# Обзор web-технологий рынка IT.

В последнее время наблюдается бурный рост рынка IT: за последние несколько лет количество вакансий на рынке IT-труда увеличилось на 40% (!).

Рост вакансий на рынке IT-труда связан со следующими показателями:

* **Рост и развитие самих компаний** (причем, рост и развитие – это разные показатели, которые рассмотрим позже). Т.е. открываются новые вакансии под новые web-технологии.
* **Текучка кадров**. В сравнении с другими рынками, на рынке IT – самая большая текучка кадров.

Текучка кадров бывает двух видов:

* **Лучшие специалисты** уезжают на ПМЖ в Москву, Европу, США. Это утечка продуктивных программистов, или, как раньше было модно говорить, утечка “мозгов”.
* IT-компаниям под новый проект либо под замену мигрировавшим специалистам, проще найти новых специалистов на рынке труда, чем переобучать своих. Но в такой сложившейся ситуации виноваты сами программисты этих компаний, которые вскоре пополнят ряды **безнадежных безработных**. Именно в силу свой высокой квалификации, они не могут пересилить себя, согласиться на **добровольное отречение от устаревших знаний**, которыми они владеют.

Следует различать два вида программирования, а соответственно и программистов, занятых в этой сфере:

* **Репродуктивное программирование**. Это люди, которые закончили какие-нибудь курсы, получили сертификат по изученной технологии, сидят на готовых продуктах и работают только в русле однажды изученной технологии. Это копирайтеры или пользователи. Они могут быть хорошими специалистами, но работают в узко-специализированной области.
* **Продуктивное программирование**. Это специалисты, которые постоянно нацелены на изучение новых технологий, поиск новых решений. При необходимости могут стать специалистами практически в любой необходимой области, но нацелены на создание продукта. Это разработчики.

Компании находятся в постоянном поиске и тех и других специалистов. Но если репродуктивные программисты – это программисты одного проекта (узконаправленные специалисты ищутся под конкретный заказ/проект), то продуктивные – вскоре становятся незаменимыми сотрудниками в любом проекте.

Отличительная черта продуктивных программистов – способность к развитию и саморазвитию. Они обретают *способность решать всё более сложные самоорганизационные задачи, что выражается, прежде всего, возрастающей сложностью используемых для этого мыслительных моделей (web-технологии, паттерны проектирования)*. Причем, они не только саморазвиваются, но и подталивают к развитию компанию.

**Развитие** – это способность адекватно реагировать на изменения во внутренней и внешней среде, самоизменяться, реорганизовываться на ходу на основе системного анализа. Развитие (в отличии от роста) предполагает постоянное изменение формы, содержания, сущности, качества любого явления.

Так рост компании не подразумевает её развитие: можно нанять 100 человек, которые будут делать одни и те же механические дейтсвия, и это будет ростом компании, т.к. увеличивается её численность. Можно нанять одного продуктивного программиста, который одним технологическим решением, сможет заменить 100 человек, и это будет развитием компании.

Следует помнить, что **в IT один год равен пяти**. Это значит, что какими бы мы крутыми специалистами на данный момент не были, через несколько лет наши знания устареют. Если мы за это время не успеем перестроиться, получить новые навыки, то останемся в неудел. Но для того чтобы перестраиваться, нужно знать куда.

Чтобы **определить направление этого рынка** нужно обладать развитой интуицией и хорошими аналитическими способностями.

Задача данного семинара – обзор текущего состояния рынка IT. И прогноз востребованных технологий на ближайшие 3-4 года.

Стремимся к **мозаичному типу знаний**.

FrontEnd, BackEnd и смежные технологии

У хороших сайтов есть 2 версии разработки:

* Для разработчиков, или **Develop**. Медленнее работает, но проще отлаживать, с выводом всех ошибок, несжатые файлы.
* Для пользователей, или **Production**

2 направления: frontend и backend. И 3 группы технологий (для backend, для frondend и общий).

**Инструментарий**

1. IDE: NetBeanse (+PHPDoc), Adobe Breackets, PHPStorm, Sablime Text 2, и WebMatrix компании Microsoft
2. OpenServer
3. NotePad++
4. FireFox, Firebug
5. Git – система контроля версий, github.com или bitbucket.org
6. Composer – менеджер зависимостей
7. Photoshop

**FrontEnd:**

1. Гибкая блочная верстка.
2. Резиновая и фиксированная верстка, традиционная блочная и табличная
3. Адаптивная верстка.
4. JSON
5. Селекторы.
6. архитектурный шаблон MVVM
7. JavaScript, Node.js
8. JavaScript и библиотеки jQuery (библиотека запросов)
9. Backbone – MVVM (ModelView-ViewModel) библиотека. Backbone требует Underscore.js и jQuery. Если они не нужны можно использовать Exoskeleton – форк Backbone, где никаких зависимостей не нужно. Данную библиотеку рационально использвать в крупном и среднем проекте, который работает со множеством данных,
10. Angular и KnockoutJS.
11. BackBone-фрэймворк – Marionette.js (модульный фрэймворк)
12. CoffeeScript
13. Less: переменные, миксины или функции, расширения, импорт, вложенность, соединение в одно свойтсво несколько свойств.
14. Sass: переменные, вложенность, импорт файлов, миксины или функции, логические операции. Фрэймворк Compass (compass-style.org) на основе Sass.
15. Live Editor, CKEditor
16. Системы сборки FrontEnd-а.
17. Jade
18. MongoDB
19. HTML5 и API HTML5: видео и аудио, холст, перетаскивание, работа с файлами, геолокация (в том числе работа с GOOGLE и YANDEX-картами), web-хранилища, взаимодейтсвие с сервером. Вспомогательный инструментарий.
20. Bootstrap 3, API Bootstrap
21. Браузерные web-консоли. В частности FireBug для FireFox.
22. Основы SEO.
23. Schema.org, генератор shema.org микроформат данных, микроданные

**BackEnd**

1) Языки программирования:

* **PHP +**
* **Java +**
* Pithon -
* Perl -
* Ruby -
* Asp.net -

2) Сервера (это ПО, получающая запросы от клиента, либо компьютеры, где это ПО находится):

3) СУБД (это ПО предназначенное для работы с базами данных):

* MySQL (PHPMyAdmin)
* NoSQL, SQL-lite
* Oracle
* MS-SQL

4) Понятие потернов. MVC, HMVC. Маршрутизация. Взаимодействие с .htaccess.

5) ООП.

6) Один, лучше 2-3 фрэймворка. Для PHP – это Kohana, YII, ZendFramework, Symfony. Доктрина (для работы с запросами). Symfony – самый крутой. Falcon – расширение для PHP, написано на C++, чтобы поставить необходим выделенный сервер, или просить хостинговую компанию.

7) CMS Drupal и 1Cбитрикс (для PHP) MoDex. Joomla, WordPress, Magenta

8) Шаблонизатор twig

10) Системы контроля вервсий: GIT. Mercurial, SVN, Subversion. Сайты github и bitbucket

11) Менеджер зависимостей: Composer.

12) Регулярные выражения, Строковые функции

13) Платежные системы.

14) Парсинг. CURL

15) Unit Test

16) Умение делать аудит, анализ и консультирование проектируемых и уже существующих сайтов. Ajile-технологии разработки проектов. Умение разрабатывать проекты без технического задания.

17) Основы хакинга.

**PHP-инклюд** — вставка исполняемых хакерских php-скриптов на взломанном сайте/сервере. *Локальный* (php-скрипты находятся на том же сервере, что и взломанный сайт) *Удаленный* (подключение php-скриптов через протокол http://).

**SQL-инъекция**

**XSS** (Cross-Site Scripting или межсайтовый скриптинг) — это вставка html/JavaScript/VBScript-кода в сгенерированные сервером страницы. Пассивный — срабатывает только при передаче пользователю сформированной хакером ссылке. Активный — непосредственная вставка html-кода в сгенерированную страницу.

**CRLF-инклюд** — подделка сообщений в сетях и чатах.

Сервера поддерживают **две модели мультипроцессорной обработки**:

1) **Мультипроцессорный** поток. Для каждого запроса выделяется отдельный процесс, продолжающийся до тех пор, пока запрос не будет обслужен. Под каждый запрос создаются дочерние процессы. Недостаток: каждый процесс расходует память.

2) **Мультипрограммный** поток. Для каждого запроса выделяется отдельный программный поток. Такой подход эффективнее, т.к. требует меньшего расхода памяти.

**Однопоточное** приложение. В приложениях Node.js используется иной подход: создается единственный программный поток, но работающий в асинхронный режиме.

**Язык логики**

Существует только один программный язык: **язык логики**, все вышеперечисленные технологии являются частными случаями. В ходе развития IT что-то может появляться новое, что-то устаревает. Сами потребности задачи отсеивают ненужное технологическое нагромождения и приводят к новому.

Важно, для самообразования использовать не только Интернет, а задействовать как можно больше органов восприятия (ведение конспекта, схематизация, пересказ и т.д.).

Наряду с инструментальными технологиями, необходимо задействовать **мыслетехнологии**:

* Извлечение информации.
* Запоминание информации.
* Типологизация ситуаций, способов применения.
* Схематизации.
* Синтез целей.
* Обучаемость.
* Обобщение.
* Критика (самокритика и критика технологий)
* На последнем месте web-программирование.

Источниками информаци могут служить:

* Книги
* Периодические издания, журналы, газеты, статьи
* TV
* Радио
* Телефон
* Курсы
* Конференции
* Личное общение
* Интернет и местные сети.

Причем, важно, чтобы было задействована как можно больше источников информации.

**Аудит web-сайта**

Существует множество технологий, множество языков, фрэймворков, CMS… Что же выбрать?

Если делаем проект **для себя**, то ставим перед собой следующие вопросы:

* На чем бы я делал проект, если бы знал все вышеперечисленные технологии?
* На чем бы я делал проект, если бы не знал ни одной из вышеперечисленных технологий?
* На чем бы я делал проект, если бы знал только те технологии, которые сейчас реально знаю?

Начинаем разрабатывать проект только в том случае, если на все три вопроса отвечаем одинаково. Если по-разному, то необходимо определить готовы ли мы потратить определенное количество времени на изучение новой технологии.

Самое важное в рабоет с заказчиком – это сделать заказчика счастливым. Аудит делаем с заказчиком. Для этого необходимо подготовить опросник заказчика:

* Любая работа начинается с изучения тезауруса (гнездовой словарь основных понятий и терминов) заказчика.
* Имеются ли у него готовые web-ресурсы? Если да, на чем они сделаны, кто делал?..
* Определиться с БД (база данных)
* Стоимость.
* Требования к железу.
* Перспективы развития.
* Персонал. Будет ли персонал поддерживать проект, либо будут наниматься новые сотрудники. Если проект будет поддерживаться своими силами, то необходимо узнать какими технологиями пользуется персонал.
* Безопасность.

Конечно, аудит сайта может делать независимый эксперт. В любом случае, **в каждый момент времени** должно быть понятно: **кто, что, когда, в какие сроки, за счёт каких финансовых и иных материальных ресурсов будет делать**.

У программистов свое мировоззрение: люди общаются с людьми, программисты – с программами.

**Изучение новых технологий**

На что в первую очередь следует обрадить внимание:

* Особенности синтаксиса
* Платформа настройка
* Hello World
* Работа с шаблонами
* Базы данных, хранение данных
* Регистрация авторизация
* Обработка изображений
* Формы, обработка данных из форм.
* Безопасность и защита отдельных страниц. Ролевой вход
* События
* Клиент-сервер

**Аналитика контента**. Подразумевается умение выделять области сайта:

1. Предметная область, предметика.
2. Кокретика.
3. Для потребителя. Контакты и т.д.

Проект может стать успешным только в том случае, если **хороший менеджер управляет хорошей командой**.

Плохой менеджер + плохая команда = провальный проект

Плохой менеджер + хорошая команда = провальный проект.

Хороший менеджер + плохая команда (такое в принципе не возможно (!))