

Web  is a resources network.

The Request-URI is a Uniform Resource Identifier (URI) for searching and identifies the resource upon which to apply the request.

A web browser is an example of a web client.

When running a web client application the browser ﬁrst makes a request to the web server.

  The web server will then respond with the Javascript client including the resources needed to run it. Once the client is transferred it will be initialized and it will be ready to be run. When the user interacts with the client, such as clicking on graphical elements, this may require new data to be fetched from the server.

Instead of reloading the entire page the client will instead make a small request to the API on the web server. The API is simply an interface that allows the clients to communicate with the server in a data format that is easy to understand for both the client and server.

So, the interaction between a client and a Web server is made by means HTTP messages exchanging:

 - client request to the server;

 - server response to the client.

Server receives request from the client and process it. Client receives the response and displays the results to the user.

Сеть - это сеть ресурсов.

Request-URI - это унифицированный идентификатор ресурса (URI) для поиска и идентифицирует ресурс, к которому применяется запрос.

 Веб-браузер является примером веб-клиента.

При запуске приложения веб-клиента браузер сначала отправляет запрос веб-серверу.

  Затем веб-сервер ответит клиентом Javascript, включая ресурсы, необходимые для его запуска. Как только клиент передан, он будет инициализирован и готов к запуску. Когда пользователь взаимодействует с клиентом, например, щелкая графические элементы, это может потребовать получения новых данных с сервера.

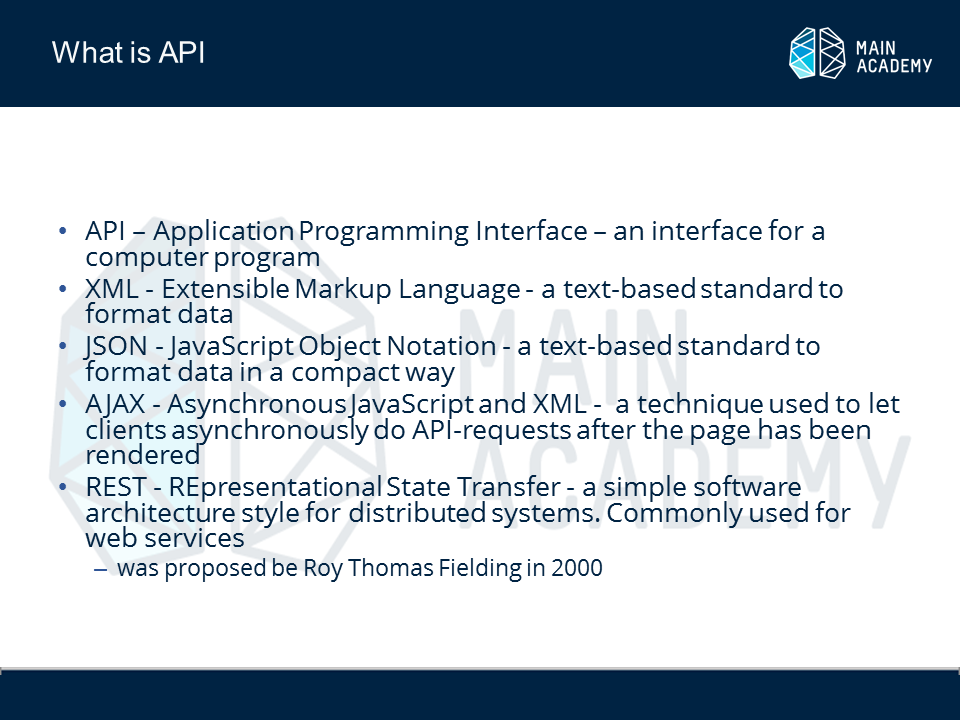
Вместо перезагрузки всей страницы клиент вместо этого сделает небольшой запрос к API на веб-сервере. API - это просто интерфейс, который позволяет клиентам взаимодействовать с сервером в формате данных, который легко понять как для клиента, так и для сервера.

Итак, взаимодействие между клиентом и веб-сервером осуществляется посредством обмена HTTP-сообщениями:

 - запрос клиента на сервер;

 - ответ сервера клиенту.

Сервер получает запрос от клиента и обрабатывает его. Клиент получает ответ и отображает результаты пользователю.



     The API describes how data is being transferred, what data format is being used and what methods are available. One of the most common ways of letting the client asynchronously communicate with the server is via AJAX.

    With the traditional approach HTML are rendered with the current content of the database. When data is changed the page must be reloaded to force the HTML to be re-generated from scratch. Even for sites using AJAX to dynamically change parts of the page this interaction is the most common case. On the other hand, an SPA allows a more ﬂexible and elegant way of dealing with data. Once the user has loaded an initial slim version of the site all the data can be fetched asynchronously.

To enable the server to push data to the client other techniques can instead be used, e.g. the websocket API in the HTML5 draft which supports bi-directional communication. This will ease the load of the server and allows a faster and more responsive system.

  When using a HTTP-based data transportation some kind of architectural structure is needed. The architecture describes how messages are formed so that both parties understand each other, e.g. what methods are available and how data is transferred. It also often includes additional information about how authentication is handled, as well as the ﬂow of communication. Common architectural structures are REST (was proposed be Roy Thomas Fielding in 2000 ) and SOAP. The data format of a message describes the syntactical form of the objects that are transferred. Common data formats used today are JSON, XML and plain text.

 The API that adhere to the REST architecture is called RESTful API.

API описывает, как данные передаются, какой формат данных используется и какие методы доступны. Один из самых распространенных способов асинхронной связи клиента с сервером - через AJAX.

   При традиционном подходе HTML отображается с текущим содержанием базы данных. Когда данные изменяются, страница должна быть перезагружена, чтобы заново генерировать HTML.

Даже для сайтов, использующих AJAX для динамического изменения частей страницы, такое взаимодействие является наиболее распространенным случаем.

С другой стороны, SPA обеспечивает более гибкий и элегантный способ работы с данными. После того, как пользователь загрузил исходную тонкую версию сайта, все данные могут быть получены асинхронно. Чтобы дать возможность серверу передавать данные клиенту, вместо этого могут использоваться другие методы, например, API websocket в проекте HTML5, который поддерживает двунаправленную связь.

Это облегчит загрузку сервера и позволит создать более быструю и более отзывчивую систему.

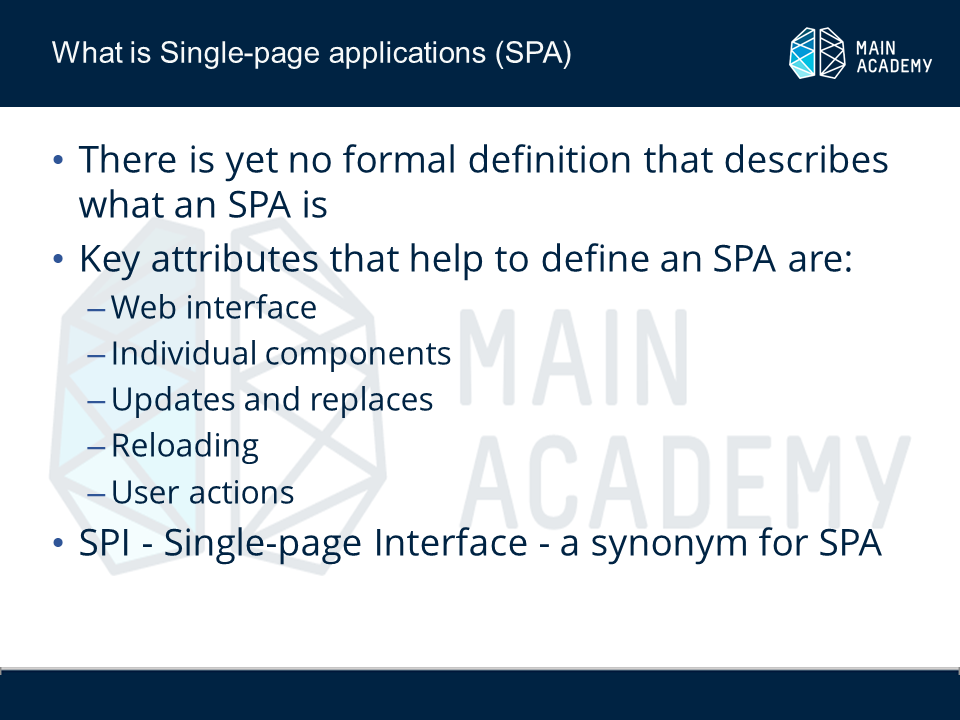
 При использовании передачи данных на основе HTTP необходима какая-то архитектурная структура.

Архитектура описывает, как сообщения формируются так, чтобы обе стороны понимали друг друга, например, какие методы доступны и как данные передаются.

Он также часто содержит дополнительную информацию о том, как обрабатывается аутентификация, а также о потоке связи.

Общими архитектурными сооружениями являются REST (был предложен Роем Томасом Филдингом в 2000 году) и SOAP.

Формат данных сообщения описывает синтаксическую форму передаваемых объектов. Обычные форматы данных, используемые сегодня, это JSON, XML и простой текст.  API, который придерживается архитектуры REST, называется RESTful API.



  According to the msdn.microsoft.com:

  Single Page Applications  (SPAs) are Web  apps  that load a single HTML page  and dynamically update that page  as the user interacts with the app.

SPAs use AJAX and HTML5 to create fluid and responsive Web apps , without constant page  reloads. However, this means much of the work happens on the client side, in JavaScript.

  There are five key attributes that describe what is SPA:

1) Web interface – used on the web, focuses on user interface.

2) Individual components – it is divided into smaller components that cope with each other.

3) Updates and replaces – a component can at any time be changed or replaced with another component.

4) Reloading – the entire page is never reloaded even though new content may be loaded into some sections.

5) User actions – the SPA is responsible for handling user actions such as input from mouse and keyboard.

По сообщению msdn.microsoft.com:

Одностраничные приложения (SPA) - это веб-приложения, которые загружают одну HTML-страницу и динамически обновляют эту страницу по мере взаимодействия пользователя с приложением.

SPA используют AJAX и HTML5 для создания гибких и отзывчивых веб-приложений без постоянной перезагрузки страницы. Однако это означает, что большая часть работы происходит на стороне клиента, в JavaScript.

Есть пять ключевых атрибутов, которые описывают, что такое SPA:

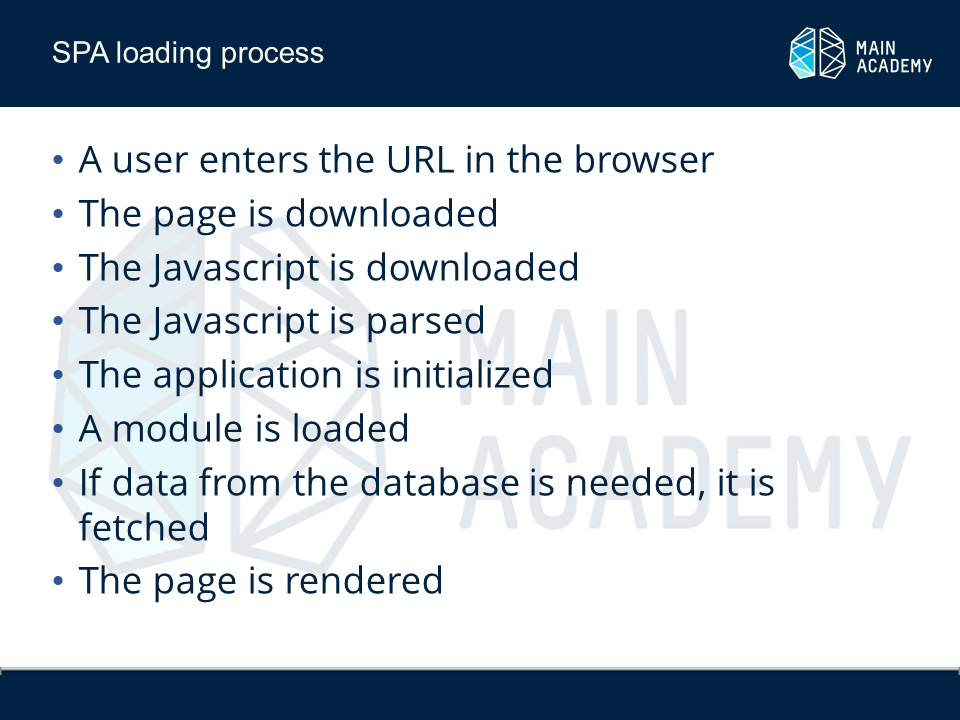
1) Веб-интерфейс - используется в Интернете, ориентирован на пользовательский интерфейс.

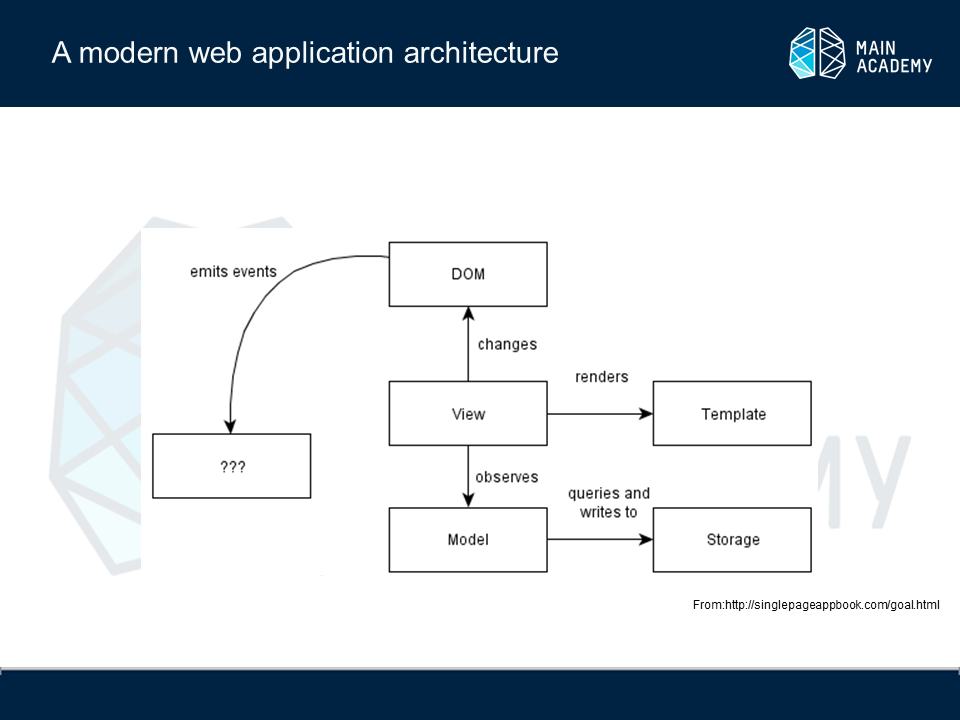
2) Отдельные компоненты - это делятся на более мелкие компоненты, которые справляются друг с другом.

3) Обновления и замены - компонент может быть в любое время заменен или заменен другим компонентом.

4) Перезагрузка - вся страница никогда не перезагружается, даже если в некоторые разделы может быть загружено новое содержимое.

5) Действия пользователя - SPA отвечает за обработку действий пользователя, таких как ввод с помощью мыши и клавиатуры.

****

****

**Write-only DOM**. No state / data is read from the DOM. The application outputs HTML and operations on elements, but nothing is ever read from the DOM. Storing state in the DOM gets hard to manage very quickly: it is much better to have one place where the data lives and to render the UI from the data, particularly when the same data has to be shown in multiple places in the UI.

**Models as the single source of truth**. Instead of storing data in the DOM or in random objects, there is a set of in-memory models which represent all of the state/data in the application.

**Views observe model changes** . We want the views to reflect the content of the models. When multiple views depend on a single model (e.g. when a model changes, redraw these views), we don't want to manually keep track of each dependent view. Instead of manually tracking things, there is a change event system through which views receive change notifications from models and handle redrawing themselves.

**Decoupled modules that expose small external surfaces** Instead of making things global, we should try to create small subsystems that are not interdependent. Dependencies make code hard to set up for testing. Small external surfaces make refactoring internals easy, since most things can changes as long as the external interface remains the same.

**Minimizing DOM dependent-code**. Why? Any code that depends on the DOM needs to be tested for cross-browser compatibility. By writing code in a way that isolates those nasty parts, a much more limited surface area needs to be tested for cross-browser compatibility. Cross-browser incompatibilities are a lot more manageable this way. Incompatibilities are in the DOM implementations, not in the Javascript implementations, so it makes sense to minimize and isolate DOM -dependent code.

Single page apps need a better word, because they have more complex state transitions than a server-side app:

- there are DOM events that cause small state changes in views;

- there are model events when model values are changed;

- there are application state changes that cause views to be swapped;

- there are global state changes, like going offline in a real time app;

- there are delayed results from AJAX that get returned at some point from backend operations.

Только для записи DOM. Состояние / данные не читаются из DOM. Приложение выводит HTML и операции над элементами, но из DOM ничего не читается. Хранение состояния в DOM становится очень сложным для управления очень быстро: гораздо лучше иметь одно место, где живут данные, и визуализировать пользовательский интерфейс из данных, особенно когда одни и те же данные должны отображаться в нескольких местах пользовательского интерфейса.

  Модели как единственный источник правды. Вместо хранения данных в DOM или в случайных объектах существует набор моделей в памяти, которые представляют все состояние / данные в приложении.  Виды наблюдают за изменениями модели. Мы хотим, чтобы взгляды отражали содержание моделей. Когда несколько видов зависят от одной модели (например, при изменении модели перерисовывают эти виды), мы не хотим вручную отслеживать каждый зависимый вид.

Вместо ручного отслеживания вещей существует система событий изменений, посредством которой представления получают уведомления об изменениях от моделей и обрабатывают их перерисовку. Разъединенные модули, которые открывают маленькие внешние поверхности.

Вместо того чтобы делать вещи глобальными, мы должны попытаться создать маленькие подсистемы, которые не являются взаимозависимыми.

Зависимости затрудняют настройку кода для тестирования. Небольшие внешние поверхности облегчают рефакторинг внутренних устройств, поскольку большинство вещей может меняться, если внешний интерфейс остается неизменным.

Минимизация DOM-зависимого кода. Почему? Любой код, который зависит от DOM, должен быть протестирован на совместимость с различными браузерами.

За счет написания кода таким образом, чтобы изолировать эти неприятные части, необходимо проверить гораздо более ограниченную площадь поверхности на совместимость с различными браузерами. Кроссбраузерная несовместимость намного более управляема таким образом.

Несовместимости есть в реализациях DOM, а не в реализациях Javascript, поэтому имеет смысл минимизировать и изолировать DOM-зависимый код.

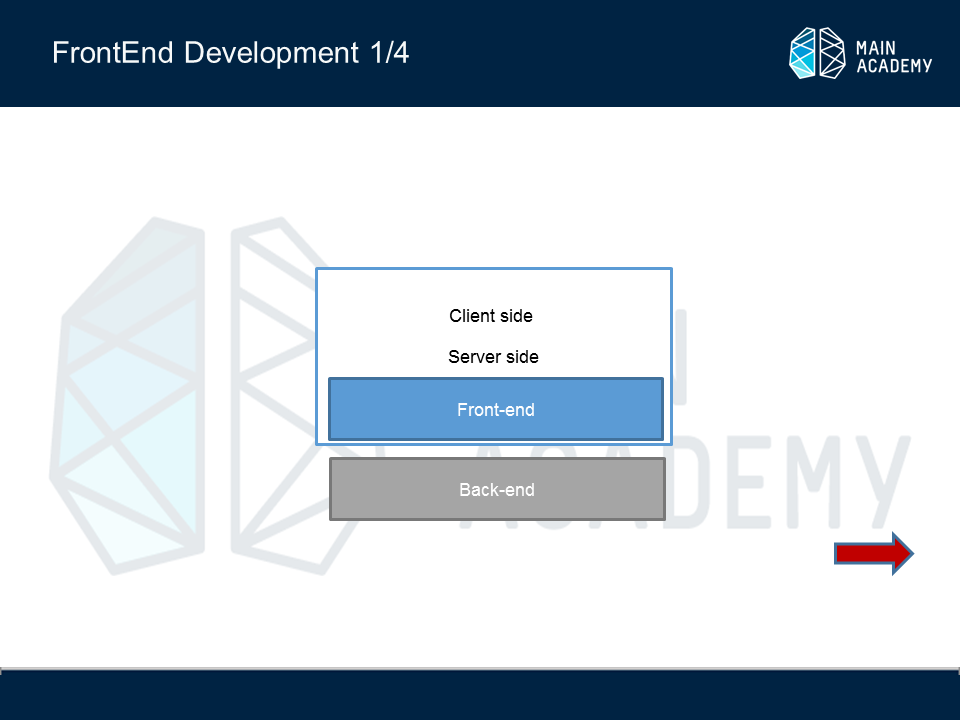
 Одностраничным приложениям нужно лучшее слово, потому что они имеют более сложные переходы состояний, чем серверные приложения: - есть события DOM, которые вызывают небольшие изменения состояния в представлениях;

- при изменении значений модели происходят модельные события;

- есть изменения состояния приложения, которые вызывают обмен местами;

- происходят глобальные изменения состояния, такие как переход в автономный режим в приложении реального времени;

 - есть задержанные результаты от AJAX, которые возвращаются в какой-то момент из бэкэнд-операций.

****

  The front-end is the part that users of the website can see and interact with.

So, front-end includes the design, images, colors, buttons, forms, typography, animations and content. It’s basically everything that you as a user of the website can see. The User Interface (UI), the animations, and usually a bunch of logic to talk to the backend are visual bits that the user interacts with.

As in every session two sides client and server interact so front-end is divided into two parts:

    - client-side;     - server-side.

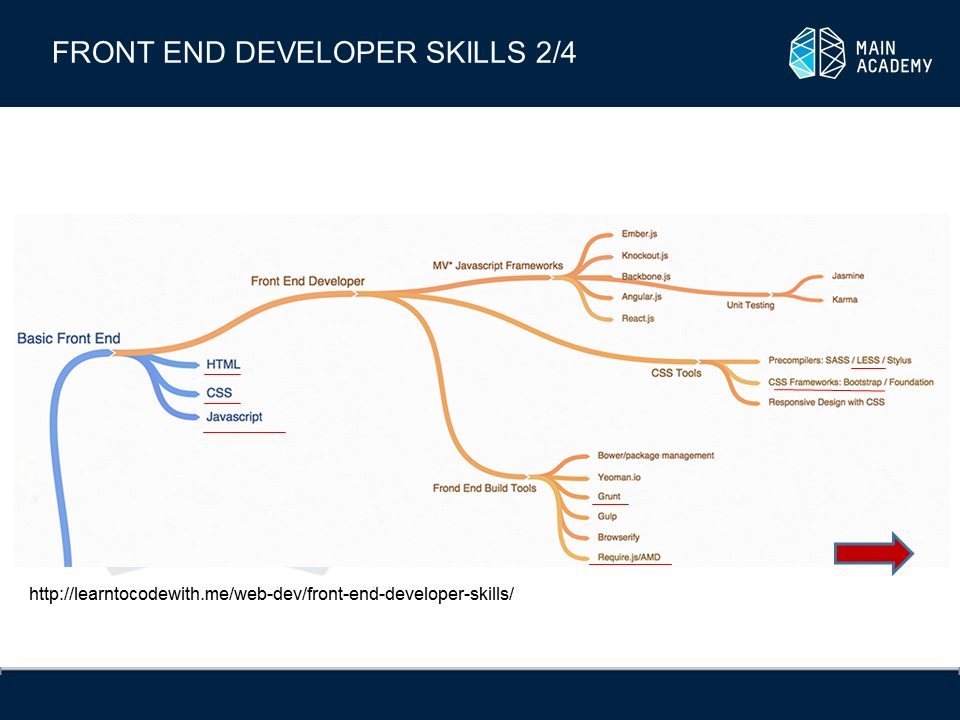
Внешний интерфейс - это та часть, с которой пользователи сайта могут видеть и взаимодействовать.

Итак, интерфейс включает в себя дизайн, изображения, цвета, кнопки, формы, типографику, анимацию и контент.

Это в основном все, что вы, как пользователь сайта, можете видеть.

Пользовательский интерфейс (UI), анимация и, как правило, связка логики для взаимодействия с бэкэндом - это визуальные биты, с которыми взаимодействует пользователь.

Как и в каждом сеансе, между клиентом и сервером взаимодействуют две стороны, так что интерфейс делится на две части:     - сторона клиента; - на стороне сервера.

****

   On this slide a front-end developers skills are generalized.

 Conditionally they can be divided into main four parts:

HTML, CSS, JavaScript - the main purpose of which is that a front-end developer architects and develops websites and applications.

MV\* JAVASCRIPT FRAMEWORKS

These help you organize code:

  - ember.js is used Yahoo! Twitch.tv Groupon

  - knockout.js is open source two but it is made by Microsoft’s emploee

- angular.js is made by Google

- backbone.js made by Jeremy Ashkenas

- react.is made by Facebook

 Unit Testing helps developers write tests for applications.

CSS TOOLS

Compared to frameworks and unit testing above, the following CSS tools are fairly easy to learn.

3.1 Precompilers.

Using a precompiler (or preprocessor) has a lot of advantages, like making code cleaner, maintaining organization, and better following DRY principles. Popular CSS precompilers are as Sass, Less, and Stylus. Only learn one.

.2 CSS frameworks. These help optimize your workflow with built-in grids and other CSS components.

Two popular examples are:  - Bootstrap (made by Twitter)  - Foundation (made by ZURB ).

3.3 Responsive design.

 Responsive design means being able to build sites that work on all screen sizes—desktop, tablet and mobile.

Responsive design is intrinsic to frameworks like Bootstrap and Foundation. Meaning if you learn one of those frameworks, you’re good to go.

4. FRONT END BUILD TOOLS

These relate to file size optimization and workflow efficiency.

4.1 Package Management. This is <https://bower.io/> When projects get large, it can be tough organizing all the libraries, assets, and so forth. Bower helps with managing all the parts.

4.2 Yeoman.io. <https://yeoman.io/>  is a scaffolding tool. It helps get you started quickly with projects, and encourages you to adopt best practices. This is run in the command line.

4.3 Task Runners. Such as <https://gruntjs.com/>  or <https://gulpjs.com/> . They run on the command line and compress files. Also act as a compiler for Sass or Less. They also have a wide range of plugins boasting other features.

4.4 Javascript Module Loader. Like <https://requirejs.org/>  and <http://browserify.org/> —these bundle up dependencies.

На этом слайде обобщены навыки фронтенд-разработчиков.

 Условно их можно разделить на четыре основных части:

HTML, CSS, JavaScript - основная цель которого заключается в том, что передовой разработчик разрабатывает и разрабатывает веб-сайты и приложения.

 РАМКИ MV \* JAVASCRIPT

Это поможет вам организовать код:

  - ember.js используется Yahoo! Twitch.tv Groupon

  - knockout.js с открытым исходным кодом два, но он сделан сотрудниками Microsoft

 - angular.js сделан Google

 - backbone.js, сделанный Джереми Ашкенасом

- response.is сделан Facebook

 Модульное тестирование помогает разработчикам писать тесты для приложений.

CSS ИНСТРУМЕНТЫ

По сравнению с фреймворками и модульным тестированием, приведенными выше, следующие инструменты CSS довольно просты в освоении.

3.1 Прекомпиляторы.

Использование прекомпилятора (или препроцессора) имеет много преимуществ, таких как повышение чистоты кода, поддержание организации и улучшение следования принципам DRY. Популярными CSS-прекомпиляторами являются Sass, Less и Stylus. Только учиться одному.

.2 CSS-фреймворки. Они помогают оптимизировать рабочий процесс благодаря встроенным сеткам и другим компонентам CSS.

Два популярных примера: - Bootstrap (сделано в Twitter) - Foundation (сделано в ZURB).

 3.3 Адаптивный дизайн.

 Адаптивный дизайн означает возможность создавать сайты, которые работают на экранах всех размеров - на настольных компьютерах, планшетах и ​​мобильных устройствах.

Адаптивный дизайн присущ фреймворкам, таким как Bootstrap и Foundation. То есть, если вы изучите одну из этих платформ, вам будет хорошо.

 4. ПЕРЕДНИЕ КОНСТРУКЦИИ ИНСТРУМЕНТЫ

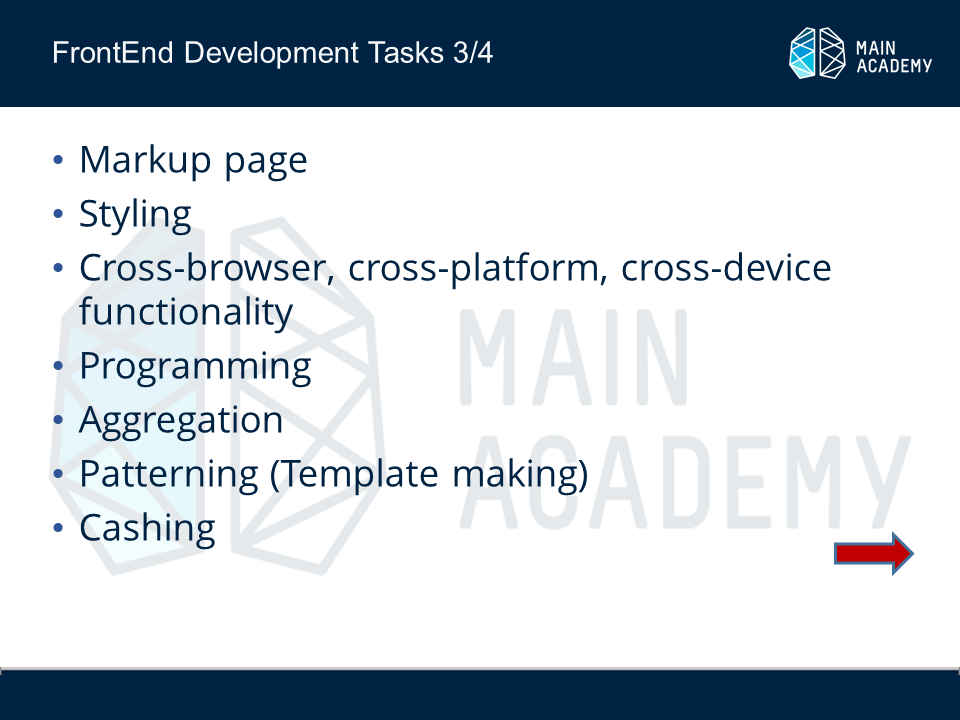
Они относятся к оптимизации размера файла и эффективности рабочего процесса.

 4.1 Управление пакетами. Это https://bower.io/ Когда проекты становятся большими, может быть сложно организовать все библиотеки, ресурсы и так далее. Бауэр помогает в управлении всеми частями.

 4.2 Yeoman.io. https://yeoman.io/ - это инструмент для создания лесов. Это помогает быстро начать работу с проектами и побуждает вас применять лучшие практики. Это запускается в командной строке.

 4.3. Бегуны. Например, https://gruntjs.com/ или https://gulpjs.com/. Они запускаются в командной строке и сжимают файлы. Также выступать в качестве компилятора для Sass или Less. У них также есть широкий спектр плагинов с другими функциями.

 4.4 Модуль загрузки Javascript. Например, https://requirejs.org/ и http://browserify.org/ - они объединяют зависимости.

****

**Markup** page – it means to create the structure of the page with markup language as HTML

**Styling with CSS**– it means to create the styles that lay out the page and give it both its unique visual flair and a clear, user-friendly view to allow readers, who never linger on pages we would like to think they do, some help to read or skim the contents quickly. Design means both how something looks and how something is structured, and in a good design, both come together.

3) **Cross-browser**, cross-platform, cross-device functionality. The browser on your computer is to remain the most advanced and feature-rich client

application to access the web for a long time, but that doesn't mean that 'snacking' with mobile browsing, or netbooks, etc.

4) **Programming** is a main part. There are not a front-end without it. Javascript is a major Web scenario language. It has fully grown up from inline commands embedded in html to full-blown asynchronous applications executed on the fly on the browser as unobtrusive rich functionality.

5) **Aggregation** — is data processing, arrived from a backend, in that look which is most convenient for the subsequent template making. For example, if the backend issues the demanded data by two separate methods, and you as a result need one list on the page, happens conveniently in advance to merge results of two methods to simplify a template. Or, for example, if the backend isn't able to sort in an order necessary to you or such orders at once a little, on a stage of aggregation also the problem of independent sorting of the obtained data can lay down.

6) **Patterning** - this is a creation of HTML from the obtained data. In different architecture templates happen different degree of complexity

7) **Cashing** – is the process of storing data in a temporary storage area. It not always problem of a frontend. It is possible to cache both ready HTML, and the answer from a backend by some popular  request.

Разметка страницы - это означает создание структуры страницы с языком разметки в виде HTML

Стилизация с помощью CSS - это означает создание стилей, которые размечают страницу и придают ей уникальный визуальный стиль и понятный, удобный для пользователя вид, позволяющий читателям, которые никогда не задерживаются на страницах, которые, как нам хотелось бы думать, они помогают быстро прочитать или просмотреть содержимое.

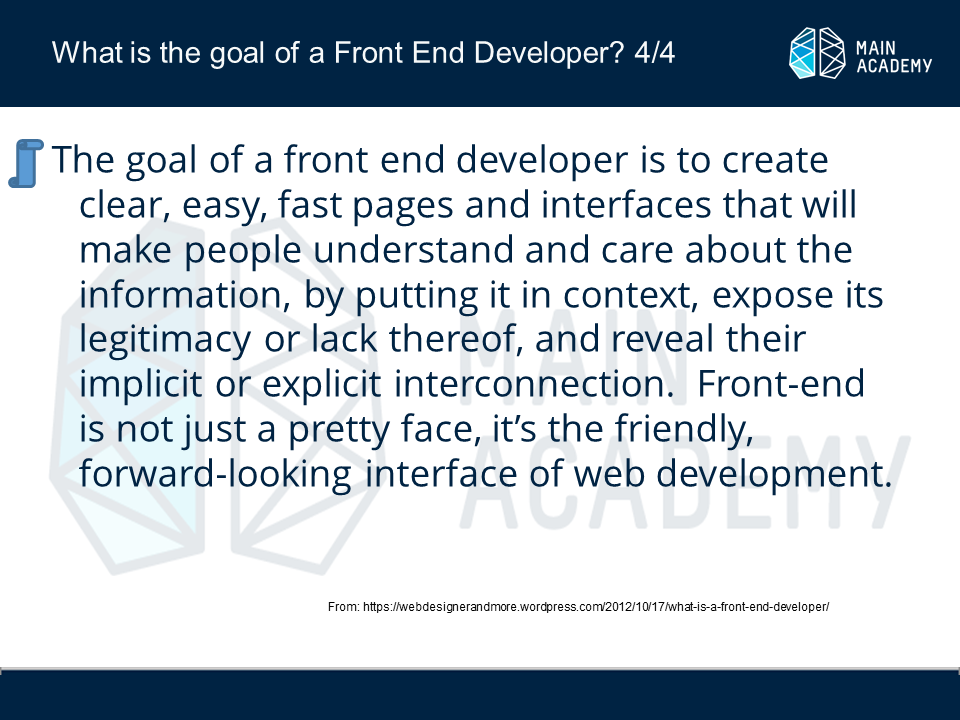
Дизайн подразумевает как то, как что-то выглядит, так и то, как что-то структурировано, и в хорошем дизайне оба сходятся.

 3) Кросс-браузер, кроссплатформенность, кросс-функциональность устройства. Браузер на вашем компьютере должен оставаться самым продвинутым и многофункциональным клиентским приложением для доступа в Интернет в течение длительного времени, но это не означает, что «перекусывать» с помощью мобильного браузера или нетбуков и т. Д.

4) Программирование является основной частью. Без него нет внешнего интерфейса. Javascript - это основной язык веб-сценариев. Он полностью вырос от встроенных команд, встроенных в html, до полноценных асинхронных приложений, выполняемых на лету в браузере, как ненавязчивая богатая функциональность.

 5) Агрегация - это обработка данных, полученных из бэкэнда, в том виде, который наиболее удобен для последующего создания шаблона. Например, если бэкэнд выдает требуемые данные двумя отдельными способами, а вам в результате нужен один список на странице, удобно заранее объединить результаты двух методов, чтобы упростить шаблон. Или, например, если бэкэнд не умеет сразу несколько сортировать в нужном вам порядке или таких заказах, на этапе агрегации может также ложиться проблема самостоятельной сортировки полученных данных. 6) Паттернинг - это создание HTML из полученных данных. В разных архитектурах шаблоны бывают разной степени сложности

 7) Обналичивание - это процесс хранения данных во временной области хранения. Это не всегда проблема фронтэнда. По некоторому популярному запросу можно кешировать как готовый HTML, так и ответ из бэкэнда.

****

Loading time is a major contributing factor to page abandonment.

The average user has no patience for a page that takes too long to load.   So, to make your SPA faster:

bundling javascript with require.js

; bundling css stylesheets with less

; optimizing images for size

; inlining images as dataurls

; build the final single html page containing the scripts and css bundles

; Optimize our production web server.

But a front-end is not only the interface of the client application. It is the solution of the client application in the context of the backend.

Время загрузки является основным фактором, способствующим отказу от страницы.

Обычный пользователь не терпит страницы, которая загружается слишком долго.

Итак, чтобы сделать ваш SPA быстрее:

связывание JavaScript с require.js  ;

объединение таблиц стилей CSS с меньшим ;

оптимизация изображений по размеру  ;

встраивание изображений в качестве dataurls  ;

создать последнюю HTML-страницу, содержащую сценарии и пакеты CSS  ;

Оптимизируйте наш производственный веб-сервер.

Но внешний интерфейс - это не только интерфейс клиентского приложения. Это решение клиентского приложения в контексте серверной части 

1. Graphic Editors are used for working with images.
2. PhotoShop is a raster graphics editor developed and published by Adobe Systems for Windows and OS X. It contains the fundamental tools, techniques and workflows for creating the web page. PhotoShop 30-Day Free Trial.
3. Adobe Photoshop Sketch is drawing app for the iPad that gives you natural drawing tools and custom brushes.

  2. Code Editors are used for writing code.

1. NotePad++ will be use for programming. Program will be run in browser.
2. An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger [https://en.wikipedia.org/wiki/Integrated\_development\_environment]. WebStorm is an IDE from JetBrains for client-side development. It is 30-Day Free Trial.
3. 3. Browser
4. A web browser (commonly referred to as a browser) is a software application for retrieving, presenting, and traversing information resources on the WWW.
5. 4. Local Web Server

Every computer on the Internet that contains a Web site must have a Web server program.

The top web servers on the Internet by Netcraft Survey April, May 2015 are:

  - Apache (by Apache) 39,26%;

- IIS (by Microsoft) 28,88%;

  - nginx (by NGINX) 14,42%;

- GWS (by Google) 2,34%.

Apache, IIS and Nginx are the most used web servers on the Internet.

5. Cloud Application platform is a platform as a service that enables developers to build and run applications entirely in the cloud.

6. Node or Node.js is an open source, cross-platform runtime environment for server-side and networking applications.

It was invented in 2009 by Ryan Dahl, and other developers working at Joyent.   Node transforms JavaScript from specialized language into general purpose language. It adds a JavaScript capability to cooperate with input/output devices by its API.   Node.js allows the creation of web servers and networking tools, using JavaScript and a collection of "modules" that handle various core functionality. The other way Node is applied to develop client applications, microcontroller

1. Графические редакторы используются для работы с изображениями.

PhotoShop - это редактор растровой графики, разработанный и опубликованный Adobe Systems для Windows и OS X. Он содержит основные инструменты, методы и рабочие процессы для создания веб-страницы. PhotoShop 30-дневная бесплатная пробная версия. Adobe Photoshop Sketch - это приложение для рисования для iPad, которое предоставляет вам естественные инструменты рисования и пользовательские кисти.

2. Редакторы кода используются для написания кода.

NotePad ++ будет использоваться для программирования. Программа будет запущена в браузере. Интегрированная среда разработки (IDE) - это программное приложение, которое предоставляет компьютерным программистам широкие возможности для разработки программного обеспечения. Обычно среда IDE состоит из редактора исходного кода, средств автоматизации сборки и отладчика [https://en.wikipedia.org/wiki/Integrated\_development\_environment]. WebStorm - это IDE от JetBrains для разработки на стороне клиента. Это 30-дневная бесплатная пробная версия.

3. Браузер

Веб-браузер (обычно называемый браузером) - это программное приложение для поиска, представления и прохождения информационных ресурсов в WWW.

4. Локальный веб-сервер

Каждый компьютер в Интернете, который содержит веб-сайт, должен иметь программу веб-сервера. Лучшие веб-серверы в Интернете по данным Netcraft Survey Апрель, май 2015 года: -

- Apache (от Apache) 39,26%;

- IIS (от Microsoft) 28,88%;

- nginx (по NGINX) 14,42%;

- GWS (от Google) 2,34%.

Apache, IIS и Nginx являются наиболее часто используемыми веб-серверами в Интернете.

5. Платформа облачных приложений - это платформа как услуга, которая позволяет разработчикам создавать и запускать приложения полностью в облаке.

6. Node или Node.js - это кроссплатформенная среда выполнения с открытым исходным кодом для серверных и сетевых приложений. Он был изобретен в 2009 году Райаном Далем и другими разработчиками, работающими в Joyent.

Node преобразует JavaScript из специализированного языка в язык общего назначения. Он добавляет возможность JavaScript для взаимодействия с устройствами ввода / вывода через его API. Node.js позволяет создавать веб-серверы и сетевые инструменты, используя JavaScript и набор «модулей», которые обрабатывают различные основные функции. Другой способ Node применяется для разработки клиентских приложений, микроконтроллеров.