**Протокол HTTP. Структура запросов и ответов, методы запросов, коды ответов сервера, заголовки запросов и ответов.**

Протокол HTTP

* Протокол прикладного уровня
* Основа – клиент сервер
* Основной объект работы – ресурс, на который указывает URI. Ресурс – все что угодно.
* Обмен информацией – вопрос ответ

Stateless протокол – отсутствие состояния. Т.е. сервер не запоминает то что один и тот же клиент обратился к нему с запросом, он всех обслуживает независимо, не запоминая сессий. Плюсы – масштабируемость, минус – состояние не сохраняется

Запрос HTTP состоит из 3х частей.

1. Стартовая строка: метод URI HTTP версия – get /spip.html HTTP/1.1
2. Заголовки - массив элементов в формате ключ значение, содержат метаинформацию для сервера о запросе: Host : cs.ifmo.ru Accept : text/html Connection : close
3. Тело сообщения – набор данных, которые мы передаем на сервер.

URI (identifier) – уникальный идентификатор ресурса, символьная строка позволяющая идентифицировать ресурс

URL (locator) – URI, позволяющий определить местоположение ресурса

URN (name) – URI содержащий единообразное имя ресурса (без местоположения)

REST

Подход к архитектуре сетевых протоколов, обеспечивающих доступ к информационным ресурсам.

Основные принципы

* Данные должны передаваться в стандартном формате – JSON, XML, HTML
* Сетевой протокол должен поддерживать кэширование, но не должен зависеть от сети, не должен сохранять состояние

Те REST приложение компонуется из набора ресурсов, каждый из которых имеет свой URI, URL.

Другой подход – RPC основан на вызове удаленных процедур. Набор процедур, которые находятся на одном уровне URL. RPC – сложная логика – информационные системы. REST – более простой – публичные сайты.

Методы HTTP

* OPTIONS – определение возможностей сервера
* GET – запрос на ресурс
* HEAD – аналог GET, в котором отсутствует тело
* POST – передача данных ресурсу
* PUT — обновляет существующий
* DELETE — удаляет

Коды состояния

* Состоят из 3х цифр
* Первая цифра – класс состояния
  + 1 - информационный
  + 2 - успешно
  + 3 - перенаправление
  + 4 – ошибка клиента
  + 5 – ошибка сервера
* Пример 404

Заголовки HTTP

Формат – ключ: значение

* General Headers – могут быть добавлены в любое сообщение
* Request Headers – используются только в запросах клиента
* Response Headers – используются только в запросах сервера
* Entity Headers – сопровождают любую сущность сообщения

**Язык разметки HTML. Особенности, основные теги и атрибуты тегов.**

Стандартный язык разметки документов в интернете, интерпретируется браузером и отображается в виде документа. Изначально проектировался для создания простых сайтов.

Браузер – программа, отображающая HTML документ в его отформатированном виде.

Структура HTML документа

* Документ состоит из элементов
* Начало и конец элемента обозначаются тегами <b>жирный текст</b>
* Теги могут быть пустыми <br>
* Теги могут иметь атрибуты <a href = …> Атрибут herf </a>
* Вложенность <b> fat text <i> ещё и курсивный </i> </b>
* Документ должен начинаться с объявлении версии HTML <!DOCTYPE HTML …>
* Начало и конец документа обозначаются тегами <html> </html>
* Внутри этих тегов находятся заголовок - <head>…</head > и тело документа <body> </body>

**Структура HTML-страницы. Объектная модель документа (DOM).**

Объектная модель документа - DOM

Интерфейс, позволяющий программам и скриптам получать доступ к содержимому HTML документов

* Стандартизирована W3C
* Представляет собой дерево узлов- корень - html
* Узлы связаны между собой отношением родитель – потомок
* Используется для динамического изменения страниц HTML

HTML-5

C 2014 года, стала более удобной для показа мультимедии.

* Новая вводная строка - <!DOCTYPE html>
* <audio> <video>
* Новые атрибуты, новые поля ввода итд.

**HTML-формы. Задание метода HTTP-запроса. Правила размещения форм на страницах, виды полей ввода.**

HTML формы

* Предназначены для обмена данными между пользователем и сервером
* Документ может содержать любое число форм, но одновременно на сервер может быть отправлена только одна из них
* Вложенные формы запрещены
* Границы теги <from> … </from>
* Атрибуты
  + Action - Адрес программы или документа, который обрабатывает данные формы.
  + Autocomplete - Включает автозаполнение полей формы.
  + Enctype - Способ кодирования данных формы.
  + Method - Метод протокола HTTP.
  + Name - Имя формы.
* Методы
  + GET - Данные видны всем в адресной строке браузера. передает данные в URL в виде пар "имя-значение" (другими словами, через ссылку)
  + POST - Данные можно увидеть только с помощью инструментов разработчика, передает данные в теле запроса

**Каскадные таблицы стилей (CSS). Структура - правила, селекторы. Виды селекторов, особенности их применения. Приоритеты правил. Преимущества CSS перед непосредственным заданием стилей через атрибуты тегов.**

CSS

* Технология описания внешнего вида языка разметки
* Используется для шрифтов цветов итд
* Цель – разделение содержимого и его представления
* Возможность представления документа в различных стилях – темный

Источники CSS

* Авторские стили
  + Inline стиль - Стиль текста указывается в его атрибуте style
  + Встроенный стиль – блок css находится внутри html кода
  + Внешняя таблица стилей – отдельный файл css
* Пользовательские стили
  + Локальный css файл, который пользователь изменил у себя = переопределение авторского
* Стиль браузера
  + Стандартный стиль, используемый браузером по умолчанию для представления элементов – если не применен свой или авторский

Структура CSS

Таблица стилей состоит из набора правил, каждое правило состоит из селекторов и блоков определений: селектор {св-во : значение;}

Набор селекторов – к каким элементам страницы будет применятся правила

Приоритеты стилей

Если к одному элементу будет подходить несколько наборов стилей, то будет применен самый приоритетный из них. Однако свойство, указанное позже, будет иметь больший приоритет.

1. !important только для больших проектов
2. Стиль прописан напрямую в теге
3. Наличие идентификаторов (#id) в селекторе
4. Количество классов .class и псевдо классов в селекторе
5. Количество имен тегов в селекторе

Плюсы CSS

* Разграничение кода и оформления
* Разное оформление для разных устройств
* Централизованное хранение
* Меньше кода

**LESS, Sass, SCSS. Ключевые особенности, сравнительные характеристики. Совместимость с браузерами, трансляция в "обычный" CSS.**

Less/Scss

Языки стилей, которые позволяют повысить уровень абстракций кода CSS и упростить структуру таблиц стилей. Для олдовых браузеров – переводятся в обычный CSS с помощью транслятора.

Особенности по сравнению с обычным CSS

* Можно использовать переменные
* Можно использовать вложенные правила
* Поддерживается вложенность
* Больше импорта
* Примеси – мы пишем под каждый браузер свою реализацию чего то. Потом вызываем эту «ф-ю» и она выбирает нужное правило из имеющихся.
* Поддержка мат. Операторов

Less – css like язык разметки, Sass – вышел из ruby SCSS – диалект Sass c синтаксисом Css.

Для того, чтобы браузер смог понять этот стиль его надо транслировать в обычный css. Обычно это делают на стадии компиляции.

**Клиентские сценарии. Особенности, сферы применения. Язык JavaScript.**

JavaScript – объектно-ориентированный скриптовый язык программирования. Используется для придания интерактивности веб-страницам

Основные черты(постулаты)

* Динамическая типизация – создание переменных без типа, и присвоение типа в процессе выполнения
* Слабая типизация – int -> string
* Автоматическое управление памятью
* Прототипное программирование – наследование на уровне объектов
* Функции – как объекты первого класса
* Регистрозависимый
* Названия переменных не могут начинаться с цифр
* Ядро (ECMAScript)
  + Встраиваемый, расширяемый, не имеющий средств ввода/вывода язык программирования (встраиваемый = не использующийся самостоятельно)
  + Примитивы – Number, String, Boolean, Null, Undefined
  + Объектный тип – object
  + 15 различных видов инструкций
  + Блок не ограничивает область видимости функции
  + Есть глобальные переменные
  + Функция – объект
* Объектная модель документа: JS позволяет производить следующие манипуляции
  + Получение узлов
  + Изменение узлов
  + Изменение связей между узлами
  + Удаление узлов
* Объектная модель браузера
  + BOM – API для взаимодействия с браузером
  + Главное назначение – управление окнами браузера и обеспечения их взаимодействия
  + Специфична для каждого браузера
  + Каждое окно браузера – объект window
  + Управление фреймами – тег, который позволяет подгрузить внешнюю страницу
  + Системные диалоги
  + Управление адресом открытой страницы
  + Поддержка cookie, так как http не умеет сохранять состояние страницы, мы можем взять куки и реализовать нужную версию страницы

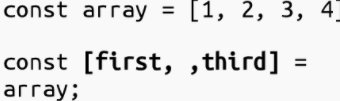
Встраивание JS в веб-страницу

* Внутри страницы
* Внутри тега
* Отделение от разметки (DOM)
* В отдельном файле

**Версии ECMAScript, новые возможности ES6 и ES7.**

Новая версия языка, поддерживается почти всеми новыми браузерами, для старых нужен транслятор

Основные особенности

* Два новых слова let, const – область видимости – блок, константа
* Новый синтаксический сахар – переменные
* A,b=x,y
* Деструктуризация – наложение маски
* Можно возвращать из ф-и несколько значений, можно накладывать маски на всё
* Появление классов и объектов и наследования – ООП
* Extends super
* Promises – вместо того чтобы вкладывать callback внутрь мы создаем promis который обрабатывает запрос.
* Стрелочные ф-и
* Появление итераторов
* Возможность задания параметров по умолчанию в ф-ях
* Spread – синтаксис, который позволяет применить какую то ф-ю ко всем э-м массива.

**Синхронная и асинхронная обработка HTTP-запросов. AJAX.**

* Синхронно: скрипт останавливается и ждет, пока сервер отправит ответ, прежде чем продолжить. Синхронная обработка запроса аналогична повторной загрузке страницы. Но вместо всей веб-страницы загружается только запрошенная информация. При этом пользователю все равно придется ждать окончания загрузки, чтобы снова взаимодействовать со страницей.
* Асинхронно: скрипт разрешает обработку страницы и обрабатывает ответ, когда и если он придет. Асинхронная обработка запроса позволяет избежать задержки в работе с сайтом. Пользователю не нужно ждать ответа с сервера, и он может продолжать выполнять различные действия на странице.

DHTML

Способ создания интерактивного веб сайта, который сочетает css js html dom.

AJAX

Подход к построению динамической веб страницы. База – фоновый обмен данными с сервером. Страница не перезагружается полностью. Использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью

* с использованием XMLHttpRequest;
* через динамическое создание дочерних фреймов;
* через динамическое создание тега <script>

XMLHTTP (XMLHttpRequest, XHR)

- набор API, позволяющий осуществлять НТТР-запросы к серверу без необходимости перезагружать страницу. Данные можно пересылать в виде XML, JSON, HTML или просто неструктурированным текстом. При пересылке используется текстовый протокол HTТР и потому данные должны передаваться в виде текста.

**Преимущества:**

* экономия трафика;
* уменьшение нагрузки на сервер;
* ускорение реакции интерфейса;

**Недостатки:**

* отсутствие интеграции со стандартными инструментами браузера;
* динамически загружаемое содержимое недоступно поисковикам; старые методы учёта статистики сайтов становятся неактуальными;
* усложнение проекта;
* требуется включенный JavaScript в браузере.

**Библиотека jQuery. Назначение, основные API. Использование для реализации AJAX и работы с DOM.**

jQuery

* JS-библиотека, предназначенная для разработки DHTML и AJAX-приложений.
* Упрощает доступ к элементам DOM с помощью кучи разных способов.
* Упрощает и унифицирует (для разных браузеров) реализацию AJAX.
* Упрощает добавление визуальных эффектов.
* Ключевым элементом API является функция (объект) $ и её синоним jQuery.

Основные особенности

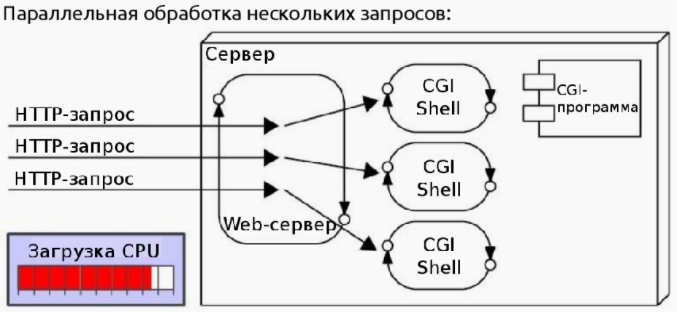
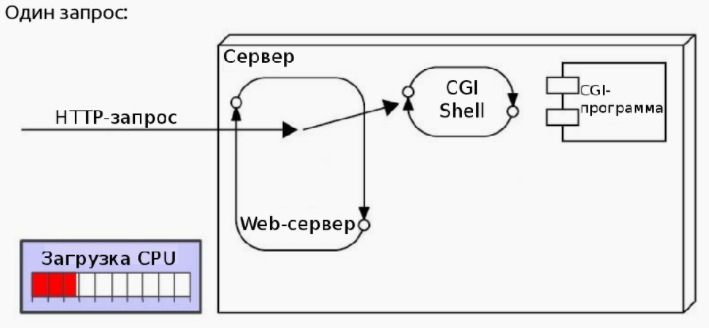
* Кроссбраузерность
* Взаимодействие с DOM
* Обработка событий
* Эффекты и анимация

**Реализация AJAX с помощью SuperAgent.**

API для реализации ajax. Более понятный синтаксис, не такой обобщенный и крутой как jQuery

**Серверные сценарии. CGI - определение, назначение, ключевые особенности.**

CGI - механизм вызова пользователем программ на стороне сервера.

* Данные отправляются программе посредством HTTР-запроса, формируемого веб-браузером.
* То, какая именно программа будет вызвана, обычно определяется URL запроса.
* Каждый запрос обрабатывается отдельным процессом CGI-программы.
* Взаимодействие программы с веб-сервером осуществляется через stdin и stdout.

Когда к серверу приходит запрос, он forkает новый процесс, передает все данные что ему пришли, процесс отрабатывает эти данные и отдает их обратно.

Достоинства:

* Программы могут быть написаны на множестве языков программирования.
* «Падение» CGI-сценария не приводит к «падению» всего сервера.
* Исключены конфликты при параллельной обработке нескольких запросов.
* Хорошая поддержка веб-серверами.

Недостатки:

* Высокие накладные расходы на создание нового процесса.
* Плохая масштабируемость.
* Слабое разделение уровня представления и бизнес-логики.
* Могут быть платформо-зависимыми.

**FastCGI - особенности технологии, преимущества и недостатки относительно CGI.**

Вместо того, чтобы форкался процесс, мы можем прослушивать какой то порт и при вызове сами решать как обрабатывать запрос.

Развитие технологии CGI. Все запросы могут обрабатываться одним процессом CGI-программы (фактическая реализация определяется программистом). Веб-сервер взаимодействует с процессом через UNIX Domain Sockets или TCP/IP (а не через stdin stdout).

**Язык PHP - синтаксис, типы данных, встраивание в веб-страницы, правила обработки HTTP-запросов. Особенности реализации принципов ООП в PHP.**

PHP (PHP: Hypertext Preprocessor) – скриптовый язык, часто используемый для написания веб-приложений.

Синтаксис

* Интерпретатор выполняет код, находящийся внутри ограничителей:
  + <?php
  + echo 'Hello, world!';
  + ?>
* Имена переменных начинаются с символа «$»:
  + $hello = 'Hello, world!';
* Инструкции разделяются символом «р»:
  + $a = 'Hello '; $b = 'world!';
  + $c = $b + $a;
* Все что вне ограничений – обычный html

Типы данных

* PHP - язык с динамической типизацией; при объявлении переменных их тип не указывается.
* 6 скалярных типов данных - integer, float, double, boolean, string и NULL. Диапазоны числовых типов зависят от платформы.
* 3 нескалярных типа - ресурс (например, дескриптор файла), массив и объект.
* 4 псевдотипа - mixed, number, callback и void.

Предопределённые массивы, имеющие глобальную область видимости:

* $\_GLOBALS - массив всех глобальных переменных.
* $\_SERVER - параметры, которые ОС передаёт серверу при его запуске.
* $\_ENV - переменные среды ОС.
* $\_GET, $\_POST - параметры GET- и POST-запроса:
* $\_FILES - сведения об отправленных методом POST файлах.
* $\_COOKIE - массив cookies.
* $\_REQUEST - содержит элементы из массивов
* $\_GET, $\_POST, $\_COOKIE и $\_FILES.
* $\_SESSION - данные HTTP-сессии.

ООП – все как джаве