**20. Дайте определение понятию «Класс» в языке JavaScript. Опишите способы создания классов и экземпляров классов, методов классов в языке JavaScript.**

Класс — в объектно-ориентированном программировании, модель для создания объектов определённого типа, описывающая их структуру (набор полей и их начальное состояние) и определяющая алгоритмы (функции или методы) для работы с этими объектами.

Иными словами, класс служит средством для введения абстрактных типов данных в программный проект. Другие описатели абстрактных типов данных — мета-классы, интерфейсы, структуры, перечисления, — характеризуются какими-то своими особенностями. Суть отличия классов состоит в том, что при задании типа данных, класс определяет одновременно как интерфейс, так и реализацию для всех своих экземпляров (т.е. объектов), поэтому вызов метода-конструктора обязателен.

В JavaScript используется модель прототипного наследования: каждый объект наследует поля (свойства) и методы объекта-прототипа. Классов, используемых в Java или Swift в качестве шаблонов или схем для создания объектов, в JavaScript не существует. В прототипном наследовании есть только объекты. Прототипное наследование может имитировать классическую модель наследования от классов.

Для определения класса используется ключевое слово class:  
class User {

// тело класса

}

Класс может не иметь названия. С помощью выражения класса можно присвоить класс переменной:  
const UserClass = class {

// тело класса

}

Классы можно экспортировать в виде модулей.

Классы используются для создания экземпляров. Экземпляр — это объект, содержащий данные и логику класса.

Экземпляры создаются с помощью оператора new: instance = new Class().

Инициализация класса производится через специальный метод constructor(param1, param2, ...) .В этом методе устанавливаются начальные значения полей экземпляра и осуществляется его настройка.

В следующем примере конструктор устанавливает начальное значение поля name:  
class User {

constructor(name) {

this.name = name

}

}

Конструктор принимает один параметр — name, который используется для установки начального значения поля this.name.

this в конструкторе указывает на создаваемый экземпляр.

Аргумент, используемый для создания экземпляра класса, становится параметром его конструктора:  
class User {

constructor(name) {

name // Печорин

this.name = name

}

}

const user = new User('Печорин')

Если не определить собственный конструктор, будет создан стандартный конструктор, представляющий собой пустую функцию, не влияющую на экземпляр.

Поля класса — это переменные, содержащие определенную информацию. Поля могут быть разделены на две группы: поля экземпляров класса и поля самого класса (статические).

Поля также имеют два уровня доступа:

1. открытые (публичные): поля доступны как внутри класса, так и в экзмеплярах;
2. частные (приватные): поля доступны только внутри класса.

Открытые поля экземпляров класса:

class User {

constructor(name) {

this.name = name

}

}

Выражение this.name = name создает поле экземпляра name и присваивает ему начальное значение. Доступ к этому полю можно получить с помощью аксессора свойства:  
const user = new User('Печорин')

user.name // Печорин

Частные поля экземпляров класса:

Инкапсуляция позволяет скрывать внутренние детали реализации класса.   
Такие классы проще обновлять при изменении деталей реализации. Хорошим способом скрыть детали является использование частных полей. Такие поля могут быть прочитаны и изменены только внутри класса, которому они принадлежат. За пределами класса частные поля недоступны. Для того, чтобы сделать поле частным, перед его названием следует поставить символ #, например, #myPrivateField. При обращении к такому полю всегда должен использоваться указанный префикс.  
Сделаем поле name частным:

class User {

#name

constructor(name) {

this.#name = name

}

getName() {

return this.#name

}

}

const user = new User('Печорин')

user.getName() // Печорин

user.#name // SyntaxError

#name — частное поле. Доступ к нему можно получить только внутри класса User. Это позволяет сделать метод getName().  
Однако, при попытке получить доступ к #name за пределами класса User будет выброшена синтаксическая ошибка

Открытые статические поля

В классе можно определить поля, принадлежащие самому классу: статические поля. Такие поля используются для создания констант, хранящих нужную классу информацию. Для создания статических полей используется ключевое слово static перед названием поля: static myStaticField.

Частные статические поля

Иногда статические поля также являются частью внутренней реализации класса. Для инкапсуляции таких полей можно сделать их частными. Для этого следует перед названием поля поставить префикс #: static #myPrivateStaticFiled.

Методы

Поля содержат данные. Возможность изменять данные обеспечивается специальными функциями, являющимися частью класса: методами.

JavaScript поддерживает как методы экземпляров класса, так и статические методы.

Методы экземпляров класса

Методы экземпляра класса могут изменять его данные. Методы экземпляра могут вызывать другие методы экземпляра, а также статические методы.

В методе класса, также как и в конструкторе, this указывает на создаваемый экземпляр. Используйте this для получения данных экземпляра: this.field, или для вызова методов: this.method().

Добавим новый метод nameContains(str), принимающий один аргумент и вызывающий другой метод:  
class User {

name

constructor(name) {

this.name = name

}

getName() {

return this.name

}

nameContains(str) {

return this.getName().includes(str)

}

}

const user = new User('Печорин')

user.nameContains('Печорин') // true

user.nameContains('Грушницкий') // false

nameContains(str) — метод класса User, принимающий один аргумент. Он вызывает другой метод экземпляра getName() для получения имени пользователя.

Метод также может быть частным. Для того, чтобы сделать метод частным, следует использовать префикс #.

Геттеры и сеттеры — это аксессоры или вычисляемые свойства. Это методы, имитирующие поля, но позволяющие читать и записывать данные.  
Геттеры используются для получения данных, сеттеры — для их изменения.

Статические методы — это функции, принадлежащие самому классу. Они определяют логику класса, а не его экземпляров. Для создания статического метода используется ключевое слово static перед названием метода: static myStaticMethod(). При работе со статическими методами, следует помнить о двух простых правилах:

1. статический метод имеет доступ к статическим полям;

2. он не имеет доступа к полям экземпляров.

Наследование: extends

Классы в JavaScript поддерживают наследование с помощью ключевого слова extends. В выражении class Child extends Parent { } класс Child наследует от класса Parent конструктор, поля и методы.

Создадим дочерний класс ContentWriter, расширяющий родительский класс User:  
 class User {

name

constructor(name) {

this.name = name

}

getName() {

return this.name

}

}

class ContentWriter extends User {

posts = []

}

const writer = new ContentWriter('Лермонтов')

writer.name // Лермонтов

writer.getName() // Лермонтов

writer.posts // []

ContentWriter наследует от User конструктор, метод getName() и поле name. В самом ContentWriter определяется новое поле posts. Частные поля и методы родительского класса не наследуются дочерними классами.

Родительский конструктор: super() в constructor()

Для того, чтобы вызвать конструктор родительского класса в дочернем классе, следует использовать специальную функцию super(), доступную в конструкторе дочернего класса.

**s**

**49. Даны два положительных числа K и N (K < N). Вывести все числа от K до N с помощью цикла while.**

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>zadanie2</title>

</head>

<body>

  <script> function dothing(){var K, N, i;

    K = Number.parseInt(document.querySelector('input[name=Name]').value);

    N = Number.parseInt(document.querySelector('input[name=Suname]').value);

    i = K + 1;

    if (K < N) {

      while (i !== N) {

        console.log(i);

        i = i + 1;

      }

    }

    }

  </script>

  <table width = "300" border="0" bordercolor = "#000000" cellpadding = "2" cellspacing = "0">

<tr>

<td align = "left">Номер 1:</td>

<td><input name="Suname" style="color:white; background-color: #99AAFF; font-style: oblique" required placeholder=""></td>

</tr>

<tr>

<td align = "left">Номер 2:</td>

<td><input name="Name" style="color:white; background-color: #99AAFF; font-style: oblique" required placeholder=""></td>

</tr>

<tr >

</tr>

</table><br/>

<input id="clickMe" type="button" value="clickme" onclick="dothing()" />

</body>

</html>

**80. Решите задачу. В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с последним элементом.**

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>zadanie2</title>

</head>

<body>

       <script>

        var arr = [1,2,3,4,5,6,7,8,9,10];

/\* вернет индекс минимального элемента из массива arr. \*/

function getMinIndex(arr)

{

var min = 0;

for(var i=1; i<arr.length; i++) if (arr[i]<arr[min]) min = i;

return min;

}

/\* вернет индекс максимального элемента из массива arr. \*/

function getMaxIndex(arr)

{

var max = 0;

for(var i=1; i<arr.length; i++) if (arr[i]>arr[max]) max = i;

return max;

}

/\* переставляет местами элементы массива arr, с индексами a и b. \*/

function exchange(arr, a, b)

{

var tmp = arr[a];

arr[a] = arr[b];

arr[b] = tmp;

}

     function dothing(){var K, N, i;

        alert("Массив до манипуляций:\n"+arr);

min = getMinIndex(arr);

max = getMaxIndex(arr);

exchange(arr,min,max);

alert("Массив после манипуляций:\n"+arr);

    }

    </script>

<input id="clickMe" type="button" value="clickme" onclick="dothing()" />

</body>

</html>

**94. Спроектировать структуру базы данных о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков, их учетные записи и пароли.**

Запросы:

а) вывести информацию о рассылках, принадлежащих определенной теме;

б) вывести информацию о рассылках, отправленных в определенный период времени.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<title>Вывод данных</title>

</head>

<body>

<fieldset>

<form method="post" action="select\_user.php">

<label for="first\_name">Тема Рассылки</label>

<br/>

<input type="text" name="first\_name" size="30"><br/>

<input id="submit" type="submit" value="Найти и вывести"><br/>

</form>

<form method="post" action="select\_date.php">

<label for="last\_name">Дата Рассылки</label>

<br/>

<input type="text" name="last\_name" size="30"><br/>

<input id="submit" type="submit" value="Найти и вывести"><br/>

</form>

</fieldset>

<fieldset>

</fieldset>

</body>

</html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<title>results</title>

</head>

<body>

<?php

$connt = mysqli\_connect("localhost", "root", "","users");

$first\_name = trim($\_REQUEST['last\_name']);

$sql\_select = "SELECT \* FROM users WHERE date='$first\_name'";

$result = mysqli\_query($connt,$sql\_select);

while ($row = mysqli\_fetch\_array($result)) {

$field1name = $row["keysp"];

$field2name = $row["date"];

$field3name = $row["topic"];

$field4name = $row["soderzhanie"];

$field5name = $row["name"];

$field6name = $row["adress"];

$field7name = $row["username"];

$field8name = $row["password"];

echo '<table border="1" cellspacing="5" cellpadding="5" width="100%"><tr>

<td>'.$field1name.'</td>

<td>'.$field2name.'</td>

<td>'.$field3name.'</td>

<td>'.$field4name.'</td>

<td>'.$field5name.'</td>

<td>'.$field6name.'</td>

<td>'.$field7name.'</td>

<td>'.$field8name.'</td>

</tr></table>';

}

?>

<a href="zadanie3.html">return to search</a><br/><br/>

</body>

</html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<title>results</title>

</head>

<body>

<?php

$connt = mysqli\_connect("localhost", "root", "","users");

$first\_name = trim($\_REQUEST['first\_name']);

$sql\_select = "SELECT \* FROM users WHERE topic='$first\_name'";

$result = mysqli\_query($connt,$sql\_select);

while ($row = mysqli\_fetch\_array($result)) {

$field1name = $row["keysp"];

$field2name = $row["date"];

$field3name = $row["topic"];

$field4name = $row["soderzhanie"];

$field5name = $row["name"];

$field6name = $row["adress"];

$field7name = $row["username"];

$field8name = $row["password"];

echo '<table border="1" cellspacing="5" cellpadding="5" width="100%"><tr>

<td>'.$field1name.'</td>

<td>'.$field2name.'</td>

<td>'.$field3name.'</td>

<td>'.$field4name.'</td>

<td>'.$field5name.'</td>

<td>'.$field6name.'</td>

<td>'.$field7name.'</td>

<td>'.$field8name.'</td>

</tr></table>';

}

?>

<a href="zadanie3.html">return to search</a><br/><br/>

</body>

</html>



