Базовый курс JavaScript. Линейные структуры, события и регулярные выражения.

JavaScript — основной язык программирования в интернете. Он позволяет расширить стандартное поведение пользователя на веб-страницах. С JavaScript страницы из сухих и статичных, которые просто показываются друг за другом, становятся интерактивными и взаимодействуют с пользователями, откликаются на их действия. Можно получать и применять данные из Сети и делать многое другое.

Вы уже умеете создавать HTML-страницы с их структурой, наполнять их и применять стили. Но пока они не обладают уникальным поведением: умеют только предоставлять навигацию по ссылкам, реагировать на наведение мышки и подобное. Но к поведению веб-страниц требований больше: они должны быть интерактивными, взаимодействовать с пользователем. В игру вступает JavaScript: посмотрим, какое место он занимает при оформлении поведения веб-страниц.

1. HTML определяет контент страниц, их структуры.

2. CSS определяет стиль оформления страницы.

3. Сервер обрабатывает HTTP-запросы, при необходимости передавая их интерпретатору.

4. Интерпретатор формирует HTML-документ, при необходимости обращаясь к БД.

JavaScript на этой схеме новичок. Он умеет определять поведение страниц. Например, странице нужно реагировать, когда пользователь кликает по кнопке «Отправить заказ». Появляется возможность наращивать программный код, выполняющий динамические операции.

Научимся подключать JavaScript-код к странице. Это делается при помощи HTML-тега <script></script>.

```
<script>
alert("Hello world!");
prompt("Напиши, что-то");
</script>
```

Команда alert только выводит сообщение на экран при запуске браузера с нашей страницей, а команда prompt выводится с текстовым полем и двумя кнопками: "ОК" и "ОТМЕНА". Мы можем прописать весь JavaScript-код в нашем файле html или создать для него отдельный файл с расширением .js и подключить его к html-страницы, на которой он должен выполнятся:

```
<script> src="/path/to/script.js" </script>
```

Структура кода

В JavaScript код состоит из команд. Сама по себе команда описывает операцию. Набор команд, из которых состоит код, и задает поведение страницы. Чтобы объявить переменную, начинают с ключевого слова var. Затем пишут имя переменной. При необходимости можно указать ее начальное значение через знак равенства, но это необязательно, так как его можно присвоить позднее.

```
var cost = 500;
var color;
```

В JavaScript регистр символов имеет значение практически везде. Поэтому будьте внимательны, давая имена переменным.

Типы данных

Типы данных в JavaScript делятся на две группы: простые типы и объекты. Простые типы — это числа, текстовые строки и логические, или булевы, значения.

Тип number (число)

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

Нужно учитывать, что тип number (число) используется как для целых, так и для дробных чисел.

Тип string (строка)

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

В JavaScript одинарные и двойные кавычки равноправны. Выберите для себя один тип и будьте последовательны.

Булевый (логический) тип boolean

У данного типа есть два значения: true (истина) и false (ложь).

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

Тип object (объекты)

Перечисленные выше типы называют примитивными. Тип «объекты» применяется для хранения коллекций данных и объявления сложных сущностей.

Операторы в JavaScript

Как и в любом языке программирования, в JavaScript есть операторы. Сам по себе оператор — это наименьшая автономная часть языка программирования, то есть команда. Например, при сложении двух чисел (3+2) работает оператор сложения с двумя операндами. Операторы бывают унарными и бинарными. Унарный оператор применяется к одному операнду:

```
var x = 1;
x = -x; // унарный минус
```

Бинарный — к двум:

```
var a = 1;
var b = 2; a + b; // бинарный плюс
```

У некоторых операторов есть особые названия:

- инкремент — означает увеличение операнда на установленный фиксированный шаг (как правило, единицу). Он же a++ или a+1;
- декремент — обратная инкременту операция: a-- или a-1;
- конкатенация — сложение строк. Обратной операции нет.

```
var a = "моя" + "строка";
```

Операторы if, if-else

Для реализации ветвления в JS используется оператор if:

```
if ( Условие ) {
    Действие;
```

}

Условие — это любое выражение, возвращающее булевское значение (true, false), то есть вопрос, на который ответить можно только «да» или «нет». Обычно условием является одна или несколько операций сравнения, объединенных логическими связками (И, ИЛИ). В результате проверки условия может выполняться сразу несколько операторов.

Разберем вариант, когда одного условия недостаточно. Рассмотрим пример ветвления, когда в случае истины выполним одно действие, а иначе — другое.

```
if ( Условие ) {
        Действие1;
}
else {
        Действие2;
}
```

Реализуем простой пример:

```
var x = 5;
var y = 42;
if ( x > y )
        alert (x + y);
else
        alert(x * y);
```

Если по условию нужно выполнять всего один оператор, можно не ставить фигурные скобки. Но не всегда можно уложить логику ветвления в две ветки. JS позволяет разделять программу на сколько угодно вариантов с помощью конструкции else if, которая позволяет анализировать дополнительное условие. При этом выполняться будет первое условие, вернувшее true. Представим следующую задачу: даны два произвольных числа, необходимо вывести на экран их соотношение друг с другом. По сути, здесь три варианта: либо первое число больше, либо второе, либо они равны.

```
        var x = 5;
        var y = 42;
        if(x > y)
           alert("x больше y");
        else if ( x < y )
           alert("x меньше y");
        else
           alert("x равно y");
```

Комбинации условий

В условном операторе можно комбинировать условия при помощи логических операций:

ИЛИ (x || y) — если хотя бы один из аргументов true, то возвращает true, иначе — false;

И (x && y) — возвращает true, если оба аргумента истинны, а иначе —– false;

НЕ (!x) — возвращает противоположное значение.

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

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

Функции

Функция — это блок кода, к которому можно обращаться из разных частей скрипта. Функции могут иметь входные и выходные параметры. Входные могут использоваться в операциях, которые содержит функция. Выходные устанавливаются функцией, а их значения используются после ее выполнения. Программист может создавать необходимые ему функции и логику их выполнения. Функция в JS объявляется с помощью ключевого слова function. За ним следует ее название, которое мы придумываем сами. Затем в круглых скобках через запятую указываются параметры, которые данная функция принимает. По сути, параметры — это входные данные для функции, над которыми она будет выполнять какую-то работу. После параметров в фигурных скобках следует тело функции. Когда функция объявлена, можем ее вызвать и посмотреть, как она работает.

```
function имя_функции(параметр1, параметр2, ...) {
    Действия
}
```

Создадим функцию, которая будет сравнивать числа:

```
function compare_numbers(x, y) {
    if (x > y)
        alert("x > y");
    else if (x < y)
        alert("x < y");
    else
        alert("x = y");
}
compare_numbers(10, 20);
compare_numbers(20, 10);
compare_numbers(20, 20);
```

При вызове функции в нее нужно передавать такое количество параметров, которое заявили при ее создании. Их может быть от нуля и более. Если параметры не переданы, при вызове функции нужно просто указать пустые скобки. Оператор return позволяет завершить выполнение функции, вернув конкретное значение. Если в функции не указано, что она возвращает, то результатом ее работы может быть только вывод текста на экран. Напишем функцию, возвращающую среднее арифметическое двух чисел:

```
function average(x, y) {
    return (x + y)/2;
}
```

```
avg = average(42, 100500);
alert(avg);
```

Так мы можем не только научить скрипт определенным навыкам, но и сохранить результат выполнения каждой функции для дальнейшего использования

События в JavaScript

Событие — это сигнал от браузера о том, что что-то произошло. При каждом нажатии кнопки, перемещении мыши, поступлении данных, изменении размеров окон генерируются события. И при новом подходе код получает возможность их обработать: выполнить действие в ответ на пользовательское. Не обязательно реагировать на все события, но всегда есть требующие этого.

Можно указать обработчик любому событию. Как правило, он представляет собой фрагмент кода, который мы определяем к выполнению, когда возникает событие. Обработчик — с точки зрения программного кода — это обычная функция. В данном контексте ее называют функцией-обработчиком. Чтобы обработчик можно было вызывать при возникновении события, его нужно зарегистрировать. Есть несколько способов — они зависят от типа события.

Браузерные события

Нам знакомы два типа событий:

1. Событие загрузки (load), происходящее при завершении загрузки страницы браузером.

2. Событие щелчка (click), которое происходит, когда пользователь щелкает на элементе страницы.

При работе с JS происходит еще множество других событий: загрузка данных из сети, геопозиционирование, таймеры. Для событий обработчик зачастую назначался свойству — onload, onclick. Рассмотрим хронометражные события. Чтобы задать их обработку, разработчик вызывает функцию setTimeout, в которой передает нужную функцию-обработчик:

```
setTimeout(reminder, 3000);

function reminder() {
   alert("Ты здесь слишком долго");
}
```

Этот код реализует таймер, который будет срабатывать каждые три секунды, выводя указанное сообщение. Обратите внимание, что здесь мы передаем одной функции другую в качестве аргумента. Функция setTimeout во время работы создает таймер обратного отсчета и связывает с ним указанный в аргументах обработчик, который будет вызываться при обнулении таймера. Чтобы сообщить setTimeout, кого следует вызывать, нужно передать ей ссылку на функцию-обработчик. Сама setTimeout сохраняет ссылку, чтобы использовать ее позднее, когда сработает таймер.

Получить детальную информацию о событии в обработчике можно посредством объекта события (Event). Данный объект создаёт браузер, когда это событие происходит. В него он помещает много различной информации. Например, для события click: какая клавиша нажата, координаты курсора и др. Обработчик назначается

вызовом addEventListener с двумя обязательными аргументами: element.addEventListener(event, handler);

- event — имя события (например, click).
- handler — ссылка на функцию, которую надо поставить обработчиком.

```
document.addEventListener("click", reminder);
```

Поиск элементов на странице с помощью JavaScript

Поиск элемента по значению id

Поиск элементов на страницу осуществляется на JavaScript посредством специальных методов. Одним из методов является поиск по идентификатору getElemetById

```
var name = document.getElementById("name").value;
console.log(name);
```

Поиск элементов по CSS селектору

В настоящее время наиболее популярным методом поиска структур является querySelector.

```
var images = document.querySelector('img');
console.log(images); // Вывод в консоль для наглядности
```

Регулярные выражения

Часто в программе нам нужно проверить, соответствует ли строка заданному формату. Представьте, что пользователь заполняет форму. Он вводит телефон и адрес электронной почты. Перед отправкой формы нужно проверить, правильно ли пользователь заполнил поля – например, не попали ли случайно в номер телефона буквы. Другая ситуация: нам нужно заменить в тексте одни символы на другие – например, двойные кавычки на одинарные. Для решения подобных задач используются регулярные выражения.

Регулярное выражение – способ записи текстовых шаблонов. По сути, это обычная текстовая строка, написанная на специальном языке, но она не содержит конкретный набор символов, а задаёт общие правила.

Вот несколько основных правил языка регулярных выражений:

```
/abc/ Идущие подряд символы abc
/[abc]/ Один из символов a, b или c
/[^abc]/ Ни один из символов, т. е. не a, не b и не c
/[a-z]/ Диапазон символов, идущих подряд в таблице Unicode
/\b/ Граница слова
/\B/ Не граница слова
/\d/ Цифра
/\D/ Не цифра
/\w/ Латинская буква, цифра или _
/\W/ Не латинская буква, не цифра и не _
/\s/ Пробельный символ
/\S/ Непробельный символ
/a{3}/ Строго 3 символа a подряд
/a{2,4}/ От 2 до 4 символов a подряд
/a+/ 1 и более символов a подряд
/a*/ 0 и более символов a подряд
```

```
/a?/ 0 или 1 символ a
/./ Один любой символ, кроме переноса строки
```

На самом деле правил значительно больше, но запоминать их не нужно. Всегда можно обратиться к справочнику. Комбинируя правила, можно создавать шаблоны:

```
/a25?0+/
```

Шаблон для строки, которая начинается с a2, затем в ней может быть цифра 5, а потом один и более нулей. То есть подойдут строки a20, a250, a2000000 и т. д. Если в шаблоне нужно использовать служебные символы вроде ? или +, то их нужно экранировать:

```
/abc\?/ Найдёт 'abc?'
```

Также в регулярных выражениях есть флаги, влияющие на поиск. Вот три основных:

- g // global Глобальный поиск, т. е. поиск всех соответствий
- i // insensitive Поиск без учёта регистра
- m // multiline Многострочный текст

Как узнать, есть ли совпадение

Для того чтобы определить, есть ли в строке совпадение с регулярным выражением, используется метод регулярного выражения test(). Он не определяет, где именно обнаружено совпадение, а лишь возвращает true или false:

```
const str = '123 abc 456';
var regexp = /abc/; regexp.test(str); // Вернёт true
var regexp2 = /xyz/; regexp2.test(str); // Вернёт false
```

Задание

1. Игра «Угадай число»

Браузер будет загадывать число, а мы будем отгадывать. Попытки отгадать число будут идти через диалоговое окно — prompt. Браузер будет сообщать в ответ, больше или меньше загаданного наш ответ.

Алгоритм будет таким:

1) Выводится окно запроса предположения, загаданного числа.
2) Браузер проверяет число и возвращает результат (больше/меньше).
3) Повторяем до тех пор, пока число не будет угадано.
4) Как только число угадано, алгоритм прекращает работу.

2. Создать таймер для страниц товаров. При нахождении пользователя на странице дольше 5 секунд выводить alert-сообщения.

3. Создайте функцию для проверки вводимых пользователем данных в форму регистрации. При нажатии на кнопку Отправить производиться проверка полей следующим образом:

a. ФИО содержит только буквы.

b. Пароль может содержать любые символы (английский алфавит, цифры), длина должна быть от 5 до 100.