**Тема 2. Введение в интернет-технологии и web-приложения**

**Содержание лекции** – Фронтенд-разработка для web-приложений: изучение основ фронтенд-разработки для web-приложений, языки разметки (HTML, CSS), язык программирования JavaScript, фреймворки и библиотеки для создания интерактивных пользовательских интерфейсов.

**Рекомендуемая литература**

1. Храмцов П.Б. Основы Web-технологий. Курс лекций [Текст] : курс лекций / П.Б. Храмцов [и др.]. - М. : ИНТУИТ.РУ, 2003. - 512 с. - (Основы информационных технологий).

2. Матросов А. HTML 4.0 [Текст] : учебное пособие для вузов / А. Матросов, А. Сергеев, М. Чаунин. - СПб. : БХВ - Петербург, 2005. - 672 с. : ил.

**Фронтенд-разработка: ключевые технологии и понятия**

Когда вы гуляете по городу, что вы видите? Взгляд часто останавливается на стильных витринах, созданных с целью привлечь внимание. Вы выбираете самую красивую из них и заходите внутрь.

То же самое происходит и в интернете. Блуждая по интернету в поисках необходимого, вы натыкаетесь на самый подходящий на первый взгляд сайт и заходите на него.

Но если витрины прозрачны и показывают, что находится внутри, приглашают рассмотреть товары поближе, то для сайтов все немного иначе. Что скрывается под видимыми элементами сайта, как они взаимодействуют с пользователем, друг с другом и с системой, стоящей за всем этим? Давайте же выясним это.

Что такое фронтенд-разработка? Фронтенд vs бэкенд

Фронтенд веб-сайта — это все, что пользователь видит и с чем может взаимодействовать при помощи браузера. Создание этой визуальной части называется фронтенд-разработкой. Дизайнеров, создающих пользовательские интерфейсы, тоже можно в какой-то смысле назвать фронтенд-разработчиками, потому что они совместно работают над этой же частью проекта.

Для разработки фронтенда в качестве базовых инструментов используются: HTML (для создания базовой структуры страниц и контента), CSS (для стилизации внешнего вида) и JavaScript (для добавления интерактивности). Такой же набор инструментов используется в процессе создания [прогрессивных веб-приложений](https://www.altexsoft.com/blog/engineering/progressive-web-apps/) — мобильных приложений, которые выглядят, как нативные, но при этом создаются с участием фронтенд-технологий. Подробнее об этом можно почитать в статье [по ссылке](https://www.altexsoft.com/blog/engineering/progressive-web-apps/).

Бэкенд — это серверная часть веб-приложения, скрытая от глаз пользователя. Это понятие включает в себя серверы, на которых расположены веб-страницы и определенную логику, которая управляет функциями и процессами сайта. [Здесь](https://www.altexsoft.com/blog/engineering/web-application-architecture-how-the-web-works/)можно почитать более подробное описание внутренней работы веб-приложений.

Бэкенд разрабатывается с использованием другого стека технологий, включая Java, PHP, Ruby, C# и иногда JavaScript, о которых мы поговорим в соответствующем разделе статьи.



Рисунок 1- Цикл запрос-ответ

Итак, базовый набор инструментов для разработки фронтенда четко определен: HTML, CSS и JavaScript. Однако этот набор может быть значительно расширен, включив в себя диспетчеры пакетов, CSS-препроцессоры, фреймворки и многое другое.

HTML: ключевая фронтенд-технология

HTML (от англ. Hypertext Markup Language) — это язык гипертекстовой разметки, предназначенный для создания веб-сайтов, которые впоследствии могут просматриваться при помощи доступа к интернету. HTML обычно используется для структурирования веб-документа. Он определяет такие элементы, как заголовки или абзацы, и позволяет вставлять изображения, видео и другие медиафайлы.

**Как работает HTML.**HTML-код представляет собой множество тегов и пишется в текстовом файле. Этот текстовый файл затем сохраняется в виде HTML-файла, который можно открыть и посмотреть в браузере. Браузер сканирует его, интерпретирует код в визуальную форму и в лучшем случае отображает страницу именно так, как задумал дизайнер.

* Гипертекст — способ, с помощью которого мы путешествуем по интернету, переходя по гиперссылкам, которые ведут на другие страницы. «Гипер» отсылает к нелинейности, которая позволяет перемещаться в любое место, поскольку этот процесса не подразумевает предопределенного порядка.
* Разметка определяет качества, которыми наделяется текст внутри HTML-тегов. Теги определяют, как браузеры форматируют и отображают содержимое страницы.
* Будучи языком, он содержит кодовые слова и синтаксис, как и любой другой язык.

Например:



Рисунок 2 - Пример HTML-кода

Парный тег <html> / </html> определяет границы веб-страницы, а текст между тегами <body> / </body> определяет видимое содержимое страницы.

<!DOCTYPE html> в самом начале объявляет тип документа для HTML5. Если его не обозначить, разные браузеры будут отображать его по-своему.

Вот так плавно мы перешли к HTML5.

***HTML5***

С момента первого своего релиза в 1991 году HTML претерпел множество обновлений. HTML5 был выпущен в 2014 году. В него были добавлены такие функции, как поддержка оффлайн хранилищ мультимедийных данных, более точные элементы контента (например, хедер, футер, навигация) и поддержка встраивания аудио и видео.

***CSS: стилизация***

CSS («каскадные таблицы стилей», от англ. Cascading Style Sheets) — это язык, используемый для стилизации страниц. Он определяет то, как HTML-элементы будут выглядеть на веб-странице с точки зрения дизайна, макета на разных устройствах с разными размерами экрана. CSS управляет макетом множества различных веб-страниц одновременно.

**Как это работает.** CSS взаимодействует с HTML-элементами, компонентами веб-страницы.

* Для взаимодействия с HTML в CSS используются **селекторы**. Селектор — это часть CSS-кода, определяющая, на какую часть HTML будут воздействовать стили CSS.
* **Объявление**содержит свойства и значения, используемые селектором.
* **Свойства**определяют размер шрифта, цвет и отступы. Каждое свойство имеет значение или набор значений.

Возьмем в качестве примера:



где P (абзац) — это селектор, { font-size:24px; color:blue; } — это объявление, font-size: и color: — свойства, а 24px; и blue; — значения.

CSS записывается в виде простого текста в текстовом редакторе. Есть три способа добавить CSS-код в HTML:

* Вместо того, чтобы добавлять CSS-код к каждому HTML-элементу, который необходимо видоизменить, можно использовать внешнюю таблицу стилей (External style sheets) — текстовый файл, содержащий весь CSS-код. Внешняя таблица стилей подключается в .html файле в контейнере <head>.
* При использовании внутренней таблицы стилей CSS-код размещается прямо в теле тега <head> конкретной .html-страницы.
* Под встроенным стилем подразумевается прописывание CSS в html-коде непосредственно того элемента, который нужно стилизовать.

**CSS-фреймворки.** CSS-фреймворк — это набор дефолтных CSS- и HTML-файлов, который расширяет возможности по части дизайна. Помимо их пользы при создании адаптивного дизайна, CSS-фреймворки помогают создавать различные макеты, что избавляет разработчиков от необходимости писать код с нуля в каждом случае. Обычно они заметно помогают при разработке веб-приложений под разные платформы и размеры экрана. Благодаря общим компонентам пользовательского интерфейса, Grid **Layout** и многим другим функциям, CSS-фреймворки значительно ускоряют рабочий процесс разработки. Существуют разные виды фреймворков:

* Полнофункциональные (Bootstrap, Foundation, Semantic UI и др.),
* Ориентированные на материальный дизайн: (Materialize and Material Design Lite) и
* Легкие (Pure).

Мы не будем описывать их все, вместо этого приведем сравнительную таблицу:



Рисунок 3 - Самые популярные CSS-фреймворки

Чтобы подробно изучить эти CSS-фреймворки, ознакомьтесь со статьей [«Самые популярные адаптивные CSS-фреймворки»](https://www.altexsoft.com/blog/engineering/most-popular-responsive-css-frameworks-bootstrap-foundation-materialize-pure-and-more/).

**Препроцессоры — Sass и LESS**. Написание CSS является рутиной, и мелкие задачи, такие как: поиск значений цвета, закрытие тегов или любые другие повторяющиеся операции, отнимают много времени. Вот где пригождается препроцессор. CSS-препроцессор представляет собой скриптовый язык и расширяет возможности CSS.

Самые распространенные препроцессоры — это Sass и LESS. У них есть некоторые общие основы:

* Синтаксические элементы и
* Обратная совместимость с обычными файлами CSS.

Однако между ними есть и различия.

* Sass расшифровывается как Syntactically Awesome Style Sheets («синтаксически превосходные таблицы стилей»). Sass работает на Ruby и обрабатывается на стороне сервера. Поскольку корни его происхождения восходят к языку Ruby, установка осуществляется через так называемые gem-ы (несколько библиотек Ruby/Rails).
* LESS расшифровывается как Leaner Style Sheets («компактная таблица стилей»). На данный момент его можно назвать скорее JavaScript-библиотекой, которая обрабатывается на стороне клиента в браузере. Разработчики гораздо чаще выбирают LESS при использовании JavaScript с таблицами стилей. Эта технология позволяет использовать фрагменты CSS-кода в файлах LESS повторно.

DOM: структура веб-страницы

Объектная модель документа (DOM, Document Object Model) — это программный интерфейс для документов HTML и XML. Он интерпретирует страницу, чтобы программы могли видоизменять структуру, стиль и содержимое документа. DOM отображает документ в виде узлов и объектов, позволяя языкам программирования подключаться к странице.



Рисунок 4 - Дерево объектов HTML DOM. Источник: W3Schools

Как это работает. Будем считать, что веб-страница — это документ, который может быть представлен либо в окне браузера, либо в качестве исходного HTML-кода. DOM является представлением этого документа, поэтому его можно изменить. DOM — это объектно-ориентированное представление веб-страницы, которое можно изменить с помощью сценарного языка, например, JavaScript.

DOM должен соответствовать стандартам спецификаций [W3C](https://dom.spec.whatwg.org/) и [WHATWG](https://dom.spec.whatwg.org/), которые исполняются в большинстве современных браузеров. Современный DOM строится с использованием нескольких [API](https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/), работающих вместе. DOM определяет объекты, которые полностью описывают документ и объекты в нем.

***JavaScript: оживляя веб***

JavaScript (JS) — один из самых популярных сценарных языков. Он в основном известен своей применимостью как для фронтенд-, так и для бэкенд-разработки. Во фронтенде он используется для придания веб-страницам динамики.

Как работает **JavaScript.**JS повышает общую интерактивность сайта. Он позволяет моделировать анимированные компоненты пользовательского интерфейса, такие как: слайдеры, всплывающие окна, расширенные меню навигации по сайту и многое другое. С помощью JavaScript веб-сайт можно наделить разными функциональностями, что не достижимо только с помощью HTML и CSS. Веб-страницы, разработанные с помощью JavaScript, реагируют на действия пользователей и обновляются динамически. Благодаря JavaScript этот процесс не требует перезагрузки страниц, чтобы отобразить изменения.

***Фреймворки и библиотеки JavaScript и зачем они нам нужны***

Новичкам в этой области знаний может показаться, что фреймворки и библиотеки делают одно и то же, благодаря чему различные визуальные элементы взаимодействуют друг с другом. Будем честны: это не так уж далеко от истины. Однако есть несколько отличительных особенностей. Итак, давайте определим, что представляет собой JS-фреймворк, что такое JS-библиотека и для какой цели они оба служат.

**Фреймворки**представляют собой шаблоны для создания веб-сайта или веб-приложения. Они обеспечивают структуру (например, основу или заготовки — scaffolding), на которой можно разместить весь проект. В то время как фреймворк устанавливает шаблоны страниц, они создают структуру с определенными выделенными областями для встраивания кода фреймворка.

Итак, фреймворки JavaScript — это полные наборы инструментов для формирования и настройки веб-сайта или веб-приложения.

**Библиотеки**— это наборы предварительно написанных фрагментов кода, которые повторно используются для реализации основных функций JavaScript. При необходимости фрагмент кода можно легко интегрировать в существующий код проекта.

Таким образом, библиотека — это специализированный инструмент для конкретных узких потребностей, а не универсальная машина для подготовки всего проекта.

***Основные фреймворки и библиотеки***

Начнем с фреймворков:

[**Angular**](https://angular.io/)— это JavaScript-фреймворк от Google, совместимый с большинством распространенных редакторов кода. Angular предназначен для создания динамических одностраничных веб-приложений (SPA — Single Page Applications) и [прогрессивных веб-приложений](https://www.altexsoft.com/blog/engineering/progressive-web-apps/). Еще с момента выпуска первоначальной версии этот фреймворк больше всего ценился за способность преобразовывать документы на основе HTML в динамический контент. Angular является одним из самых популярных фронтенд-фреймворков. Почитать про него подробнее можно [в нашей статье](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/#angular%202%20and%20angular%202+) о плюсах и минусах разработки на Angular.

[**Vue.js**](https://vuejs.org/) — еще один фреймворк с открытым исходным кодом для одностраничных приложений, который требует знания HTML и CSS. Он использует модель разработки на основе компонентов и позволяет присоединять компоненты к проекту. Vue.js — это пример библиотеки, больше похожей на фреймворк, поэтому мы отнесли его к фреймворкам. Он предлагает целую кучу шаблонов и паттернов, применяемых при разработке. Vue в первую очередь известен небольшим размером документов и синтаксисом на основе HTML. Чтобы ознакомиться подробнее с  преимуществами и недостатками Vue.js, [переходите по ссылке](https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/).

[**Ember.js**](https://emberjs.com/) — это фреймворк для разработки одностраничных, мобильных и десктопных приложений. Он использует шаблон проектирования [Model-View-ViewModel](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel) (MVVM). Инструменты Ember позволяют проектировать среду разработки, а его интерфейс командной строки предоставляет инструменты для автоматизации сценариев.

А теперь коротко о нескольких библиотеках:

[**React**](https://reactjs.org/)— библиотека с открытым исходным кодом для создания динамических пользовательских интерфейсов, разработанная Facebook. Применяется для создания веб-приложений с множественными динамическими компонентами. React основан на JavaScript и [JSX](https://reactjs.org/docs/introducing-jsx.html) и позволяет создавать HTML-элементы для многократного использования. React также включает в себя React Native, специальную кроссплатформенную среду для разработки мобильных приложений. Чтобы узнать больше о плюсах и минусах React и React Native, читайте [статью](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/) на эту тему.

[**jQuery**](https://jquery.com/), в свою очередь, предназначена для управления HTML-документами. Она обладает простым API для управления событиями и разработки анимации в браузерах. Кроме того, jQuery применяется для управления объектной моделью документа (DOM), а также служит инструментом разработки плагинов. Она также поставляется с более легкой кросс-браузерной библиотекой, [jQuery UI](http://jqueryui.com/) для мобильного фреймворка [jQuery Mobile](https://jquerymobile.com/) и для построения графического интерфейса.

[**D3.js**](https://d3js.org/) — это управляемая данными библиотека для визуализации данных. С помощью привязывания временных данных к DOM и внедрения в документ изменений, управляемых данными, библиотека позволяет управлять данными и создавать динамическую [визуализацию данных](https://www.altexsoft.com/blog/data-visualization-tools-types-techniques/). Она может поддерживать и обрабатывать большие наборы данных и динамические ответы для взаимодействия и анимации. Функциональный стиль D3 допускает повторное использование кода и работает с CSV и HTML.

Как мы видим, библиотеки JavaScript обладают широкой функциональностью, обеспечивая фронтенд-разработчиков универсальными решениями. Но мы не говорим о замене фреймворков библиотеками или наоборот — всегда найдется место как для одного, так и для другого.

Если вы начинаете свой путь во фронтенде, было бы разумно начать с библиотек, так как они имеют более мягкую кривую обучения. После изучения основ некоторых из них можно переходить к фреймворкам, поскольку они требуют более глубокого понимания JS.

JavaScript для бэкенда

Говоря о JavaScript как о комплексной среде разработки, мы не можем не упомянуть его значимость для бэкенд-разработки. Поскольку JavaScript очень популярен, мир разработки программного обеспечения адаптировал JS к специфике бэкенда.

Для разработки на стороне веб-сервера с помощью JavaScript одним из самых значимых инструментов, вероятно, стал Node.js. Однако, это не фреймворк и не библиотека. Node.js — это среда выполнения, работающая на основе ядра JavaScript V8. Чтобы больше узнать о Node.js, прочитайте [статью](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/) о плюсах и минусах разработки веб-приложений на Node.js. Или посмотрите видео:

Взаимодействие между фронтендом и бэкендом

Фронтенд не существует изолированно от бэкенда или сервера, который фактически предоставляет данные. Давайте посмотрим на это с точки зрения клиент-серверного взаимодействия.

Асинхронные операции

В 2010-х годах одной из основных концепций, описывающих работу интернета, был AJAX. Термин расшифровывается как «асинхронный JavaScript и XML» (от англ Asynchronous JavaScript and XML). По сути, это комплекс средств для создания веб-сайтов и веб-приложений. Давайте рассмотрим ключевые компоненты, из которых состоит AJAX.

**Асинхронность**— фундаментальный аспект всего инструментария. Ключевая роль подхода заключается в асинхронном обновлении веб-контента. Это означает, что браузер не перезагружает всю веб-страницу целиком, когда изменений требует небольшая часть контента на странице. Например, не нужно перезагружать страницу со всеми продуктами, чтобы применить фильтр к результатам поиска. Вам просто нужно, чтобы результаты обновились.

**JavaScript**— JavaScript использует процессы автоматизации сайтов, поэтому разработчикам не приходится отдельно редактировать каждый процесс, отображаемый на странице. Он используется, в частности, для создания, добавления и управления динамическим контентом веб-сайта. После обработки всех процессов HTML и CSS JavaScript запускает обновления в реальном времени, пока посетитель просматривает страницу с интерактивным контентом.

**XML**(или «расширяемый язык разметки», от англ **Extensible Markup Language**) позволяет передавать данные, размещенные на странице, в браузеры, которые ее просматривают. Это важно для определенных систем, которые зачастую не умеют взаимодействовать с данными, отформатированными другой системой. В этом случае XML позволяет избежать несовместимости при сохранении данных в текстовом формате между XML-тегами. Таким образом, XML предлагает средства хранения, перемещения и обмена данными, которые не зависят от какой-то одной программной или аппаратной системы. Поскольку XML устаревает, уступая новым форматам данных, можно запускать AJAX с JSON, который является более коротким и удобным для чтения людьми.

На сегодняшний день AJAX не так часто обсуждается во фронтенд-сообществе из-за того, что асинхронный рендеринг веб-страницы входит в стандартную комплектацию всех основных фронтенд-фреймворков, которые мы обсуждали выше.

**Как работают асинхронные обновления**. Предположим, пользователь нажимает на кнопку. Клиент в фоновом режиме отправляет запрос серверу в формате XML/JSON, пока пользователь продолжает просматривать страницу. Серверная сторона получает данные от JavaScript, при необходимости обращается к базе данных и обрабатывает данные. Данные в формате XML/JSON отправляются обратно на исходную страницу на стороне клиента, которая выполнила запись. Коллбэк, выполняемый JavaScript, собирает данные и обновляет компонент веб-страницы, который требует изменений.

***REST и GraphQL***

REST расшифровывается как «передача репрезентативного состояния» (от англ Representational State Transfer). По сути, это упрощенный архитектурный стиль, применяемый для обмена сообщениями между клиентом (фронтенд) и сервером (бэкенд). Службы RESTful и API — это веб-службы, соответствующие архитектуре REST.

Например, разработчик должен разработать веб-приложение, которое показывает всех друзей в социальных сетях в определенном порядке. Фронтенд может создавать запросы к RESTful API Facebook для просмотра списка друзей и передачи этих данных обратно. Любая служба, использующая RESTful API, похожа на этот общий процесс, разница только в том, что данные извлекаются и возвращаются. REST — это простой набор руководств и практик, устанавливающих правила взаимодействия с веб-службой.

Основная идея REST заключается в том, что серверу безразлично, что происходит с клиентом. REST-сообщения содержат всю необходимую информацию для сервера, чтобы вернуть необходимые данные и забыть об этой операции. Узнать больше о различных форматах обмена сообщениями и более старом стандарте SOAP можно в [этой статье](https://www.altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/).

GraphQL представляет современную версию обмена сообщениями. Это язык запросов к базам данных из клиентских приложений. GraphQL определяет на стороне сервера, как отображать данные клиенту, и обслуживает различные типы клиентов и их потребности в данных. Чтобы узнать больше на эту тему, читайте [статью](https://www.altexsoft.com/blog/engineering/graphql-core-features-architecture-pros-and-cons/) о возможностях ядра GraphQL.

Больше практик, которые используются во фронтенд-разработке

Итак, мы обозначили базовые составляющие фронтенд-разработки. Однако есть еще много терминов и понятий, в которых фронтенд-разработчик должен разбираться.

***Отзывчивый дизайн***

Поскольку все больше людей пользуются интернетом с мобильных устройств, а не с настольных компьютеров, обязательной характеристикой веб-приложений стала отзывчивость дизайна. Отзывчивый дизайн означает, что макет приложения (включая функциональность и контент) подстраивается под размер экрана и вид устройства.

Например, когда сайт посещается с настольного компьютера с большим монитором, дизайн включает несколько колонок, тяжелую графику и пользовательский интерфейс, разработанный с учетом использования мыши и клавиатуры. На мобильных устройствах контент того же сайта отображается в одну колонку, сайт адаптирован для сенсорного взаимодействия, но содержит те же базовые данные. Чтобы узнать об отзывчивости больше, читайте [эту статью](https://www.altexsoft.com/blog/mobile/making-your-ecommerce-site-mobile-friendly-and-responsive/).

***Доступность и инклюзивность***

Говоря о доступности, мы имеем в виду, что сайт должен быть доступен как можно большему количеству людей с различными видами нарушений, таких как: нарушения зрения, когнитивных функций, слуха или подвижности. Этот термин также охватывает вопросы права, стандартов соответствия, различные мобильные устройства и различные типы сетевых подключений. Подробнее на эту тему можно прочитать [в статье](https://www.altexsoft.com/blog/uxdesign/reach-your-audience-with-accessible-and-inclusive-design/) «Как создавать интерфейсы, которые принесут пользу всем: тестирование доступности и принципы инклюзивного дизайна».

***Заключение***

**HTML**(Hypertext Markup Language) — это язык разметки для создания веб-сайтов. Обычно он применяется для структурирования веб-документа.

**CSS**(Cascading Style Sheets) — это язык таблиц стилей, который используется для стилизации HTML-элементов на веб-странице. Область действия CSS включает в себя дизайн, макет и варианты отображения для различных устройств и размеров экрана.

**DOM**(Document Object Model) — это программный интерфейс для HTML- и XML-документов. Он содержит информацию о странице, чтобы программы могли изменять структуру, стиль и содержимое документа.

**JavaScript**— сценарный язык. Во фронтенде он используется для придания веб-страницам динамики. Кроме того, он предоставляет фулстек-технологии с библиотеками и фреймворками для написания скриптов как на стороне клиента, так и на стороне сервера.

**AJAX.**AJAX (Asynchronous JavaScript and XML) — это комплекс методов для создания веб-сайтов и веб-приложений с динамически загружаемым контентом без создания новой записи данных.

Конечно, есть еще много других технологий, навыков и общих знаний, которые предстоит освоить новичку. Тем не менее, мы надеемся, что эта статья проложила тропинку в лес фронтенд-разработки, чтобы путник не сбился с пути.

**Вопросы для самостоятельного изучения**

1. Особенности применения HTML.

2. Где до сих пор используется HTML?

3. Проблемы усложнения web-ресурсов.

4. Потребительские оценки построения сайтов.

5. Перечислите основные фреймворки и библиотеки, их особенности.