Оценочный материал для промежуточной аттестации по дисциплине «Основы WEB-программирования» 1 семестр

Код, направление подготовки	09.03.02 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ
Направленность (профиль)	Информационные системы и технологии
Форма обучения	очная
Кафедра-разработчик	Информатики и вычислительной техники
Выпускающая кафедра	Информатики и вычислительной техники

Типовые задания для контрольной работы:

Часть 1. Основы HTML и CSS

Задание 1. Создайте HTML-документ с соблюдением стандартов HTML5, содержащий следующие элементы:

- Семантические блоки <header>, <main>, <footer> и <article>.
- Вложенные списки (неупорядоченный и упорядоченный).
- Мини-формы с полем ввода и кнопкой отправки.
- Внутреннюю гиперссылку на секцию внутри страницы.

Результат: валидный HTML-документ, отображаемый в любом современном браузере.

Задание 2. Верстка главной страницы новостного портала. Оформите страницу с использованием CSS, добавьте флексбокс для равномерного распределения карточек новостей по ширине экрана, установите фоновый цвет и цвет ссылок.

- Технические требования:
- Ширина контейнера новостей — 1200px.
- Между карточками новостей — расстояние 20 пикселей.
- Цвет фона — светло-голубой (#dfeffc).

Фон карточки новостей — белый, рамка толщиной 1px серого цвета.

Результат: готовая страница, оформленная с помощью CSS, с правильной структурой и оформлением.

Часть 2. JavaScript и браузерные API

Задание 3. Напишите скрипт на JavaScript, который проверяет корректность заполнения электронной почты и телефонного номера на стороне клиента. Поля обязательны для заполнения. Сделайте так, чтобы кнопка отправляла форму только при успешном прохождении проверок.

Проверочные критерии:

- Электронная почта должна содержать знак "@" и иметь допустимый домен (.ru, .com, .org).
- Телефонный номер должен начинаться с "+7" и содержать ровно 11 цифр.

Результат: рабочий JavaScript-код, обеспечивающий валидацию формы.

Задание 4. Используя метод fetch(), получите JSON-данные с внешнего ресурса (например, https://jsonplaceholder.typicode.com/users) и выведите их на страницу в виде списка пользователей с их именем и городом проживания.

Требования:

- Использовать промисы или async/await.
- Использовать шаблонные строки для вывода имени и города.

Результат: демонстрационный сайт, отображающий полученные данные.

Часть 3. Серверная часть (Node.js)

Задание 5. Настройте и запустите простейший HTTP-сервер на Node.js, который возвращает статичную HTML-страницу с сообщением «Добро пожаловать!» при обращении к корневому адресу (/).

Дополнительные требования:

- Сервер должен прослушивать порт 3000.

- Настройка маршрута /about, возвращающая другую статичную страницу.

Результат: запущенный сервер с функционирующим маршрутом.
Задание 6. Создайте веб-сервис, принимающий POST-запрос с данными формы и сохраняющий их в файл data.json. Форма должна содержать поля email и message.
Требования:

- Используется модуль fs для работы с файлами.
- Проверяйте наличие данных перед сохранением.

Результат: рабочая серверная часть с обработчиком формы и функционалом сохранения данных.
Часть 4. Веб-приложения и работа с базами данных
Задание 7. Подключите приложение Node.js к базе данных MySQL или MongoDB и осуществите CRUD-операции (Create, Read, Update, Delete) для таблицы Users. Реализуйте регистрацию и просмотр профиля пользователя.
Требования:

- Создание пользователя с именем и email.
- Вывод списка зарегистрированных пользователей.
- Редактирование данных пользователя.
- Удаление пользователя.

Результат: полнофункциональное веб-приложение с поддержкой CRUD-операций.
Задание 8. Постройте архитектуру своего веб-приложения, придерживаясь MVC-подхода. Предоставьте описание ролей компонентов (Model, View, Controller) и продемонстрируйте разделение ответственности.
Требования:

- Ясное разделение кода на Model, View и Controller.
- Показать примеры классов и функций для каждого компонента.

Результат: чётко организованное приложение с описанием структуры и компонентов.
Часть 5. Документирование и финализация проекта
Задание 9. Оформите документацию вашего проекта в виде README.md, содержащего:
описание проекта,
инструкцию по установке и запуску,
технические требования,
возможные расширения и дальнейшее развитие.
Требования:

- Соблюдение стандарта Markdown.
- Удобочитаемость и подробность документации.

Результат: полноценный README-файл с указанием всех важных моментов проекта.

Вопросы к зачету.

Часть 1. Основы web, протокол HTTP, архитектура Client-Server

1. Что такое HTTP-протокол и какова его роль в архитектуре Client-Server?
2. Какие основные составляющие входят в HTTP-запрос и HTTP-ответ?
3. Раскройте основные функции браузера и сервера в клиент-серверной архитектуре.
4. Как устроен HTTP GET-запрос и в каких случаях он используется?
5. В чем основное отличие синхронных и асинхронных HTTP-запросов?
6. Перечислите основные HTTP-методы и раскройте их назначения.
7. Чем отличается состояние сессии (stateful) от состоянияless коммуникации в HTTP?
8. Опишите жизненный цикл HTTP-запроса от отправки клиентом до получения ответа сервером.

Часть 2. Основы HTML и CSS, разметка и верстка

9. Какие основные семантические блоки HTML5 вы знаете и для чего они предназначены?
10. Назовите основные HTML-элементы, используемые для формирования структуры страницы.
11. Что такое Flexbox и как его использовать для расположения элементов на странице?
12. Расскажите о селекторах CSS и их применении для стилей элементов.
13. Объясните концепцию адаптивной верстки и роль медиа-запросов в CSS.
14. Покажите пример базовой адаптации страницы с использованием сетки Grid Layout.
15. Докажите необходимость использования семантических тегов в HTML-кодировании и объясните преимущества этого подхода.

Часть 3. JavaScript, браузерные API и обработка данных

16. Что такое JavaScript и какую роль он играет в современном веб-дизайне?
17. Основное назначение и синтаксис переменных в JavaScript.
18. Описание основных структур данных в JavaScript (объекты, массивы, примитивы).
19. Особенности event loop в JavaScript и его взаимодействие с DOM.
20. Важнейшие методы Browser API и их использование для взаимодействия с пользователями.
21. Главные различия между JavaScript в браузере и на сервере (Node.js).
22. Поясните принцип обработки ошибок в JavaScript и как их ловить с помощью try-catch блока.

Часть 4. Серверная сторона: Node.js, обработка HTTP-запросов

23. Что такое Node.js и каковы его основные возможности?
24. Принцип работы модуля fs в Node.js и основные методы для работы с файлами.
25. Приведите примеры обработки ошибок и CLI-интерфейса в Node.js.
26. Как реализовать простейшее HTTP-серверное приложение на Node.js?
27. Охарактеризуйте модуль http и его роль в Node.js.
28. В чем суть маршрутизации в веб-приложениях и как её настроить в Node.js?
29. Подробно расскажите о механизме работы с формой на стороне сервера (валидация, обработка данных).

Часть 5. Работа с базами данных и архитектура приложения

30. С какими проблемами сталкивается разработчик при подключении базы данных к веб-проекту?
31. Опишите основные приемы работы с асинхронностью в JavaScript при взаимодействии с БД.
32. Продемонстрируйте, как экспортировать и импортировать модули в Node.js (export/import).
33. Какие существуют паттерны архитектуры в разработке веб-приложений и какие у них плюсы и минусы?
34. Структуру каталогов типичного веб-проекта: как она выглядит и почему важна?
35. Назовите и поясните основные преимущества и риски использования open-source решений в разработке ПО.
36. Докажите значимость README.md файла и отметьте, что должно содержаться в нём обязательно.

Часть 6. Практическая интеграция фронтенда, бэкенда и базы данных

37. Расскажите, как соединить фронтенд и бэкенд в единую экосистему веб-приложения.
38. Представьте сценарий взаимодействия фронта и бэка на примере AJAX-запроса.
39. Перечислите основные типы запросов к БД, применяемые в веб-приложениях.
40. Проверьте и докажите, что ваше веб-приложение защищено от типичных уязвимостей (XSS, CSRF, SQL-injection).