1. **Поясните понятие Шаблонизация.**

Механизм генерации текстового на основе готового шаблона.

1. **Поясните понятие Шаблон (template).**

Статический текст с вкраплениями специальных символов (комбинацией символов), предназначенных для динамической генерации новых фрагментов текста.

1. **Поясните понятие Контекст шаблона (template context).**

Шаблон — это готовая вёрстка страницы или блока, которая состоит только из оформления, и не содержит никакого контента (полезной информации). Шаблон показывает динамическую информацию. Прилагательное «динамическая» означает, что эти данные могут меняться и показываться в зависимости от различных условий.

Контекст шаблона – данные, которые передаются в шаблон.



1. **Поясните понятие Макет (layout).**

Лейаут — это шаблон, который содержит HTML-код, общий для всех страниц сайта. В нём могут содержаться подключение стилей, мета-теги, шапка, подвал. Также лейаут содержит область для вставки уникального содержимого каждой страницы.

1. **Поясните понятие Частичное представление (partial-view).**

Нередко вeб-страницы в приложении используют какие-то общие элементы. Это может быть меню, шапка сайта, футер, другие элементы. Однако здесь встает проблема: если потребуется поменять этот общий элемент, то придется вносить изменения на все веб-страницы, которые используют этот элемент. И было бы гораздо проще определить этот элемент в одном месте, а затем подключать на все страницы.

Решить эту проблему помогают частичные представления (partial views), которые представляют разделяемые общие элементы, которые можно добавлять на обычные представления.

1. **Поясните понятие Хелпер (helper).**

Хелперы фактически представляют функции, которые возвращают некоторую строку. После получения эту строку можно добавить в любое место представления. Строка может представлять собой в том числе и код html.

Хелперы позволяют оптимизировать создание кода представлений. В частности, мы можем один раз определить функцию хелпера, а затем многократно применять ее в самых различных местах для генерации кода.

Хелпер определяется с помощью функции hbs.registerHelper(). Первый параметр функции - название хелпера, а второй - функция, которая возвращает строку. В данном случае возвращается текущее время.

1. **Перечислите встроенные хелперы Handlebars.**

if — используйте этот хелпер для вывода блока по условию.

unless — используйте этот хелпер как обратный хелперу if. Блок будет выведен, если выражение вернет ложное значение.

each — используйте этот хелпер для перебора списков. Внутри блока Вы можете использовать this для ссылки на элемент списка.

with — используйте этот хелпер для сдвига контекста секции handlebars-шаблона.

log — позволяет логировать состояние контекста во время выполнения шаблона

1. **Поясните понятие Статический файл.**

Cтатические это те, которые отдаются в неизменном виде, которые серверу не нужно обрабатывать. Например, JS, CSS, jpg, png, html, txt и т. д. Динамические файлы это те которые сервер обрабатывает: php, pl, cgi и т. д.

файлы, расположенные на стороне сервера и предназначенные для считывания их без изменения с помощью HTTP GET-запроса по имени ресурса, включающего имя файла.

<link>, <script>, <img>, <audio>, <video>, <a>, <form>, <frame> (не поддерживается в HTML5).

1. **Поясните назначение http-заголовков: Cache-control, ETag, Last-Modified, If-Modified-Since, If-None-Match, Expires.**

HTTP/1.1 ввел новый класс заголовков, заголовки ответа Cache-Control, чтобы дать веб-мастерам больший контроль над контентом и решить ограничения, связанные с Expires.

Заголовки ответов Cache-Control включают:

* max-age=[секунды] - описывает максимальный период времени, в течение которого контент остается свежим. Аналогично Expires, эта директива указывает время, относительное моменту запроса, а не абсолютную величину. [секунды] - количество секунд от момента запроса, в течение которых вы хотите, чтобы контент трактовался как свежий.
* s-maxage=[секунды] - подобен max-age, отличаясь тем, что применяется только к общему кэша (т.е. прокси).
* public - помечает авторизованные запросы, как кэшируемые; это нормально, если требуется HTTP-аутентификация, ответы автоматически становятся приватными.
* private - позволяет кэшу, который действует для определенного пользователя (т.е. в браузере) хранить ответ; общему кэшу (т.е. прокси) - нет.
* no-cache - принуждает кэш отправлять запрос на исходный сервер каждый раз для валидации, прежде чем выдать кэшированную копию. Это полезно, когда необходимо гарантировать, что аутентификация принята во внимание (в сочетании с public) или для поддержания жесткой свежести без потери преимуществ кэширования.
* no-store - указывает кэшу не сохранять копию контента, ни при каких условиях.
* must-revalidate - сообщает кэшу, что он должен подчиниться любой свежей информации, что вы ему предоставляете о контенте. HTTP позволяет кэшу хранить устаревший контент при определенных условиях; упомянув этот заголовок, вы сообщаете кэшу, что вы хотите, чтобы он строго следовал вашим правилам.
* proxy-revalidate - подобен must-revalidate, кроме того, что применяется только к прокси.

В секции “[**Как работает кэш**](http://habrahabr.ru/post/203548/#cache-works)” мы сказали, что валидация используется серверами и кэшами для взаимодействия, когда контент был изменен. Используя её, кэш избегает необходимости скачивания контента целиком, когда он уже имеет локальную копию, но не уверен в том, что она все еще свежая.

Валидаторы очень важны; если нет ни одного и не доступна любая информация о свежести (Expires или Cache-Control), кэш не будет хранить контент вообще.

Наиболее распространенный валидатор это время, когда документ был в последний раз именен, о чем сообщено в заголовке Last-Modified. Когда кэш хранит контент, который содержит заголовок Last-Modified, он (кэш) может использовать его для того, чтобы опросить сервер, чтобы узнать был ли изменен контент со времени его последнего просмотра, с помощью If-Modified-Since запроса.

HTTP/1.1 ввел новый вид валидатора, названный ETag. ETag - это уникальные идентификаторы, которые генерируются сервером и изменяются каждый раз, когда запрашивается контент. Поскольку сервер управляет тем, как сгенерированы ETag, кэш может быть уверен в том, что, если ETag совпадают по результатам запроса If-None-Match, контент действительно совпадает.

Почти все кэши используют время из Last-Modified как валидатор; ETag также становятся распространенными.

Большинство современных веб-серверов могут автоматически генерировать и ETag, и Last-Modified заголовки, чтобы использовать в качестве валидаторов для статического контента (т.е. файлов); вам не придется ничего делать. Однако, они не знают достаточно о динамическом контенте (таком, как CGI, ASP, базы данных сайтов), чтобы их генерировать.

HTTP-заголовок Expires - основной способ управления кэшем; он сообщает всем кэшам, как долго контент трактуется как свежий. По истечении этого времени, кэш всегда будет опрашивать исходный сервер, чтобы узнать, изменился ли контент. Заголовки Expires поддерживаются практически любым кэшем.

Заголовки Expires особенно хороши для кэширования статических изображений (таких как навигация и кнопки). Потому что они не изменяются часто и вы можете устанавливать им чрезвычайно долгое время истечения, делая ваш сайт более отзывчивым к пользователям. Они также полезны для управления кэшированием страниц, которые изменяются регулярно. Если вы обновляете страницу с новостями единоразово около шести часов утра, вы можете установить время истечения для контента на этот час - так кэш будет знать о том, когда необходимо получить свежую копию, без необходимости ручного обновления пользователями с помощью нажатия “Обновить”.

**Cache-control** используется для задания инструкций кэширования как для запросов, так и для ответов.

**ETag** (или entity tag) — один из механизмов кэширования в HTTP. По сути это идентификатор, который присваивается файлу сервером для последующей проверки

**Last-Modified** в ответе HTTP содержит дату и время, в которую, по мнению удаленного сервера, запрашиваемый ресурс был изменен.

***If***-**Modified-Since** делает запрос условным: сервер отправит обратно запрошенный ресурс с статусом [200](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/200), только если он был изменен после указанной даты. Если запрос не был изменен после указанной даты, ответ будет [304](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/304) без какого-либо тела; заголовок Last-Modified при этом будет содержать дату последней модификации.

**If-None-Match** Для методов [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET) и [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD) сервер отправляет запрошенный ресурс только в том случае, если он соответствует одному из перечисленных **ETags**. Для [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) и других небезопасных методов он будет загружать только ресурс в этом случае.

1. **Поясните свойство \_\_dirname объекта global.**

\_\_dirname - возвращает путь к каталогу текущего исполняемого файла (**\_\_dirname использует локацию выполняемого скрипта (файла)**).

1. **Поясните назначение и принцип работы метода response.download().**

Функция **res.download()** передает файл по пути в виде " вложения’. Как правило, браузеры запрашивают у пользователя загрузку.

**Параметры:** Имя файла-это имя файла, который должен быть загружен как вложение, а fn-это функция обратного вызова.

**Возвращаемое значение:** оно ничего не возвращает.

По умолчанию Content-Disposition используется параметр заголовка “filename=” path (обычно он появляется в диалоговом окне браузера). Переопределите это значение по умолчанию с filename помощью параметра.

1. **Поясните назначение и принцип работы метода response.attachment().**

Устанавливает Content-Disposition в поле заголовка HTTP - ответа значение “вложение”. Если filename задан a, то он устанавливает тип содержимого на основе имени расширения via res.type() и задает параметр Content-Disposition “filename=”.

Этот метод используется для отправки файла в качестве вложения в HTTP-ответе.

1. **Поясните назначение http-заголовка: Content-Disposition: attachment.**

Attachment – указание http-клиенту не отображать содержимое body, а сохранить в файловой структуре.

В обычном HTTP-ответе заголовок Content-Disposition является индикатором того, что ожидаемый контент ответа будет отображаться в браузере, как вэб-страница или часть вэб-страницы, или же как вложение, которое затем может быть скачано и сохранено локально.

Первым параметром в контексте HTTP должен быть или inline (это значение по умолчанию, указывающее, что контент должен быть отображен внутри вэб-страницы или как вэб-страница) или attachment (указывает на скачиваемый контент; большинство браузеров отображают диалог "Сохранить как" с заранее заполненным именем файла из параметра filename, если он задан).

https://github.com/express-handlebars/express-handlebars