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 № 14**

Инструкционно-технологическая карта

Тема: Конструкторы. Создание объекта через new.

Цель: Научиться использовать конструкторы. Научиться создавать объекты через оператор new.

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

Обычный синтаксис {...} позволяет создать только один объект. Но зачастую нужно создать множество однотипных объектов, таких как пользователи, элементы меню и т.д.

Это можно сделать при помощи функции-конструктора и оператора "new".

## Функция-конструктор

Функции-конструкторы являются обычными функциями. Но есть два соглашения:

Имя функции-конструктора должно начинаться с большой буквы.

Функция-конструктор должна вызываться при помощи оператора "new".

Например:

function User(name) {

this.name = name;

this.isAdmin = false;

}

let user = new User("Вася");

alert(user.name); // Вася

alert(user.isAdmin); // false

Когда функция вызывается как new User(...), происходит следующее:

1. Создаётся новый пустой объект, и он присваивается this.
2. Выполняется код функции. Обычно он модифицирует this, добавляет туда новые свойства.
3. Возвращается значение this.

Другими словами, вызов new User(...) выполняет следующие действия:

function User(name) {

// this = {}; (неявно)

// добавляет свойства к this

this.name = name;

this.isAdmin = false;

// return this; (неявно)

}

То есть, результат вызова new User("Вася") – это тот же объект, что и:

let user = {

name: "Вася",

isAdmin: false

};

Теперь, когда нам необходимо будет создать других пользователей, возможно использование new User("Маша"), new User("Даша") и т.д. Данная конструкция гораздо удобнее и читабельнее, чем каждый раз создавать литерал объекта. Это и является основной целью конструкторов – удобное повторное создание однотипных объектов.

Технически любая функция может быть использована как конструктор. То есть, каждая функция может быть вызвана при помощи оператора new, и выполнится алгоритм, указанный выше в примере. Заглавная буква в названии функции является всеобщим соглашением по именованию, она как бы подсказывает разработчику, что данная функция является функцией-конструктором, и её нужно вызывать через new.

# new function() { … }

Если в нашем коде большое количество строк, создающих один сложный объект, можно обернуть их в функцию-конструктор следующим образом:

let user = new function () {

this.name = "Вася";

this.isAdmin = false;

// ...другой код для создания пользователя

// возможна любая сложная логика и выражения

// локальные переменные и т. д.

};

Такой конструктор не может быть вызван дважды, так как он нигде не сохраняется, просто создаётся и тут же вызывается. Таким образом, такой метод создания позволяет инкапсулировать код, который создаёт отдельный объект, но без возможности его повторного использования.

## Проверка на вызов в режиме конструктора: new.target

Используя специальное свойство new.target внутри функции, можно проверить, вызвана ли функция при помощи оператора new или без него.

В случае, если функция вызвана при помощи new, то в new.target будет сама функция, в противном случае undefined.

function User() {

alert(new.target);

}

// без "new":

User(); // undefined

// с "new":

new User(); // function User { ... }

Это можно использовать, чтобы отличить обычный вызов от вызова «в режиме конструктора». В частности, чтобы функцию можно было вызывать как с, так и без new:

function User(name) {

if (!new.target) { // в случае, если вызов без оператора new

return new User(name); // ...добавим оператор new за вас

}

this.name = name;

}

let vasya = User("Вася"); // переадресовывает вызовы на new User

alert(vasya.name); // Вася

Такой подход иногда используется в библиотеках для создания более гибкого синтаксиса, который позволяет разработчикам вызывать функции при помощи оператора new или без него.

Впрочем, это не очень хорошая практика, так как отсутствие new может ввести разработчика в заблуждение. С оператором new точно, в итоге будет создан новый объект.

## Возврат значения из конструктора return

Обычно конструкторы ничего не возвращают явно. Их задача – записать все необходимое в this, который в итоге станет результатом.

Но если return всё же есть, то применяется простое правило:

1. При вызове return с объектом, будет возвращён объект, а не this.
2. При вызове return с примитивным значением, примитивное значение будет отброшено.

Другими словами, return с объектом возвращает объект, в любом другом случае конструктор вернёт this.

В примере ниже return возвращает объект вместо this:

function BigUser() {

this.name = "Вася";

return { name: "Godzilla" }; // <-- возвращает этот объект

}

alert(new BigUser().name); // Godzilla, получили этот объект

пример с пустым return

function SmallUser() {

this.name = "Вася";

return; // <-- возвращает this

}

alert(new SmallUser().name); // Вася

Обычно у конструкторов отсутствует return. В данном блоке упомянуто особое поведение с возвращаемыми объектами, чтобы не оставлять пробелов в изучении языка.

### Отсутствие скобок

Скобки после new можно не ставить, если вызов конструктора идёт без аргументов.

let user = new User; // <-- без скобок

// то же, что и

let user = new User();

Пропуск скобок считается плохой практикой, но синтаксис языка такое позволяет.

## Создание методов в конструкторе

Использование конструкторов для создания объектов даёт большую гибкость. Можно передавать конструктору параметры, определяющие, как создавать объект, и что в него записывать.

В this можно добавлять не только свойства, но и методы.

Например, в примере ниже, new User(name) создаёт объект с данным именем name и методом sayHi:

function User(name) {

this.name = name;

this.sayHi = function () {

alert("Меня зовут: " + this.name);

};

}

let vasya = new User("Вася");

vasya.sayHi(); // Меня зовут: Вася

/\*

vasya = {

name: "Вася",

sayHi: function() { ... }

}

\*/

Для создания сложных объектов есть и более «продвинутый» синтаксис – классы.

## Итого

* Функции-конструкторы или просто конструкторы являются обычными функциями, именовать которые следует с заглавной буквы.
* Конструкторы следует вызывать при помощи оператора new. Такой вызов создаёт пустой this в начале выполнения и возвращает заполненный в конце.

Можно использовать конструкторы для создания множества похожих объектов.

JavaScript предоставляет функции-конструкторы для множества встроенных объектов языка: например, Date, Set и других, которые нам ещё предстоит изучить.

# Контрольные вопросы

1. При помощи какого оператора вызывается функция-конструктор?
2. Что происходит при вызове функции new?
3. В каком случае после new не нужно ставить скобки?

# Задания

***Задание:*** Создать конструктор, создающий следующие поля и методы:

| **№** | **Поля** | **Метод 1** | **Метод 2** |
| --- | --- | --- | --- |
| 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 |
| 8 | Координаты изображения прямоугольника: *x*1, *y*1, *x*2, *y*2 | Вычислить длину диагонали прямоугольника в пикселях | Изобразить прямоугольник и его диагональ на странице |
| 9 | Координаты изображения прямоугольника: *x*1, *y*1, *x*2, *y*2 | Определить, является ли прямоугольник квадратом? | Изобразить прямоугольник на странице, 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: действ тельная 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 |
| 16 | Работник: фамилия, оклад, год рождения | Вычислить возраст работника | Сколько календарных дней до исполнения работнику 50 лет |
| 17 | Вектор на плоскости: координаты вектора на плоскости (*x*1, *y*1, *x*2, *y*2) | Вычислить длину вектора | Изобразить линию вектора на странице с толщиной линии 2 пикселя |
| 18 | Вектор на плоскости: координаты вектора на плоскости (*x*1, *y*1, *x*2, *y*2) | Вычислить координаты середины вектора | Равен ли угол наклона  вектора 45 градусов? |
| 19 | Вектор на плоскости: координаты вектора на плоскости (*x*1, *y*1, *x*2, *y*2) | Вычислить координаты вектора, удвоенной длины | Вычислить площадь прямоугольного треугольника, образованного вектором и прямыми, параллельн. осям О*x*, О*y.* |
| 20 | Цилиндр:  диаметр основания, высота | Вычислить объем цилиндра | Изобразить круг заданного диаметра на форме (Image), закрашенный красным цветом. |
| 21 | Параллелепипед: длины сторон | Вычислить объем параллелепипеда | Вычислить длину наибольшей диагонали параллелепипеда |
| 22 | Параллелепипед: длины сторон | Вычислить площадь поверхности | Вычислить сумму длин всех ребер параллелепипеда. |
| 23 | Четыре целых числа: *a, b, c, d* | Вычислить среднее арифметическое чисел | Определить максимальное из чисел |
| 24 | Три вещественных числа *x, y, z* | Вычислить среднее геометрическое чисел | Определите, сколько цифр содержит сумма заданных трех чисел. |
| 25 | Товар: наименование, цена, год выпуска | Определить, сколько лет назад был выпущен товар | Увеличить цену товара на 20%, если в наименовании товара есть слово «TV». |
| 26 | Товар: наименование, цена в гривне, изготовитель | Пересчитать цену товара в долларах | Увеличить цену товара в долларах, если название товара содержит слово «Toyota». |
| 27 | Координаты изображения эллипса: *x*1, *y*1, *x*2, *y*2 | Определить, является ли эллипс окружностью? | Изобразить эллипс на странице зеленым цветом. |
| 28 | Книга: название,  количество страниц, цена | Увеличить количество страниц на 10 | Уменьшить цену в два раза, если количество страниц больше 100 (после увеличения) |
| 29 | Комната: длина, ширина, высота (в метрах) | Площадь стен (вместе с окнами и дверьми) | Площадь стен без окна (размер 2×15 м) и двери (размер 2 ×8 м). |
| 30 | Работник: фамилия, должность, оклад | Увеличить оклад на 15% (каждому работнику) | Работникам, у которых фамилия начинается с сочетания букв «Иван», присвоить должность  «инженер». |

# Литература

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**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