**Лабораторная работа: JS**

**Введение**

JavaScript — язык сценариев, или скриптов. Скрипт представляет собой программный код — набор инструкций, который не требует предварительной обработки (например, компиляции) перед запуском.

JavaScript – язык интерпретируемый, но также оснащен JIT-компиляцией.

**JIT** или компиляция «на лету» (Just-in-time compilation) — технология увеличения скорости работы программных систем, использующих байт-код, путём компиляции байт-кода в машинный код непосредственно во время работы программы.

Код JavaScript интерпретируется движком браузера во время загрузки веб-страницы. Интерпретатор браузера выполняет построчный анализ, обработку и выполнение исходной программы или запроса.

Более 90% сайтов и web-проектов в той или иной степени используют JavaScript. От выпадающих элементов меню и до управления базой данных; от Ajax-запросов (вызов серверных скриптов без перезагрузки страницы) и до Node.js (серверный JavaScript).

В 1996 году язык JavaScript был стандартизован компанией Ecma, которая занимается стандартизацией информационных и коммуникационных технологий. Сама спецификация была названа ECMAScript или сокращённо ES. По сути, JavaScript является реализацией спецификации ECMAScript.

В настоящее время язык **JavaScript используется не только для создания web-приложений, с** помощью него можно написать обычные приложения для десктопных и мобильных операционных систем, использовать его в роли серверного языка (node.js) и др.

  С появлением HTML5 расширились и возможности JavaScript, появилась API video и audio. С появлением API холста (canvas) отпала необходимость для двухмерной графики или анимации использовать флэш. Появились API перетаскивания, геолокации, индексированных баз данных, работы с файлами, истории, автономной работы, управления рабочими процессами и коммуникационный API.

JavaScript обзавелся своей базой данных – MongoDB, данные хранятся в JSON-формате.

**JavaScript**, «самостоятельный» язык программирования, ведь прав на него нет ни у одной компании или группы разработчиков.

Любой современный браузер основывается на **движке**. **Движок** – это часть браузера, которая преобразует HTML, CSS, JavaScript, изображения и другую информацию в **интерактивную картинку**.

Основные современные движки и браузеры, которые их используют:

* [**Trident**](https://ru.wikipedia.org/wiki/Microsoft_Trident) — движок Microsoft Internet Explorer;
* [**Gecko**](https://ru.wikipedia.org/wiki/Gecko)— открытый движок проекта **Mozilla**. Используется в браузере Firefox, почтовом клиенте Thunderbird и других.
* [**EdgeHTML**](https://ru.wikipedia.org/wiki/EdgeHTML)— движок от компании Microsoft для браузера Microsoft Edge.
* [**WebKit**](https://ru.wikipedia.org/wiki/WebKit)— движок для браузера Apple Safari.
* [**Blink**](https://ru.wikipedia.org/wiki/Blink_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)) — движок браузера Google Chrome**.** Является ответвлением WebKit.

**Подключение JS к Html-странице**

2 способа подключения JavaScript к странице HTML:

* <script>//JavaScript код</script>
* <script src="/my/script.js"></script>

1. Для того, чтобы добавить сценарий на страницу нужно дописать следующий код между тегами script

Пример:

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

<script>

let x=5;

let y=x+3;

alert("Привет, head)!");

document.write(y);

</script>

</head>

<body>

<h1>Заголовок h1</h1>

...

<script>

document.write("Привет, body)!");

</script>

</body>

</html>

Второй метод:

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

<script type="text/javascript" src="js/script.js"></script>

<script>

var x=5;

var y=x+3;

alert("Привет, head)!");

document.write(y);

</script>

</head>

<body>

<h1>Заголовок h1</h1>

...

<script>

document.write("Привет, body)!");

</script>

</body>

</html>

Script.js

alert("Привет, внешний файл)!");

document.write("Привет1, внешний файл)!");

Когда браузер при чтении страницы встречает на ней элемент script, браузер **останавливает дальнейшую загрузку страницы** и выполняет, подключенный с помощью этого элемента JavaScript код. После его выполнения, он приступает к дальнейшей загрузке страницы.

**Ввод – вывод данных в JS**

В JavaScript используются три стандартных метода для ввода и вывода данных: alert(), prompt() и confirm().

**alert**

Данный метод позволяет выводить диалоговое окно с заданным сообщением и кнопкой ОК. Синтаксис соответствующего выражения имеет следующий вид:

alert ("сообщение”)

Если ваше сообщение представляет собой определенный набор символов, то его необходимо заключить в двойные или одинарные кавычки. Например, **alert("Hello, World")**

Cообщение представляет собой данные любого типа: последовательность символов, заключенную в кавычки, число (в кавычках или без них), переменную или выражение.

Диалоговое окно, выведенное на экран методом alert(), можно убрать, щелкнув на кнопке ОК. До тех пор пока вы не сделаете этого, переход к ранее открытым окнам невозможен. Окна, обладающие свойством останавливать все последующие действия пользователя и программ, называются ***модальными***. Таким образом, окно, создаваемое посредством alert(), является модальным.

**confirm**

Метод confirm позволяет вывести диалоговое окно с сообщением и двумя кнопками — ОК и Отмена (Cancel). Этот метод возвращает логическую величину, значение которой зависит от того, на какой из двух кнопок щелкнул пользователь. Если он щелкнул на кнопке ОК, то возвращается значение true; если же он щелкнул на кнопке Отмена, то возвращается значение false. Возвращаемое значение можно обработать в программе. Синтаксис применения метода confirm имеет следующий вид:

confirm(сообщение)

Если ваше сообщение представляет собой определенный набор символов, то его необходимо заключить в кавычки, двойные или одинарные. Например, confirm("Bы действительно хотите завершить работу?")

Cообщение представляет собой данные любого типа: последовательность символов, заключенную в кавычки, число (в кавычках или без них), переменную или выражение.

Диалоговое окно, выведенное на экран методом confirm(), можно убрать щелчком на любой из двух кнопок — ОК или Отмена. До тех пор пока вы не сделаете этого, переход к ранее открытым окнам невозможен. Окно, создаваемое посредством confirm(), является модальным.

**prompt**

Метод prompt позволяет вывести на экран диалоговое окно с сообщением, а также с текстовым полем, в которое пользователь может ввести данные

Кроме того, в этом окне предусмотрены две кнопки: ОК и Отмена **(Cancel). В**отличие от методов **alert()**и **confirm()**данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию. Если пользователь щелкнет на кнопке ОК, то метод вернет содержимое поля ввода данных, а если он щелкнет на кнопке Отмена, то возвращается логическое значение false. Возвращаемое значение можно затем обработать в программе. Синтаксис применения метода prompt имеет следующий вид:

prompt(сообщение, значение\_поля\_ввода\_данных);

Параметры метода prompt() не являются обязательными. Если вы их не укажете, то будет выведено окно без сообщения, а в поле ввода данных подставлено значение по умолчанию — undefined (не определено). Если вы не хотите, чтобы в поле ввода данных появлялось значение по умолчанию, то подставьте в качестве значения второго параметра пустую строку "". Например,

prompt("Введите Ваше имя, пожалуйста", "");

Пример

alert("Начало работы скрипта");

qu = confirm("Закрыть окно?");

document.writeln(qu+';');

uname = prompt("Введите имя", "Anonymous");

document.writeln("Имя " + uname)

## [Переменная](https://learn.javascript.ru/variables" \l "peremennaya)

Для создания переменной в JavaScript используйте ключевое слово let.

В JavaScript есть два ограничения, касающиеся имён переменных:

1. Имя переменной должно содержать только буквы, цифры или символы $ и \_.
2. Первый символ не должен быть цифрой.

Переменная в JavaScript может содержать любые данные

В JavaScript есть 8 основных типов.

* number для любых чисел: целочисленных или чисел с плавающей точкой; целочисленные значения ограничены диапазоном ±(253-1).
* bigint для целых чисел произвольной длины.
* string для строк. Строка может содержать ноль или больше символов, нет отдельного символьного типа.
* boolean для true/false.
* null для неизвестных значений – отдельный тип, имеющий одно значение null.
* undefined для неприсвоенных значений – отдельный тип, имеющий одно значение undefined.
* object для более сложных структур данных.
* symbol для уникальных идентификаторов.

let message = "hello";

message = 123456;

message = 12.345;

Cтрока (string) в JavaScript должна быть заключена в кавычки.

В JavaScript существует три типа кавычек.

1. Двойные кавычки: "Привет".
2. Одинарные кавычки: 'Привет'.
3. Обратные кавычки: `Привет`.

Обратные кавычки позволяют встраивать выражения в строку, заключая их в ${…}. Например:

// Вставим выражение

alert( `результат: ${1 + 2}` ); // результат: 3

Специальное значение null :

let age = null;

Некоторые встроенные функции

Функция представляет собой подпрограмму, которую можно вызвать для выполнения, обратившись к ней по имени. В JavaScript есть встроенные функции, которые можно использовать в программах.

**parseInt(строка)** – преобразует указанную строку в целое число:

a = parseInt("2.5")

b = parseInt("10 рублей")

c = parseInt("рубли 10") d = parseInt("-15.87")

document.writeln(a)

document.writeln(b)

document.writeln(c)

document.writeln(d)

**parseFloat(строка)** – преобразует указанную строку в число с плавающей разделительной (десятичной) точкой:

a = parseFloat("2.5")

b = parseFloat("10 рублей")

d = parseFloat("-15.87")

document.writeln(a)

document.writeln(b)

document.writeln(d)

**isNaN (значение)** – возвращает true, если указанное в параметре значение не является числом, иначе – false:

a = isNaN("2.5")

b = isNaN("10 рублей")

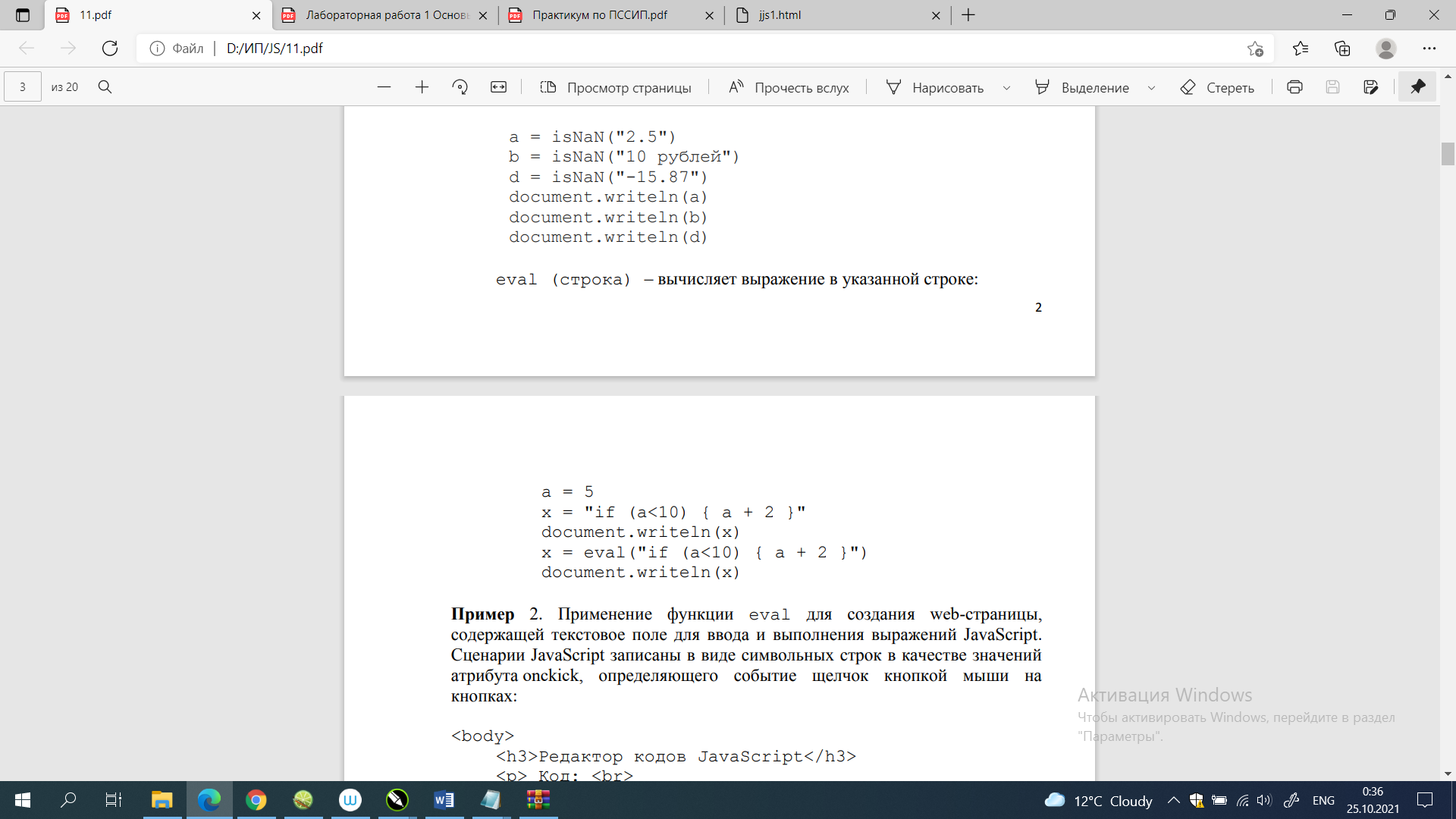
d = isNaN("-15.87")

document.writeln(a)

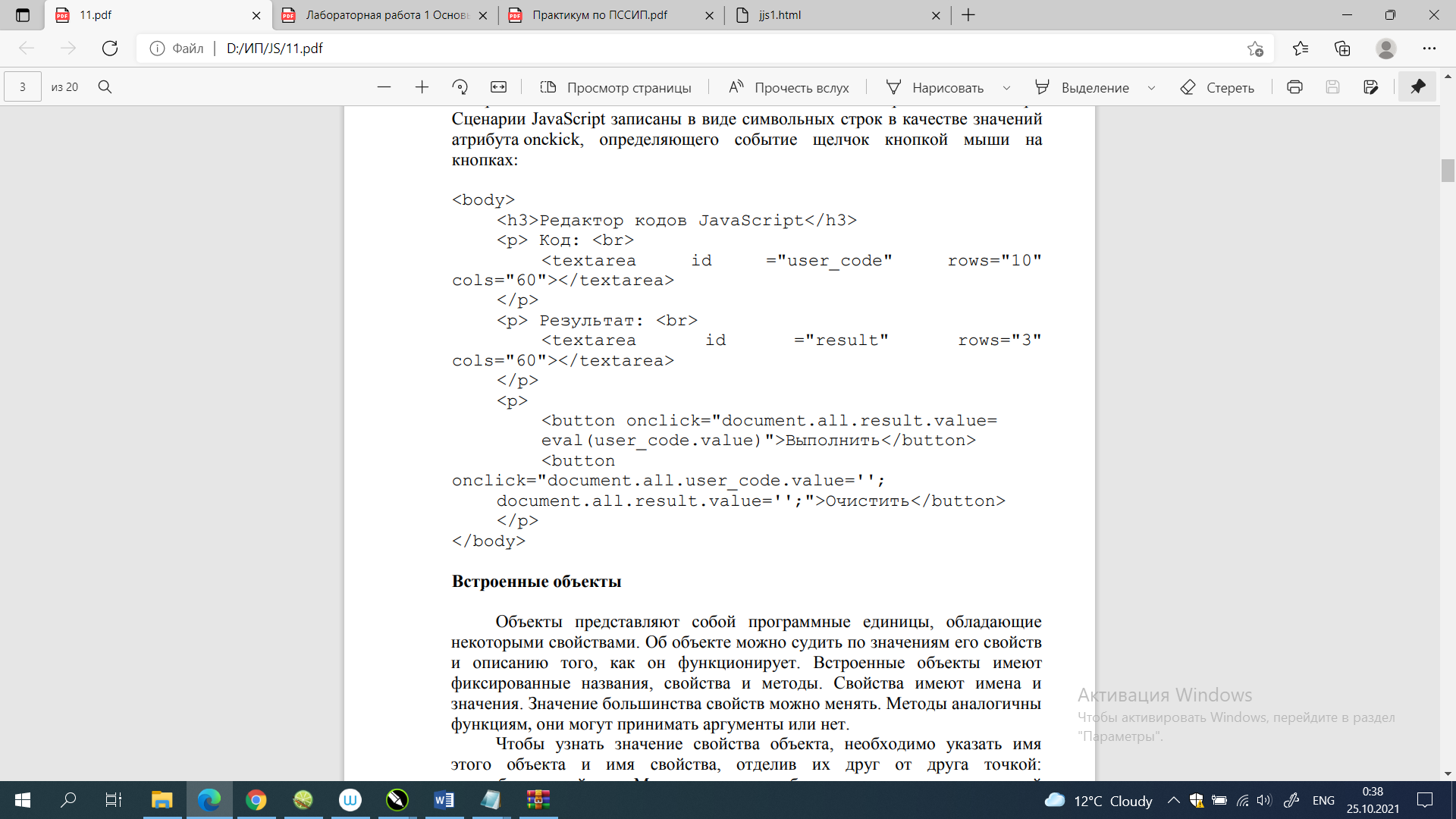
document.writeln(b)

document.writeln(d)

**eval (строка)** – вычисляет выражение в указанной строке:



Применение функции eval для создания web-страницы, содержащей текстовое поле для ввода и выполнения выражений JavaScript. Сценарии JavaScript записаны в виде символьных строк в качестве значений атрибута onckick, определяющего событие щелчок кнопкой мыши на кнопках:



**Встроенные объекты Объекты** представляют собой программные единицы, обладающие некоторыми свойствами. Об объекте можно судить по значениям его свойств и описанию того, как он функционирует. Встроенные объекты имеют фиксированные названия, свойства и методы. Свойства имеют имена и значения. Значение большинства свойств можно менять. Методы аналогичны функциям, они могут принимать аргументы или нет. Чтобы узнать значение свойства объекта, необходимо указать имя этого объекта и имя свойства, отделив их друг от друга точкой: имя\_объекта.свойство. Можно заставить объект выполнить тот или иной присущий ему метод. В этом случае также говорят о применении метода к объекту. Синтаксис соответствующего выражения

имя\_объекта.метод(параметры)

**Объект String (строка)** Для создания строкового объекта используют выражение следующего вида:

имя\_переменной = new String ("строковое\_значение")

Можно создавать строковый объект с помощью обычного оператора присваивания:

имя\_переменной = "строковое\_значение"

Пример использования свойства length (длина строки):

s1 = "Тестовая строка"

x = s1.length

document.writeln(x)

**Методы обработки строк**

**charAt (индекс)** – возвращает символ, занимающий в строке указанную позицию. Синтаксис:

строка.charAt(индекс)

Индекс – число, индекс первого символа равен 0.

s = "Привет мир"

a = s.charAt(0)

b = s.charAt(1)

c = s.charAt(9)

document.writeln(a,b,c)

**concat(строка) – конкатенация («склеиваение») строк**. Синтаксис: строка1.concat(строка2)

Этот метод действует так же, как и оператор "+" сложения для строк: к строке строка1 добавляется справа строка2.

s1 = new String("Привет ") s2 = "мир"

document.writeln(s1.concat(s2))

*substr(индекс [, длина])* – возвращает подстроку исходной строки, начальный индекс и длина которой указываются параметрами. Синтаксис: строка.substr(индекс [, длина])

Пример Анализ адреса адрес = "petrov.vi@mail.ru"

i = адрес.indexOf("@")

пользователь = адрес.substr(0,i)

домен = адрес.substr(i+1)

document.writeln("Пользователь: ", пользователь, " Домен: ",домен)

При обработке строк часто требуется вставить или заменить подстроки. Создадим функцию вставки строки в исходную строку. Данная функция должна принимать три параметра: исходную строку s1, вставляемую строку s2 и индекс позиции вставки n. Пример определения функции и ее вызова: function вставка\_строки(s1, s2, n)

{ return s1.slice(0,n) + s2 + s1.slice(n) }

x = вставка\_строки ("Привет студентам", " , ", 16)

y = вставка\_строки ("обучающимся в группе!", " 2527", 13) document.writeln(x) document.writeln(y)

**Объект массив Массив** представляет собой упорядоченный набор данных. Существует несколько способов создания массива, сначала создается новый объект массива: имя\_массива = new Array ([длина массива])

Рассмотрим создание массива товар, содержащего в качестве элементов некоторые характеристики:

товар = new Array(4)

товар [0] = "Компьютер"

товар [1] = 4

товар [2] = "Windows 7"

товар [3] = 240

document.writeln(товар [0]) document.writeln(товар [3])

**Методы Array**

concat (массив) – конкатенация массивов, объединяет два массива в третий массив. Синтаксис: имя\_массива1.concat(массив2)

a1 = new Array(1, 2, " товары ")

a2 = new Array(" категории: ","a", "b", "c", "d")

a3 = a1.concat(a2)

document.writeln(a3)

join(разделитель) – создает строку из элементов массива с указанным разделителем между ними. Синтаксис: имя\_массива.join(строка) document.writeln(a3.join(" - "))

splice(индекс, количество [, элемент1 [, элемент2 [, ...элементN]]]) – удаляет из массива несколько элементов и возвращает массив из удаленных элементов или заменяет значения элементов. Синтаксис: имя\_массива.splice (индекс, количество [, элемент1 [, элемент2 [, ...элементN]]])

массив = new Array("Красный", "Синий", "Зеленый", "Желтый", "Белый") /массив.splice(1,3, "Сиреневый", "Черный")

document.writeln("/",массив)