Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 27**

**Инструкционно-технологическая карта**

Тема: Класс. Синтаксис. Что такое синтетический сахар. Class Expression. Свойства классов.

Цель: Сформировать умения по созданию и использованию классов в JavaScript.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Теоретические сведения

На практике часто надо создавать много объектов одного вида, например пользователей, товары или что-то ещё.

В современном JavaScript есть продвинутая конструкция «class», которая предоставляет новые возможности, полезные для объектно-ориентированного программирования.

## Синтаксис «class»

Базовый синтаксис выглядит так:

class MyClass {

    // методы класса

    constructor() { ... }

    method1() { ... }

    method2() { ... }

    method3() { ... }

    ...

}

Для создания нового объекта используется вызов со всеми перечисленными методами **new MyClass()**.

При этом автоматически вызывается метод constructor(), в нём можно 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"):

1. Создаётся новый объект.
2. **constructor** запускается с заданным аргументом и сохраняет его в **this.name**.

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

## Что такое класс?

Это не полностью новая языковая сущность, как может показаться на первый взгляд.

В JavaScript класс – это разновидность функции.

class User {

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

    sayHi() { alert(this.name); }

}

// доказательство: User - это функция

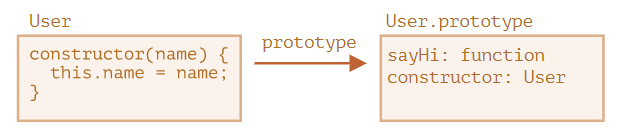
alert(typeof User); // function

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

1. Создаёт функцию с именем **User**, которая становится результатом объявления класса. Код функции берётся из метода **constructor** (она будет пустой, если такого метода нет).
2. Сохраняет все методы, такие как **sayHi**, в **User.prototype**.

При вызове метода объекта **new User** он будет взят из прототипа. 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); // 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 можно сделать всё то же самое без конструкции **class**:

// 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:

1. 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 { ... }

1. Методы класса являются неперечислимыми. Определение класса устанавливает флаг **enumerable** в **false** для всех методов в **"prototype"**.

И это хорошо, так как если необходимо проходиться циклом **for..in** по объекту, то обычно не желательно при этом получать методы класса.

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

## 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

Даже можно 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

## Свойства классов

В приведённом выше примере у класса User были только методы. Добавление свойства:

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.

# Контрольные вопросы

1. Что такое класс?
2. Приведите пример описания класса.
3. Приведите пример Class Expression.
4. Что такое конструктор и в какой момент он вызывается?

# Задания

Задание 1. создать класс с полями, указанными в задании

Реализовать в классе методы: конструктор по умолчанию; конструктор перезагрузки с параметрами.

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | Класс родитель и его поля | Функция-метод 1 обработки данных | Функция-метод 2 обработки данных |
| 1 | Дата (три числа): день, месяц, год | Определить, является ли год високосным (кратным 4) | Увеличить дату на 5 дней |
| 2 | Дата (три числа): день, месяц, год | Увеличить год на 1 | Уменьшить дату на 2 дня |
| 3 | Дата (три числа): день, месяц, год | Определить, совпадают ли номер месяца и число дня | Увеличить дату на один месяц |
| 4 | Время (три числа): часы, минуты, секунды | Вычислить количество секунд в указанном времени | Увеличить время на 5 секунд |
| 5 | Время (три числа): часы, минуты, секунды | Вычислить количество полных минут в указанном времени | Уменьшить время на 10 минут |
| 6 | Время (три числа): часы, минуты, секунды | Определить количество минут до полу ночи (24:00:00) | Увеличить время 100 минут |
| 7 | Координаты изображения прямоугольника: *x*1, *y*1, *x*2, *y*2 | Вычислить площадь прямоугольника в пикселях | Изобразить прямоугольник на форме (Image) с толщиной линии 2 пикселя |
| 8 | Координаты изображения прямоугольника: *x*1, *y*1, *x*2, *y*2 | Вычислить длину диагонали прямо- угольника в пикселях | Изобразить прямоугольник и его диагональ на форме (Image) |
| 9 | Координаты изображения прямоугольника: *x*1, *y*1, *x*2, *y*2 | Определить, является ли прямоугольник квадратом? | Изобразить прямоугольник на форме (Image), закрашенный зеленым цветом |
| 10 | Правильная дробь: числитель, знаменатель | Выразить значение дроби в процентах | Найти сумму цифр значения знаменателя |
| 11 | Комплексное число: действительная (*a*1) и мнимая (*b*1) части числа | Вычислить модуль комплексного числа | Найти комплексное число, обратное заданному |
| 12 | Комплексное число: действительная и мнимая часть числа | Вычислить произведение комплексного числа на число, вводимое пользователем | Вычислить аргумент комплексного числа в градусах |
| 13 | Книга: название, количество страниц, цена | Вычислить среднюю стоимость одной страницы | Увеличить цену книги в два раза, если название начинается со слова «Программирование» |
| 14 | Книга: название, автор, год издания | Вычислить, сколько лет книге | Количество дней, прошедших после года издания книги |
| 15 | Работник: фамилия, оклад, год поступления на работу | Вычислить стаж работы работника на данном предприятии | Сколько дней прошло после года поступления на работу |

# Литература

1. **Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.
2. **Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.
3. **Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич