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 — метаклассы, 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.

**Синтаксис «class»**

Базовый синтаксис выглядит так:

class MyClass {

// методы класса

constructor() { ... }

method1() { ... }

method2() { ... }

method3() { ... }

...

}

Затем используйте вызов new MyClass() для создания нового объекта со всеми перечисленными методами.

При этом автоматически вызывается метод constructor(), в нём мы можем инициализировать объект.

Например:

class User {

constructor(name) {

this.name = name;

}

sayHi() {

alert(this.name);

}

}

// Использование:

let user = new User("Иван");

user.sayHi();

Когда вызывается new User("Иван"):

Создаётся новый объект.

constructor запускается с заданным аргументом и сохраняет его в this.name.

…Затем можно вызывать на объекте методы, такие как user.sayHi().

**Методы в классе не разделяются запятой**

Частая ошибка начинающих разработчиков – ставить запятую между методами класса, что приводит к синтаксической ошибке.

Синтаксис классов отличается от литералов объектов, не путайте их. Внутри классов запятые не требуются.

**Что такое класс?**

Итак, что же такое class? Это не полностью новая языковая сущность, как может показаться на первый взгляд.

Давайте развеем всю магию и посмотрим, что такое класс на самом деле. Это поможет в понимании многих сложных аспектов.

В JavaScript класс – это разновидность функции.

Взгляните:

class User {

constructor(name) { this.name = name; }

sayHi() { alert(this.name); }

}

// доказательство: User - это функция

alert(typeof User); // function

Вот что на самом деле делает конструкция class User {...}:

Создаёт функцию с именем User, которая становится результатом объявления класса. Код функции берётся из метода constructor (она будет пустой, если такого метода нет).

Сохраняет все методы, такие как sayHi, в User.prototype.

При вызове метода объекта new User он будет взят из прототипа, как описано в главе F.prototype. Таким образом, объекты new User имеют доступ к методам класса.

На картинке показан результат объявления class User:

Можно проверить вышесказанное и при помощи кода:

class User {

constructor(name) { this.name = name; }

sayHi() { alert(this.name); }

}

// класс - это функция

alert(typeof User); // function

// ...или, если точнее, это метод constructor

alert(User === User.prototype.constructor); // true

// Методы находятся в User.prototype, например:

alert(User.prototype.sayHi); // sayHi() { alert(this.name); }

// в прототипе ровно 2 метода

alert(Object.getOwnPropertyNames(User.prototype)); // constructor, sayHi

**Не просто синтаксический сахар**

Иногда говорят, что class – это просто «синтаксический сахар» в JavaScript (синтаксис для улучшения читаемости кода, но не делающий ничего принципиально нового), потому что мы можем сделать всё то же самое без конструкции class:

// перепишем класс User на чистых функциях

// 1. Создаём функцию constructor

function User(name) {

this.name = name;

}

// каждый прототип функции имеет свойство constructor по умолчанию,

// поэтому нам нет необходимости его создавать

// 2. Добавляем метод в прототип

User.prototype.sayHi = function() {

alert(this.name);

};

// Использование:

let user = new User("Иван");

user.sayHi();

Результат этого кода очень похож. Поэтому, действительно, есть причины, по которым class можно считать синтаксическим сахаром для определения конструктора вместе с методами прототипа.

Однако есть важные отличия:

Во-первых, функция, созданная с помощью class, помечена специальным внутренним свойством [[IsClassConstructor]]: true. Поэтому это не совсем то же самое, что создавать её вручную.

В отличие от обычных функций, конструктор класса не может быть вызван без new:

class User {

constructor() {}

}

alert(typeof User); // function

User(); // Error: Class constructor User cannot be invoked without 'new'

Кроме того, строковое представление конструктора класса в большинстве движков JavaScript начинается с «class …»

class User {

constructor() {}

}

alert(User); // class User { ... }

Методы класса являются неперечислимыми. Определение класса устанавливает флаг enumerable вfalse для всех методов в "prototype".

И это хорошо, так как если мы проходимся циклом for..in по объекту, то обычно мы не хотим при этом получать методы класса.

Классы всегда используют use strict. Весь код внутри класса автоматически находится в строгом режиме.

Также в дополнение к основной, описанной выше, функциональности, синтаксис class даёт ряд других интересных возможностей, с которыми мы познакомимся чуть позже.

**Class Expression**

Как и функции, классы можно определять внутри другого выражения, передавать, возвращать, присваивать и т.д.

Пример Class Expression (по аналогии с Function Expression):

let User = class {

sayHi() {

alert("Привет");

}

};

Аналогично Named Function Expression, Class Expression может иметь имя.

Если у Class Expression есть имя, то оно видно только внутри класса:

// "Named Class Expression"

// (в спецификации нет такого термина, но происходящее похоже на Named Function Expression)

let User = class MyClass {

sayHi() {

alert(MyClass); // имя MyClass видно только внутри класса

}

};

new User().sayHi(); // работает, выводит определение MyClass

alert(MyClass); // ошибка, имя MyClass не видно за пределами класса

Мы даже можем динамически создавать классы «по запросу»:

function makeClass(phrase) {

// объявляем класс и возвращаем его

return class {

sayHi() {

alert(phrase);

};

};

}

// Создаём новый класс

let User = makeClass("Привет");

new User().sayHi(); // Привет

**Геттеры/сеттеры, другие сокращения**

Как и в литеральных объектах, в классах можно объявлять вычисляемые свойства, геттеры/сеттеры и т.д.

Вот пример user.name, реализованного с использованием get/set:

class User {

constructor(name) {

// вызывает сеттер

this.name = name;

}

get name() {

return this.\_name;

}

set name(value) {

if (value.length < 4) {

alert("Имя слишком короткое.");

return;

}

this.\_name = value;

}

}

let user = new User("Иван");

alert(user.name); // Иван

user = new User(""); // Имя слишком короткое.

При объявлении класса геттеры/сеттеры создаются на User.prototype, вот так:

Object.defineProperties(User.prototype, {

name: {

get() {

return this.\_name

},

set(name) {

// ...

}

}

});

Пример с вычисляемым свойством в скобках [...]:

class User {

['say' + 'Hi']() {

alert("Привет");

}

}

new User().sayHi();

**Свойства классов**

**Старым браузерам может понадобиться полифил**

Свойства классов добавлены в язык недавно.

В приведённом выше примере у класса User были только методы. Давайте добавим свойство:

class User {

name = "Аноним";

sayHi() {

alert(`Привет, ${this.name}!`);

}

}

new User().sayHi();

Свойство name не устанавливается в User.prototype. Вместо этого оно создаётся оператором new перед запуском конструктора, это именно свойство объекта.

**Итого**

Базовый синтаксис для классов выглядит так:

class MyClass {

prop = value; // свойство

constructor(...) { // конструктор

// ...

}

method(...) {} // метод

get something(...) {} // геттер

set something(...) {} // сеттер

[Symbol.iterator]() {} // метод с вычисляемым именем (здесь - символом)

// ...

}

MyClass технически является функцией (той, которую мы определяем как constructor), в то время как методы, геттеры и сеттеры записываются в MyClass.prototype.

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(){

let i = document.querySelector('input[name= numer1]').value;

let n = document.querySelector('input[name= numer2]').value;

while (i <= n) {

console.log(i);

i++;

}}

</script>

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

<tr>

<td align = "left">число 1:</td>

<td><input name="numer1" style="color:white; background-color: #99AAFF; font-style: oblique" required placeholder=""></td>

</tr>

<tr>

<td align = "left"> число 2:</td>

<td><input name="numer2" style="color:white; background-color: #99AAFF; font-style: oblique" required placeholder=""></td>

</tr>

</table><br/>

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

</body>

</html>

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

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

**var** max = array[0]; **var** maxIndex = 0;

**var** min = array[0]; **var** minIndex = 0;

**for** (**var** i = 1; i < array.length; i++)

            {

**if** (array[i] < min)

                {

                    min = array[i];

                    minIndex = i;

                }

**if** (array[i] > max)

                {

                    max = array[i];

                    maxIndex = i;

                }

            }

            array[minIndex] = max;

            array[maxIndex] = min;

console.log(array);

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>Поиск</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="Отправить запрос"><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=" Отправить запрос "><br/>

</form>

</fieldset>

<fieldset>

</fieldset>

</body>

</html>

<?php

mysqli\_connect("localhost", "root", "","users")//параметры в скобках ("хост", "имя пользователя", "пароль")

or die("<p>Ошибка подключения к базе данных! " . mysql\_error() . "</p>");

?>

<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>