Правила создания документа JavaScript

1. Любой скрипт в HTML начинается с пары:

```
<script>
…
</script>
```

2. JS различает регистр.
3. Текст внутри скобок находится в кавычках, который представляет собой HTML.
4. Двойные и одинарные кавычки равноправны.
5. Внутри двойных кавычек ставятся одинарные.
6. Так как JS – объектно-ориентированный язык программирования, то при составлении скрипта будут присутствовать объекты (object), методы и свойства.

```
<script>
document.write (“<FONT COLOR = ‘red’>Это красный текст</FONT>”)
</script>
```

7. Вставка комментариев:

```
// в каждой строке;
/* в начале и в конце*/
```

8. Прописывая строку JS ее необходимо умещать на одной строке редактора.
9. Каждая строка JS заканчивается «;»
10. Сочетание текста и команд требует «+» между элементами.

Типы данных в JavaScript

В JavaScript существуют следующие примитивные типы данных:

1. Число «number». Единый тип «число» используется как для целых, так и для дробных чисел.

```
var n = 123;
n = 12.345;
```

2. Строка «string». Тип «символ» не существует, есть только «строка».

```
var str = "Мама мыла раму";
str = 'Одинарные кавычки тоже подойдут';
```

3. Булевый (логический) тип «boolean». У него всего два значения: true (истина) и false (ложь). Как правило, такой тип используется для хранения значения типа да/нет.

```
var checked = true; // поле формы помечено галочкой
checked = false;    // поле формы не содержит галочки
```

4. Специальное значение «null». Значение null не относится ни к одному из типов выше, а образует свой отдельный тип, состоящий из единственного значения null. В JavaScript null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках. Это просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».

```
var age = null;
```

В частности, код говорит о том, что возраст age неизвестен.

5. Специальное значение «undefined». Значение undefined, как и null, образует свой собственный тип, состоящий из одного этого значения. Оно имеет смысл «значение не присвоено». Если переменная объявлена, но в неё ничего не записано, то её значение как раз и есть undefined. В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого» или «неизвестного» значения используется null.

```
var x;
alert( x ); // выведет "undefined"
```

Можно присвоить undefined и в явном виде, хотя это делается редко:

```
 var x = 123;
x = undefined;
alert( x ); // "undefined"
```

6. Объекты «object», не является примитивным. Это отдельный тип. Он используется для коллекций данных и для объявления более сложных сущностей. Объявляются объекты при помощи фигурных скобок {...}.

```
var user = { name: "Вася" };
```

7. В стандарте ECMAScript 6 определен новый тип данных Symbol.

Переменные в JavaScript

Тип переменной зависит от того, какой тип информации в ней хранится. *JavaScript* не является жестко типизированным языком. Это означает, что не нужно точно определять тип данных переменной, в момент ее создания. Тип переменной присваивается автоматически в процессе выполнения скрипта.

Если во время присвоения значения переменной число заключается в кавычки, то оно рассматривается как строковое, а не числовое значение.

Можно определить переменную следующим образом:

```
let answer = 42
```

А позже, можно присвоить той же переменной следующее значение:

```
answer = "Уже текст..."
```

Так как *JavaScript* не поддерживает никаких методов и свойств для определения типа текущего значения переменной, очень важно внимательно отслеживать типы переменных во избежание неожиданных результатов.

На имя переменной в JavaScript наложены всего два ограничения.

1. Имя может состоять из: букв, цифр, символов $ и _.
2. Первый символ не должен быть цифрой.

Целочисленные переменные могут быть десятичными (24), шестнадцатеричными (0x24) или восьмеричными (024), с плавающей запятой (5.14E6), булевы переменные (true, false).

Переменные могут быть локальными и глобальными.

Основные операторы

Сложение строк, бинарный +

Обычно при помощи плюса '+' складывают числа. Но если бинарный оператор '+' применить к строкам, то он их объединяет в одну, т.е. выполняет конкатенацию строк:

```
var a = "моя" + "строка";
alert( a ); // моястрока
```

Если хотя бы один аргумент является строкой, то второй будет также преобразован к строке, причем не важно, справа или слева находится операнд-строка.

```
alert( '1' + 2 ); // "12"
alert( 2 + '1' ); // "21"
```

Остальные арифметические операторы работают только с числами и всегда приводят аргументы к числу.

```
alert( 2 - '1' ); // 1
alert( 6 / '2' ); // 3
```

Преобразование к числу, унарный плюс +

Унарный, то есть применённый к одному значению, плюс ничего не делает с числами. Однако он широко применяется, так как он позволяет преобразовать значения в число.

Например, когда полученные значения в форме строк из HTML-полей или от пользователя необходимо сложить.

```
let apples = "2";
let oranges = "3";

alert (apples + oranges); //  результат "23", так как бинарный плюс складывает строки

let apples = "2";
let oranges = "3";

alert (+apples + +oranges); // 5, число, оба операнда предварительно преобразованы в числа.
```

Присваивание

Принцип работы такой же, как и в изученных ранее языках программирования. Возможно присваивание по цепочке:

```
let a, b, c;
a = b = c = 2 + 2;
alert( a ); // 4
alert( b ); // 4
alert( c ); // 4
```

Такое присваивание работает справа-налево, то есть сначала вычисляется самое правое выражение 2+2, присвоится в c, затем выполнится b = c и, наконец, a = b.

Взятие остатка %

Его результат a % b – это остаток от деления a на b.

Деление на цело

Для взятия целой части от деления необходимо выполнить округление. Варианты округления:

```
num1 = 10, num2 = 3;
Math.floor(num1/num2); // 3 (округление в меньшую сторону)
Math.ceil(num1/num2); // 4 (округление в большую сторону)
Math.round(num1/num2); // 3 (математическое округление)
parseInt((num1/num2)); // 3 (приведение к числу, будет отброшена дробная часть)
```

```
<script>
        //из числа 126 получаем 621
        let a = 126;
        a1 = (Math.floor(a/100));
        a2 = (Math.floor(a/10))%10;
        a3 = a%10;
        alert ("Исходное число: " + a);
        alert ("Результат: " + (a3*100 + a2*10 + a1));
</script>
```

Инкремент/декремент: ++, --

Одной из наиболее частых операций в JavaScript, как и во многих других языках программирования, является увеличение или уменьшение переменной на единицу. Для этого существуют специальные операторы:

Инкремент ++ увеличивает на 1:

```
let i = 2;
i++;      // более короткая запись для i = i + 1.
alert(i); // 3
```

Декремент -- уменьшает на 1:

```
let i = 2;
i--;      // более короткая запись для i = i - 1.
alert(i); // 1
```

Инкремент/декремент можно применить только к переменной. Код 5++ даст ошибку.

Вызывать эти операторы можно не только после, но и перед переменной: i++ (называется «постфиксная форма») или ++i («префиксная форма»). Обе эти формы записи делают одно и то же: увеличивают на 1. Постфиксная форма i++ отличается от префиксной ++i тем, что возвращает старое значение, бывшее до увеличения.

```
// префиксная форма
let i = 1;
let a = ++i; // увеличит переменную, а затем вернёт ее значение в a
alert(a); // 2

// постфиксная форма
let i = 1;
let a = i++;
alert(a); // 1
```

Сокращённая арифметика с присваиванием

```
let n = 2;
n += 5;
n *= 2;
alert( n ); // 14
```

Оператор запятая

Обычно он используется в составе более сложных конструкций, чтобы сделать несколько действий в одной строке. Например:

```
// три операции в одной строке
for (a = 1, b = 3, c = a*b; a < 10; a++) {
 ...
}
```

Работа с консолью

Консоль даёт разработчику возможность писать JavaScript-код прямо в браузере, наблюдать за тем, что происходит на страницах, и управлять этими процессами.

Чтобы в браузере открыть консоль:
Chrome – Ctrl + Shift + i
Yandex - Ctrl + Shift + j
Или в меню → Дополнительно → Дополнительные инструменты → Консоль

Методы для вывода в консоль:

console.log(переменная)

```
<script>
        //из числа 126 получаем 621
        let a = 126;
        a1 = (Math.floor(a/100));
        a2 = (Math.floor(a/10))%10;
        a3 = a%10;
        console.log("Исходное число: " + a);
        console.log("Результат: " + (a3*100 + a2*10 + a1));
</script>
```

console.table(переменная)

```
<script>

      // Я в соцсетях
      const mySocial = {
            facebook: true,
            linkedin: true,
            flickr: true,
            instagram: true,
            VKontaktebadoo: false
            };
      console.log("Я в соцсетях");
      console.table(mySocial);

</script>
```

Задание: в данный скрипт добавьте еще 2-3 соцсети с соответствующим значением

КТ № 1

Варианты заданий

Вариант 1. Известно, что X кг конфет стоит A рублей. Определить, сколько стоит 1 кг и Y кг этих же конфет.

Вариант 2. Известно, что X кг шоколадных конфет стоит A рублей, а Y кг ирисок стоит B рублей. Определить, сколько стоит 1 кг шоколадных конфет, 1 кг ирисок, а также во сколько раз шоколадные конфеты дороже ирисок.

Вариант 3. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T_1 ч, а по реке (против течения) — T_2 ч. Определить путь S, пройденный лодкой (путь = время • скорость). Учесть, что при движении против течения скорость лодки уменьшается на величину скорости течения.

Вариант 4. Скорость первого автомобиля Vi км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга. Данное расстояние равно сумме начального расстояния и общего пути, проделанного автомобилями; общий путь = время • суммарная скорость.

Вариант 5. Скорость первого автомобиля Vi км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу. Данное расстояние равно модулю разности начального расстояния и общего пути, проделанного автомобилями; общий путь = время • суммарная скорость.

Вариант 6. Дано расстояние L в сантиметрах. Используя операцию деления нацело, найти количество полных метров в нем (1 метр =100 см).

Вариант 7. Дана масса M в килограммах. Используя операцию деления нацело, найти количество полных тонн в ней (1 тонна = 1000 кг).

Вариант 8. Дан размер файла в байтах. Используя операцию деления нацело, найти количество полных килобайтов, которые занимает данный файл (1 килобайт = 1024 байта).

Вариант 9. Даны целые положительные числа A и B (A > B). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Используя операцию деления нацело, найти количество отрезков B, размещенных на отрезке A.

Вариант 10. Даны целые положительные числа A и B (A > B). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Используя операцию взятия остатка от деления нацело, найти длину незанятой части отрезка A.

Вариант 11. Дано двузначное число. Вывести вначале его левую цифру (десятки), а затем — его правую цифру (единицы). Для нахождения десятков использовать операцию деления нацело, для нахождения единиц — операцию взятия остатка от деления.

Вариант 12. Дано двузначное число. Найти сумму и произведение его цифр.

Вариант 13. Дано двузначное число. Вывести число, полученное при перестановке цифр исходного числа.

Вариант 14. Дано трехзначное число. Используя одну операцию деления нацело, вывести первую цифру данного числа (сотни).

Вариант 15. Дано трехзначное число. Вывести вначале его последнюю цифру (единицы), а затем — его среднюю цифру (десятки).

Вариант 16. Дано трехзначное число. Найти сумму и произведение его цифр.

Вариант 17. Дано трехзначное число. Вывести число, полученное при прочтении исходного числа справа налево.

Вариант 18. Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее справа. Вывести

Вариант 19. Дано трехзначное число. В нем зачеркнули первую справа цифру и приписали ее слева. Вывести полученное число.

Вариант 20. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа (например, 123 перейдет в 213).

Вариант 21. Дано трехзначное число. Вывести число, полученное при перестановке цифр десятков и единиц исходного числа (например, 123 перейдет в 132).

Вариант 22. Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду сотен в записи этого числа.

Вариант 23. Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду тысяч в записи этого числа.

Вариант 24. С начала суток прошло N секунд (N — целое). Найти количество полных минут, прошедших с начала суток.

Вариант 25. С начала суток прошло N секунд (N — целое). Найти количество полных часов, прошедших с начала суток.

Вариант 26. С начала суток прошло N секунд (N — целое). Найти количество секунд, прошедших с начала последней минуты.

Вариант 27. С начала суток прошло N секунд (N — целое). Найти количество секунд, прошедших с начала последнего часа.

Вариант 28. С начала суток прошло N секунд (N — целое). Найти количество полных минут, прошедших с начала последнего часа.

Вариант 29. Дни недели пронумерованы следующим образом: 0 — воскресенье, 1 — понедельник, 2 — вторник, ..., 6 — суббота. Дано целое число K, лежащее в диапазоне 1-365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было понедельником.

Вариант 30. Дни недели пронумерованы следующим образом: 0 — воскресенье, 1 — понедельник, 2 — вторник, . . . , 6 — суббота. Дано целое число K, лежащее в диапазоне 1-365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было четвергом.

Вариант 31. Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, ..., 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1-365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было вторником.

Вариант 32. Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, ..., 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1-365. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было субботой.

Вариант 33. Дни недели пронумерованы следующим образом: 1 — понедельник, 2 — вторник, ..., 6 — суббота, 7 — воскресенье. Дано целое число K, лежащее в диапазоне 1-365, и целое число N, лежащее в диапазоне 1-7. Определить номер дня недели для K-го дня года, если известно, что в этом году 1 января было днем недели с номером N.

Вариант 34. Даны целые положительные числа A, B, C. На прямоугольнике размера A x B размещено максимально возможное количество квадратов со стороной C (без наложений). Найти количество квадратов, размещенных на прямоугольнике, а также площадь незанятой части прямоугольника.

Вариант 1 1. Составить программу вычисления функции: $Y=(x+1)^3+2(x-2)$.	Вариант 2 1. Составить программу вычисления функции: $Y=2x^2+3(x^2-2)^2$.	Вариант 3 1. Составить программу вычисления функции: $Y=2(x^2+3)+ (x-3)^2$.
Вариант 4 1. Составить программу вычисления функции: $Y=x^3 - (x^2+1)$.	Вариант 5 1. Составить программу вычисления функции: $Y=2x^2-3(x^2+1)^2$.	Вариант 6 1. Составить программу вычисления функции: $Y=3(x-2)^2+2(x^2+1)$.
Вариант 7 1. Составить программу вычисления функции: $Y=(x+1)^2+3(x+1)$.	Вариант 8 1. Составить программу вычисления функции: $Y=6x^2+3(x^2+1)2$.	Вариант 9 1. Составить программу вычисления функции: $Y=2(x+3)+3(x+3)^2$.
Вариант 10 1. Составить программу вычисления функции: $Y=x^2(x^2+1)$.	Вариант 11 1. Составить программу вычисления функции: $Y=4x^2+2(x^2+1)^2$.	Вариант 12 1. Составить программу вычисления функции: $Y=3(x+1)^2+2(x+1)+2$.
Вариант 13 1. Составить программу вычисления функции: $Y=x^2+2(x+3)$.	Вариант 14 1. Составить программу вычисления функции: $Y=\frac{x}{2}+\left(\frac{x}{2}\right)^2$.	Вариант 15 1. Составить программу вычисления функции: $Y=\frac{x+1}{5}+(x+1)^2$.
Вариант 16 1. Составить программу вычисления функции: $Y=x^2(x^2+1)$.	Вариант 17 1. Составить программу вычисления функции: $Y=\frac{(x+1)^2+2(x+1)}{4}$.	Вариант 18 1. Составить программу вычисления функции: $Y=\frac{x^2}{2}+\left(\frac{x^2}{2}\right)^2+3$.
Вариант 19 1. Составить программу вычисления функции: $Y=\frac{x+1}{3}+2(x+1)^2$.	Вариант 20 1. Составить программу вычисления функции: $Y=\frac{x+1}{3}+(x+1)^2$.	Вариант 21 1. Составить программу вычисления функции: $Y=\frac{x^2+1}{2}+\frac{27}{x^2+1}$.
Вариант 22 1. Составить программу вычисления функции: $Y=\frac{x-1}{5}+(x+1)^3$.	Вариант 23 1. Составить программу вычисления функции: $Y=\frac{x}{3}+\left(\frac{x}{2}\right)^2$.	Вариант 24 1. Составить программу вычисления функции: $Y=x^2+2x+3$.

Операторы сравнения и логические значения.

Операторы сравнения следующие:

- Больше/меньше: a > b, a < b.
- Больше/меньше или равно: a >= b, a <= b.
- Равно a == b.
- Не равно !=.

Логические значения. Как и другие операторы, сравнение возвращает значение. Это значение имеет логический тип.

```
alert( 2 > 1 ); // true, верно
alert( 2 == 1 ); // false, неверно
alert( 2 != 1 ); // true
```

Логические значения можно использовать и напрямую, присваивать переменным, работать с ними как с любыми другими:

```
var a = true; // присваивать явно

var b = 3 > 4; // или как результат сравнения
alert( b ); // false

alert( a == b ); // (true == false) неверно, выведет false
```

Сравнение строк. Строки сравниваются побуквенно. При этом сравниваются численные коды символов. JavaScript использует кодировку Unicode. В кодировке Unicode обычно код у строчной буквы больше, чем у прописной. Для корректного сравнения символы должны быть в одинаковом регистре.
В частности, код у символа Б больше, чем у А, поэтому и результат сравнения такой.

```
alert( 'Б' > 'А' ); // true
```

Если строка состоит из нескольких букв, то сначала сравниваются первые буквы, потом вторые, и так далее, пока одна не будет больше другой.
Если первая буква первой строки больше – значит первая строка больше, независимо от остальных символов. Если одинаковы – сравнение идёт дальше. При этом любая буква больше отсутствия буквы.

```
alert( 'Привет' > 'Прив' ); // true, так как 'е' больше чем "ничего"
```

Обычно значения, получаемые от посетителя, представлены в виде строк. Поэтому числа, полученные таким образом сравнивать нельзя, результат будет неверен.

```
alert( "2" > "14" ); // true, неверно, ведь 2 не больше 14
```

2 оказалось больше 14 , потому что строки сравниваются посимвольно, а первый символ 2 больше 1.

Правильно было бы преобразовать их к числу явным образом, поставив перед ними +

alert(+"2" > +"14"); // false, теперь правильно

Сравнение разных типов. При сравнении значений разных типов, используется числовое преобразование. Оно применяется к обоим значениям.

alert('2' > 1); // true, сравнивается как 2 > 1
alert('01' == 1); // true, сравнивается как 1 == 1
alert(false == 0); // true, false становится числом 0
alert(true == 1); // true, так как true становится числом 1.

Строгое равенство. Для проверки равенства без преобразования типов используются операторы строгого равенства === (тройное равно) и !==. Если тип разный, то они всегда возвращают false.

alert(0 === false); // false, т.к. типы различны

Взаимодействие с пользователем.

Метод alert. Выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК». Окно сообщения, которое выводится, является модальным окном. Слово «модальное» означает, что посетитель не может взаимодействовать со страницей, нажимать другие кнопки и т.п., пока не нажмёт на «ОК».

alert(сообщение)

alert ("Привет");

Метод prompt выводит окно со строкой ввода.

Prompt ("текст в окне", "текст в строке ввода")

Чтобы строка ввода оставалась чистой, не заполнять вторую пару кавычек. Если вторая пара кавычек отсутствует, то в строке появиться слово undefined. Если текст, имеющийся в строке ввода не изменить и нажать ОК, то он отобразится на странице. Если текст в строке отсутствует и ее не заполнять, то после нажатия Отмена на странице появится слово null.

```
<script>
/* Скрипт предназначен для того, чтобы получить
от пользователя информацию и поместить ее на страницу */

let user_name = prompt ("Напишите свое имя","Здесь");
document.write("Привет, " + user_name + "! Милости просим!");
</script >
```

Метод confirm выводит сообщение в модальном окне с двумя кнопками: "ОК" и "ОТМЕНА".

```
<script>
        if (confirm("Сказать привет?")) {
          alert("Привет!")
        } else {
          alert("Вы нажали кнопку отмена")
        }
    </script >
```

Конкретное место, где выводится модальное окно с вопросом – обычно это центр браузера, и внешний вид окна выбирает браузер. Разработчик не может на это влиять.

Задание: Создайте скрипт, который спрашивает имя и выводит его.

Условный оператор if

Условный оператор if имеет формат

```
If (условие)
        {
        Команда(ы)
        }
else
        {
        Команда(ы)
        }
```

Оператор if вычисляет и преобразует выражение в скобках к логическому типу. В логическом контексте: число 0, пустая строка "", null и undefined, а также NaN являются false, остальные значения – true.

```
if (0) { // 0 преобразуется к false, такое условие никогда не выполнится
  ...
}

if (1) { // 1 преобразуется к true, такое условие выполнится всегда
  ...
}
```

Можно передать уже готовое логическое значение, к примеру, заранее вычисленное в переменной:

JavaScript. Раздаточный материал №39

```
let cond = (year != 2011); // true/false

if (cond) {
  ...
}
```

Пример 1

```
<script>
let year = prompt('В каком году появилась спецификация ECMA-262 5.1?', '');
if (year < 2011) {
  alert('Это слишком рано..');
} else if (year > 2011) {
  alert('Это поздновато..');
} else {
  alert('Да, точно в этом году!');
}
  </script>
```

Решение задач:

1. Скрипт из ***Пример 1*** оформить как внешний скрипт и вызвать его на выполнение.

2. Используя конструкцию if..else, напишите код, который получает значение prompt, а затем выводит alert:
   1, если значение больше нуля,
   -1, если значение меньше нуля,
   0, если значение равно нулю.

3. Напишите код, который будет спрашивать логин (prompt).
   Если посетитель вводит «Админ», то спрашивать пароль, если нажал отмена (escape) – выводить «Вход отменён», если вводит что-то другое – «Я вас не знаю».

   Пароль проверять так. Если введён пароль «Чёрный Властелин», то выводить «Добро пожаловать!», иначе – «Пароль неверен», при отмене – «Вход отменён».

КТ № 2

Вариант 1

2. Составить программу вычисления функции:

$$y=\begin{cases}x^3, если X>0\\ 4x, если X\le 0\end{cases}$$

3. Составить программу вычисления функции:

$$y=\begin{cases}|x|, если X<-2\\ 2-x, если -2\le x\le 3\\ x, если X>3\end{cases}$$

Вариант 2

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2, если X<2\\ 2+3x, если X\ge 2\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}3x, если X<-4\\ 2+x, если -4\le x\le -2\\ x-1, если X>-2\end{cases}$$

Вариант 3

1. Составить программу вычисления функции:

$$y=\begin{cases}x^4, если X>1\\ 4-x, если X\le 1\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x+3, если X<-3\\ x-4, если -3\le x\le 3\\ 2x, если X>3\end{cases}$$

Вариант 4

1. Составить программу вычисления функции:

$$y=\begin{cases}x^3, если X\le 4\\ 5x, если X>4\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x+5, если X<-5\\ 2-x, если -5\le x\le 1\\ 4, если X>1\end{cases}$$

Вариант 5

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2, если X>-5\\ 6x, если X\le -5\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}6+x, если X<-6\\ \sin(x), если -6\le x\le 0\\ 5x, если X>0\end{cases}$$

Вариант 6

1. Составить программу вычисления функции:

$$y=\begin{cases}x, если X<6\\ 7x^2, если X\ge 6\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x+5, если X<-7\\ 7-x, если -7\le x\le 3\\ x^2, если X>3\end{cases}$$

Вариант 7

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2, если X>-7\\ 7-x, если X\le -7\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x-1, если X<-8\\ 3x, если -8\le x\le 1\\ 7x, если X>1\end{cases}$$

Вариант 8

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2, если X\le -8\\ 8x, если X>-8\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}2x, если X<-9\\ 8-x, если -9\le x\le 3\\ 9, если X>3\end{cases}$$

Вариант 9

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2+1, если X>9\\ 2x, если X\le 9\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x, если X<-1\\ 6x, если -1\le x\le 3\\ 10x^2, если X>3\end{cases}$$

Вариант 10

1. Составить программу вычисления функции:

$$y=\begin{cases}x^2+1, если X>-9\\ 4+x, если X\le -9\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}|x+1|, если X<-1\\ 2x-1, если -1\le x\le 2\\ 1+\frac{x}{2}, если X>2\end{cases}$$

Вариант 11

1. Составить программу вычисления функции:

$$y=\begin{cases}x-1, если X<12\\ 5x-2, если X\ge 12\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}3x^2, если X<-3\\ 12+x, если -3\le x\le 1\\ |x|-1, если X>1\end{cases}$$

Вариант 12

1. Составить программу вычисления функции:

$$y=\begin{cases}2x^3, если X>11\\ 2x-3, если X\le 11\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}1-x, если X<-6\\ \frac{x-1}{4}, если -6\le x\le -4\\ 2x-5, если X>-4\end{cases}$$

Вариант 13

1. Составить программу вычисления функции:

$$y=\begin{cases}x, если X\le 6\\ \frac{4}{x}(x+1), если X>6\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x+1, если X<1\\ \sqrt{2+x}, если 1\le x\le 2\\ 4x, если X>2\end{cases}$$

Вариант 14

1. Составить программу вычисления функции:

$$y=\begin{cases}1-x^2, если X>-15\\ \frac{4x}{3}, если X\le -15\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}x, если X<-1\\ 2-x, если -1\le x\le 2\\ |5-x|, если X>2\end{cases}$$

Вариант 15

1. Составить программу вычисления функции:

$$y=\begin{cases}\frac{x^3}{6}, если X>-5\\ \sqrt{4x^2+1}, если X\le -5\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}5-x, если X<-4\\ \frac{2x}{4}, если -4\le x\le 1\\ 2x^2, если X>1\end{cases}$$

Вариант 16

1. Составить программу вычисления функции:

$$y=\begin{cases}2x^2+3, если X>-2\\ 4x, если X\le -2\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}\sqrt{x^2+1}, если X<-6\\ 2+x, если -6\le x\le 2\\ x-5, если X>2\end{cases}$$

Вариант 17

1. Составить программу вычисления функции:

$$y=\begin{cases}1-x^2, если X>-3\\ 6+x, если X\le -3\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}2-x, если X<1\\ x+2, если 1\le x\le 4\\ 2x, если X>4\end{cases}$$

Вариант 18

1. Составить программу вычисления функции:

$$y=\begin{cases}x-1, если X>1\\ 2x^2, если X\le 1\end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases}1+x, если X<-1\\ 6x, если -1\le x\le 2\\ 1-x^2, если X>2\end{cases}$$

Вариант 19

1. Составить программу вычисления функции:

$$y=\begin{cases} x^3+5, если X>1 \\ 4-x, если X\le 1 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} |x-1|, если X<0 \\ 2x, если 0\le x\le 2 \\ x+6, если X>2 \end{cases}$$

Вариант 20

1. Составить программу вычисления функции:

$$y=\begin{cases} 1-x^2, если X<0 \\ 3x, если X\ge 0 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} 3-x, если X<-3 \\ 2+x^2, если -3\le x\le 2 \\ |1-x|, если X>2 \end{cases}$$

Вариант 21

1. Составить программу вычисления функции:

$$y=\begin{cases} x^3, если X>2 \\ 4-3x, если X\le 2 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} 3x+1, если X<3 \\ 4x, если 3\le x\le 5 \\ 2-x, если X>5 \end{cases}$$

Вариант 22

1. Составить программу вычисления функции:

$$y=\begin{cases} x^3-2, если X\le 3 \\ 4-x, если X>3 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} x+5, если X<-4 \\ 3x^2, если -4\le x\le 2 \\ 2x-1, если X>2 \end{cases}$$

Вариант 23

1. Составить программу вычисления функции:

$$y=\begin{cases} 3x^2, если X>-1 \\ 6-x, если X\le -1 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} 6x, если X<2 \\ |2-x|, если 2\le x\le 6 \\ 5-x, если X>6 \end{cases}$$

Вариант 24

1. Составить программу вычисления функции:

$$y=\begin{cases} x-3, если X<2 \\ 2x^2-1, если X\ge 2 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} x+5, если X<-4 \\ 7x-1, если -4\le x\le 5 \\ 1-x^2, если X>5 \end{cases}$$

Вариант 25

1. Составить программу вычисления функции:

$$y=\begin{cases} 2x-x^2, если X>-1 \\ 2x, если X\le -1 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} x^2-1, если X<-3 \\ 3-x, если -3\le x\le 0 \\ 4x, если X>0 \end{cases}$$

Вариант 26

1. Составить программу вычисления функции:

$$y=\begin{cases} x^2-1, если X\le -6 \\ 8+x, если X>-6 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} 2-x, если X<-7 \\ 4x, если -7\le x\le 1 \\ 9-x^2, если X>1 \end{cases}$$

Вариант 27

1. Составить программу вычисления функции:

$$y=\begin{cases} 1+x^2, если X>5 \\ 2-x, если X\le 5 \end{cases}$$

2. Составить программу вычисления функции:

$$y=\begin{cases} 2x, если X<-1 \\ \sin x-1, если -1\le x\le 3 \\ x^2, если X>3 \end{cases}$$

Смотрим видео:

https://www.youtube.com/watch?v=dcnCI_-Uq4E&list=PLA0M1Bcd0w8x9TltCzZDhw0SatK1d10yy&index=1

https://www.youtube.com/watch?v=3UEoc5iWZUw&list=PLA0M1Bcd0w8x9TltCzZDhw0SatK1d10yy&index=2

Тема: Логические операторы.

Логические операторы могут применяться к значениям любого типа и возвращают также значения любого типа.

Логический оператор || (ИЛИ)

Оператор ИЛИ возвращает то значение, на котором остановились вычисления, причём, не преобразованное к логическому типу, т.е. оператор ИЛИ вычисляет ровно столько значений, сколько необходимо – до первого true.

Раздаточный материал №43

```
result = a || b;

alert( true || true ); // true
alert( false || true ); // true
alert( true || false ); // true
alert( false || false ); // false
```

Если значение не логического типа – то оно к нему приводится в целях вычислений.

Раздаточный материал №44

```
if (1 || 0) { // сработает как if( true || false ), число 1 будет воспринято как true, а 0 – как false
  alert( 'верно' );
}
```

Обычно оператор ИЛИ используется в if, чтобы проверить, выполняется ли хотя бы одно из условий.

Раздаточный материал №45

```
 let hour = 12,
  isWeekend = true;

if (hour < 10 || hour > 18 || isWeekend) {
  alert( 'Офис до 10 или после 18 или в выходной закрыт' );
}
```

JavaScript вычисляет несколько ИЛИ слева направо. Если первый аргумент – true, то результат заведомо будет true (хотя бы одно из значений – true), и остальные значения игнорируются. Если все значения «ложные», то ИЛИ возвратит последнее из них

Раздаточный материал №46

```
let x;
true || (x = 1);
alert(x); // undefined, x не присвоен

let x;
false || (x = 1);
alert(x); // 1
```

```
let undef; // переменная не присвоена, т.е. равна undefined
let zero = 0;
let emptyStr = "";
let msg = "Привет!";
let result = undef || zero || emptyStr || msg || 0;
alert( result ); // выведет "Привет!" - первое значение, которое является true


alert( undefined || '' || false || 0 ); // 0
```

Логический оператор && (И)

Раздаточный материал №47

```
result = a && b;

alert( true && true ); // true
alert( false && true ); // false
alert( true && false ); // false
alert( false && false ); // false
```

Раздаточный материал №48

```
 let hour = 12,
  minute = 30;

if (hour == 12 && minute == 30) {
  alert( 'Время 12:30' );
}
```

В логическом И допустимы любые значения. Если левый аргумент – false, оператор И возвращает его и заканчивает вычисления. Иначе – вычисляет и возвращает правый аргумент. Можно передать и несколько значений подряд, при этом возвратится первое «ложное» (на котором остановились вычисления), а если его нет – то последнее. Оператор && вычисляет операнды слева направо до первого «ложного» и возвращает его, а если все истинные – то последнее значение. Приоритет оператора И больше, чем ИЛИ.

Раздаточный материал №49

```
 if (1 && 0) { // вычислится как true && false
  alert ('не сработает, т.к. условие ложно');
}

// Первый аргумент - true,
// Поэтому возвращается второй аргумент
alert( 1 && 0 ); // 0
alert( 1 && 5 ); // 5

// Первый аргумент - false,
// Он и возвращается, а второй аргумент игнорируется
alert( null && 5 ); // null
alert( 0 && "не важно" ); // 0
```

```
alert( 1 && 2 && null && 3 ); // null

alert( 1 && 2 && 3 ); // 3

alert( 5 || 1 && 0 ); // 5
```

Логический оператор ! (НЕ) получает один аргумент. Сначала приводит аргумент к логическому типу true/false. Затем возвращает противоположное значение.

Раздаточный материал №50

```
let result = !value;

alert( !true ); // false
alert( !0 ); // true
```

Тема: Работа с формами

Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга.

Устанавливают форму на веб-странице теги <form> и </form>. Тег имеет следующие атрибуты:

accept-charset - устанавливает кодировку, в которой сервер может принимать и обрабатывать данные, например Windows-1251, UTF-8 и др.

action - адрес программы или документа (обработчик), который обрабатывает данные формы. В качестве обработчика может выступать CGI-программа или HTML-документ, который включает в себя серверные сценарии. После выполнения обработчиком действий по работе с данными формы он возвращает новый HTML-документ. Если атрибут action отсутствует, текущая страница перезагружается, возвращая все элементы формы к их значениям по умолчанию. В качестве значения принимается полный или относительный путь к серверному файлу (URL).

HTML. Раздаточный материал №53

```
<form action="handler.php">
  <p>...</p>
 </form>
```

HTML. Раздаточный материал №54

```
<form action="handler.php" method="post">
```

name – уникальное имя формы, которое чаще всего используется для обращения к форме из скриптов.

Допускается внутрь контейнера <form> помещать другие теги, при этом сама форма никак не отображается на веб-странице, видны только ее элементы и результаты вложенных тегов.

Элементы формы

На форме могут быть определены следующие элементы управления: кнопка, флажок, радиокнопка, меню, строка текста, пароль, скрытое поле, файл, объект. Основной атрибут тега <input>, определяющий вид элемента — type.

Строка текста

Позволяет вводить короткие фрагменты текста – имена, адреса и т.п. При его создании используется name для указания имени, которое поможет различить его в наборе данных на сервере. Наименование поля может состоять из букв, цифр и _.

Атрибуты:

size – ширина строки текста.

maxlength – ограничивает емкость строки текста.
value - начальный текст отображаемый в поле.

HTML. Раздаточный материал №55

```
<br> Ключевые слова: <input type=”text” name=”key” size=”30” maxlength=”25”>
```

type=”password” - строка текста для хранения конфиденциальной информации.

Многострочный текст
<textarea> - создает область ввода многострочного текста.
name - имя области ввода.
rows, cols – ширина и высота области ввода
wrap= задает правила переноса целых слов текста:

soft – включает функцию переноса, но не воздействует на внешний вид данных, передаваемых на сервер.
hard – форматирование данных как на экране, так и во внутреннем буфере браузера.

HTML. Раздаточный материал №56

```
<textarea name=”unit” rows=”5” cols=”45” wrap=”hard”>
</textarea>
```

Флажок
type=”checkbox”
name – имя группы элементов-флажков.
value – служит для назначения имени каждому флажку в группе. Имя будет включено в итоговый набор данных формы, если пользователь выберет соответствующий флажок.
В значениях name и value используются только буквы, цифры и _.
checked – позволяет установить флажок предварительно, чтобы при воспроизведении страницы опция выбиралась автоматически.

HTML. Раздаточный материал №57

```
<br> Установите атрибуты файла
<br><input type=” checkbox” name=”flag” value=”sys” checked> системный
<br><input type=” checkbox” name=”flag” value=”archive”>архивный
<br><input type=” checkbox” name=”flag” value=”read”>только чтение
```

Переключатель
Используется при выборе только одной опции. Создание аналогично флажку. Type=”radio”.

Меню
name, value – аналогично флажку.
size – число элементов меню отображаемых единовременно (по умолчанию – один элемент).
<option> - определяет отдельный элемент меню.
selected – автоматический выбор опции.

HTML. Раздаточный материал №58

```
<br> Какой цвет Вам нравится?
<br><select name=”color”  size=”3”>
 <option value=”blue”>Синий
 <option value=”red” selected >Красный
 <option value=”yellow”>Желтый
</select>
 <br><br><br><input type=”submit” value=”отправить”> 
<input type=”reset” value=”очистить”>
```

Кнопка submit служит для завершения пользовательского сеанса заполнения формы и отправки данных на сервер.
Кнопка reset очищает все поля формы и возвращает их к исходному состоянию.
Value – определяет текст, отображаемый на поверхности кнопки (по умолчанию submit/reset).

Кроме кнопки Submit для отправки формы на сервер можно нажать клавишу Enter в пределах формы. Если кнопка Submit отсутствует в форме, клавиша Enter имитирует ее использование, но только в том случае, когда в форме имеется только один элемент <input>. Если таких элементов два и более, нажатие на <Enter> не вызовет никакого результата.

Второй способ создания кнопки основан на использовании тега <button>.
Button позволяет отображать разнообразные элементы в виде кнопок. Кнопки могут быть созданы с использованием всех возможностей, доступных в HTML и таблицах стилей (Поддерживается не всеми браузерами).

HTML. Раздаточный материал №59

```
<button style="font-family:Arial; font-size:16pt; color:navy">добавьте информацию<span style="font-style:italic; color:green> Пожалуйста!
</span>
</botton>
```

HTML. Раздаточный материал №60

Вариант 1	https://professorweb.ru/my/html/html5/level2/files/img46023.jpg
Вариант 2	https://cf.ppt-online.org/files/slide/v/vRAlxeaYzgFDVC2krBdPXQtZbW5McoLJ08In6g/slide-21.jpg
Вариант 3	https://lh5.googleusercontent.com/-wG_YHAIbVZU/Ud696wJg0FI/AAAAAAAACP4/eaIzPTZRixE/w596-h642-no/4_3.png
Вариант 4	https://www.devlounge.net/wp-content/uploads/2011/02/24ways.jpg
Вариант 5	https://i.pinimg.com/originals/73/c6/0d/73c60def8c55043f9fd27b370530a9cf.jpg
Вариант 6	https://smashinghub.com/wp-content/uploads/2011/08/html5-forums-14.jpg
Вариант 7	https://soulcompas.com/wp-content/uploads/2020/06/html5-admin-template-free.jpg
Вариант 8	https://www.webasyst.ru/wa-data/public/updates/img/09/209/7355/7355.970.jpg
Вариант 9	https://www.zoho.com/creator/images/subpages/htmlforms/workshop_registration.gif
Вариант 10	https://bramus.github.io/ws1-sws-course-materials/assets/03/testform.jpg
Вариант 11	https://www.bestfree.ru/uslugi/constructors/SozdanieSaytaNaUcoz_3.png
Вариант 12	http://ip-whois.net/forma-obratnoj-svyazi/img/form1.jpg
Вариант 13	https://www.digiseller.ru/preview/125077/p1_30116215520413.JPG
Вариант 14	https://mob25.com/images/Perl/images/ris150_1.jpg
Вариант 15	https://i2.wp.com/ps.w.org/ultimate-form-builder-lite/assets/screenshot-1.png
Вариант 16	https://i.stack.imgur.com/px0jW.png

Тема: Функции в JavaScript.

При создании скрипта разумно в нем выделить логически независимые части – функции.

Раздаточный материал №61

```
function F(V)
{
S
}
```

где F – имя функции, по которому к ней можно обращаться,
V – список параметров, разделенных запятыми,
S – тело функции (операторы).

Описание функции не может быть вложено в описание другой функции. Начальные значения параметрам присваиваются при обращении к функции. Если описание функции имеет вид

Раздаточный материал №62

Function F(V1,V2,…,Vn) {S},
то вызов функции должен иметь вид F(e1,e2,…,en),
где e1,e2,…,en - выражения, задающие фактические значения параметров.

Параметры V1,V2,…,Vn – формальные параметры, которые получают смысл только после задания в функции фактических параметров e1,e2,…,en, с которыми функция затем и работает.

Если в функции параметры отсутствуют, то описание функции следующее:

Раздаточный материал №63

```
Function F()
{
S
}
```

где F – имя функции, по которому к ней можно обращаться,
S – тело функции (операторы).

Вызов функции:
1. Указав ее имя в качестве оператора JavaScript.
2. Указав ее имя в HTML-ссылке
   <a href=”javascript:hello()”> … </A>
3. В адресной строке
   javascript:hello()
4. В форме при помощи атрибута action

Раздаточный материал №64

```
<html>
<head>
<script>
function hello() {
document.write("Привет и добро пожаловать");
}
</script>
</head>
<body>
<form action="javascript:hello()">
<input type="submit" value="Отобразить приветствие">
</form>
</body>
</html>
```

Решение задач. *Раздаточный материал №65*

1. Переработать функцию из раздаточного материала № 64 так, чтобы она сначала спрашивала имя пользователя, а затем выводила сообщение "Привет *имя_пользователя* и добро пожаловать".

2. Создать скрипт, в котором функция спросит имя пользователя и поприветствует его, а если пользователь не ввел имя, т.е. нажал «Отмена», то выдаст сообщение об этом.