к строке строка1 приписывается справа строка2.

substr (индекс [, длина]) - возвращает подстроку исходной строки, начальный индекс и длина которой указываются параметрами.

Синтаксис: строка.substr(индекс [, длина])

При обработке строк часто требуется вставить или заменить подстроки.

Создадим сами функцию вставки строки в исходную строку. Назовем ее insstr. Данная функция должна принимать три параметра: исходную строку s1, вставляемую строку s2 и индекс позиции вставки n.

ЛИСТИНГИ ПО УРОКУ №6

Листинг 18. Свойство length

```
<!DOCTYPE HTML>
<html>
```

```
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            mystr="What is it?"
            x=mystr.length
            document.writeln(x)
        </script>
    </body>
</html>
```

Листинг 19. Метод charAt

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            hi="Hello world!"
            a=hi.charAt(0)
            b=hi.charAt(6)
            c=hi.charAt(11)
            document.writeln(a,b,c)
        </script>
    </body>
</html>
```

Листинг 20. Метод concat

```
<!DOCTYPE HTML>
<html>
```

```
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            l=new String("abcd")
            b="ABCD"
            document.writeln(l.concat(b))
        </script>
    </body>
</html>
```

Листинг 21. Метод substr

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
    //анализ адреса
        <script>
            adr="petersmith@mail.com"
            i=adr.indexOf("@")
            user=adr.substr(0,i)
            domen=adr.substr(i+1)
            document.writeln("User: ", user, " Domen: ",
domen)
        </script>
    </body>
</html>
```

Листинг 22. Функция вставки строки в исходную строку

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function insstr(s1, s2, n)
            {
                return s1.slice(0,n)+s2+s1.slice(n)
            }
            x=insstr("abcd", " ! ",2)
            y=insstr("Sun Earth", " Moon", 3)
            document.writeln(x)
            document.writeln(y)
        </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №6

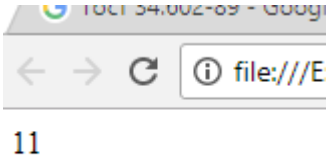


Рисунок 23 – Результат выполнения свойства length

Рисунок 24 – Результат работы метода charAt

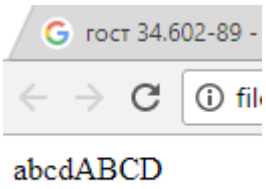


Рисунок 25 – Результат работы метода concat

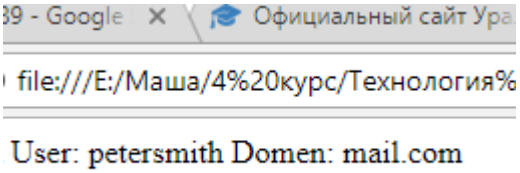


Рисунок 26 – Результат работы метода substr

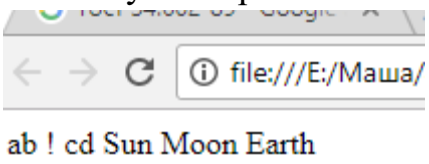


Рисунок 27 – Результат выполнения функции вставки строки в исходную строку

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №7: ОБЪЕКТ ARRAY (МАССИВ)

Массив представляет собой упорядоченный набор данных.

Существует несколько способов создания массива. В любом случае прежде всего создается новый объект массива:

имя_массива = new Array ([длина массива])

Методы Array

concat (массив) - конкатенация массивов, объединяет два массива в третий массив.

Синтаксис: имя_массива1.concat(массив2)

join(разделитель) - создает строку из элементов массива с указанным разделителем между ними.

Синтаксис: имя_массива.join(строка)

splice(индекс, количество [, элем1 [, элем2 [, ...элемN]]]) - удаляет из массива несколько элементов и возвращает массив из удаленных элементов или заменяет значения элементов.

Синтаксис: имя_массива.splice (индекс, количество [, элем1 [, элем2 [, ...элемN]]])

Функции обработки числовых массивов

Во многих приложениях требуется получить статистические характеристики числовых данных, хранящихся в виде массива: сумму всех чисел, среднее, максимальное и минимальное значение.

ЛИСТИНГИ ПО УРОКУ №7

Листинг 23. Создание массива earth

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            planet=new Array (4)
            planet[0]="Earth"
            planet[1]=24
            planet[2]=6378
            planet[3]=365.25
            document.writeln(planet[0])
            document.writeln(planet[3])
        </script>
    </body>
```

```
</html>
```

Листинг 24. Метод concat

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            a1=new Array(1,2,"star")
            a2=new Array("a","b","c","d")
            a3=a1.concat(a2)
            document.writeln(a3)
        </script>
    </body>
</html>
```

Листинг 25. Метод join

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            a1=new Array(1,2,"star")
            a2=new Array("a","b","c","d")
            a3=a1.concat(a2)
            document.writeln(a3.join("-"))
        </script>
    </body>
```

```
</html>
```

Листинг 26. Метод splice

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            arr=new
Array("Yellow","Red","Green","Blue", "White")
            arr.splice(1,3,"Darkgreen","Black")
            document.writeln("<br>", arr)
        </script>
    </body>
</html>
```

Листинг 27. Функция, возвращающая сумму (или конкатенацию) значений всех элементов

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function sum(arr)
            {
                var s=arr[0]
                for(i=1; i<arr.length; i++)
                    s+=arr[i]
```

```
            return s
        }
        a=new Array(1,5,"hello")
        b=new Array(2,3.2, 100, 21.3)
        document.writeln("<br>", sum(a))
        document.writeln(sum(b))
    </script>
</body>
</html>
```

Листинг 28. Функции, возвращающие минимальное и максимальное значение среди элементов массива

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function amin(arr)
            {
                var m=arr[0]
                for(i=1; i<arr.length; i++)
                    if(arr[i]<m)
                        m=arr[i]
                return m
            }
            function amax(arr)
            {
                var m=arr[0]
                for(i=1; i<arr.length; i++)
```

```
                    if(arr[i]>m)
                        m=arr[i]
                return m
            }
            a=new Array(15,5,89,12,104,45,34,9,110)
            document.writeln("<br>", amin(a))
            document.writeln(amax(a))
        </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №7

Рисунок 28 – Массив earth

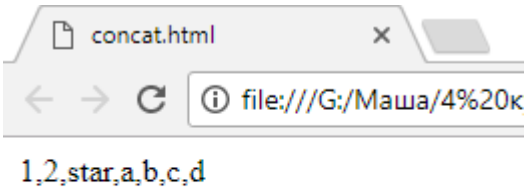


Рисунок 29 – Результат выполнения метода concat

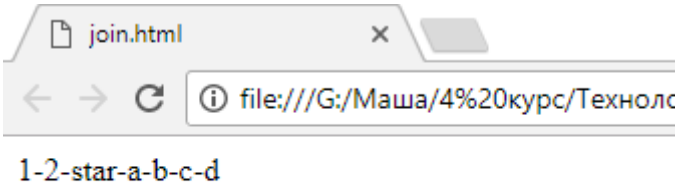


Рисунок 30 – Результат выполнения метода join

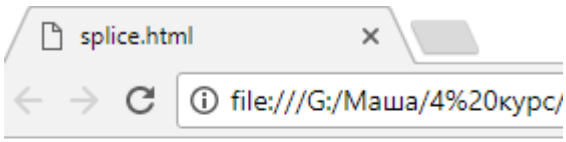


Рисунок 31 – Результат выполнения метода splice

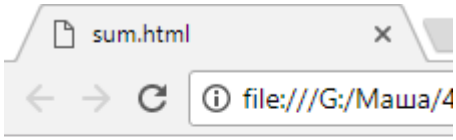


Рисунок 32 – Результат работы функции, возвращающей сумму (или конкатенацию) значений всех элементов

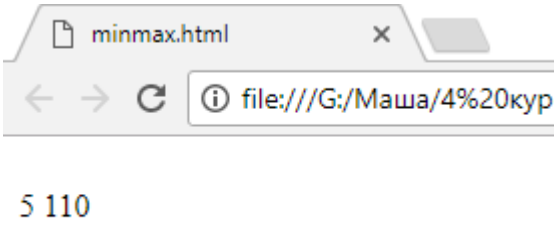


Рисунок 33 – Результат работы, функций, возвращающих минимальное и максимальное значение среди элементов массива

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №8: ОБЪЕКТЫ NUMBER, MATH И DATE

Числа в JavaScript можно представить в различных системах счисления, то есть в системах с различными основаниями: 10 (десятичной), 16 (шестнадцатеричной) и 8 (восьмеричной).

Объект Math (математика)

Объект Math предназначен для хранения некоторых математических констант (например, числа пи) и выполнения преобразования чисел с

помощью типичных математических функций. Доступ к свойствам и методам объекта Math обеспечивается следующими выражениями:

Math.свойство

Math.метод (параметры)

Объект Date (Дата)

Отображает дату и время.

ЛИСТИНГИ ПО УРОКУ №8

Листинг 29. Функция преобразования из десятеричной в шестнадцатеричную форму

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function to16(n10)
            {
                hchars="0123456789abcdef"
                if (n10>255) return null
                i=n10%16
                j=(n10-i)/16
                result="0x"
                result+=hchars.charAt(j)
                result+=hchars.charAt(i)
                return result
            }
            document.writeln(to16(16), " ", to16(100),
" ", to16(178))
        </script>
```

```
    </body>
</html>
```

Листинг 30. Функция преобразования из десятеричной в двоичную форму

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function to2(n10)
            {
                hchars="0123456789abcdef"
                if (n10<2) return ""+n10//чтобы результат был строковый
                i=n10%2
                j=(n10-i)/2
                return to2(j)+i
            }
            document.writeln("<p>", to2(7), " ", to2(15), " ", to2(178))
        </script>
    </body>
</html>
```

Листинг 31. Функция для решения квадратного уравнения

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
```

```
</head>
<body>
    <script>
        function q_eq(a,b,c)
        {
            var root=new Array()
            var d=b*b-4*a*c
            if(a==0)
            {
                if(!(b==0))
                {
                    root[0]=-c/b
                    root[1]=null
                }
                return  root//единственный  корень
или нет корней
            }
            if (d==0)//одинаковые корни
            {
                root[0]=((-b)/2*a)
                root[1]=root[0]
            }

            if (d>0)//различные корни
            {
                root[0]=((-b)-Math.sqrt(d))/2/a
                root[1]=((-b)+Math.sqrt(d))/2/a
            }
            return root
```

```
            }
            document.writeln("<p>(",  q_eq(0,2,6),  ");
(", q_eq(1,-2,1), ");<br> (")
            document.writeln(q_eq(3,4,-2.5),   ");   (",
q_eq(2,0,5), ");</p>")
        </script>
    </body>
</html>
```

Листинг 32. Объект Date

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            now=new Date()
            document.writeln("Time:                    "+
now.getHours()+":"+now.getMinutes()+"<br>")
            document.writeln("Date:                    "+
(now.getMonth()+1)+"/"   +   now.getDate()   +   "/"   +
(now.getYear()+1900))
        </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №8

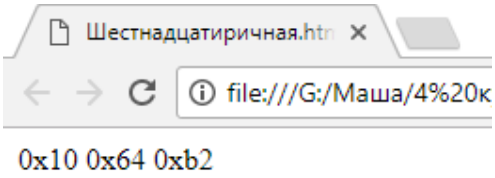


Рисунок 34 – Результат выполнения функции преобразования из десятеричной в шестнадцатеричную форму

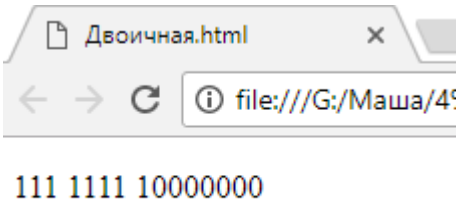


Рисунок 35 – Результат выполнения функции преобразования из десятеричной в двоичную форму

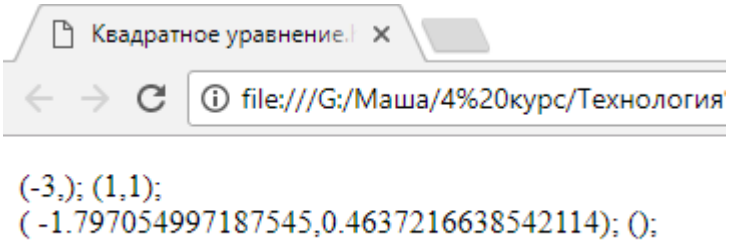


Рисунок 36 – Результат выполнения функции для решения квадратного уравнения

Рисунок 37 – Результат выполнения функции date

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №9: ПОНЯТИЕ СОБЫТИЯ

Каждое действие пользователя (нажатие на клавишу, щелчок кнопкой мыши и т.п.) формирует некоторое событие, т.е. сообщение о произошедшем.

Свойства события

Сообщение о событии формируется в виде объекта, т.е. контейнера для хранения информации.

В объектной модели имеется объект event, являющийся подобъектом объекта окна window. Он содержит информацию о том, какое событие произошло, какой элемент должен на него реагировать, и ряд других характеристик.

Свойство srcElement возвращает ссылку на объект элемента HTML-документа, который инициировал событие. При получении такой ссылки можно узнать или изменить значения свойств этого объекта и применить к нему любые из его методов.

У объекта event имеется изменяемое свойство returnValue (возвращаемое значение). С его помощью можно отменять действия принятые по умолчанию. Для этого достаточно присвоить ему значение false.

Например, щелчок на ссылке по умолчанию означает переход по указанному адресу. Однако вы можете отменить это действие или запросить у пользователя подтверждение перехода.

Нередко требуется разместить в документе элемент, который внешне выглядит как обычная ссылка (при наведении указателя мыши его вид изменяется), но щелчок должен приводить не к переходу на другой документ, а просто к выполнению некоторого сценария.

Прохождение событий

Большинство тегов в HTML являются контейнерными и, следовательно, могут содержать в себе другие теги. При этом может оказаться так, что одно и тоже событие будет обозначено в различных, но вложенных друг в друга тегах.

Особенность этого примера в том, что если пользователь щелкает на кнопке, сработает обработчик не только кнопки, но и блока div.

Листинг 33. Работа с событиями

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <script>
            function test()
            {
                str=""
                str+="x = "+window.event.x+"\n"
                str+="y = "+window.event.y+"\n"
                str+= "Вы нажали клавишу: "
                if (window.event.shiftKey) str+="Shift"
                if (window.event.ctrlKey) str+="Ctrl"
                if (window.event.altKey) str+="Alt"
                alert(str)
            }
        </script>
    </head>
    <body>
        <button onclick="test()">
            Нажми эту кнопку
        </button>
    </body>
</html>
```

Листинг 34. Свойство srcElement

```
<!DOCTYPE HTML>
<html>
    <head>
```

```
        <meta charset="utf-8">
        <script>
            function chtext()
            {
                x=window.event.srcElement//ссылка    на
объект
                x.innerText="Hahahahaha!!!"//замена
текста
            }
        </script>
    </head>
    <body>
        <button onclick="chtext()">
            Button One
        </button>
        <button onclick="chtext()">
            Button Two
        </button>
    </body>
</html>
```

Листинг 35. Свойство returnValue

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function myref()
            {
```

```
                ref=confirm("Are you sure?")
                if(!ref) window.event.returnValue=false
            }
        </script>
        <a                              onclick="myref()"
href="http://inf1.info">Информатика</a>
    </body>
</html>
```

Листинг 36. Свойство returnValue

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            function myref1()
            {
                alert("Был щелчок на ссылке")
                window.event.returnValue=false
            }
        </script>
        <a   onclick="myref1()"   href="">Здесь   ничего
нет</a>
    </body>
</html>
```

Листинг 37. Применение контейнеров

```
<!DOCTYPE HTML>
<html>
    <head>
```

```
        <meta charset="utf-8">
    </head>
    <body>
        <div onclick="alert('Щелчок по блоку')">
            <img src="G:/Маша/4 курс/Технология разработки интернет приложений/Лабораторная работа №3/Урок 9/cat1.jpg">
            <button onclick="alert('Щелчок по кнопке')"> Button </button>
        </div>
    </body>
</html>
```

Листинг 38. Применение контейнеров

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <div onclick="alert('Щелчок по блоку')">
            <img src="G:/Маша/4 курс/Технология разработки интернет приложений/Лабораторная работа №3/Урок 9/cat1.jpg">
            <button onclick="alert('Щелчок по кнопке')"; window.event.cancelBubble=true> Button </button>
        </div>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №9

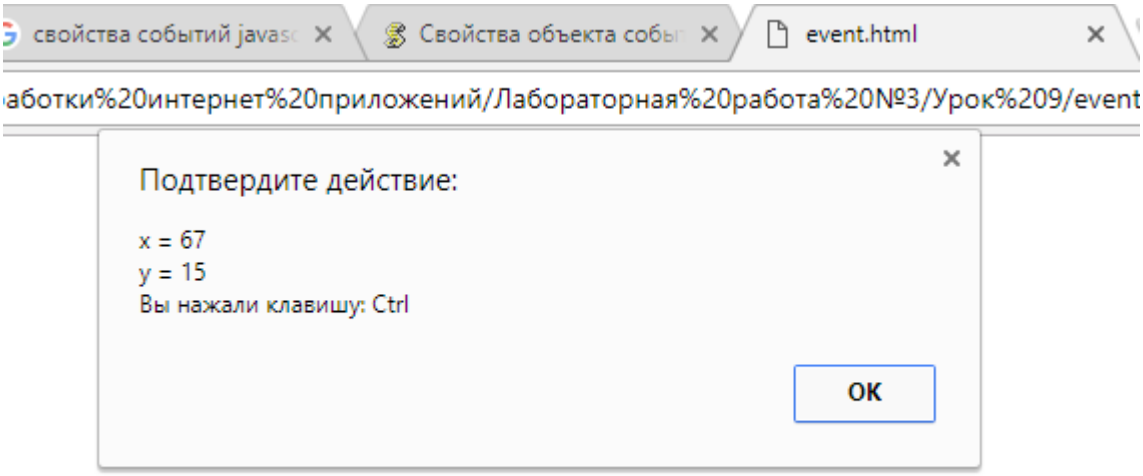


Рисунок 38 – Работа с событиями event

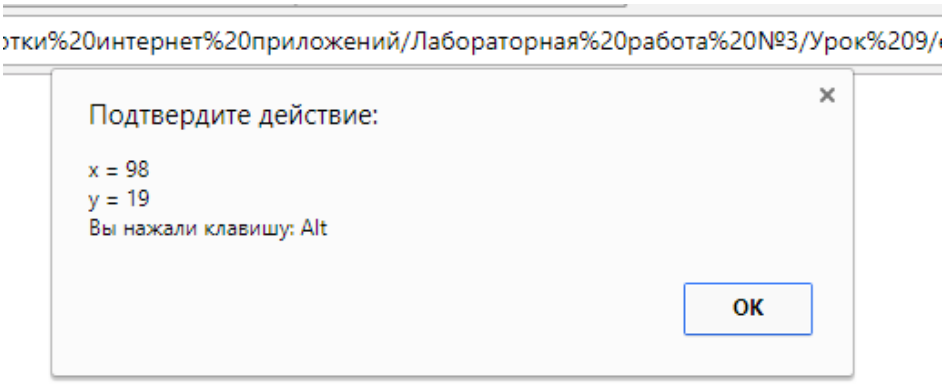


Рисунок 39 – Работа с событиями event

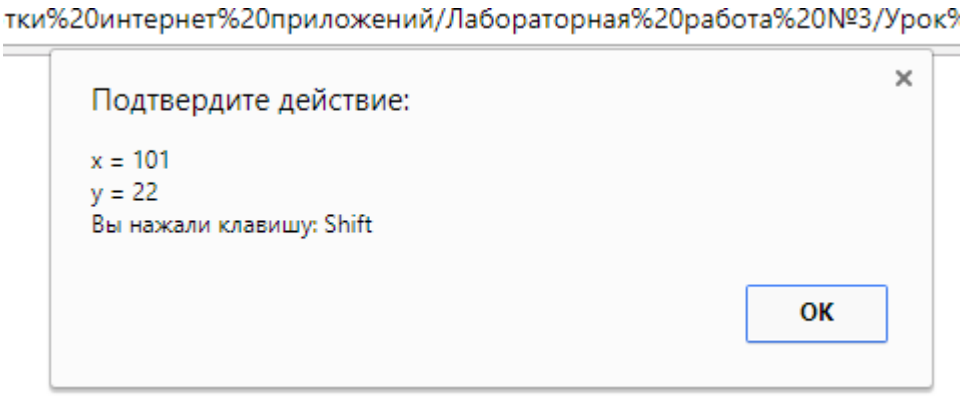


Рисунок 40 – Работа с событиями event

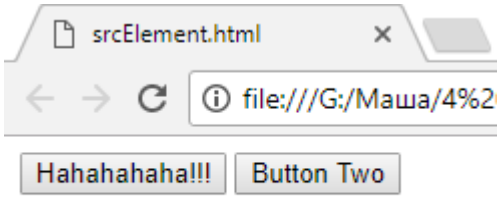


Рисунок 41 – Свойство srcElement

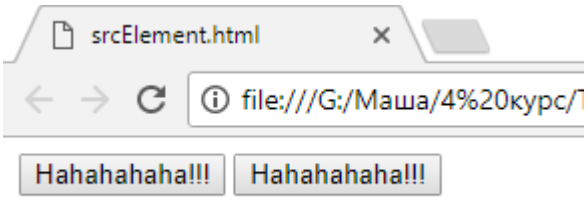


Рисунок 42 – Свойство srcElement

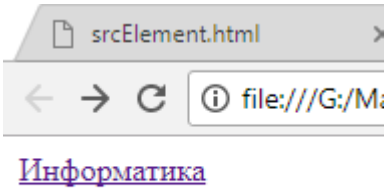


Рисунок 43 – Свойство returnValue

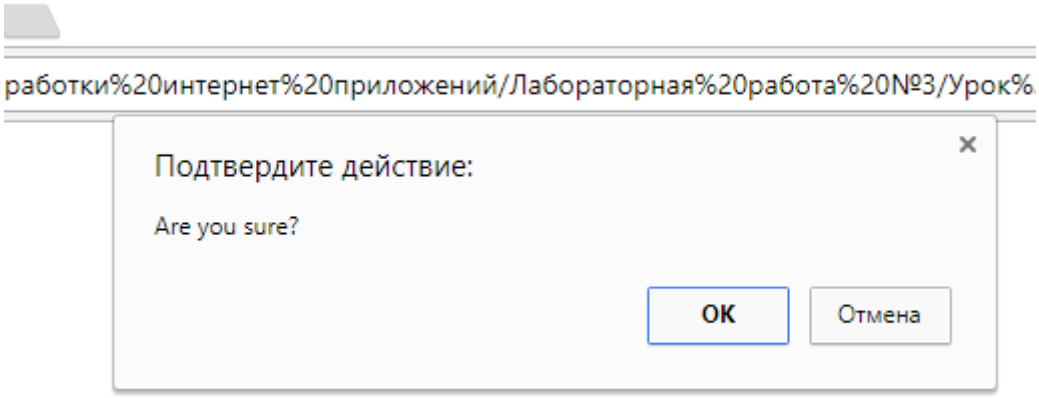


Рисунок 44 – Свойство returnValue

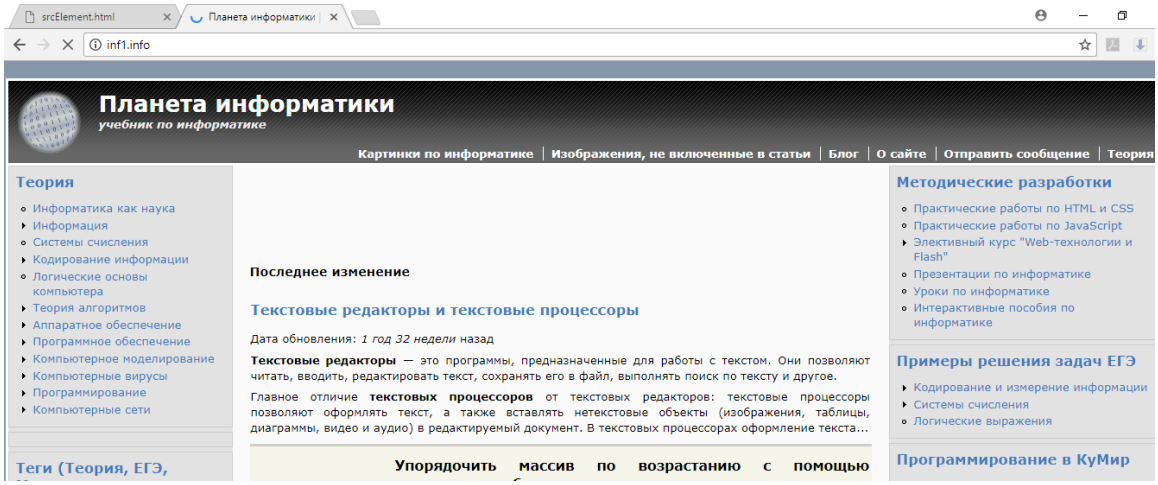


Рисунок 45 – Свойство returnValue

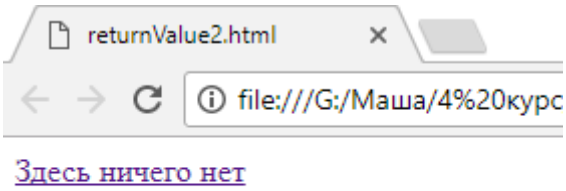


Рисунок 46 – Свойство returnValue

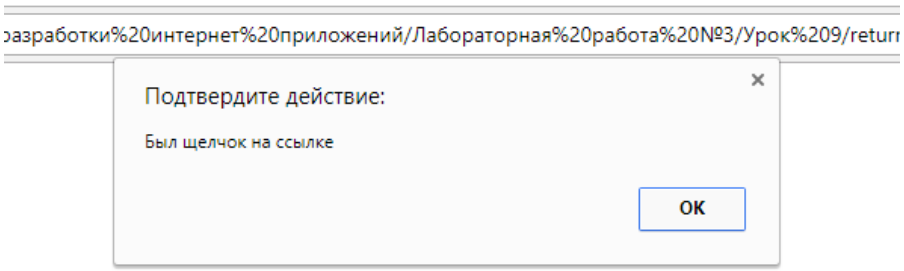


Рисунок 47 – Свойство returnValue

Рисунок 48 – Работа с контейнерами

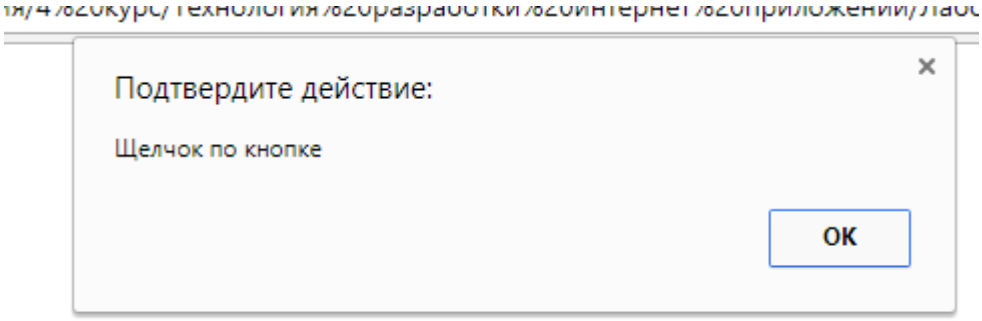


Рисунок 49 – Работа с контейнерами

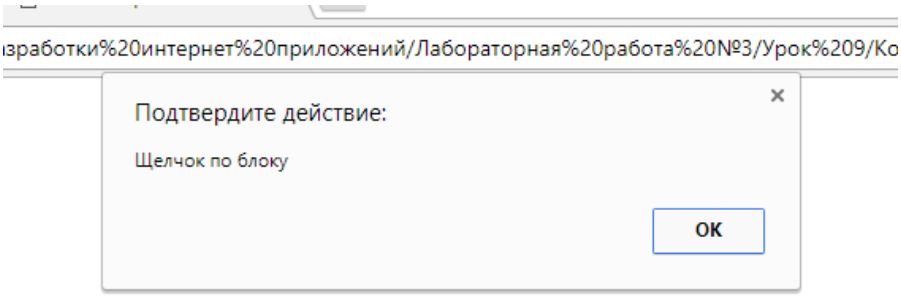


Рисунок 50 – Работа с контейнерами

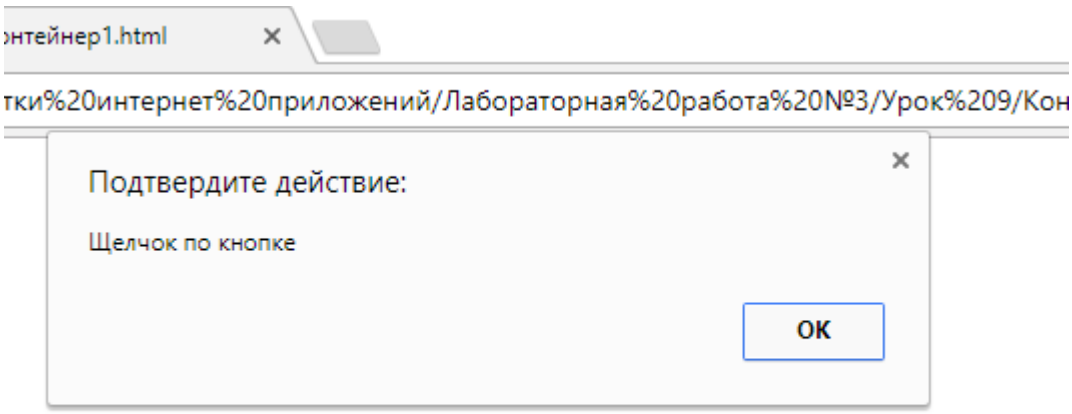


Рисунок 51 – Работа с контейнерами

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №10: ИЕРАРХИЯ ОБЪЕКТОВ В JAVASCRIPT

В языке JavaScript все элементы на web-странице выстраиваются в иерархическую структуру. Каждый элемент предстается в виде объекта. И каждый такой объект может иметь определенные свойства и методы. Язык JavaScript позволяет легко управлять объектами web-страницы, хотя для этого очень важно понимать иерархию объектов, на которые опирается разметка HTML.

С точки зрения языка JavaScript окно браузера - это некий объект window. Этот объект также содержит в свою очередь некоторые элементы оформления, такие как строка состоятия.

Внутри окна мы можем разместить документ HTML (в общем случае файл любого типа). Такая страница является ни чем иным, как объектом document. Это означает, что объект document представляет в языке JavaScript загруженный на настоящий момент документ HTML. Объект document является очень важным объектом в языке JavaScript. К свойствам объекта document относятся, например, цвет фона для web-страницы. Однако для нас гораздо важнее то, что все без исключения объекты HTML являются свойствами объекта document. Примерами объекта HTML являются, к примеру, ссылка или заполняемая форма.

Разумеется, мы должны иметь возможность получать информацию о различных объектах в этой иерархии и управлять ею. Для этого мы должны знать, как в языке JavaScript организован доступ к различным объектам. Как видно, каждый объект иерархической структуры имеет свое имя. Следовательно, если Вы хотите узнать, как можно обратиться к первому рисунку на нашей HTML-странице, то обязаны сориентироваться в иерархии объектов. И начать нужно с самой вершины. Первый объект такой структуры называется document. Первый рисунок на странице представлен как объект images[0]. Это означает, что отныне мы можем получать доступ к этому объекту, записав в JavaScript document.images[0].

Если же, например, Вы хотите знать, какой текст ввел читатель в первый элемент формы, то сперва должны выяснить, как получить доступ к этому объекту. И снова начинаем мы с вершины нашей иерархии объектов. Затем прослеживаем путь к объекту с именем elements[0] и последовательно записываем названия всех объектов, которые минуем. В итоге выясняется, что доступ к первому полю для ввода текста можно получить, записав document.forms[0].elements[0]. Хотя чаще всего доступ к объекту получают по его имени.

ЛИСТИНГИ ПО УРОКУ №10

Листинг 39. Страница с различными объектами

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <title> Page with image</title>
    </head>
    <body>
        <center>
```

```
            <img src="D:/Internet Explorer/Архипова
Мария/4 курс/Технология разработки интернет
приложений/Лабораторная работа №3/Урок 10/cat1.jpg"
            name="cat" width="128" height="128">
        </center>
        <p>
            <form name="form1">
                Name:
                <input type="text" name="uname"
value=""><br>
                e-mail:
                <input type="text" name="email"
value=""><br><br>
                <input type="button" name="but_push"
value="Push me" onClick="alert('Hi')"><br><br>
                <input type="text" name="input_text"
value="Привет"><br><br>
                <input type="button" value="На
английском"
onClick="document.form1.input_text.value='Hello'">
            </form>
        </p>
        <p>
            <a href="http://inf1.info">Информатика</a>
        </p>
    </body>
</html>
```

Листинг 40. Доступ к объектам на странице

```
<!DOCTYPE HTML>
<html>
```

```
    <head>
        <meta charset="utf-8">
        <title> Objects</title>
        <script>
            function out()
            {
                alert(document.form1.fieldtext.value);
            }
            function check()
            {
                //проверка состояния флажка
                var str="The checkbox is ";
                if(document.form1.box.checked)
                    str+="checked"
                else
                    str+="not checked";
                alert(str);
            }
        </script>
    </head>
    <body>
        <form name="form1">
            Name:
            <input     type="text"     name="fieldtext"
value="Non">
            <input     type="button"     name="button1"
value="Button 1" onClick="out()">
            <br>
            <input type="checkbox" name="box" CHECKED>
```

```
            <input     type="button"     name="button2"
value="Button 2" onClick="check()">
        </form>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №10

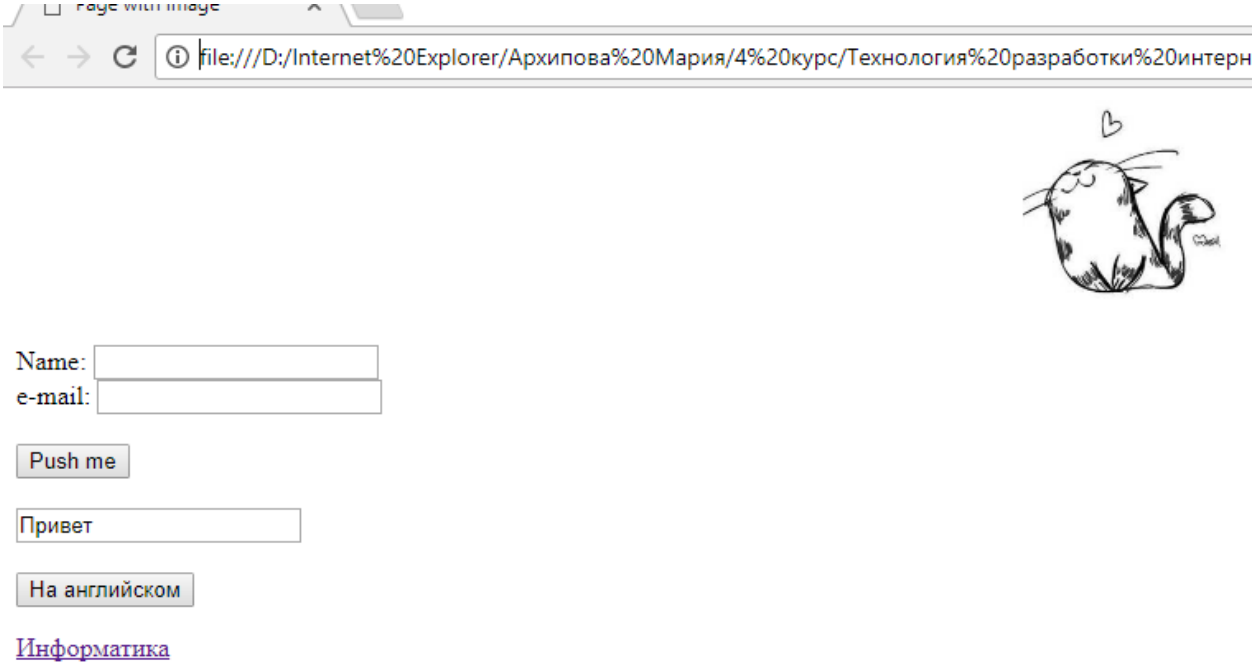


Рисунок 52 – Страница с различными объектами

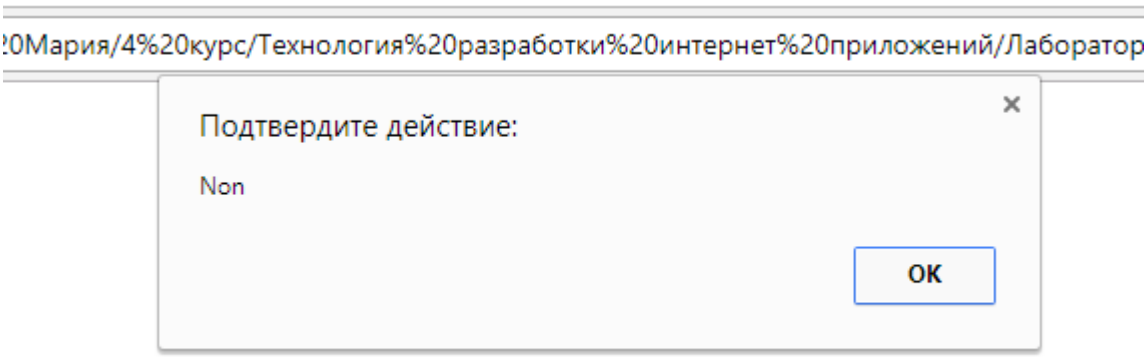


Рисунок 53 – Доступ к объектам на странице

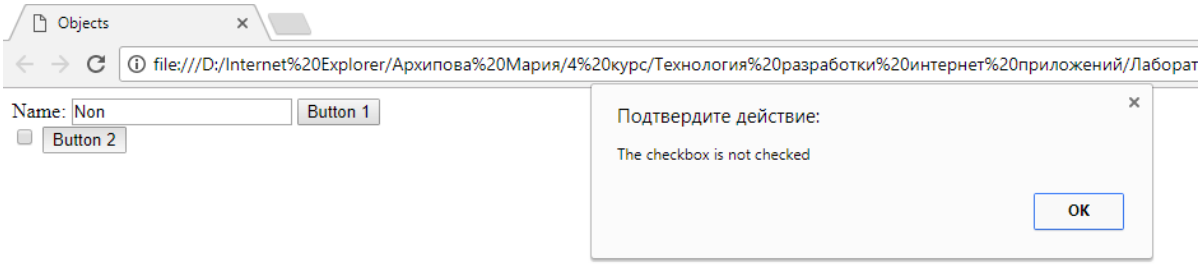


Рисунок 54 – Доступ к объектам на странице

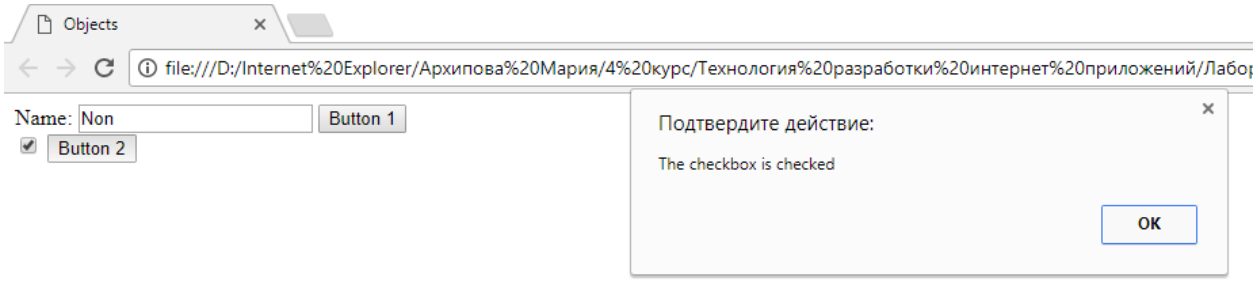


Рисунок 55 – Доступ к объектам на странице

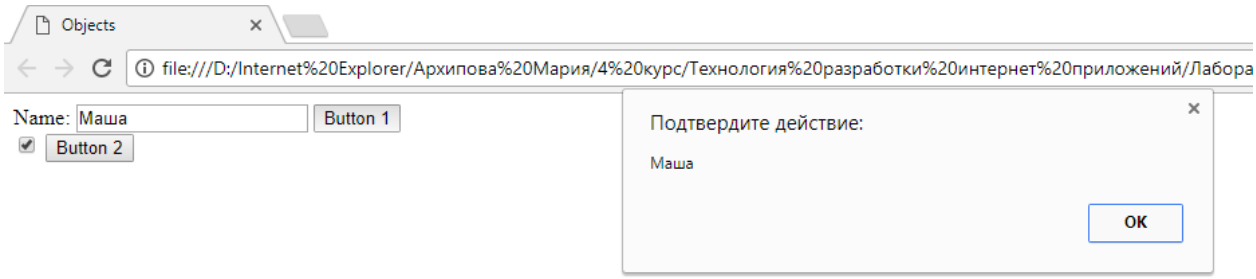


Рисунок 56 – Доступ к объектам на странице

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №11: ПОНЯТИЕ СЦЕНАРИЯ

Сценарий на языке JavaScript - это программа, работающая с объектами HTML-документа.

Обработка событий

Одним из главных (но далеко не единственным) назначений сценариев в HTML-документе является обработка событий, таких как щелчок кнопкой мыши на элементе документа, помещение указателя мыши на элемент, перемещение указателя с элемента, нажатие клавиш и т.п.

Значением таких атрибутов-событий в тегах HTML является строка, содержащая сценарий, выполняющий роль обработчика события.

Объекты, управляемые сценариями

Приведенный ниже HTML-код формирует простую веб-страницу, содержащую заголовок первого уровня, изображение, ссылку и форму с двумя элементами - полем ввода и кнопкой. Сценарий основан на использовании свойства tagName, которое возвращает название тега, с помощью которого был создан соответствующий объект.

ЛИСТИНГИ ПО УРОКУ №11

Листинг 41. Обработчик щелчка по изображению

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <title> Page with image</title>
        <script>
            function clickimage()
            {
                alert("Hello");
            }
        </script>
    </head>
    <body>
        <p>Вариант 1
            <img src="D:/Internet Explorer/Архипова Мария/4 курс/Технология разработки интернет приложений/Лабораторная работа №3/Урок 11/cat0.jpg"
            onClick="clickimage()">
        </p>
        <p>Вариант 2
            <img src="D:/Internet Explorer/Архипова Мария/4 курс/Технология разработки интернет приложений/Лабораторная работа №3/Урок 11/cat1.jpg"
```

```
            onClick='alert("olleH")'>
        </p>
    </body>
</html>
```

Листинг 42. Использование свойства tagName

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <title> Page with image</title>
        <script>
            function analysis()
            {
                msg=""
                for(i=0; i<document.all.length; i++)
                    msg+=i+"
"+document.all[i].tagName+"            name            =
"+document.all[i].name+"\n"
                return msg
            }
        </script>
    </head>
    <body>
        <h1>The page</h1>
        <img src="D:/Internet Explorer/Архипова Мария/4
курс/Технология разработки интернет
приложений/Лабораторная работа №3/Урок 11/cat1.jpg">
        <p><a  href="http://younglinux.info">Уроки  по
GNU/Linux</a></p>
```

```
        <textarea name="field" rows="10"
cols="30"></textarea>
        <br>
        <button
onClick="document.all.field.value=analysis()">Press
here</button>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №11

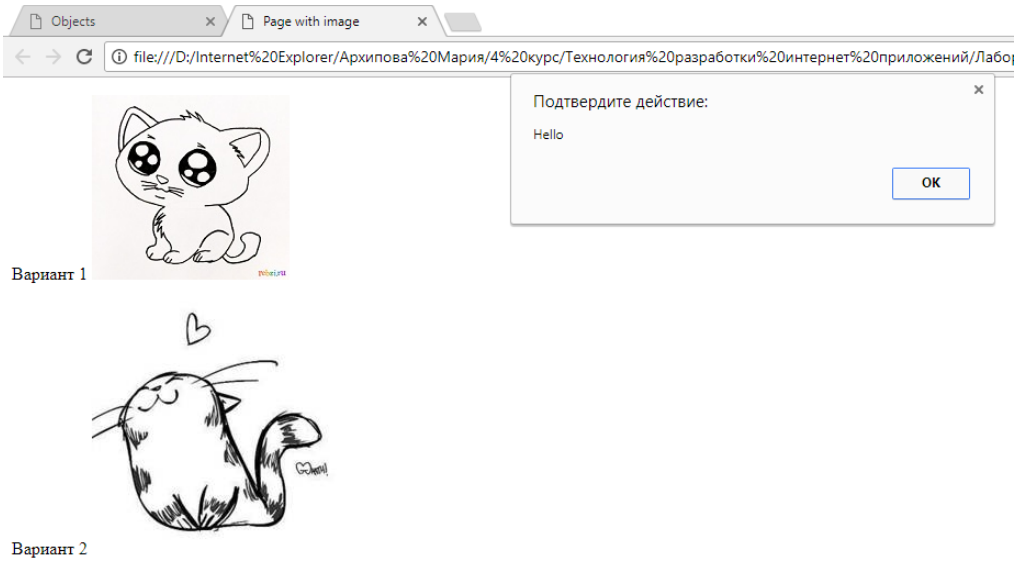


Рисунок 57 – Обработчик щелчка по изображению

Рисунок 58 – Обработчик щелчка по изображению

Рисунок 59 – Использование свойства tagName

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №13: ЦВЕТОВЫЕ ЭФФЕКТЫ

Подсветка кнопок и текста

Рассмотрим задачу изменения цвета кнопки при наведении на нее указателя мыши. При удалении указателя с кнопки должен вернуться ее первоначальный цвет.

В нашем примере три элемента-кнопки находятся в контейнере формы FORM. К этому контейнеру привязываются обработчики событий onmouseover (наведение указателя мыши на объект) и onmouseout (сведение указателя мыши с объекта).

Аналогичным образом можно изменять цвет и других элементов, например фрагментов текста. Если требуется подсвечивать текст, то он должен быть заключен в какой-нибудь контейнер, например в теги P, B, I или DIV.

Мигающая рамка

Вы можете создать прямоугольную рамку, окаймляющую некий текст, которая периодически изменяет цвет. Рамка создается тегами одноячеечной таблицы с заданием нужных атрибутов и параметров стиля. Далее необходимо создать функцию, изменяющую цвет рамки таблицы на другой, и передать ее в качестве первого параметра методу setInterval().

Второй параметр этого метода задает период в миллисекундах, с которым вызывается функция, указанная в первом параметре.

Переливающиеся цветами ссылки

Суть задачи состоит в том, чтобы случайным образом выбирать и устанавливать цвет ссылок. В приведенном ниже листинге подмножества цветов, из которых происходит выбор, различаются для уже использованных и еще не использованных ссылок. Эти множества цветов задаются в виде массивов.

ЛИСТИНГИ ПО УРОКУ №13

Листинг 43. Изменение цвета кнопок при наведении курсора

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <style>
            .color_but{font-weight: bold; background-
color: #a0a0a0;}
        </style>
        <script>
            function colorchange(color)
            {
                if (event.srcElement.type == "button")

    event.srcElement.style.backgroundColor=color;
            }
```

```
        </script>
    </head>
    <body>
        <form       onmouseover="colorchange('yellow')"
onmouseout="colorchange('#a0a0a0')">
            <input       type="button"       value="One"
class="color_but">
            <input       type="button"       value="Two"
class="color_but">
            <input      type="button"      value="Three"
class="color_but">
        </form>
    </body>
</html>
```

Листинг 44. Изменение цвета фрагментов текста

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <script>
            function colorch(color)
            {
                event.srcElement.style.color=color;
            }
        </script>
    </head>
    <body>
        <p>
```

```
            Этот    <b    onmouseover="colorch('red')"
onmouseout="colorch('blue')"> жирный текст </b> при
наведении на него указателя мыши изменяет цвет.
        </p>
    </body>
</html>
```

Листинг 45. Создание мигающей рамки

```
<html>
    <head>
        <meta charset="utf-8">
        <script>
            function flash()
            {
                if (!document.all) return null;
                if(tab.style.borderColor=='yellow')
                    tab.style.borderColor=='red'
                else
                    tab.style.borderColor=='yellow'
            }
        </script>
    </head>
    <body>
        <table    id="tab"    border="1"    width="150"
style="border:10 solid:yellow">
            <tr>
                <td>Мигающая рамка</td>
            </tr>
        </table>
        <script>
            setInterval("flash()", 500);
```

```
        </script>
    </body>
</html>
```

Листинг 46. Переливающиеся цветами ссылки

```
<html>
    <head>
        <meta charset="utf-8">
        <script>
            a_link=naw      Array()//массив      цветов
неиспользованных ссылок
            a_link[0]='yellow'
            a_link[1]='#80ff80'
            a_link[2]='#ffff80'
            a_link[3]='#408000'
            a_vlink=naw     Array()//массив      цветов
использованных ссылок
            a_vlink[0]='blue'
            a_vlink[1]='purple'
            a_vlink[2]='black'
            a_vlink[3]='red'
            function colorch()
            {

    alink=Math.round((a_link.length+0.1)*Math.random())

    vlink=Math.round((a_vlink.length+0.1)*Math.random()
)
                document.linkColor=a_link[alink]
                document.linkColor=v_link[vlink]
            }
```

```
        </script>
    </head>
    <body>
        <ul>
            <li> <a href="http://younglinux.info">Уроки
по СПО</a></li>
            <li>                                    <a
href="http://inf1.info">Информатика</a></li>
            <li>                                    <a
href="http://pas1.ru">Pascal</a></li>
        </ul>
        <script>
            setInterval("colorch()", 500);
        </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №13

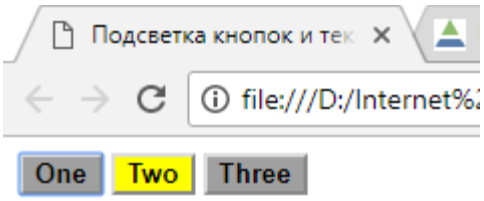


Рисунок 60 – Изменение цвета кнопок при наведении курсора

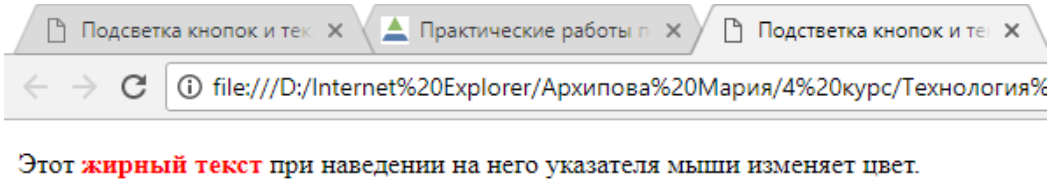


Рисунок 61 – Изменение цвета фрагментов текста

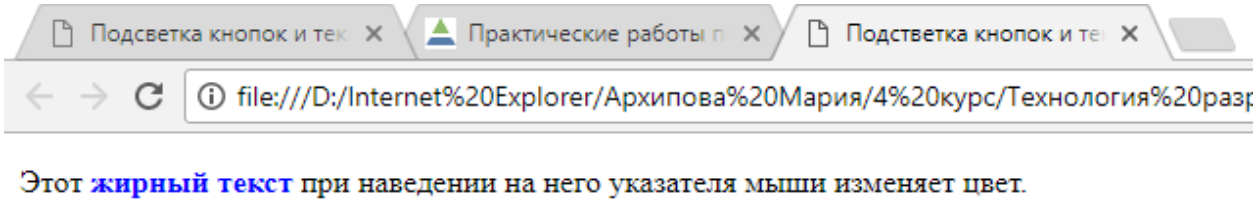


Рисунок 62 – Изменение цвета фрагментов текста

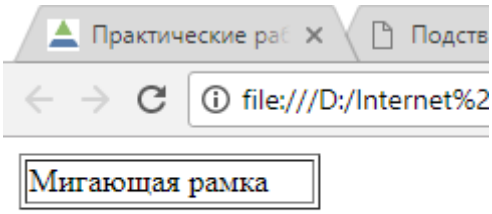


Рисунок 63 – Создание мигающей рамки

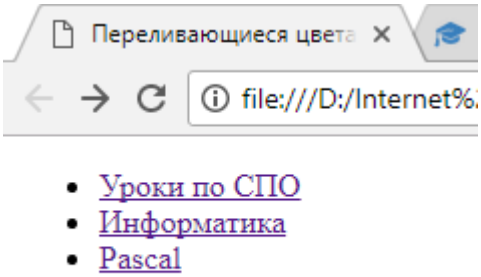


Рисунок 64 – Переливающиеся цветами ссылки

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №14: ОБЪЕМНЫЕ ЗАГОЛОВКИ

Идея создания объемного заголовка достаточно проста: достаточно несколько надписей с одинаковым содержанием наложить друг на друга с некоторым сдвигом по координатам.

ЛИСТИНГИ ПО УРОКУ №14

Листинг 47. Создание объёмного заголовка

```
<html>
      <head>
      <title>3D-эффект</title>
          <style>
```

```
            p  {font-family:  sans-serif;  font-size:
72px; font-weight: 800; color: #00aaaa;}
            p.highlight {color: silver}
            p.shadow {color: black}
        </style>
    </head>
    <body bgcolor="#aeb98c">
        <!-- Тень -->
        <div style="position:absolute; top:5; left:5">
            <p class="shadow">Объёмный заголовок</p>
        </div>
        <!-- Подсветка -->
        <div style="position:absolute; top:0; left:0">
            <p class="highlight">Объёмный заголовок</p>
        </div>
        <!-- Передний план -->
        <div style="position:absolute; top:2; left:2">
            <p>Объёмный заголовок</p>
        </div>
    </body>
</html>
```

Листинг 48. Различные заголовки

```
<html>
    <head>
        <title>3D-эффект</title>
    </head>
    <body>
        <script>
            function
d3(text,x,y,tcolor,fsize,fweight,ffamily,zind) {
```

```
            /* text - текст заголовка
            x - горизонтальная координата (left)
            y - вертикальная координата (top)
            tcolor - цвет переднего плана
            fsize - размер шрифта (pt)
            fweight - вес (толщина шрифта)
            ffamily - название семейства шрифтов
            zind - z-Index */
            if (!text) return null;
            // значение параметров по умолчанию:
            if (!ffamily) ffamily = 'arial';
            if (!fweight) fweight = 800;
            if (!fsize) fsize = 36;
            if (!tcolor) tcolor = '#00aaff';
            if (!y) y = 0;
            if (!x) x = 0;
            var sd = 5, hd = 2 // сдвиг тени и подсветки
            var xzind = ""
            if (zind) xzind = "; z-Index:" + zind
            var xstyle = 'font-family:' + ffamily + ';
font-size:' + fsize + '; font-weight:' + fweight + ';'
            var xstr = '<div style="position:absolute;
top:' + (y + sd) + ';left:' + (x + sd) + xzind + '">'
            xstr += '<p style="' + xstyle +
'color:darkred">' + text + '</p></div>'
            xstr += '<div style="position:absolute;
top:' + y + ';left:' + x + xzind + '">'
            xstr += '<p style="' + xstyle +
'color:silver">' + text + '</p></div>'
```

```
                xstr  +=  '<div  style="position:absolute;
top:' + (y + hd) + ';left:' + (x + hd) + xzind + '">'
                xstr += '<p style="' + xstyle + 'color:' +
tcolor + '">' + text + '</p></div>'
                document.write(xstr)
                }
                d3("Привет!",  50,  15,  'red',  40,  800,
'sans-serif')
                d3("Это не графика", 50, 50, 'blue', 72,
800, 'Times')
                d3("Это текст", 10, 80, '#00aa00', 92, 900,
'Courier New')
          </script>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №14

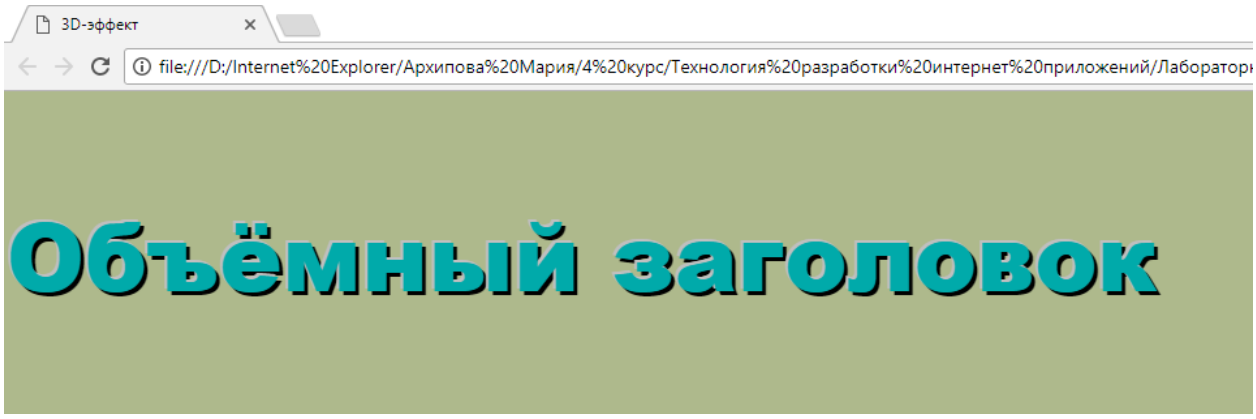


Рисунок 65 – Создание объёмного заголовка

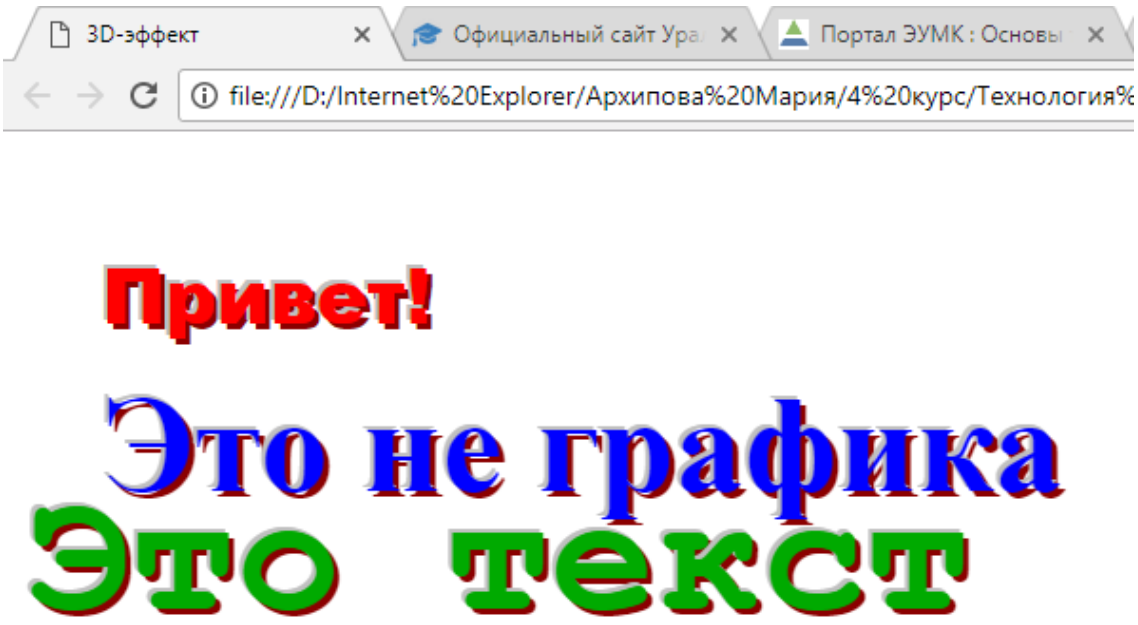


Рисунок 66 – Различные заголовки

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №15: ОБРАБОТКА ДАННЫХ ФОРМ

Чтобы отправить данные, достаточно в теге FORM указать атрибут ACTION, а в самой форме установить кнопку типа Submit.

Если перед отправкой данных формы требуется предварительно их проверить, то для этого необходим сценарий.

Если необходимо затруднить выявление адреса электронной почты различными программами-роботами, сканирующими веб-страницы, то следует предпринять некоторые меры.

Самый простой рецепт - хранить отдельные компоненты адреса в различных переменных и собирать их с помощью выражения конкатенации (склейки), которое присваивается свойству action.

ЛИСТИНГИ ПО УРОКУ №15

Листинг 49. Отправка данных

```
<!DOCTYPE HTML>
<html>
    <head>
```

```
        <meta charset="utf-8">
        <title> Отправка данных</title>
    </head>
    <body>
        <form                                method="post"
action="mailto:pypath@yandex.ru" enctype="text/plain">
            <textarea   name="let"   value=""   rows="4"
cols="40"></textarea>
            <input      name="subm"      type="submit"
value="Отправить">
        </form>
    </body>
</html>
```

Листинг 50. Проверка адреса

```
<html>
    <head>
        <title>Форма 2</title>
            <script>
                function mes_go() {
                    var            nomail            =
mess.m_to.value.indexOf('@') == -1
                    var notext = !mess.let.value
                    var   xtext   =   "\nПисьмо   не
отправлено"
                    if (nomail || notext) {
                       event.returnValue = false
                    if (nomail)
                       alert("Неправельный      адрес
получателя" + xtext)
                    else
```

```
                        alert("Поле сообщения пусто" +
xtext)
                    } else
                        mess.action = "mailto:" +
mess.m_to.value
                }
            </script>
    </head>
    <body>
        <form name="mess" method="GET"
action="mailto:pypath@yandex.ru" enctype="text/plain">
            <p>Кому: <input name="m_to" type="text"
value=""></p>
            <p>От кого: <input name="m_from"
type="text" value=""></p>
            <p>Сообщение:<br>
            <textarea name="let" value="" rows="4"
cols="40"></textarea></p>
            <p><input name="subm" type="submit"
value="Отправить" onclick="mes_go()"></p>
        </form>
    </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №15

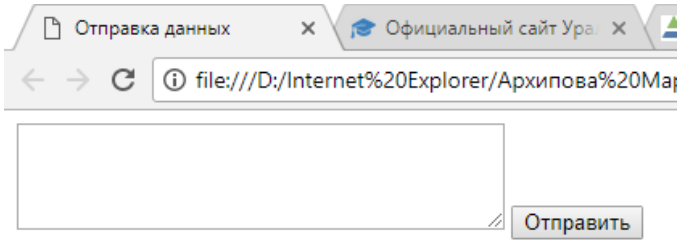


Рисунок 67 – Отправка данных

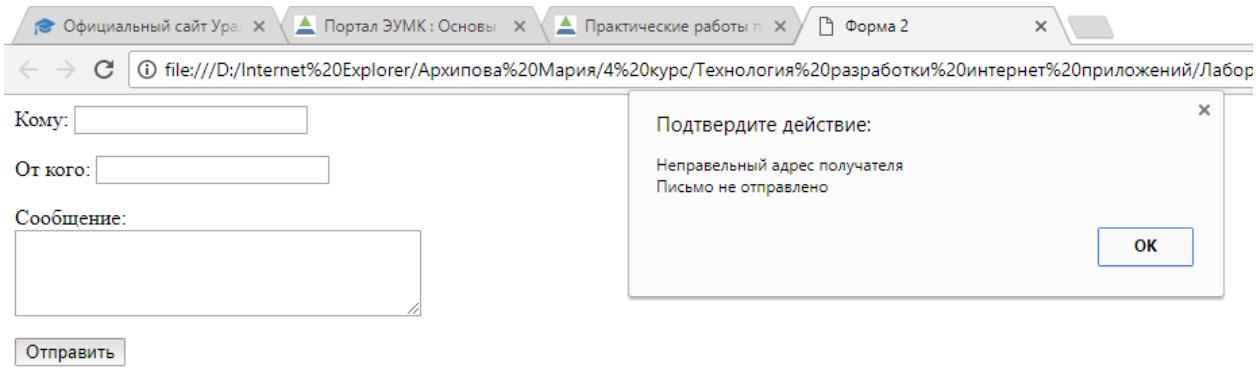


Рисунок 68 – Проверка адреса

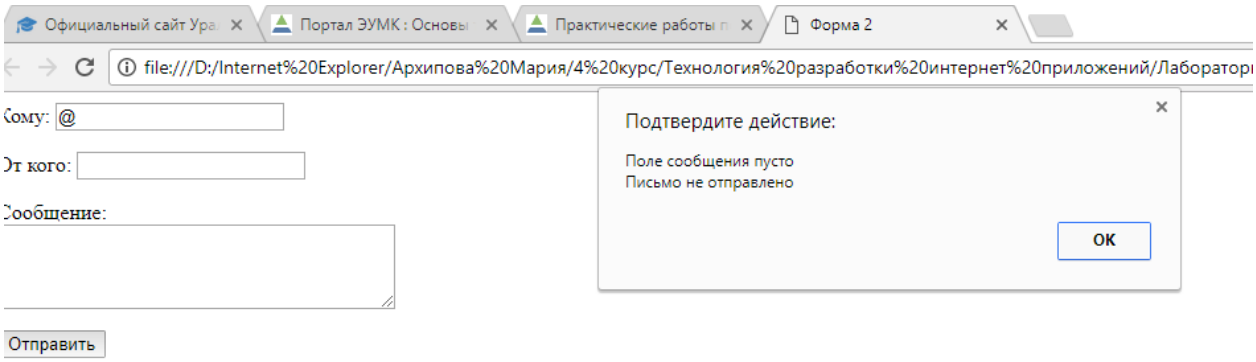


Рисунок 69 – Проверка адреса

ОСНОВНАЯ ИНФОРМАЦИЯ ПО УРОКУ №16: ДВИЖЕНИЕ ОБЪЕКТОВ

Типы движения:

- Линейное движение,
- Движение по кривой,
- Движение по кривой с возвратом назад.

ЛИСТИНГИ ПО УРОКУ №16

Листинг 51. Линейное движение

```
<html>
    <head>
        <title>Линейное движение картинки</title>
        <script>
            function init_move() {
```

```
                    dx = 8 // приращение по осям
                    dy = 3
                    setInterval("move()",200)
                }
            function move() { // изменение координат
изображения
                    // текущие координаты
                    var y =
parseInt(document.all.pic.style.top)
                    var x =
parseInt(document.all.pic.style.left)
                    // новые координаты
                    document.all.pic.style.top = y +
dy
                    document.all.pic.style.left = x +
dx
                }
        </script>
    </head>
<body>
     <img name="pic" src="D:/Internet
Explorer/Архипова Мария/4 курс/Технология разработки
интернет приложений/Лабораторная работа №3/Урок
16/cat1.jpg" style="position:absolute; top:10;
left:20;">
     <script>
        init_move()
    </script>
</body>
</html>
```

Листинг 52. Движение по кривой

```
<html>
    <head>
        <title>Движение по кривой</title>
        <script>
            function curve_move(xid, yexpr, xexpr,
ztime) {
                if (!xid) return null
                if (!yexpr) yexpr = "x"
                if (!xexpr) xexpr = "x"
                if (!ztime) ztime = 100
                x = 0
                setInterval("move('" + xid + "','"
+ yexpr + "','" + xexpr + "')",ztime)
             }
                function move(xid, yexpr, xexpr) {
                x += 1
                document.all[xid].style.top     =
eval(yexpr)
                document.all[xid].style.left    =
eval(xexpr)
                }
        </script>
    </head>
    <body>
         <img     name="pic"     src="D:/Internet
Explorer/Архипова Мария/4 курс/Технология разработки
интернет приложений/Лабораторная работа №3/Урок
16/cat1.jpg" style="position:absolute;">
         <script>
```

```
                curve_move("pic","100 + 50 *
    Math.sin(0.03*x)","50+x",30)
                 </script>
            </body>
</html>
```

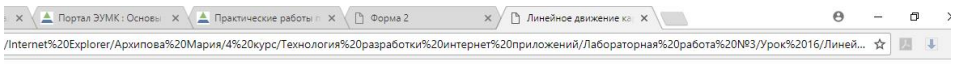
Листинг 53. Движение по кривой с возвратом назад

```
<html>
        <head>
            <title>Движение по кривой с
    возвратом</title>
            <script> // движение буквой V
                var action, coef_px=-1, coef_pt=1,
    p_move=5;
                function startmove() {
                    dynamic.style.left =
    parseInt(document.body.offsetWidth)/3 + 200;
                    dynamic.style.top = 0;
                    action =
    window.setInterval("move()",100);
                    }
                function move() {
                    px = parseInt(dynamic.style.left);
                    px = px + coef_px * p_move;
                    dynamic.style.left = px;
                    if (px <=
    parseInt(document.body.offsetWidth)/3 - 200 ||
                    px >=
    parseInt(document.body.offsetWidth)/3 + 200) {
                       coef_px = -coef_px
                    }
```

```
                    pt = parseInt(dynamic.style.top);
                    pt = pt + coef_pt * p_move;
                    dynamic.style.top = pt;
                   if(pt <= 0 || pt >= 200) {
                       coef_pt = -coef_pt
                    }
                }
            </script>
        </head>
        <body onload=startmove()>
            <div                          id="dynamic"
    style="position:absolute;">
            <img  src="D:/Internet  Explorer/Архипова
    Мария/4  курс/Технология  разработки  интернет
    приложений/Лабораторная  работа  №3/Урок
    16/cat1.jpg"></div>
        </body>
</html>
```

СНИМКИ ЭКРАНА ПО ВЫПОЛНЕННОМУ УРОКУ №16

Рисунок 70 – Линейное движение



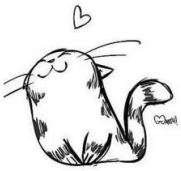
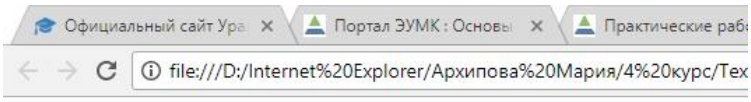


Рисунок 71 – Линейное движение



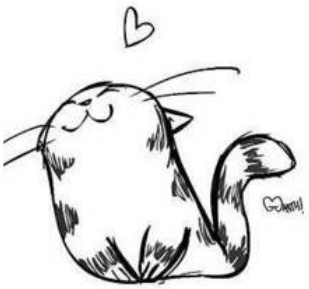
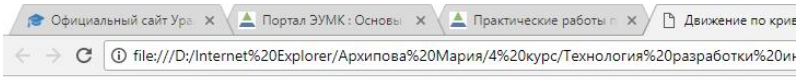


Рисунок 72 – Движение по кривой



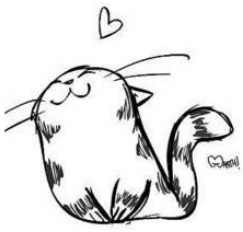


Рисунок 73 – Движение по кривой

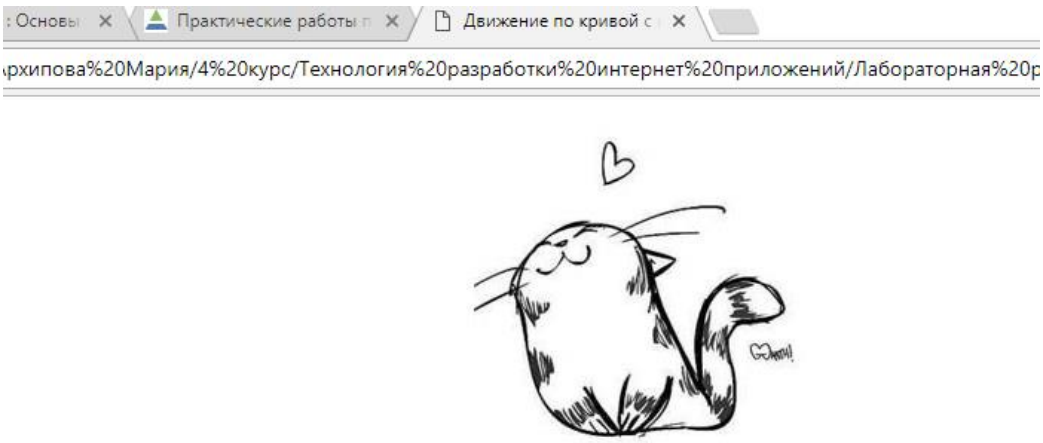


Рисунок 74 – Движение по кривой с возвратом назад

Рисунок 75 – Движение по кривой с возвратом назад