Федеральное государственное образовательное бюджетное учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финансовый университет)

Колледж информатики и программирования

ПМ.08 Разработка кода информационных систем

Группа: 3ИСИП-622

УТВЕРЖДАЮ
Председатель предметно-цикловой комиссии информационных систем и программирования
_____ Т.Г. Аксёнова
«____» _____2025 г.

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

Разработка калькулятора геометрических фигур

Преподаватель
_____Р. Р. Абзалимов
Исполнитель
_____Л. Д. Слепцов
Оценка: _____
«____» _____2025 г.

Москва
2025

Цель работы

Создать веб-приложение для расчета площадей различных геометрических фигур с использованием HTML, CSS и JavaScript, применяя современные подходы к разработке.

Ход работы

1. Почему в современном JavaScript рекомендуется использовать addEventListener вместо onclick?

- Множественные обработчики: addEventListener позволяет добавлять несколько обработчиков на одно событие, тогда как onclick перезаписывает предыдущий обработчик.
- Гибкость: С addEventListener можно управлять фазой события (захват или всплытие) с помощью третьего параметра ({ capture: true }).
- Удаление обработчиков: С addEventListener можно легко удалить обработчик с помощью removeEventListener, что невозможно с onclick.
- Читаемость кода: Использование addEventListener делает код более структурированным и понятным, особенно в больших проектах.

2. Какие преимущества дает использование объекта для хранения шаблонов полей ввода?

- Централизация: Все шаблоны хранятся в одном месте, что упрощает их поиск и изменение.
- Масштабируемость: Добавление новых шаблонов становится проще, так как не нужно изменять логику кода.
- Читаемость: Код становится более понятным, так как шаблоны отделены от основной логики.
- Упрощение тестирования: Легче тестировать и отлаживать код, так как шаблоны изолированы.

3. Почему важно проверять результат на NaN и неположительные значения?

- Корректность данных: Проверка на NaN помогает избежать ошибок при выполнении математических операций с некорректными данными.
- Защита от ошибок: Неположительные значения (например, отрицательные длины или нулевые радиусы) могут привести к некорректным результатам или ошибкам в логике программы.
- Пользовательский опыт: Вывод сообщений об ошибках помогает пользователю понять, что введенные данные некорректны.
- Надежность: Проверка данных делает приложение более устойчивым к ошибкам и непредвиденным ситуациям.

4. Какую роль играет метод toFixed() в выводе результата?

- Округление: toFixed() округляет число до указанного количества знаков после запятой, что полезно для финансовых расчетов или измерений.
- Читаемость: Возвращает строку, которая выглядит более аккуратно и понятно для пользователя.
- Унификация формата: Гарантирует, что все числа будут выводиться в одинаковом формате.
- Ограничение точности: Позволяет избежать вывода избыточных данных.

5. Зачем используется оператор optional chaining (?.) в функции getInputValue?

- Защита от ошибок: Если элемент не существует, ?. возвращает undefined вместо выброса исключения.
- Упрощение кода: Избавляет от необходимости проверять наличие элемента с помощью условий.
- Безопасность: Позволяет безопасно обращаться к свойствам вложенных объектов или DOM-элементов.
- Читаемость: Делает код более лаконичным и понятным.

6. Почему для хранения функций расчета используется объект calculations?

- Организация кода: Все функции хранятся в одном месте, что упрощает их поиск и управление.
- Масштабируемость: Добавление новых функций становится проще, так как не нужно изменять основную логику.
- Упрощение тестирования: Легче тестировать отдельные функции, так как они изолированы.
- Читаемость: Код становится более структурированным и понятным.

7. Какие преимущества дает использование const при объявлении переменных DOM-элементов?

- Неизменяемость ссылок: const предотвращает случайное переопределение переменной, что особенно важно для DOM-элементов.
- Надежность: Гарантирует, что ссылка на элемент останется неизменной на протяжении всего выполнения кода.
- Читаемость: Показывает, что переменная не будет изменяться, что упрощает понимание кода.
- Лучшие практики: Использование const соответствует современным стандартам JavaScript.

8. Почему важно очищать результат при смене типа фигуры?

- Актуальность данных: Очистка результатов предотвращает отображение устаревших данных, которые могут ввести пользователя в заблуждение.
- Пользовательский опыт: Улучшает восприятие интерфейса, так как пользователь видит только актуальную информацию.
- Предотвращение ошибок: Исключает возможность использования некорректных данных в дальнейших расчетах.
- Чистота интерфейса: Делает интерфейс более аккуратным и понятным.

9. Какие методы можно использовать для валидации пользовательского ввода помимо проверки на NaN?

- Регулярные выражения: Для проверки формата данных (например, email, телефон).
- Минимальные/максимальные значения: Для ограничения диапазона вводимых данных.
- Проверка на целые числа: С помощью Number.isInteger().
- Проверка на пустые значения: С помощью trim() и проверки длины строки.
- Валидация в реальном времени: С использованием событий input или change.

10. Как можно улучшить обработку ошибок в данном приложении?

- Подробные сообщения: Вывод понятных сообщений об ошибках с указанием, что именно пошло не так.
- Визуальная индикация: Подсветка некорректных полей с помощью CSS.
- Валидация в реальном времени: Проверка данных сразу после ввода, а не только при нажатии кнопки.
- Логирование ошибок: Запись ошибок в консоль или на сервер для дальнейшего анализа.
- Резервные значения: Использование значений по умолчанию в случае ошибок.