JavaScript

4.0 Введение

ЧТО ЭТО ТАКОЕ

JavaScript – это интерпретируемый язык программирования. Код JavaScript представляет собой последовательность строк, которые могут заканчиваться точкой с запятой. Такой код называется скриптом. Документ JavaScript (JavaScript document) может содержать любое количество встроенных скриптов.
Интерпретаторы JavaScript учитывают регистр символов. В отличие от языка Java код JavaScript не требует компиляции, а запускается во время выполнения.
Последовательность строк с кодом называется скриптом. Блоки кода выделяются фигурными скобками {}.

JavaScript и Java
Несмотря на схожесть названий, языки JavaScript и Java значительно отличаются друг от друга. Основное отличие заключается в том, что JavaScript нельзя использовать для системного программирования (разработки ПО). Java – строго типизированный язык, а в JavaScript типизация переменных осуществляется динамически (подробнее об этом читайте ниже). По сути, в JavaScript применяются динамические объекты: элементы данных и методы объекта могут изменяться во время выполнения.

ПРИМЕНЕНИЕ

Код JavaScript можно подключать через внешний файл или встраивать в HTML с помощью элемента <style>.
Код можно встраивать в HTML, но это не очень удобно, поскольку его нужно писать полностью для каждой кнопки. Рекомендуется определять все функции JavaScript между тегами

заголовка (`<head>` и `</head>`). Также скрипт можно держать в отдельном файле.

ПОПРОБУЙТЕ

Ниже приведен пример встроенного скрипта, который создает элемент Button (кнопку) и при нажатии показывает сообщение.

```
<input type ="button" id="button1" value="Нажми меня!"
onclick="alert('Вы нажали кнопку!')">
```

Событие onclick происходит при нажатии мышью на этом элементе.

> ВНИМАНИЕ! Не используйте атрибуты событий. В таком случае HTML трудно отличить от JavaScript, что усложняет чтение и отладку кода и увеличивает его объем.

Ниже показано, как правильно оформлять только что приведенный код.

```
<head>
  <script>
              function doSomething() {
                            alert('Вы нажали кнопку!');
              }
      var button1 = document.getElementById('button1');
      button1.onclicked = doSomething;
  </script>
</head>

<input type ="button" id="button1" value="Нажми!" />
```

Этот фрагмент JavaScript аналогичен приведенному выше, но здесь мы использовали модульный подход – предпочтительный способ написания кода.

4.1 Доступ к элементам в JavaScript

ЧТО ЭТО ТАКОЕ

В JavaScript есть несколько способов обращения к объекту, связанному с элементом формы XHTML. В объектной модели документа (Document Object Model, DOM) используются массивы `Forms` и `Element` объекта `Document` на который ссылается объект `Window`.

Напомним, что объект `Window` – это окно, в котором отображается контент XHTML. У любого объекта `Window` есть свойство `Document`. У любого объекта `Document` есть массив `Forms`, каждый элемент которого соответствует одной из форм в документе. В

массиве `Forms` в качестве свойства используется массив `Element`, который содержит объекты, соответствующие элементам формы XHTML, например кнопки, меню и т. д.

ПРИМЕНЕНИЕ

Элементами документа являются объекты с данными и операциями. Данные называются свойствами, а операции – методами.

Рассмотрим этот пример:

```
<input type="text" name="age">
```

Здесь type и name – свойства объекта, а text и age – его значения.

Адрес в модели DOM – это обращение к объекту JavaScript.. Доступ к элементам нижних уровней в дереве DOM осуществляется через точку. Указать нужный элемент можно при помощи его имени или уникального идентификатора. Для этого используются методы getElementByName() и getElementById()соответственно.

Рассмотрим этот пример:

```
var dom = document.getElementById("button1");
```

Этот код осуществляет доступ к элементу с уникальным идентификатором *button1*, который является частью документа.

ПОПРОБУЙТЕ

<u>Задача: подсчитать количество установленных флажков в форме.</u>

Скрипт создает неявный массив из трех элементов – флажков разного цвета. Всем элементам присвоены одинаковые имена (идентификаторы), что упрощает последующий доступ к ним. Доступ к элементам осуществляется по их индексам. Индексация начинается с нуля. Последний элемент имеет индекс на 1 меньше, чем длина массива.

Поскольку мы используем флажки (checkbox), у каждого элемента будет дополнительное свойство `checked` *("установлен").* Это свойство принимает значение true, если флажок установлен, и значение false – в противном случае.

```
<form id="FavoriteColor">
  <input type="checkbox" name="color"
     value="red" /> Красный
  <input type="checkbox" name="color"
     value="blue" /> Синий
  <input type="checkbox" name="color"
     value="green" /> Зеленый
</form>


var numChecked = 0;
var dom = document.getElementById("FavoriteColor");
for (i = 0; i < dom.color.length; i++)
```

```
if (dom.car[i].checked)
      numChecked++;
```

4.2 Переменные, операторы и комментарии

4.2.1 Переменные

ЧТО ЭТО ТАКОЕ

Попросту говоря, переменные используются для хранения значений. Прежде чем использовать переменную, ее нужно объявить.

JavaScript – язык с динамической типизацией. Это значит, что переменные в нем могут принимать значения разных типов и использоваться для ссылок на объекты.

Тип переменной определяется типом присвоенного ей значения.

Типы переменных в JavaScript делятся на две группы:

простые (примитивные) типы: number, string, boolean, undefined и null;

ссылочные типы: object, array и function.

Важные замечания по переменным:

- В отличие от HTML и CSS, язык JavaScript чувствителен к регистру.
- Служебные слова (например, var, document и другие) нельзя использовать в качестве имен переменных.
- Если значение переменной не присвоено, то по умолчанию для нее используется значение undefined.
- Строковые переменные (тип string) должны объявляться в кавычках.

Область видимости переменной

- Локальные переменные

  Переменная, объявленная в функции, доступна только в ней и действует только во время работы этой функции
- Глобальные переменные

  Переменная, объявленная на верхнем уровне (вне функции) файла со скриптом, доступна в любом месте программы JavaScript.
- Необъявленные переменные по умолчанию считаются глобальными.

КАК

Значение можно присваивать без объявления переменной (в таком случае интерпретатор неявно объявляет переменную) или в объявлении переменной, которое начинается словом `var`.

ПОПРОБУЙТЕ

```
var counter;
      pi = 3.14;
      flag = true;                    //Логическая переменная.
```

shape = "circle"; //Строковая переменная.

4.2.2 Операторы

ЧТО ЭТО ТАКОЕ

Различные действия, совершаемые со значениями, например сложение, вычитание и т. д., называются операторами. Некоторые операторы (например, +) можно также применять к строковым переменным.

КАК

В JavaScript используются различные виды операторов.

- Арифметические.
  - Они служат для выполнения основных арифметических действий.

+	Сложение
–	Вычитание
*	Умножение
/	Деление
%	Функция вычисления остатка
++, --	Инкремент, декремент переменной (унарный оператор)

Оператор + также можно использовать для конкатенации строковых величин.

```
var a = "Hello" + "World"
```

В этом примере переменной *a* присваивается строковое значение *Hello World*.

- Знак равенства
  - Используется для сравнения двух значений или типов.

=	Оператор присвоения `var a = 2;` Присваивает переменной a значение 2.
==	Оператор равенства `var a = "2" == 2;` //верно. Используется для сравнения типов или значений двух переменных.

- Сравнение

○ Используется для сравнения двух значений (в том числе – объектов String).

< , >	Меньше, больше.
<= , >=	Меньше или равно, больше или равно.

4.2.3 Комментарии

ЧТО ЭТО ТАКОЕ

Комментарии – это обычный текст в коде, который не является частью программы. Они пишутся в свободной форме и служат для того, чтобы понимать код было легче.

ПРИМЕНЕНИЕ

Однострочные комментарии добавляются после двойной косой черты //.
Многострочные комментарии отделяются от кода символами /* и */.

ПОПРОБУЙТЕ

```
var a = 2;                    //Этот комментарий нельзя переносить на другую строку.
var b = "Hello";          /*Это многострочный комментарий.
Он занимает несколько строк.*/
```

4.3 Функции, объекты и события

4.3.1 Функции

ЧТО ЭТО ТАКОЕ

Функция – это отдельный модуль кода, который выполняет определенное действие. Функции обеспечивают многократное применение кода, когда нужно несколько раз выполнить различные действия, например получить данные, запустить определенную процедуру и т. п.

ПРИМЕНЕНИЕ

Определение функции состоит из заголовка и тела – набора команд.
Заголовок функции начинается со служебного слова function. Функции также присваивается уникальное имя, которое описывает ее назначение. Тело состоит из команд, которые выполняют действия, определенные функцией.
Функции в JavaScript являются объектами, поэтому ссылающиеся на них переменные можно рассматривать как обычные ссылки на объекты.
Функции могут иметь параметры – переменные в заголовке функции, которым присвоены значения. При вызове функции ей могут передаваться аргументы – значения, используемые в качестве исходных данных, которые затем используются в функции.

Следует отметить, что в функцию передается только значение, а не ссылка на объект. Такой метод называется передачей параметров по значению.

```
function functionName(var1, var2) {
              /* Здесь указывается код самой функции.
              …
              */
              Return returnVal; // Возвращение значения
(необязательно).
}
```

ПОПРОБУЙТЕ

Задача: создать функцию, которая объединяет две строковых переменных.

```
function returnConcatString(var1 var2) {
              var result = "";                          //Добавляем
пустую строку на выходе.
              for (var i = 0; i < arguments.length; i++) {
                            result = result + " " + arguments[i];
              }
      return result;
}

var b = returnConcatString("Hello", "World");
```

Переменная *b* будет содержать строку *Hello World.*

4.3.2 Объекты

ЧТО ЭТО ТАКОЕ

Объект – это набор свойств. Свойство, значением которого является функция, называется методом.

Объект может состоять из двух элементов – свойства и метода.

Все объекты одного типа имеют одинаковые свойства, но их значения могут быть разными.

ПРИМЕНЕНИЕ

Чтобы создать объект, нужно определить его свойства и методы. Доступ к свойствам и методам осуществляется с помощью оператора-точки (.).

Доступ к свойствам объекта осуществляется так:

```
       objectName.propertyName;
              objectName["propertyName"];
```

Доступ к методам объекта осуществляется так:

```
objectName.methodName();
```

В этом примере в конструкторе объявляется пустой объект:

```
var car = new Object();
```

После создания объекта можно инициализировать его свойства.

```
car.type = "седан";
car.color = "красный";
```

ПОПРОБУЙТЕ

Задача: создать объект типа car (автомобиль) со свойствами, описывающими тип, цвет и модель автомобиля.

```
var car = {type: 'седан',
                          color: 'баклажан',
                          model: 'Лада'};
```

4.3.3 События

ЧТО ЭТО ТАКОЕ

Событие – это действие пользователя или оповещение о чем-либо. Код JavaScript взаимодействует со страницами HTML через события. Например, нажатие кнопки, движение мыши и т. д. – это всё события.

ПРИМЕНЕНИЕ

Обработчик события – это скрипт, который неявно вызывается сразу после события. У каждого элемента есть связанный с ним атрибут события. У разных элементов могут быть одинаковые атрибуты событий.

Это означает, что у события onclick (событие клика) одинаковые действия, будь то клик по ссылке или клик по текстовому полю.

Ниже перечислены некоторые события и их атрибуты.

СОБЫТИЕ	АТРИБУТ ТЕГА
blur (потеря фокуса)	onblur
click (клик)	onclick
dblclick (двойной клик)	ondblclick
focus (получение	onfocus

фокуса)	
keydown (нажатие клавиши)	onkeydown
keypress (удержание клавиши)	onkeypress
keyup (отпускание клавиши)	onkeypress
load (загрузка)	onload
mousedown (нажатие кнопки мыши)	onmousedown
mousemove (движение мыши)	onmousemove
mouseover (наведение мыши)	onmouseover
reset (сброс)	onreset
select (выбор)	onselect
submit (отправка)	onsubmit
unload (уход со страницы)	onunload

ПОПРОБУЙТЕ

Задача: показать оповещение после загрузки тела документа.

```
<html>
<head>
              <title> load.html </title>
              <script type = "text/javascript"  src = "load.js">
              </script>
</head>
<body onload = load_greeting();">
              <p> Страница загрузилась! </p>
</body>
</html>

//load.js
```

```
function load_greeting() {
              alert("Это всплывающее сообщение с приветствием. Добро
пожаловать на нашу страницу!");
}
```

4.4 Строки, числа, математические функции и функции для работы с датами

4.4.1 Строки (объекты String)

ЧТО ЭТО ТАКОЕ

Строковые величины в JavaScript хранятся не как массивы символов, а как скалярные значения.

ПРИМЕНЕНИЕ

Со строками можно проделывать различные операции с помощью функций.
Методы строковых переменных могут возвращать значения элементарных типов, подобно тому, как это выполняется для объектов.
Например, количество символов в строке хранится в ее свойстве length.

В таблице ниже перечислены некоторые распространенные методы строковых объектов.

Метод	Параметр	Результат
length	Нет	Возвращает количество символов в строковом объекте.
charAt	Число	Возвращает символ, который находится в строке на указанной позиции.
indexOf	Односимвольная строка	Возвращает позицию параметра в строковом объекте.
substring	Два числа	Возвращает подстроку строкового объекта между позициями, указанными первым и вторым параметром.
toLowerCase	Нет	Преобразует все буквы в строке в нижний регистр.
toUpperCase	Нет	Преобразует все буквы в строке в верхний регистр.

Помимо этих методов также используются т. н. экранированные символы.

\n	Новая строка
\t	Табуляция (8 пробелов)
\r	Возврат каретки

ПОПРОБУЙТЕ

```
var str = "Hello World";

str.charAt(2);                          // Возвращает l .
str.indexOf('o');              // Возвращает 4
str.substring(2,4);              // Возвращает llo
str.toLowerCase();              // Возвращает hello world
```

4.4.2 Числа (объекты Number)

ЧТО ЭТО ТАКОЕ

Объекты Number в JavaScript имеют ряд свойств, содержащих постоянные значения.

КАК

Некоторые из этих свойств перечислены в таблице ниже.

Свойство	Значение
MAX_VALUE	Наибольшее представимое число.
MIN_VALUE	Наименьшее представимое число.
NaN	Не числовое.
POSITIVE_INFINITY	Любое значение, превышающее наибольшее представимое число (плюс-бесконечность).
NEGATIVE_INFINITY	Любое значение, которое меньше наименьшего представимого числа (минус-бесконечность).
PI	Значение числа π (3,14).

4.4.3 Функции объекта Math (математические функции)

ЧТО ЭТО ТАКОЕ

В объекте Math есть методы, позволяющие вычислять тригонометрические (синус, косинус и т. п.) и другие распространенные математические функции.

ПРИМЕНЕНИЕ

В объекте Math есть тригонометрические функции, например sin (sine, синус) и cos (cosine,

косинус), а также другие математические функции, такие как `floor` и `round` (округляют число в меньшую и большую сторону соответственно), `max` (возвращает большее из двух чисел) и т. д.
Все математические функции вызываются в объекте Math.

ПОПРОБУЙТЕ

```
Math.floor(4.2343);                    //Возвращает 4.
```

4.4.4 Функции для работы с датами (объект Date)

ЧТО ЭТО ТАКОЕ

Функции объекта Date представляют дату и время.

ПРИМЕНЕНИЕ

Даты в JavaScript указываются с помощью объекта Date.
Его можно создать с помощью такой команды:

```
var today = new Date();
```

Ниже перечислены некоторые распространенные методы объекта Date.

Метод	Возвращаемое значение
toLocaleString	Локализованные дата или время
getDate	День месяца
getMonth	Месяц (от 0 до 11)
getDay	День недели (от 0 до 6)
getFullYear	Год
getTime	Количество миллисекунд с 1 января 1970 г.
getHours	Час (от 0 до 23)
getMinutes	Минута (от 0 до 59)
getSeconds	Секунда (от 0 до 59)

4.5 Массивы

ЧТО ЭТО ТАКОЕ

Массив – это набор значений. Доступ к любому элементу массива осуществляется по индексу элемента. Индексация начинается с нуля. Последний элемент имеет индекс на 1 меньше, чем длина массива.
Отметим, что в JavaScript, в отличие от других языков программирования, допускается

использование в массиве элементов различных типов.

ПРИМЕНЕНИЕ

Объекты Array, в отличие от большинства других объектов JavaScript, можно создавать двумя способами.

В первом способе используется тот же оператор new, что и при создании других объектов.

```
var list = new Array (1, 2, "три", "четы ре");
```

Во втором способе используется литерал массива.

```
var list_2 = [1, 2, "три", "четыре"];
```

Индексация элементов в любом массиве всегда начинается с нуля.

Доступ к элементам массива осуществляется по индексу, указанному в имени массива.

Длина массива равна количеству элементов в нем. Поэтому последний индекс массива всегда на единицу меньше его длины.

Для работы с массивами используются различные методы. Некоторые из них перечислены ниже.

Метод	Назначение
join	Преобразует все элементы объекта Array в объекты String и объединяет их в строку.
reverse	Упорядочивает элементы объекта Array в обратном порядке.
sort	Преобразует все элементы объекта Array в объекты String и упорядочивает их в алфавитном порядке.
concat	Объединяет объект Array с другими массивами.
slice	Возвращает часть объекта Array, заключенную между параметрами.
pop	Удаляет из объекта Array последний элемент и возвращает его.
push	Добавляет элемент в конец массива Array.

ПОПРОБУЙТЕ

Задача: создать массив и показать его значения, затем добавить в него новые элементы и показать их значения.

```
//array_example.js
var list = [2, 4, 6];

for (var i = 0; i < list.length; i++) {
              document.write(list[i] + "  ");
```

```
}
document.write("<br />");

var new_list = list.concat(8, 10);

for (var j = 0; j < new_list.length; j++) {
              document.write(new_list[j] + "  ");
}
```

Результат:
2 4 6
2 4 6 8 10

4.6 Логические значения (boolean), условные операторы и циклы

4.6.1 Объекты Boolean

ЧТО ЭТО ТАКОЕ

Объекты Boolean – это специальный тип данных в JavaScript. Они могут иметь два значения:

True и False.

ПРИМЕНЕНИЕ

Объекты Boolean объявляются так же, как и другие объекты в JavaScript.

```
var bool_t = new Boolean();
       bool_t = true;
```

4.6.2 Условия

ЧТО ЭТО ТАКОЕ

Условные операторы играют важную роль, поскольку они определяют последовательность выполнения команд. Они принимают определенные параметры и возвращают логические значения. Если логическое значение верно, тогда выполняется блок команд, следующий непосредственно за условием.
В противном случае выполняется фрагмент кода, определяемый условными операторами.

ПРИМЕНЕНИЕ

Выражения, с помощью которых можно управлять последовательностью выполнения команд, включают элементарные значения, выражения сравнения и составные выражения. Такие выражения всегда принимают значение True или False.
В JavaScript также есть логические операторы: && (И), || (ИЛИ) и ! (НЕ).

Есть два типа условных операторов.

1. Операторы выбора.

Эти операторы содержат условия if-then и if-then-else.

У таких конструкций следующий синтаксис:

```
if (condition) {
                /*Если условие соблюдается,
   выполняются эти команды:
                …
                */  }
                else {
                /*Если условие не соблюдается,
   выполняются эти команды:
                …
                */  }
```

2. Конструкция switch

Используется, когда нужно поочередно проверить несколько условий и выполнить одно из них. У таких конструкций следующий синтаксис:

```
switch (выражение) {
                            case value_1:
 //Одна или несколько команд.
                            case value_2:
 //Одна или несколько команд.

   …
   case value_n:                          //Одна или
несколько команд.
   [default:                            //Одна или
несколько команд.]
}
```

Выражение в конструкции switch поочередно проверяется на соответствие разным значениям. Если соответствие найдено, тогда соответствующая директива case выполняется до конца конструкции переключения или до ближайшего оператора break. В большинстве случаев требуется найти одно соответствие, после чего управление передается из конструкции switch другим командам.
В этом случае можно использовать оператор завершения *break*.
Он прерывает выполнение конструкции *switch*.

ПОПРОБУЙТЕ

Задача: сравнить два числа с помощью простой конструкции if-else и вывести большее.

```
//if_then.js
```

```
var a = 5;
var b = 8;

if (a > b)
   document.write("A > B <br/> ");
else
   document.write("A < B <br/> ");
```

4.6.3 Циклы

ЧТО ЭТО ТАКОЕ

Циклы в основном используются для выполнения одной задачи с несколькими элементами в массиве или списке. По сути, в цикле фрагмент кода выполняется несколько раз.

ПРИМЕНЕНИЕ

В циклах обычно есть контрольное выражение, за которым следует фрагмент кода в скобках. Есть два вида циклов.

1. Циклы while

Циклы while выполняют фрагмент кода до тех пор, пока контрольное выражение истинно. У таких конструкций следующий синтаксис:

```
while (условие)
        {
                        /*Набор команд.
                        …
                        */
        }
```

2. Циклы for

В циклах for используется начальное значение, условное значение и шаг приращения. Если логическое значение верно, тогда выполняется блок команд, следующий непосредственно за условием. Как только значение условия перестает быть истинным, цикл завершается. У таких конструкций следующий синтаксис:

```
for (начальное выражение; контрольное вы ражение; приращение)
{
                        /*Набор команд.
                        …
                        */
        }
```

Циклы обоих видов можно принудительно завершить командой break.

ПОПРОБУЙТЕ

Задача: написать код JavaScript, который определит произведение всех чисел от 1 до 50.

```
//loops_while.js

var i = 1;
var sum = 0;

while (i <= 50) {
sum = sum * i;
}

document.write("Произведение равно " + sum);

//loops_for.js
var sum = 0;
for (var i = 0; i < 51; i++)
sum = sum * i;

document.write("Произведение равно" + sum);
```

4.7 Регулярные выражения (объекты RegExp)

ЧТО ЭТО ТАКОЕ

JavaScript предоставляет мощные средства поиска строк по маске. Эту задачу можно решать также с помощью функций объекта String, однако возможности объекта RegExp гораздо шире.

ПРИМЕНЕНИЕ

Все символы в регулярных выражениях делятся на две группы – обычные и специальные (метасимволы). К числу специальных или метасимволов относятся следующие:

```
\ | ( ) [ ] { } ^ $ * + ? .
```

Если нужно найти метасимвол, в регулярном выражении перед ним добавляется обратная косая черта (\).

Точка (.) обозначает любой символ, кроме новой строки. Чтобы обозначить точку, в регулярном выражении перед ней нужно поставить обратную косую черту.
Пример:
По выражению /кот./ могут быть найдены строки "коты", "коту", "кота" и т. д.

Циркумфлекс (^) перед первым символом регулярного выражения означает отрицание или инверсию ("все символы, кроме указанных").
Пример:
Выражение [^abc] [^abc] задает соответствие любым символам, кроме латинских букв a, b

и с

Диапазоны обозначаются дефисом (-).

В таблице ниже перечислены некоторые классы символов.

Имя	Эквивалент	Совпадения
`\d`	`[0-9]`	Цифровой символ
`\D`	`[^0-9]`	Нецифровой символ
`\w`	`[A-Za-z_0-9]`	Буквенный или цифровой символ
`\W`	`[^A-Za-z_0-9]`	Любой символ, кроме буквенного или цифрового
`\s`	`[ \r\t\n\f]`	Пробельный символ
`\S`	`[^ \r\t\n\f]`	Непробельный символ

ПОПРОБУЙТЕ

[\d\.\d\d] //Цифра, точка и еще две цифры.
[\D\d\D] //Одна цифра.
[\d+\.\d*] //Одна или несколько цифр, точка, а затем 0 или более цифр.
[\w\w\w] //Три подряд буквенно-цифровых символа.