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:

```
var 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.

Локальными являются переменные, объявленные внутри функции с помощью var. Они будут доступны только внутри функции и не будут распознаны вне ее, то есть значение переменной будет неизвестно.

Глобальными являются переменные, объявленные внутри функции без var. Переменные, объявленные вне функции даже при использовании var, так же будут глобальными и их можно использовать в любом месте текущего документа, как внутри функции, так и вне ее. Как правило, перед именем глобальной переменной ставится буква g (например, gVar1).

Для объявления переменной используется ключевое слово Var

```
Var  имя_переменной = значение
```

Значением может быть команда, другая переменная, выражение.

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.

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

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

var apples = "2";
var 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:

```
var 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)
```

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:

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

Декремент -- уменьшает на 1:

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

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

Сокращённая арифметика с присваиванием

```
var 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++) {
 ...
}
```

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.

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
```