

Виктор Белозор

СОЗДАНИЕ И ПРОЕКТИРОВАНИЕ ВЭБ-ИНТЕРФЕЙСОВ

ВВЕДЕНИЕ

Добрый день, меня зовут *Виктор*. Сегодняшний доклад будет посвящен созданию и проектированию веб-интерфейсов. Для удобства мы разделили доклад на **4** последовательные части:

1. Психология дизайна
2. Интерфейс и юзабилити
3. Практика веб-дизайна
4. Новое в веб-дизайне

В конце трех из четырех частей приводятся практические примеры для закрепления материала и, выражаясь научным языком, «снятия когнитивной нагрузки», что является важным критерием при разработке любого интерфейса.

Таким образом, мы плавно переходим к первой части доклада – *психологии дизайна*.

ЧАСТЬ 1 - ПСИХОЛОГИЯ ДИЗАЙНА

Говоря о дизайне и веб-дизайне в частности, мы всегда должны обращаться к *психологии*. Попадая в какую-либо мало знакомую среду, например на сайт, человек не должен теряться, а должен использовать свой предыдущий опыт, чтобы понять, как ему следует себя вести и что он может сделать в новой для него среде.

СЛАЙД 04 - АФФОРДАНС

В связи с этим при проектировании систем и сред важным понятием является *аффорданс*. Как несложно догадаться из перевода данного термина (*англ. afford* - позволять) аффорданс – это то, что предмет или среда «*позволяет* сделать с собой».

Колесо круглое (а не квадратное), потому что только таким образом можно достичь оптимального способа передвижения или транспортировки чего-либо. Если аффорданс объекта соответствует выполняемой им функции, то данный объект будет эффективным и удобным в использовании.

Копирование физических объектов и сред во многом облегчает использование разработок, в особенности, если речь заходит о поведении в «нематериальных» средах, как, например, в вебe. Так, в частности, не стоит недооценивать известные всем «вэбдвандольные» кнопки. Вряд ли мы полностью от них откажемся, хотя их повсеместное использование выглядит достаточно нарочито.

ПЕРЕХОД К СЛАЙДУ 05

Аналогичную функцию выполняют пиктограммы в операционных системах, значки на приборной панели автомобиля и пр. Вообще правильное использование аффорданса позволяет пользователю выработать *интуицию*. Однако интуиция пользователя и дизайнера это не всегда одно и то же.

СЛАЙД 05 - КОНЦЕПТУАЛЬНАЯ И МЕНТАЛЬНАЯ МОДЕЛИ

Дело в том, что понимание людьми систем и сред и взаимодействие с ними происходит путем сравнения *умозрительных моделей* с *концептуальными моделями*, т.е. реальными системами, с которыми происходит взаимодействие.

Как правило, создавая концептуальную модель, дизайнеры имеют о ней *практически полные данные, но слабо представляют о том, как с такой системой будут взаимодействовать пользователи*. Пользователи же наоборот имеют весьма скудные представления о работе с системой, но накапливают более полный опыт (по сравнению с дизайнерами) по мере работы с системой.

NB: Чем больше концептуальная модель дизайнеров соответствует ментальной модели пользователей, тем более успешным является продукт.

ПЕРЕХОД К СЛАЙДУ 06

По аналогии с концептуальной и ментальной моделями, для вэба можно обнаружить интересное совпадения, а именно соответствие популярной маркетинговой модели **AIDA** (**A**ttention **I**nterest **D**esire **A**ction) поведению людей в *стрессовой ситуации*.

СЛАЙД 06 - ЗАМРИ-БЕГИ-ДЕРИСЬ-СДАВАЙСЯ И AIDA

Когда люди попадают в стрессовую ситуацию (попадание в другую среду или на сайт, в особенности для пожилых людей это стресс) их поведение можно описать последовательностью действий «*дерись и беги*» (или более точно «*замри-беги-дерись-сдавайся*»).

Предположим, произошла стрессовая ситуация. В первые минуты человек *замирает*. Затем, еще не до конца понимая, что происходит, он *бежит*, чтобы избежать угрозы. Если угрозу избежать не удастся – человек *борется*. Ну, а если угрозу никак не получается нейтрализовать, то наступает этап *капитуляции*.

«Дерись и беги» – врожденная реакция человека. Не смотря на то, что пусковые механизмы данной реакции у всех разные, она должна учитываться при разработке систем, в которых пользователь в той или иной степени принимает решения (например, решение о покупке в интернет или просто магазине).

В маркетинге эта реакция нашла отражение в модели **AIDA** (*англ.* - **В**нимание **И**нтерес **Ж**елание-**Д**ействие). Ну а в вэбе ее можно наблюдать уже сейчас, как, например, на рисунке справа (так называемая *Z схема*, которую мы рассмотрим в третьей части доклада).

ПЕРЕХОД К СЛАЙДУ 07

Когда все препятствия для удобного ориентирования в системе (или на сайте) устранены, пользователь должен четко себе представлять, *зачем он сюда попал*, что ему делать и куда двигаться дальше. А вот об этом должны были заранее подумать дизайнеры – какую *функцию* (и будет ли вообще какая-либо функция) у данной системы.

СЛАЙД 07 - ФОРМА И ФУНКЦИЯ

При постановке такого вопроса всегда важно понимать, *какие аспекты дизайна будут являться решающими для выполнения поставленной задачи*. В решении одних вопросов можно поступиться эстетическими составляющими (например, когда речь идет о безопасности), а при решении других – функциональными. Главным остается тот фактор, который лучше отвечает запросам потребителей.

NB: В дизайне нет законов, поэтому не стоит использовать т.н. «рекомендательную формулировку» данного правила: первое место уделять функции, а второе форме. *Главным критерием любого проекта является успех.*

ПЕРЕХОД К СЛАЙДУ 08

Но чтобы передать успех потребителю важно направить его по верному адресу – через парадный вход, а не через заднюю дверь. Таким входом является *точка входа*. В дизайн.

СЛАЙД 08 - ТОЧКА ВХОДА В ДИЗАЙН

Как известно, встречают по одежке. Если форма и функция уже определены, а сама среда или система не представляет сложности для ориентирования, то вам нечего бояться. Хотя подстраховаться, конечно, нужно, направив пользователя к главному входу. Для этого вам потребуется *точка входа*.

NB: Первое впечатление о системе или среде серьезно влияет на последующее отношение пользователя и формируется при входе в нее, или иными словами, во время начала работы.

Идеальная точка входа должна обладать следующими характеристиками:

1. обладать *минимальными барьерами* (т.е. она должна быть видна из далека)
2. должна подсказывать, *что делать дальше* (т.е. не мешать обзору)
3. должна *привлекать внимание* и завлекать внутрь (самое эффективное, когда «приманки» появляются постепенно и увлекают пользователя за собой)

ВЫВОД ДЛЯ ЧАСТИ 1

Хорошо продуманная эргономика, разработанная с учетом свойств системы или среды, понимание возможного поведения пользователя, правильное позиционирование системы для решения конкретной задачи, а также наличие удобного входа *обеспечат начало комфортного взаимодействия пользователя с системой.*

ПЕРЕХОД К ПРИМЕРУ 1

Как нам удалось выяснить из первой части, *психология играет важную роль в дизайне*. Не стоит забывать об этом, занимаясь повседневными дизайнерскими мелочами, как например, выбирая или создавая иконки для вашего проекта. Правильно подобранная иконка способна оказать необходимое воздействие и послужить хорошим помощником в навигации.

ПРИМЕР 1 - ПСИХОЛОГИЯ ИКОНОК

Какую иконку выбрать в зависимости от поставленной задачи? Существует **4** типа знаковых представлений:

1. Знаки подобия
2. Знаки-примеры
3. Знаки-символы
4. Условные знаки

NB: В зависимости от того насколько сложна абстракция, которую требуется передать при помощи графического символа, следует выбирать тот или иной тип знака.

Знаки подобия эффективны для отображения *простых действий, объектов и идей*. Они менее эффективны для сложных понятий. Например, знаки подобия реализованы в правилах дорожного движения, указателях поворота, обновить страницу в браузере и пр.

Знаки-примеры используют для отображения более сложных понятий – *сложных действий, объектов или идей*. Например, иконка самолета может означать аэропорт, а вилка и ложка – ресторан и т. д.

Еще более сложные понятия помогут объяснить *знаки-символы*. Они эффективны, когда действия или объект имеют «*метафору*» или *хорошо узнаваемы* (например, электричество – это молния, хрупкий – это рюмочка и пр.)

Особняком стоят *условные знаки*. Они имеют слабое отношение или вообще никак не связаны с действием – эту связь нужно запомнить (например, знак радиации, биологической опасности, зеркальце *Венеры*). Такие знаки хорошо использовать в качестве стандартов.

ПЕРЕХОД К ЧАСТИ 2

Надеюсь, данный пример поможет вам правильно подобрать иконки для ваших следующих проектов, ну а мы переходим ко второй части доклада.

ЧАСТЬ 2 - ИНТЕРФЕЙС И ЮЗАБИЛИТИ

«Затянув» пользователя «в среду» / систему / на сайт и создав ему комфортные условия, важно разобраться, *как при этом должна работать система, как поддержать интерес пользователя на каждом этапе принятия решения* (например, опять же при совершении покупки в интернет-магазине), что свойственно пользователям, а чего пользователи предпочитают избегать.

СЛАЙД 11 - ОПРЕДЕЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Для начала, рассмотрим определения пользовательского интерфейса. Вне зависимости от формулировки, главным всегда остается *взаимодействие* пользователя и системы.

Цумата: «Всякий раз, когда мы говорим о пользователе как о потребителе, интерфейс становится продуктом»

Джеф Раскин

В частности, например, вопреки всеобщему заблуждению, нужно сказать, что пользовательский интерфейс – это не то, каким образом следует расставить кнопки в меню, а нужно ли вообще меню реализовывать при помощи кнопок.

Рассмотрим *факторы, определяющие хороший интерфейс*. Всего **8** факторов, их нет в презентации, я быстро по ним «пробежусь»:

1. *Лаконичность* (правильный перевод с иностранного языка, иерархия элементов, использование понятных «метафор»)
2. *Целостность* (необходимо отразить только самые нужные элементы)
3. Интерфейс должен быть *знаком пользователю*
4. *Отзывчивость* (интерфейс должен быстро отвечать на запросы пользователей)
5. Интерфейс должен *передавать логику* на малознакомые участки системы
6. *Эстетика*
7. *Эффективность*
8. Интерфейс должен *уметь прощать* [пользователя, разумеется]

ПЕРЕХОД К СЛАЙДУ 12

Конечно, все перечисленные факторы целиком опираются на *привычки пользователей*. Поэтому для разработки удобных систем немаловажным является понимание того, как пользователи ведут себя в мало знакомой среде.

СЛАЙД 12 - ПРИВЫЧКИ ПОЛЬЗОВАТЕЛЕЙ

Пользователь – это, прежде всего человек, такой же, как и мы с вами. Поэтому никаких сверхсложных задач перед ним ставить не нужно.

1. Раз пользователь пришел на сайт, значит *ему там что-то нужно*. Даже если на сайте будут располагаться плохо оформленные рекламные объявления, а часть контента будет плохо оформлена сама по себе, но если на сайте располагается ровно та информация, за которой он пришел – пользователь стерпит все. *Дизайн всегда следует за содержанием*.

2. Сайт это не художественная книга, поэтому *пользователь никогда не читает интернет страницу – он ее просматривает*. А по сему – структурируйте информацию, выдавайте ее пользователю постепенно, дозированно. Вообще, у тех, кто, так или иначе, «попал» в интернет всегда мало времени, так что это тоже приходится учитывать.
3. Мало того, что пользователи всегда торопятся, так они еще и *не хотят думать, когда что-то ищут*. Такова уж природа человека, а точнее его биохимия (за поиск информации отвечает нейромедиатор *дофамин*, он «держит» пользователя сосредоточенным при решении конкретной задачи поиска информации). Так что, чем меньше человек думает на сайте, тем лучше. Не следует его «нагружать» еще больше.
4. *Пользователи исследуют вэб так как им удобно* и не всегда, то, что они выбирают, является оптимальным с точки зрения дизайнера, проектировавшего интерфейс. Пользователи выберут такой вариант, который будет достаточным и удовлетворит их потребность в поиске. *Любая оптимизация сложна сама по себе, пользователи идут по более быстрому пути*.
5. Пользователь живет по правилу: *«Если что-то работает (не важно, как), то я воспользуюсь тем, что есть»*. Пользователь идет к своей цели, и он совершенно не хочет понимать механизмы, которые его туда приводят.
6. Ну и, конечно же, неотъемлемой частью человеческой природы является желание *ощущать контроль* над ситуацией и интернет здесь не исключение.

ПЕРЕХОД К СЛАЙДУ 13

В связи со всем вышесказанным перед дизайнером стоит нелегкая задача – *удержать посетителя на сайте*. В современной литературе по интерфейсам часто можно встретить словосочетание **«чистый и четкий дизайн»**. Конечно, все аспекты данного термина выходят за рамки данного доклада, и мы рассмотрим только наиболее значимые из них, имеющие существенное практическое отношение к проектированию вэб-интерфейсов.

СЛАЙД 13 - ЧИСТЫЙ И ЧЕТКИЙ ДИЗАЙН 1 - ВАЖНЫЕ МЕЛОЧИ

Красивый дизайн всегда скрывается в *мелочах*. А раз мы говорим о «чистом и четком» дизайне, то мелочам необходимо уделять особое внимание. Многие путают чистый дизайн с «минималистичным», что не правильно, т.к. последний – *это стиль, характеризующийся лаконичностью выразительных средств*.

Если создать «комфорт» пользователю можно более замысловатым, интересным, придающим проекту законченный образ способом, то это также будет «чистый и четкий» дизайн.

Переходя к практическим аспектам проектирования сайтов, можно отметить факторы:

1. *Точные, не отвлекающие пользователя детали*, границы и прочие элементы, спроектированные «пиксель-в-пиксель»
2. Не пренебрегайте *пустым пространством*. Оно создает ритм в композиции и вэб-сайт не является исключением
3. Следствием из второго пункта является позиционирование элементов. В этом дизайнерам помогают *фреймворки*, о них мы поговорим далее.
4. Развитие технологий позволяет в настоящее время активно использовать *шрифты*. Но следует помнить, что вэб – это среда, отличная от той, где мы всегда привыкли видеть шрифты (книги, журналы и пр.). Поэтому следует тщательно подбирать шрифты не только для выделения, но и для набора больших текстов.
5. *Цвет* я не случайно поставил на последнее место, т.к., если все сделать правильно, то первых четырех пунктов хватит для того, чтобы создать «чистый и четкий» дизайн. Это как с логотипами – если логотип не смотрится в черно-белом варианте, то и цвет его не спасет.

ПЕРЕХОД К СЛАЙДУ 14

Данные мелочи относятся к эстетической составляющей «чистого и четкого» дизайна. Куда более важными являются такие вопросы, как *проектирование меню*, где также нужно учитывать мелочи, но и не забывать про интуицию пользователя.

СЛАЙД 14 - ЧИСТЫЙ И ЧЕТКИЙ ДИЗАЙН 2 - ПРОЕКТИРОВАНИЕ МЕНЮ

Во второй части «чистого и четкого» дизайна я привожу *закон Хика*. Запоминать формулу не нужно, я привел ее потому, что данный закон часто обсуждается в литературе и вам могло бы быть интересно, как он выглядит на самом деле.

Суть закона предельно проста: *чем больше альтернатив, тем сложнее сделать выбор*. И это нужно всегда помнить, проектируя меню. Закон *Хика* находит применение в разработке любой системы или процесса, которые требуют принятия простых решений при наличии множественного выбора.

Любую задачу можно представить как последовательность действий:

1. Определение проблемы
2. Оценка вариантов решения
- 3. Принятие решения**
4. Реализация варианта

Первые два пункта мы так или иначе рассмотрели ранее (проектируя систему), а вот на третьем этапе наступает очередь пользователя, тут нужно использовать закон *Хика*.

Примечание к слайду: во втором примере на слайде закон *Хика* не применим; важно понимать, что чем сложнее меню, тем сложнее пользователю (в прямом смысле). Поэтому сейчас на сайтах мало где можно найти сложносоставные меню.

ПЕРЕХОД К СЛАЙДУ 15

Ну и напоследок рассмотрим, что еще можно сделать, чтобы повысить *юзабилити* вашего сайта.

СЛАЙД 15 - ЧИСТЫЙ И ЧЕТКИЙ ДИЗАЙН 3 - ЮЗАБИЛИТИ

Во-первых, что такое юзабилити?

Определение: Юзабилити – это мера удобства достижения пользователем конечного результата; иными словами – практичность системы.

Сразу вспоминается *закон Фиттса* – время, необходимое для наведения на цель, является функцией размера цели и расстояния до нее.

Понятно, почему кнопка **купить** всегда яркого красного или зеленого цвета и больше всех остальных. Так удобно.

NB: Закон *Хика* отвечает за время принятия решения, а закон *Фиттса* отвечает за «направленность» данного решения.

Учет данных законов позволит пользователям быстрее и эффективнее ориентироваться на сайте, что повысит степень его удобства.

ВЫВОД ДЛЯ ЧАСТИ 2

Кроме законов еще есть ряд т.н. «правил», которыми мы резюмируем данную часть доклада.

1. С точки зрения удобства пользователя, бывшие некогда популярными *заставки* – это «*кошмар юзабилити*». Они не несут практически никакой смысловой нагрузки, но зато здорово вырывают пользователя из контекста всего содержания сайта.
2. По этой же причине *ссылки всегда должны открываться в том же окне* (и даже ссылки на внешние источники). Если пользователь захочет уйти с сайта – он и так уйдет, а лишнее переключение между ссылками только мешает сосредоточенной работе.
3. Раз уж мы заговорили про ссылки, то не лишним будет сказать, что *ссылки должны выглядеть как ссылки* (т.е. пользователь должен понимать, что перед ним ссылка, а что – просто текст). Ведь вэб – это в той или иной степени «набор ссылок» (гиперссылок).
4. Как мы уже говорили, *использование метафор из реальной жизни* сильно упрощает работу пользователя и стирает грани между реальным и виртуальным.

5. Хороший интерфейс не должен наказывать своих пользователей, а *должен уметь их прощать*, тем более что они всегда спешат и могут нажать не ту кнопку.
6. Логика, которую пользователь перенимает при помощи *интуиции*, должна пронизывать весь проект и сохраняться везде, даже в отдельных элементах дизайна.

ПЕРЕХОД К ПРИМЕРУ 2

Пройдя две части доклада, мы подходим к практике вэб-дизайна. Но перед тем как мы приступим к обсуждению практических аспектов вэб-дизайна, рассмотрим еще один пример. Кроме иконок, в вэбе можно часто встретить *текстуры, кнопки и пр.* Их появление не случайно – *они помогают пользователю «нащупать» грань между двумя реальностями, вспомнить опыт из реальной жизни.*

ПРИМЕР 2 - ИСПОЛЬЗОВАНИЕ ТЕКСТУР И КНОПОК

Контактная форма обычно «*гладкая*» – там есть поля для ввода данных, поэтому излишнее усложнение здесь ни к чему. Представив «гладкое» на «*шершавой* текстуре» мы делаем форму ближе к пользователю.

Немного поработав над формой полей, создадим *легкое подобие окошка*, куда нужно вводить информацию. Плюс людям *свойственно отдавать предпочтение объектам с плавными, а не с острыми контурами.*

«Шершавость» уголка текстового блока пришла в вэб из операционных систем. Так пользователю дают понять, что данное окно масштабируется или его можно переместить (как это было в случае операционных систем).

Ну и конечно же *метафора кнопки* говорит о том, что после того как все поля заполнены для завершения операции нужно нажать кнопку.

Примечание к примеру: рассмотренный выше пример является лишь одним из способов оформления контактной формы. Он скорее «оправдывает» использование дополнительных элементов, а не создает правило оформления контактных форм. Контактная форма, сделанная средствами HTML и базовыми стилями, также будет являться функциональной.

ПЕРЕХОД К ЧАСТИ 3

Таким образом, мы переходим от эстетических правил хорошего дизайна непосредственно к архитектуре сайта. Но не стоит откладывать *привычки пользователей* далеко, т.к. они скоро нам понадобятся, когда речь пойдет о схемах сайтов.

ЧАСТЬ 3 - ПРАКТИКА ВЭБ-ДИЗАЙНА

В данной части мы коснемся сугубо *практических аспектов вэб-дизайна*, рассмотрим инструменты для реализации факторов хорошего дизайна, которые мы обозначили в предыдущей части и рассмотрим, *как поведение пользователей влияет на выбор того или иного типа сайта.* Поэтому в данной части у нас не будет практического примера, но большой пример будет в заключительной четвертой части.

СЛАЙД 18 - HTML КАК ИНСТРУМЕНТ ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ

Прежде всего, необходимо помнить, что за красотой сайта (как эстетической, так и функциональной) скрывается *код*. Этот факт нужно всегда иметь в виду. Он позволит более точно решать задачи, если вы используете в своих проектах *аутсорсинг* (привлекаете дизайнеров или программистов). Да и чтобы клиенты не ставили перед вами невыполнимых задач.

С технической точки зрения HTML это не совсем код. Он *не передает логику операций, а определяет порядок генерирования страницы*. Это, если хотите, следующий этап формирования полиграфической сетки, но только для вэба.

NB: HTML переносит ваш дизайн в браузер *количественно*. Поэтому, к нему можно предъявить ряд требований:

1. Используйте **классы** только когда нужно. Если необходимо пометить элемент, посмотрите на структуру, повторяется ли она еще где-либо, и тогда обратитесь к элементу при помощи CSS, а если этого сделать нельзя – используйте **id**. Использование классов может также быть продиктовано тем или иным *фреймворком*, который вы используете в качестве вспомогательного агента.
2. Лучше, чтобы *HTML был доступен всегда*. Вы не покажете всей красоты проекта, но пользователь сможет получить, то, за чем пришел на сайт.
3. *Валидность* – это не панацея хорошего дизайна и создатели HTML5 эту проблему устранили. Рекомендуется жертвовать валидностью ради расширения функционала сайта.
4. *Семантика* поможет вам эффективно продвигать сайт

ПЕРЕХОД К СЛАЙДУ 19

Как уже было сказано, *HTML – это следующий этап полиграфической сетки для сайта*. Но какой предыдущий? Правильно – сама полиграфическая сетка. Правда, в случае проектирования вэб-интерфейса это целая система сеток, «поддерживающая» дизайн на различных этапах.

СЛАЙД 19 - 960 И BLUEPRINTS

Определение: Полиграфическая сетка – это вспомогательный элемент дизайна печатной продукции, который определяет логику представления содержания на страницах того или иного издания в зависимости от его типа.

В случае вэба мы тоже используем сетку, но сложность состоит в том, что мы должны сохранить эту сетку еще и на этапе программирования.

В вэбе на сегодняшний момент эту задачу решают *960* и *Blueprints*. Более популярен *960*. Почему 960? Потому что это самый оптимальный размер контейнера для контента, использующийся в 99%

случаев в вэбе. И его удобно делить на 12, 16 и 24 части в зависимости от сложности проекта. Все соотношения в 960 основаны на принципе *золотого сечения*.

Примечание о золотом сечении:

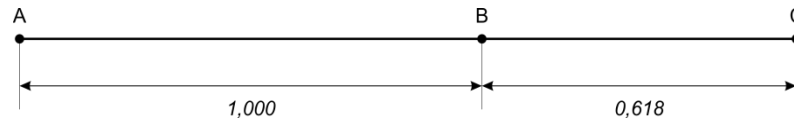


Рис. 1 Золотое сечение

Золотое сечение – это такое отношение, в котором меньший отрезок относится к большему также как больший относится к сумме двух отрезков (Рис. 1). $BC / AB = AB / AC \approx 0,618$. Золотое сечение является иррациональным бесконечным числом, вычисляемым по формуле $(\sqrt{5} - 1)/2 \approx 0,618$ (Формула *Бине*). Число $\phi \approx 1+0,618 \approx 1,618$ также является пределом последовательности *Фибоначчи*.

Blueprint немного отличается от 960. Здесь ширина равняется 950 px, а не 960 px, он также использует 24 колонки, но, что самое главное *Blueprint* создавался как уже готовая таблица стилей с заготовленными свойствами под каждый элемент. Основным недостатком *Blueprint* является то, что данная система очень сложна для внедрения на этапе дизайна и прототипирования. Поэтому большинство людей используют 960 совместно с рядом других «*фреймворк-плагинов*».

ПЕРЕХОД К СЛАЙДУ 20

Начав прототипирование сайта, не лишим будет задуматься: *а можно ли применить какой-либо поведенческий паттерн пользователей, отдав предпочтение той или иной схеме?* Да, можно, но как мы уже говорили, четких законов в дизайне нет – поэтому **рассматривать данные схемы можно скорее как следствие предпочтений пользователей, а не как предписание к конкретным действиям.**

СЛАЙД 20 - Z СХЕМА - ДИАГРАММА ГУТЕНБЕРГА

Так в частности, *Z схема* говорит нам, что логотип должен быть слева в верхнем углу. Вовсе нет. Где вы поставите логотип, там он и будет (справа, по центру и пр.).

Z схема (еще называемая диаграммой *Гутенберга*) отражает привычки *европейца* (мы читаем слева → направо, сверху ↓ вниз). Поэтому с точки зрения удобства восприятия и подачи информации, лучше направить пользователя по привычной для него «проторенной дороге». По такому принципу, например, строится структура газетных полос, оформляются детские книги.

Направление глаз вдоль центрального штриха горизонтально отраженной буквы **Z** называют еще «*гравитацией чтения*». Расположение элементов, раскрывающих логическую ось ориентации, позволяют ускорить чтение и лучше понять материал. Чем сложнее и тяжелее для осмысления текст (или содержимое сайта), тем в большей степени нужно стремиться раскрывать потенциал данной схемы.

Из-за этого факта *Z схема* стала очень популярной в качестве заглавных страниц и промо-сайтов (показать максимум информации пользователю, не загрузив его и сохранив интерес к дальнейшему просмотру).

ПЕРЕХОД К СЛАЙДУ 21

Z схема, а точнее диаграмма Гутенберга, лежит в основе еще одной схемы *F* схемы.

СЛАЙД 21 - F СХЕМА - ТЕПЛОВАЯ КАРТА КЛИКОВ

F схема появилась в результате исследования блогосферы и была подтверждена *тепловой картой кликов*.

Определение: Тепловая карта кликов – это карта, показывающая, на что пользователи в большей степени обращают внимание.

Во-первых, это заголовок, во-вторых, – верхняя часть (как в *Z* схеме), а дальше пользователи сканируют картинки и заголовки, выбирая только самое интересное, и уже читают только то, что представляет для них наибольший интерес.

В *F* схеме часто присутствует боковая колонка, которая служит для «бытовых» нужд сайта (меню, реклама и пр.). Однако в верхней части колонки зачастую располагают достаточно значимую информацию, т.к. она также способна привлечь внимание пользователя.

Вообще, разделение контента на колонки удобно (вспомните хотя бы газету). В действительности люди читают быстрее длинные строки, но предпочитают разбивать их на более мелкие отрезки.

Примечание о том, как читают люди:

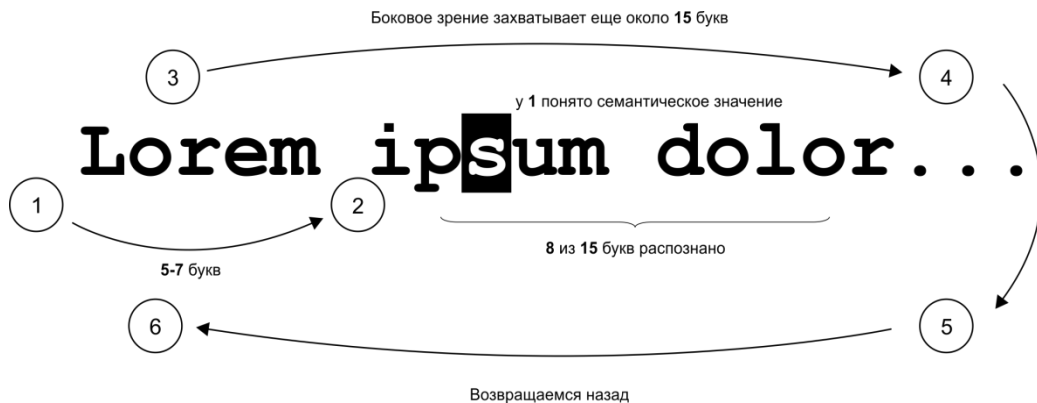


Рис. 2 Схема движения глаз при чтении

Люди читают *саккадами*, т.е. совершают резкие скачкообразные движения, распознают буквы с запасом, возвращаются назад и т.д. Соответственно, прерывать эту последовательность им неудобно. Поэтому длинную строку человек прочитает быстрее, чем короткую (*оптимальной считается строка в 100 символов*). Однако человек натура тонкая, и он предпочтет разбить строку на более удобные 45-72 символа. Длинные строки не прерывают *саккады* – отсюда и скорость. Но человек всегда предпочтет колонки сплошному массиву текста. Такова его природа.

ПЕРЕХОД К СЛАЙДУ 22

Деление на колонки плюс скачкообразное движение глаз приводит нас к еще одной схеме, появившейся и набирающей популярность в вэбе достаточно недавно.

СЛАЙД 22 - S СХЕМА - НОВАЯ ЖИЗНЬ ПРОМО-САЙТОВ

S схема использует «гравитацию чтения» от диаграммы Гутенберга и удобство чтения, связанного с делением контента на колонки. Такая схема способна моментально донести содержание до пользователя. Обычно ее применяют в шахматном порядке: *изображение - текст* и так далее.

Технические возможности HTML5 делают эту схему достаточно популярной в настоящее время и, по-видимому, в будущем.

ВЫВОД ДЛЯ ЧАСТИ 3

Следует отметить, что четких указаний, когда какую схему использовать нет, да и не может быть. Наверное, глупо представить *F* схему для промо-сайта, а *S* схема не очень компактна для размещения товаров в интернет-магазине. В любом случае, понимание того, как будет сгенерирована страница в браузере, на всех этапах создания сайта подскажет, какой шаблон лучше использовать, а современные инструменты сайтостроения помогут реализовать проект.

ПЕРЕХОД К ЧАСТИ 4

Как уже говорилось, примера у нас в этой части не будет, т.к. она и без того получилась слишком «практической», так что переходим к заключительной части. В ней мы поговорим об HTML5, коротко коснемся тенденций и рассмотрим интересный практический пример.

ЧАСТЬ 4 - НОВОЕ В ВЭБ-ДИЗАЙНЕ

Вот уже практически два года у всех на слуху новое слово HTML5. Да-да именно слово! Мало, кто обращал внимание на правописание нового интернет-термина, *пятерка пишется слитно*. Почему спросите вы. А для этого рассмотрим историю становления HTML5.

СЛАЙД 24 - ИСТОРИЯ ВОЗНИКНОВЕНИЯ HTML5

Все началось с HTML 4.01. После очередного пересмотра документации была разработана версия XHTML 1.0 ($X \neq \text{eXtreme}$, а $X = \text{eXtensible}$, не *экстремальный*, а *расширенный*). Элементы XHTML 1.0 были такими же, как и в HTML 4.01, разница заключалась в *более сложном синтаксисе*. Это не было так уж плохо и способствовало эстетике кода (например, чтобы все элементы были записаны в нижнем регистре).

Публикация XHTML 1.0 совпала с развитием CSS и пришлось на начало «браузерных войн», так что строгий синтаксис XHTML 1.0 был рекомендован W3C. Далее W3C выпустила XHTML 1.1. Он был *до мозга костей XML* (если вы пишете `mime-type XML`, то IE документ вам не выдаст).

W3C видели будущее интернета за XML, поэтому после того как вышла 4 версия HTML, они сели за разработку XHTML 2.0. Основной упор при разработке нового языка был сделан на точность, а о прошлой истории XHTML не хотели даже упоминать. На практике получился кошмар. Внутри W3C назревал раскол, вызванный «теоретизацией» языка и уходом от практических реалий. Представители «Эппл», «Оперы» и «Мозиллы» также были недовольны таким решением и хотели сделать акцент на разработке *вэб-приложений*.

Все решилось на семинаре в 2004 году. Айан Хиксон («Опера Софтвэз») предложил **идею о «расширении» HTML с возможностью создавать приложения**. Предложение было отклонено, и тогда люди создали рабочую группу под неблагозвучным для английского языка названием **WHATWG (Web Hypertext Application Technology Working Group)**. Айан Хиксон стал директором данной группы. Он принимал и утверждал решения, выносимые на повестку дня (нужно отметить, что в W3C такого человека не было). **WHATWG** разделилась на группы: *Web Applications 1.0* и *Web Forms 2.0*. Все они в дальнейшем были объединены в HTML5.

В это время W3C продолжала работу над XHTML 2.0. Однако без должного руководства они двигались «в никуда» и направлялись туда они очень медленно. В октябре 2006 W3C решили, что разработки **WHATWG** лягут в основу новой версии HTML, какими бы они ни были. Все это привело к тому, что W3C работали одновременно над HTML 5 (с пробелом) и XHTML 2.0, а **WHATWG** работали над спецификацией HTML5.

В 2009 году W3C объявила о том, что чартер XHTML 2 не будет пересмотрен, чем фактически был подписан смертный приговор новой версии языка. Те же, кто использовал XHTML 1, считали, что синтаксис нового HTML5 будет содержать ошибки. Это не так – *HTML5 имеет настолько точный синтаксис, насколько вы этого захотите*.

В настоящее время над HTML5 работают две группы **WHATWG** и **W3C**. **2022 год** считается годом «предложения к рекомендации» HTML5, а в **2012 году** HTML5 стал «кандидатом для рекомендации». Но использовать его можно уже сейчас.

ПЕРЕХОД К СЛАЙДУ 25

HTML5 не революционен, а эволюционен. Он является следующим этапом развития гипертекста. Если сейчас вы используете какую-либо версию HTML в ваших проектах, то, фактически, вы уже используете HTML5. Рассмотрим особенности HTML5.

СЛАЙД 25 - ОСОБЕННОСТИ HTML5

Основным кредо HTML5 является *поддержка существующего контента, т.е. не разрушать старое*. XHTML 2 хотел все уничтожить и на обломках создать новый синтаксис, а HTML5 хочет все модернизировать и ускорить. Этим HTML5 заботится о пользователях, да что там говорить – *нацелен на пользователей*, где последним представляется возможность самим становиться дизайнерами и разработчиками.

Переходим сразу к делу. Что изменилось в HTML5?

1. Новый доктайп

```
<!DOCTYPE HTML PUBLIC "-//W3C// DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C// DTD XHTML 1.0//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html>
```

2. Упрощенный способ задания кодировки

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta charset=UTF-8>
```

3. Упрощенный способ подключения скриптов

```
<script type="text/javascript" src="file.js"></script>
```

```
<script src="file.js"></script>
```

4. Упрощенный способ подключения таблиц стилей

```
<link rel="stylesheet" type="text/css" href="file.css"></script>
```

```
<link rel="stylesheet" href="file.css"></script>
```

В некоторых языках программирования «белое пространство» не играет никакой роли, а в некоторых играет очень серьезную роль и скрипт может просто перестать работать. В гипертексте все просто, особенно в пятой версии. Пишите, как хотите – вас поймут:

```
<img> = <IMG> = <img />
```

```
<br> = <BR> = <br />
```

Выражаясь современным языком, у HTML5 *синтаксис «в стиле кэжуал»*. И многим это нравится. Однако в новом HTML есть ряд устаревших элементов.

Итак, **устаревшие элементы** и короткое описание к ним:

Frame – определяет свойства отдельного фрейма, на которые делится окно браузера. Этот элемент должен располагаться в контейнере **frameset**, который к тому же задает способ разметки страницы на отдельные области. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью атрибута **src**. Хотя обязательных атрибутов у тега **frame** и нет, рекомендуется задавать каждому фрейму его имя через атрибут **name**. Это особенно важно, если требуется по ссылке из одного фрейма загружать документ в другой.

Содержимое тега **noframes** отображается в браузере, когда он не поддерживает фреймы и не умеет их интерпретировать. Браузеры, которые работают с фреймами, полностью игнорируют содержимое тега **noframes**. Как правило, внутри этого тега располагается текст, информирующий пользователя о том, что его браузер фреймы не поддерживает или с предложением перейти на страницу без фреймов.

Тег **acronym** указывает на то, что текст является акронимом. В отличие от аббревиатуры, акроним – это устоявшееся сокращение, которое применяется как самостоятельное слово. К акронимам, например, можно отнести следующие слова: СПИД, ликбез, замполит, США, DOS и др.

По умолчанию, текст заключенный в контейнере `acronym`, подчеркивается пунктирной линией.

NB: Все акронимы – аббревиатуры, но не все аббревиатуры акронимы.
Поэтому используйте `abbr`.

Также не используются теги `font`, `big` и `center` и атрибуты `bgsolor`, `cellspacing`, `cellpadding` и `valign`.

Вместо элементов `b` и `i` используйте `strong` и `em`.

Переопределены элементы `cite` и `a` (последний элемент буквально «на стероидах»)

```
<h2><a href="/about">About me</a></h2>
<p><a href="/about">Find out what makes me tick.</a></p>
```

Если контента много, его можно «завернуть» в тег `a`:

```
<a href="/about">
  <h2>About me</h2>
  <p>Find out what makes me tick</p>
</a>
```

ПЕРЕХОД К СЛАЙДУ 26

Мы рассмотрели много «удобных» изменений в HTML5. Настало время узнать, что в HTML5 стало совсем новым.

СЛАЙД 26 - НОВОЕ В HTML5

HTML5 – не просто язык гипертекстовой разметки, а *такая же технология как Flash и Silverlight, прямой конкурент так сказать.*

Новые элементы позволяют быть HTML5 максимально гибким.

1. Элемент `canvas` – среда для создания динамического контента. Сам элемент очень прост:

```
<canvas id="my-canvas" width="360" height="240">
</canvas>
```

Однако, к сожалению, не все браузеры поддерживают `canvas`.

Например, если нужно нарисовать прямоугольник 100 px на 50 px в **canvas** нужно записать:

```
context.strokeRect(20,30,100,50);
```

Особенностью **canvas** является то, что у него нет DOM. На сегодняшний момент, это является проблемой HTML5 и мешает его активному продвижению.

2. Элементы **audio** и **video**. Долгое время *Flash* была технологией, которая отвечала за «плееры» в вэбе. HTML5 исправляет это.

```
<audio src="song.mp3"></audio>
```

```
<audio src="song.mp3" autoplay></audio>
```

- так делать нельзя, играть ведь будет :)

```
<audio src="song.mp3" autoplay loop></audio>
```

- а так делать вообще никогда нельзя 8)

Примечание: (это булевы операции, их можно еще записать так **autoplay=true loop=true**)

```
<audio src="song.mp3" controls></audio>
```

- добавим управление.

```
<audio src="song.mp3"></audio>
```

```
<div><button onclick="document.getElementById(`player`).play()">play</button>
<button onclick="document.getElementById(`player`).pause()">pause</button>
<button onclick="document.getElementById(`player`).volume += 0.1">Volume Up</button>
<button onclick="document.getElementById(`player`).volume -= 0.1">Volume Down</button></div>
```

- управление с кнопками.

```
<video src="movie.mp4" controls width="360" height="240" poster="placeholder.jpg"></video>
```

- проиграет видео.

3. Атрибут **placeholder** пригодится для того, чтобы поместить текст в графу поиска. К сожалению, его поддерживают не все браузеры. **Required** обозначит поля, необходимые для заполнения, а **тип поля** выдаст удобную раскладку клавиатуры для ввода на мобильном устройстве.

```
<label for="Your hobby">Увлечения</label>
<input id="Your hobby" name="Your hobby" type="text" placeholder="футбол">
```

```
<label for="pass">Ваш пароль</label>
    <input id="pass" name="pass" type="password" required>
```

- для полей, необходимых для заполнения.

```
<form action="/selfdestruct" autocomplete="off">
```

- автозаполнение отключено

В зависимости от типа поля в мобильном устройстве будет отображаться тем или иным образом клавиатура.

```
<label for="email">Email address</label>
<input id="email" name="email" type="email">
```

- квэрти клавиатура.

```
<label for="website">Website</label>
```

```
<input id="website" name="website" type="url"> - кверти клавиатура с .com и пр.
```

```
<label for="phone">Phone number</label>
```

```
<input id="phone" name="phone" type="phone"> - телефонная клавиатура для набора номера.
```

4. Удобство работы с датами и временем:

```
<label for="dstart">Start date</label>
```

```
<input id="dtstart" name="dtstart" type="phone">
```

5. Семантика. Элемент **mark** не добавляет важности к элементу, он *служит для формирования ссылки на элемент, учитывая его важность и присутствие в другом контексте* (типа как *тэг*).

```
<h1>Результаты поиска для `термин такой-то`</h1>
```

```
<ol>
```

```
  <li><a href="http://www.site.com">
```

```
    Необходимо рассмотреть <mark>термин такой-то</mark> в контексте всего интернета
```

```
  </a></li>
```

```
</ol>
```

6. Элемент **time** используется для отображения дат и пр.

```
<time datetime="17:00">5pm</time>
```

```
<time datetime="2010-04-07">April 7th</time>
```

```
<time datetime="2010-04-07T17:00">5pm on April 7th</time>
```

7. Элемент **meter** для измерения (часть чего-то, что имеет минимальное и максимальное значение)

```
<meter low="-273" high="100" min="12" max="30" optimum="21" value="25">
```

```
Сегодня достаточно тепло
```

```
</meter>
```

8. Элементы структуры **header** (используется для отображения содержания, являющегося введением или навигацией), **footer** (содержит информацию об авторе и пр.; может присутствовать и в секции), **section** (группирует содержание связанное одинаковой тематикой, обладает семантикой), **nav**, **article**, **aside** (служит для отображения навигации, статей и боковых элементов). *Вэб-дизайнеры слишком часто использовали термины header, footer и пр., что их решили включить в HTML5:*

```
<section>

  <header>

    <h1>Создание и проектирование веб-интерфейсов</h1>

  </header>

  <p>Этот отчет предназначается для дизайнеров и программистов</p>

</section>

<footer>

  <p>Автор: Виктор Белозор</p>

</footer>

<article>

  <header>

    <h1>DOM Scripting review</h1>

  </header>

  <p>A small lighthouse for what has been a long and sometimes dark voyage with JS</p>

  <footer>

    <p>Published

      <time datetime="2005-10-08T15-13" pubdate>3:13pm on October 8th, 2005</time>

    by Glenn Jones</p>

  </footer>

</article>
```

ПЕРЕХОД К СЛАЙДУ 27

Кроме новых «логичных и логических» элементов, а также упрощенного синтаксиса HTML5 по-новому позволяет использовать SVG. Это, как предсказывал *Артемий Лебедев*, приближает смерть пиксельной графики и периодически выражается в «модности» больших векторных иллюстраций в трендах 2010-х годов.

СЛАЙД 27 - 10 ФАКТОВ ПРО SVG

SVG – масштабируемая векторная графика. На этом слайде, я не делаю отдельного выступления и просто коротко расскажу о векторной графике, используемой в HTML5.

Работая с векторной графикой, вы можете легко и просто получить ее код и использовать его для вэба. Нарисовав что-либо в векторе, сохраните изображение в формате **.svg** и откройте файл в текстовом редакторе. Например, если открыть в редакторе изображение простейшей кривой, то можно получить вот такой код:

```

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<!-- Creator: CorelDRAW X5 -->

<svg xmlns="http://www.w3.org/2000/svg" xml:space="preserve" width="210mm" height="297mm"
style="shape-rendering:geometricPrecision; text-rendering:geometricPrecision; image-
rendering:optimizeQuality; fill-rule:evenodd; clip-rule:evenodd"

viewBox="0 0 21000 29700"

xmlns:xlink="http://www.w3.org/1999/xlink">

<defs>

<style type="text/css">

<![CDATA[

.str0 {stroke:black;stroke-width:290.61;stroke-linecap:round;stroke-linejoin:round}

.fil0 {fill:none}

]]>

</style>

</defs>

<g id="Layer_x0020_1">

<metadata id="CorelCorpID_0Corel-Layer"/>

<path class="fil0 str0" d="M10000 9029c2906,0 10306,781 10171,10171"/>

</g>

</svg>

```

Этот код, по сути, является XML. Он может быть встроен в DOM и быть проиндексирован браузерами, что позволяет существенно экономить трафик (хотя сейчас это и не такая проблема, как раньше) и всегда иметь качественное изображение вне зависимости от масштаба и разрешения экрана.

ПЕРЕХОД К СЛАЙДУ 28

Постепенное внедрение HTML5, доступность его новых элементов и возможностей выражается в появлении *тенденций в веб-дизайне*, интенсивно меняющихся каждый год. Не исключением становятся и 2012-2013 гг.

СЛАЙД 28 - ТЕНДЕНЦИИ В ВЭБ-ДИЗАЙНЕ

Вообще, если судить по публикациям тенденций в логотипах, *к такому типу информации не следует относиться серьезно*. Ведь важно, чтобы логотип точно передавал зашифрованное в нем послание той аудитории, которой оно направляется, а не следовал тому, что сейчас «модно». Конечно, практические аспекты «производства» логотипов (новые возможности векторной графики, полиграфия и пр.) рассматривать можно и нужно, но слепо следовать тенденциям глупо.

То же самое и с тенденциями в вэбе. *Прислушиваться и знать их нужно, а вот стремиться, чтобы сайт удовлетворял всем им – не следует.* Самой важной тенденцией является **адаптивность под различные разрешения**. Эта тенденция настолько сильна, что ряд экспертов предсказывает даже *смерть мобильных приложений – зачем делать что-то новое поверх существующего сайта, когда можно просто открыть сайт на телефоне и все.*

Вторая важная тенденция – **реализация графического замысла на чистом коде CSS и SVG** (фреймворки вам в помощь!).

Ну а все что остается – большие изображения, минимализм (или, напротив, дизайн в стиле ретро) может меняться в зависимости от индивидуального замысла дизайнера и становиться или не становиться «трендом года».

ВЫВОД ДЛЯ ЧАСТИ 4

В заключительной части мы рассмотрели возможности HTML5, которые год за годом становятся **революционными** (простите, *эволюционными*) и безграничными. Интернет становится более понятной и одновременно с тем более сложной на уровне архитектуры среды, внимание к которой приковывается не только специалистами, но и простыми пользователями.

ПЕРЕХОД К ПРИМЕРУ 3

Новые инструменты позволяют нам по-новому взглянуть на возможности вэба и совместить технологии для решения различных задач. В качестве такого проекта рассмотрим *возможность создания инфорграфики средствами SVG и Python*. Проект также можно разместить в интернете или сделать интерактивным.

ПРИМЕР 3 - ИСПОЛЬЗОВАНИЕ SVG

Задача: нанести данные на карту и сделать инфографику. По желанию можно разместить карту в интернете.

Решение:

1. Для начала нам понадобятся *данные*. Это может быть все что угодно. В данном примере, взятом с сайта *The Flowing Data*, были использованные данные по безработице в США.
2. Данные необходимо подготовить. Помещаем данные в формат **.csv**. Данный формат *предназначен для представления табличных данных*. Каждая строка такого файла – это одна строка таблицы. Значения отдельных колонок разделяются разделительным символом (**delimiter**) – запятой (,).

Вот как выглядит фрагмент такого файла:

```
CN010010,01,001,"Autauga County, AL",2009,"23,288 ","21,025 ","2,263 ","9.7
PA011000,01,003,"Baldwin County, AL",2009,"81,706 ","74,238 ","7,468 ","9.1
CN010050,01,005,"Barbour County, AL",2009,"9,703 ","8,401 ","1,302 ","13.4
CN010070,01,007,"Bibb County, AL",2009,"8,475 ","7,453 ","1,022 ","12.1
CN010090,01,009,"Blount County, AL",2009,"25,306 ","22,789 ","2,517 ","9.9
CN010110,01,011,"Bullock County, AL",2009,"3,527 ","2,948 ","579 ",16.4
CN010130,01,013,"Butler County, AL",2009,"8,884 ","7,403 ","1,481 ",16.7
PA010250,01,015,"Calhoun County, AL",2009,"52,055 ","46,422 ","5,633 ","10.8
CN010170,01,017,"Chambers County, AL",2009,"13,726 ","11,171 ","2,555 ","18.6
CN010190,01,019,"Cherokee County, AL",2009,"11,583 ","10,220 ","1,363 ","11.8
CN010210,01,021,"Chilton County, AL",2009,"18,697 ","16,854 ","1,843 ","9.9
CN010230,01,023,"Choctaw County, AL",2009,"4,925 ","4,299 ","626 ",12.7
CN010250,01,025,"Clarke County, AL",2009,"10,300 ","8,552 ","1,748 ",17.0
CN010270,01,027,"Clay County, AL",2009,"5,474 ","4,603 ","871 ",15.9
```

3. Скачиваем из свободного доступа файл **.svg** той территории, которую хотим анализировать (в нашем примере США). Открываем его в текстовом редакторе.

Вот как выглядит один из контуров данной карты:

```
<path style="font-size:12px;fill:#d0d0d0;fill-rule:nonzero;stroke:#000000;stroke-
opacity:1;stroke-width:0.1;stroke-miterlimit:4;stroke-dasharray:none;stroke-linecap:butt;marker-
start:none;stroke-linejoin:bevel" d="М 62.678745,259.31235 L 63.560745,258.43135 L
64.220745,257.99135 L 64.439745,258.43135 L 64.000745,258.65135 L 64.439745,258.65135 L
66.643745,257.99135 L 68.626745,255.56635 L 70.388745,256.44835 L 70.388745,256.89035 L
69.727745,257.54935 L 69.727745,258.21235 L 70.388745,257.99135 L 70.829745,256.89035 L
71.269745,256.44835 L 71.930745,257.10835 L 72.150745,257.99135 L 72.811745,258.21235 L
73.030745,257.77135 L 74.131745,257.54935 L 75.894745,257.54935 L 76.113745,257.77135 L
75.673745,258.43135 L 75.673745,258.65135 L 76.996745,258.87235 L 76.774745,259.53235 L
77.656745,259.53235 L 78.757745,258.87235 L 81.180745,258.65135 L 82.722745,259.09235 L
83.386745,259.09235 L 84.044745,259.31235 L 84.267745,259.53235 L 85.148745,259.53235 L
86.249745,259.31235 L 87.572745,259.31235 L 89.114745,259.75435 L 89.554745,259.53235 L
90.436745,258.87235 L 90.655745,258.65135 L 91.096745,258.21235 L 92.639745,258.43135 L
96.163745,259.53235 L 97.264745,263.05835 L 97.925745,265.26135 L 88.893745,267.46435 L
89.334745,269.88635 L 87.572745,270.32735 L 82.945745,271.21135 L 82.722745,271.21135 L
72.371745,272.31135 L 69.947745,272.31135 L 69.947745,271.87035 L 68.186745,271.87035 L
68.186745,271.42935 L 66.423745,271.42935 L 64.661745,271.64935 L 63.338745,271.64935 L
63.338745,271.21135 L 62.678745,271.21135 L 62.678745,271.42935 L 60.696745,271.42935 L
60.255745,271.21135 L 60.034745,271.21135 L 60.034745,271.42935 L 59.154745,271.42935 L
58.932745,270.98935 L 57.831745,270.98935 L 57.831745,271.42935 L 57.389745,271.42935 L
54.304745,271.21135 L 54.304745,272.08935 L 52.762745,272.08935 L 51.441745,271.42935 L
50.780745,270.54735 L 51.220745,269.22735 L 51.441745,267.68335 L 52.983745,267.90535 L
54.967745,267.68335 L 55.626745,267.46435 L 56.948745,265.92135 L 57.611745,263.93935 L
58.932745,261.95735 L 59.814745,261.07435 L 60.474745,261.29735 L 61.356745,260.85535 L
62.678745,259.31235" id="02185" inkscape:label="North Slope, AK"/>
```

Примечание: в данном примере нас будет интересовать **id** контура в **.svg** и **FIPS** (**F**ederal **I**nformation **P**rocessing **S**tandard) в **.csv**. Это один и тот же номер, т.е. это *нить* между географическим контуром региона в файле контурной карты и данными для отображения.

4. Чтобы соединить и обработать два файла используем «*BeautifulSoup*». Это HTML/XML парсер для *Python*, который может превратить даже невалидную разметку в удобное дерево для парсинга. Он предоставляет простые, идиоматические пути навигации, поиска и изменения дерева для парсинга. Данная библиотека способна сэкономить многие часы работы программиста.
5. Теперь нам нужно получить файл **.svg**, раскрашенный в соответствии со статистическими данными, т.е. нам нужно
 - а. продумать *шкалу цветов* в зависимости от статистического значения
 - б. назначить *цвет контуру*
 - в. сохранить *новый файл .svg*
6. Пишем скрипт, запускаем его и на выходе получаем новый файл **.svg**.

Вот как выглядит полная версия скрипта с комментариями:

```

7.  ### colorize_svg.py
8.
9.  import csv
10. from BeautifulSoup import BeautifulSoup
11.
12. # Read in unemployment rates
13. unemployment = {}
14. min_value = 100; max_value = 0
15. reader = csv.reader(open('unemployment09.csv'), delimiter=",")
16. for row in reader:
17.     try:
18.         full_fips = row[1] + row[2]
19.         rate = float( row[8].strip() )
20.         unemployment[full_fips] = rate
21.     except:
22.         pass
23.
24.
25. # Load the SVG map
26. svg = open('counties.svg', 'r').read()
27.
28. # Load into BeautifulSoup
29. soup = BeautifulSoup(svg, selfClosingTags=['defs', 'sodipodi:namedview'])
30.
31. # Find counties
32. paths = soup.findAll('path')
33.
34. # Map colors
35. colors = ["#F1EEF6", "#D4B9DA", "#C994C7", "#DF65B0", "#DD1C77", "#980043"]
36.
37. # County style
38. path_style = 'font-size:12px;fill-rule:nonzero;stroke:#FFFFFF;stroke-opacity:1;
39. stroke-width:0.1;stroke-miterlimit:4;stroke-dasharray:none;stroke-linecap:butt;
40. marker-start:none;stroke-linejoin:bevel;fill:'
41.
42. # Color the counties based on unemployment rate
43. for p in paths:
44.
45.     if p['id'] not in ["State_Lines", "separator"]:
46.         # pass
47.         try:
48.             rate = unemployment[p['id']]
49.         except:
50.             continue
51.
52.
53.         if rate > 10:
54.             color_class = 5
55.         elif rate > 8:
56.             color_class = 4
57.         elif rate > 6:
58.             color_class = 3
59.         elif rate > 4:
60.             color_class = 2

```

```
61.         elif rate > 2:
62.             color_class = 1
63.         else:
64.             color_class = 0
65.
66.         color = colors[color_class]
67.         p['style'] = path_style + color
68.
69.     print soup.prettify()
70.
```

Инфографика готова.

ЗАКЛЮЧЕНИЕ

В данном докладе мы прошли большой путь от рассмотрения того, *что движет человеком* во время его пребывания в незнакомой среде, *чего он хочет* и *как сделать сайт эффективнее до последних тенденций и технологий*, используемых в вэбе, которые существенно упрощают жизнь не только простых людей, но и дизайнеров и вэб-разработчиков.

Надеюсь, вам понравилась эта презентация и доклад. До свидания.

P.S.

На самом последнем слайде я привожу список источников, которые я использовал при подготовке презентации и доклада.