**Веб-технологии. ОТВЕТЫ**

1. **Краткая история развития World Wide Web. Основные составляющие WWW.**

Текст.

1. **Язык разметки гипертекста HTML. История развития. Основные понятия. Структура документа**

Текст.

1. **Язык HTML. Теги заголовочной̆ части документа. Теги форматирования текста. Графика в HTML-документах. Форматирование таблиц.**

Текст.

1. **Каскадные таблицы стилей CSS. Селекторы. Специфичность селекторов. Применение селекторов для выбора DOM-элементов.**

Текст.

1. **Каскадные таблицы стилей CSS. Позиционирование элементов.**

Текст.

1. **Каскадные таблицы стилей CSS. Flexible Box Layout Module. Свойства flex-элемента и flex-контейнера.**

Текст.

1. **Каскадные таблицы стилей CSS. Медиа запросы. Медиа функции.**

Текст.

1. **Каскадные таблицы стилей CSS. Единицы измерения.**

Текст.

1. **Каскадные таблицы стилей CSS. Блочная модель.**

Текст.

1. **Язык разметки гипертекста HTML. Элемент DIV и его свойства. Гипертекстовые ссылки. URL.**

Текст.

1. **Язык разметки гипертекста HTML. Формы в HTML. Теги форм.**

Текст.

1. **Понятие DOM (Document Object Model) и его связь с HTML-документом. Методы доступа к элементам DOM и различия между ними.**

Текст.

1. **Понятие события DOM. Способы их обработки с помощью JavaScript.**

Текст.

1. **Функции JavaScript для работы с DOM-элементами, их применение.**

Текст.

1. **Понятие делегирования событий (event delegation). Применение делегирования событий при обработке событий.**

Текст.

1. **Понятие объектно-ориентированного программирования (ООП), его основные принципы и подходы в JavaScript. Понятие прототипа объекта (prototype), его применение в JavaScript.**

Текст.

1. **Способы создания нового объекта с помощью конструктора в JavaScript. Основные особенности конструктора в JavaScript. Способы создания методов и свойств у объектов в JavaScript.**

Текст.

1. **Понятие наследования в ООП. Способ реализации наследования с помощью прототипов в JavaScript.**

Текст.

1. **Понятие класса в JavaScript. Основные различия класса и прототипа.**

Текст.

1. **Принципы работы ключевого слова this в JavaScript, его связь с контекстом выполнения.**

Текст.

1. **Основные возможности ES6 (ECMAScript 2015) в сравнении с классическим JavaScript. Примеры использования этих возможностей. Типы данных в ES6 их применение.**

Текст.

1. **Понятие стрелочных функций (arrow functions) в JavaScript их отличия от обычных функций.**

Текст.

1. **Основные принципы работы модульной системы (module system) в ES6. Подходы и практики её применения в организации кода.**

Текст.

1. **Понятие генератора (generator) в JavaScript, способы их применения для создания итераторов.**

Текст.

1. **Основные принципы работы деструктуризации (destructuring) в JavaScript, области её применения.**

Текст.

1. **Принципы и подходы в работе с промисами (promises) в управлении асинхронным кодом в JavaScript.**

Текст.

1. **Понятие фреймворка. Основные особенности и преимущества Vue.js для разработки веб-приложений. Способы создания компонентов в Vue.js. Основные методы жизненного цикла компонента.**

Текст.

1. **Способы передачи данных из родительского компонента в дочерний компонент в Vue.js.**

Текст.

1. **Понятие реактивности в Vue.js, её связь с обновлением DOM.**

Текст.

1. **Методы обработки событий в компонентах Vue.js.**

Текст.

1. **Способы обработки данных пользовательского ввода в компонентах Vue.js.**

Текст.

1. **Принципы работы системы компонентов в Vue.js и организация их взаимодействия.**

Текст.

1. **Понятие глобальных миксинов (global mixins) в Vue.js, их применение.**

Текст.

1. **Принципы работы модификатора v-model в Vue.js, его применение.**

Текст.

1. **Понятие асинхронных компонентов (async components) в Vue.js. Применение асинхронных компонентов в оптимизации загрузки приложения.**

Текст.

1. **Понятие Docker, основные преимущества его использования. Порядок установки и настройки Docker на локальной машине.**

Текст.

1. **Порядок действия для написания простого docker-compose. Основные команды для сборки и запуска контейнеров, описанных в docker-compose.yml файле.**

Текст.

1. **Подходы к настройке сетей для контейнеров в docker-compose.yml файле. Способы настройки связи между контейнерами в docker-compose.yml файле.**

Текст.

1. **Способы просмотра состояния и журнала контейнеров, запущенных с использованием docker-compose. Способы остановки и удаления контейнеров, запущенные с использованием docker-compose.**

Текст.

1. **Понятие, особенности и возможности языка PHP для разработки веб-приложений.**

PHP (Hypertext Preprocessor) — язык серверного программирования, который широко используется для разработки веб-приложений.

Особенности:

* Чувствителен к регистру;
* Динамическая типизация;
* Кроссплатформенность
* Поддержка серверами

Области применения:

1. Разработка динамических веб-сайтов и веб-приложений (взаимодействие с пользователем и данными, интерактивные функции);
2. Работа с базами данных (интеграция с СУБД);
3. Серверная логика для API (обработка запросов, взаимодействие с БД, преобразование результатов в надлежащий формат)
4. Работа с файлами (создание, чтение, запись и удаление файлов);
5. Интеграция с внешними сервисами
6. Обработка данных на стороне сервера;
7. Создание и управление сессиями и аутентификацией
8. Автоматизация и скриптинг (автоматические задачи на сервере: резервное копирование данных, генерация отчетов или рассылка уведомлений по расписанию);
9. **Основные методы взаимодействия между клиентской и серверной частями при использовании PHP.**

PHP-код размещается внутри HTML-страницы с помощью специальных тегов <?php ... ?>. Эти теги позволяют внедрять PHP-инструкции в HTML-код. При обработке страницы веб-сервером, PHP-код выполняется на сервере, а результаты его работы выводятся как часть HTML, отправляемого браузеру пользователя.

* **GET-запрос**: передача данных через URL-строку. Используется для получения информации с сервера. Данные передаются как параметры запроса, видимые в строке браузера.
  + Пример: http://example.com/page.php?name=John&age=25
  + данные доступны через суперглобальный массив $\_GET['key'].
* **POST-запрос**: передача данных через тело запроса. Обычно используется для отправки больших объемов данных или данных, которые должны быть скрыты от пользователя (например, при отправке формы).
  + В PHP данные доступны через массив $\_POST['key'].

Массив **$\_REQUEST** объединяет данные из массивов $\_GET и $\_POST. Он может использоваться для получения данных вне зависимости от метода отправки.

1. **Способы загрузки файлов на сервер с помощью PHP. Функции, используемые для обработки загруженного файла. Основные методы безопасности, применяемые при обработке загружаемых файлов с помощью PHP.**

Текст.

1. **Подходы, функции, правила обработки ошибок к установке соединения с базой данных из PHP.**

Текст.

1. **Понятие ООП в языке PHP. Основные преимущества ООП в разработке программного обеспечения на PHP.**

Текст.

1. **Основные элементы класса в PHP. Способы создания объекта в PHP. Операции с объектами. Основные подходы и принципы при проектировании классов и взаимодействии между ними в PHP.**

Текст.

1. **Понятие наследования в ООП и его применение в PHP для создания иерархии классов.**

Текст.

1. **Понятие абстрактных классов и интерфейсы в PHP, их применение для определения общего поведения и структуры классов.**

Текст.

1. **Понятие полиморфизма в ООП. Способы реализации полиморфизма в PHP с помощью переопределения методов.**

Текст.

1. **Подходы к использованию пространства имён (namespaces) в PHP для организации кода и предотвращения конфликтов имен.**

Текст.

1. **Понятие асинхронного программирования. Основные преимущества для взаимодействия с сервером в PHP. Методы и средства PHP для асинхронного взаимодействия с сервером.**

Текст.

1. **Понятие фреймворка в PHP. Преимущества фреймворка Laravel для разработки веб-приложений.**

Текст.

1. **Компоненты для установки и настройки Laravel на локальном сервере.**

Текст.

1. **Способы, функции и классы Laravel для загрузки и обработки файлов на сервере.**

Текст.

1. **Создание и применение моделей в Laravel для работы с базой данных. Методы и отношения между моделями.**

Текст.

1. **Операции хранения, поиска и удаления данных с использованием моделей и запросов в Laravel.**

Текст.

1. **Методы и функции Laravel для удобной и эффективной работы с базой данных, включая миграции, сидеры и фабрики.**

Текст.

1. **Настройка поиска и фильтрации данных в Laravel с использованием запросов Eloquent.**

Текст.

1. **Функции Laravel для обработки и валидации пользовательского ввода перед сохранением в базу данных.**

Текст.

1. **Методы и функции Laravel для управления файлами и папками на сервере, включая загрузку, хранение и удаление файлов.**

Текст.

1. **Понятие авторизации и аутентификации веб-приложений. Способы реализации в Laravel.**

Текст.

1. **Настройка и применение механизма аутентификации пользователей в Laravel.**

Текст.

1. **Создание и управление ролями и правами пользователей в Laravel.**

Текст.

1. **Методы и функции Laravel для создания и обработки токенов аутентификации (API-токенов).**

Текст.

1. **Создание и настройка API-маршрутов в Laravel для обеспечения взаимодействия с внешними системами через API.**

Текст.

1. **Методы и функции Laravel для валидации и фильтрации входных данных при использовании API.**

Текст.

1. **Инструменты и практики для обеспечения безопасности при работе с авторизацией, разграничением прав и созданием API в Laravel.**

Текст.

1. **Применение командной строки (CLI) в Laravel для выполнения задач и управления приложением. Применение Artisan для выполнения команд и задач из командной строки.**

Текст.

1. **Создание и регистрация собственных команд в Laravel с помощью Artisan.**

Текст.

1. **Методы и функции Laravel для работы с расписанием (Scheduling) и выполнения задач по расписанию. Способы настройки расписания задач в Laravel, указывая время и интервалы выполнения задач.**

Текст.

1. **Применение механизма очередей (Queue) в Laravel для выполнения задач асинхронно или в фоновом режиме. Способы добавления и настройки драйверов очереди в Laravel.**

Текст.