Разработка веб-сайта с играми. Функции.

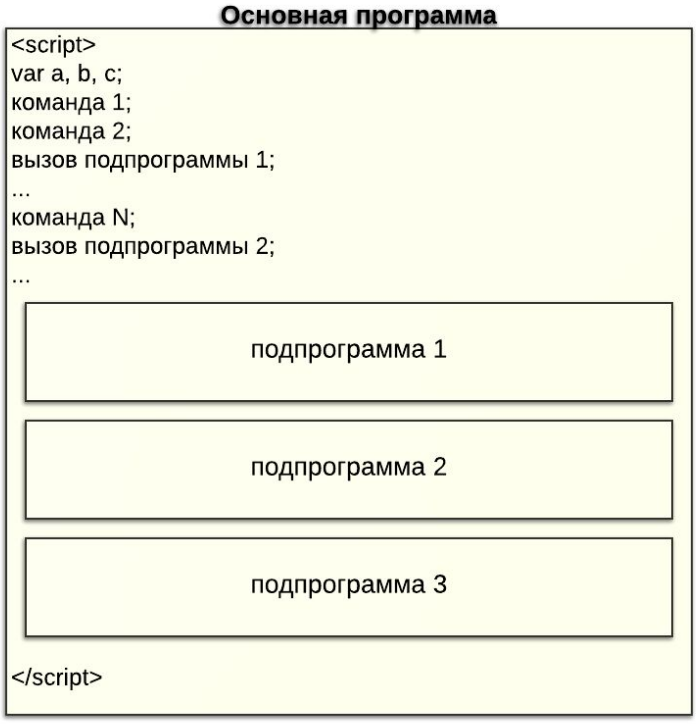
Функции. Рекурсии. Разработка интерактивных игр на сайте.

Функции

Функции — это рабочие лошадки JavaScript. Функции сами по себе играют те роли, которые в других языках исполняются самыми разными средствами: процедурами, методами, конструкторами, классами и модулями. После освоения функций вы овладеете существенной частью JavaScript.

Функция - это подпрограмма

Программисты при написании своих программ постоянно используют функции. Одной из главных целей использования функций является упрощение программирования. Действительно, ведь при написании большой программы часто возникают случаи, когда один и тот же код нужно выполнять много раз. В этом случае повторяющийся код может быть оформлен в виде подпрограммы внутри программы только со своим собственным именем. Такие программы внутри программы в общем случае называются подпрограммами, а в разных языках программирования эти подпрограммы могут носить разное название: функции, процедуры, методы - в зависимости от назначения подпрограммы.



На самом деле, когда мы писали alert или document.write и другие команды, мы уже использовали стандартные функции JavaScript.

Скажем привет функциям

Наберите программу, которая показывает, как описать функцию, и как её вызвать. В отличие от некоторых языков программирования, описание функции может идти после её вызова, но лучше старайтесь сначала декларировать функцию, а потом вызывать ее.

```
<script>
    function SayHello() {
        alert("Hello!");
    }
    SayHello();
</script>
```

SayHello() - это вызов функции. Отличительной особенностью функций от переменных является наличие скобок после её названия.

```
function SayHello() {
    alert("Hello!");
}
```

Это описание функции. Описание функции начинается с ключевого слова function, за которым идет название функции, скобки, в которых могут быть параметры функции, и фигурные скобки, в которых заключено тело функции.

Параметры функции

Мы написали простую функцию, которая выводит на экран приветствие. Но что если нам нужно, чтобы в приветствии так же было имя человека с которым мы хотим поздороваться? Писать функции с разными названиями? Для этих целей служат параметры функции.

```
<script>
    function SayHello(name)
    {
        document.write("Hello!"+name+"<br>");
    }
    SayHello("Phil");
    SayHello("Fred");
</script>
```

Если мы хотим, чтобы была возможность передать данные из основной программы в функцию, то можно при описании функции в скобках перечислить названия параметров функции. Теперь при вызове функции мы можем в скобках писать данные, которые будут переданы через параметры внутрь функции. Можно воспринимать параметры как специальные переменные, необходимые для передачи данных внутрь функции. Можно описать сколько угодно много параметров, перечисляя их через запятую.

Функции вычисляют и возвращают значения

Довольно часто необходимо, чтобы главная программа могла получить результат работы функции. В этом случаем можно использовать механизм возврата значения из функции. Тогда сама функция в конце своей работы возвращает какой-то результат в то место где она была вызвана, и мы можем использовать её возвращаемое значение.

Пример функции, возвращающей квадрат числа x:

```
<script>
    function degree2(x) {
```

```
        return x * x;
    }
    alert(degree2(5));
</script>
```

Упрощение логики программы

Как уже было сказано, функции позволяют сделать программу более читаемой. Давайте рассмотрим это на примере решения следующей задачи.

Написать программу: Определить значение z=max(a,2*b)*max(2*a-b,b), где max(x,y) – максимальное значение из чисел x,y.

Решение без функции	Решение с функцией

<script>
a = parseInt(prompt("a:"));
b = parseInt(prompt("b:"));
 	
max1 = a;
if (2 * b > max1)
max1 = 2 * b;
 	
max2 = 2 * a - b;
if (b > max2)
max2 = b;
 	
z = max1 * max2;
alert(z);
</script>
	<script>
	a = parseInt(prompt("a:"));
	b = parseInt(prompt("b:"));
	
	function Max(x, y) {
	if (x > y)
	return x;
	else
	return y;
	}
	
	z = Max(2 * b, a) * Max(2 * a - b, b);
	alert(z);
	</script>

Не знаю как вам, но нам кажется, что с функцией программа становится более простой для понимания.

Рекурсии

Рекурсия - это вызов функции самой себя.

Рассмотрим простой пример:

```
<script>
    function Print10(n) {
        document.write(n + "<br>");
        if (n < 10)
            Print10(n + 1);
    }
    Print10(1);
</script>
```

Здесь функция вызывает саму себя, пока n<10. Таким образом на экран выведется последовательность чисел от 1 до 10.

Рекурсия является альтернативой циклам, и с её помощью часто можно написать весьма элегантные алгоритмы решения задач.

Задание

Попробуйте переместить document.write после условия. Объясните получившийся результат.

Рекурсивный факториал

Факториал F(n) это произведение чисел от 1 до n. Например, F(5)=1*2*3*4*5=120.

Факториал хорошо решается с помощью рекурсии, так как факториал можно описать рекуррентно:

F(n)=F(n-1)*n, n>1
F(0)=1, n=0. Где n - целые числа

Если мы можем построить рекуррентное соотношение, то это рекуррентное соотношение легко реализуется с помощью рекурсии

```
<script>
    function factorial(n) {
        if (n == 0) {
            return 1;
        } else if(n < 0) {
            alert('Факториал рассчитывается только для натуральных чисел.');
        } else {
            return factorial(n - 1) * n;
        }
    }
    alert(factorial(5));
</script>
```

При этом вычисления происходят в следующем порядке:

1. F(5)=F(4)*5 2. F(4)=F(3)*4 3. F(3)=F(2)*3 4. F(2)=F(1)*2 5. F(1)=F(0)*1 6. F(0)=1	7. F(1)=1*1=1 8. F(2)=1*2=2 9. F(3)=2*3=6 10. F(4)=6*4=24 11. F(5)=24*5=120

Домашнее задание

1. **Функции для работы с массивами**
    a. Написать функцию, которая принимает в качестве параметра число n. Результатом работы функции является массив из N элементов со значениями 1, 2, 3… n.
    b. Написать функцию, которая принимает массив чисел. Результатом работы функции является сумма чисел этого массива.
    c. * Написать функцию, которая на вход получает массив целых чисел, и в качестве результата возвращает максимальное число.
    d. * Написать функцию, которая на вход получает массив целых чисел, и в качестве результата возвращает минимальное число
    e. * Написать функцию, которая на вход получает массив целых чисел, и в качестве результата возвращает только четные числа из этого массива. Чтобы определить четность числа, воспользуйтесь оператором для подсчета остатка от деления: x % 2. Если остаток от деления числа на 2 равен 0, то число - четное.
2. *** Добавить в программу по слепой печати еще несколько уровней.**
    a. **Уровень 2:** пользователь видит сообщение: “Поставьте мизинец левой руки на букву Ф, безымянный палец - на Ы, средний - на В, указательный - на А. Мизинец правой руки на букву Ж, безымянный палец - на Д, средний - на Л, указательный - на О Запомните расположение пальцев.. А теперь, повторяйте за мной”. Пользователь получает последовательность из букв Ы и В длиной в 10 символов, и должен её воспроизвести. Если ему это не удалось - попытка повторяется. В случае успеха переходим к следующему уровню.
    b. **Уровень 3:** пользователь видит сообщение: “Поставьте мизинец левой руки на букву Ф, безымянный палец - на Ы, средний - на В, указательный - на А. Мизинец правой руки на букву Ж, безымянный палец - на Д, средний - на Л, указательный - на О Запомните расположение пальцев.”. Пользователь получает последовательность из букв О и Ж длиной в 10 символов, и должен её воспроизвести. Если ему это не удалось - попытка повторяется. В случае успеха переходим к следующему уровню.

3. **Доработать игру в загадки:**
    a. Избавится в игре в загадки от дублирования кода, используя массивы и функции.
    b. Сделать так, чтобы программа поддерживала несколько правильных ответов на один и тот же вопрос. Все возможные варианты задаются программистом в коде программы.