# Матричный навигатор для маленького сайта

Сразу оговорюсь. Я не являюсь специалистом по изготовлению веб-страниц. Последний профессиональный сайт я разрабатывал в первые годы нашего века.
Описанный здесь опыт касается моей приватной страницы.
Дело было так. Более пятнадцати тому назад завел я себе веб сайт www.sirotin.eu, поместил туда информацию о моих книгах и ... не прикасался к нему с тех пор. А прогресс не стоял на месте. Сайт мой  безнадежно устарел. Вот и решил я его так модернизировать, чтобы:

1. Страницы сайта можно было бы читать и на мобильных телефонах и на больших экранах (поддерживался Responsive Design)
2. Поскольку на сайте предполагается иметь информацию на трех языках (русском, английском, немецком), пользователь должен иметь возможность выбрать подходящий ему язык.
3. Входная страница определяет основной (default) язык пользовательского бровзера и автоматически включает для него этот язык.

Как это сделать? Скачивать и вникать в сложные библиотеки не хотелось. Интуиция подсказывала, что кода должно быть строчек 50-60.
Погуглил - ничего не нашел. Порылся на GitHub - решений море, но все вплетено в набор других, мне не нужных фичей. И выплести их оттуда не очень понятно как. Задал вопрос на StackOverflow - но никто на него не ответил.

Что делать - засучиваем рукава и беремся делать сами.
Разумеется, совсем без сторонних библиотек дело не обошлось. Проблему Responsive Design решает замечательно Bootstrap. Про него я слышал и представлял себе как графический инструмент, который генерирует страницы. Оказалось - это относительно компактная и понятная Javascript библиотека к которой есть гениальный вводный курс на W3School (http://www.w3schools.com/bootstrap/ ). Скачиваем,  разбираемся, пробуем. Странички с одноязыковыми меню скалируются от малюсенького экрана смартфона до самого большого из доступных дисплеев. Берём.

Проблема определения языка браузера и переключения языка решена много раз в самых разных GitHub проектах. В конце-концов путем отбрасывания лишнего и добавления своего необходимого получился вот такой код (страница index.shtml в главной папке проекта).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Redirection</title>
</head>
<body>
<div class="container">
    <p>You will be redirected to your language variant. If you see this page 3
seconds please click <a href="rub_main/lang_en/index.shtml">here</a>.</p>
</div>
<script>
    const DEFAULT_LANG_FOLDER = "lang_en";
    const SUPPORTED_LANGS =["en", "de", "ru"];
    var folder = "lang_en";
    var lang = window.navigator.userLanguage || window.navigator.language;
    if(lang){
        var pos_undl = lang.indexOf("-");
        if(pos_undl != -1){
            lang = lang.substring(0, pos_undl);
        }
     };
```

```
     if(SUPPORTED_LANGS.indexOf(lang) >=0){
        folder = "lang_" + lang;
     }
    window.location = "rub_main/" + folder + "/index.shtml";
</script>
</body>
```

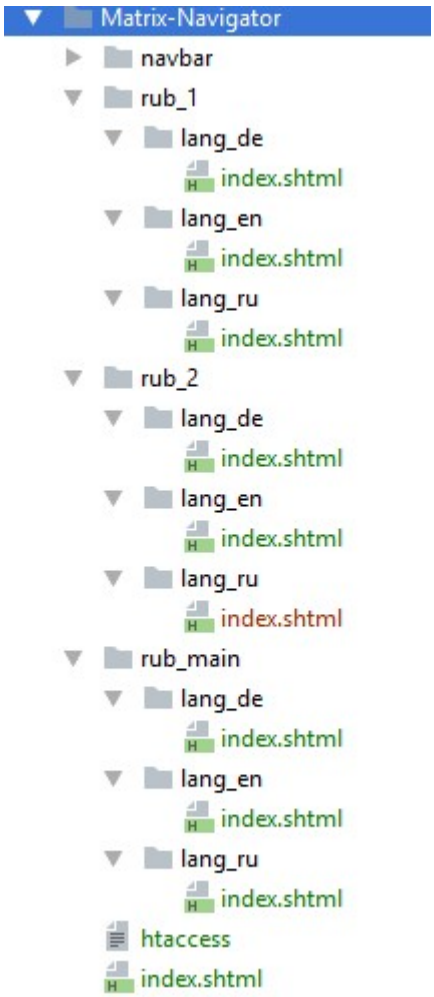
Идея кода проста. Сначала пытаемся определить код страны, который задан в браузере, вызвавшим эту страницу и записать его в переменную lang.
Если это удалось, сравниваем его с кодами, поддерживаемыми сайтом. Если одно из сравнений закончилось положительно, используем код страны для формирования имени папки (переменная folder). В случае неудачи с определением кода или если установленный в браузере язык не поддерживается сайтом, пользователь будет перенаправлен на англоязычную страницу.

Осталась главная проблема - многоязычных меню. После нескольких неудачных наскоков вдруг кристаллизовалось понимание, что предметная область может быть абстрагирована до матрицы или таблицы. Собственно поэтому я и назвал навигатор матричным.

Колонками в ней являются тематические рубрики, строками - языки, а в "клетках" таблицы находятся вводные страницы в тему рубрики на соответствующем языке. Например - для рубрики (колонки нашей воображаемой таблицы) существует три клеточки - по одной на каждый из  языков. А для каждого языка существует столько "клеток" таблицы, сколько рубрик имеется на сайте. При этом для каждой клетки (комбинации рубрика/язык) существует страница на которую осуществляется переход в случае выбора соответствующего пункта меню.Все эти страницы имеют одно и тоже имя -  index.shtml

| | Главная рубрика | Рубрика 1 | Рубрика 2 |
|---|---|---|---|
| Русский язык | index.shtml | index.shtml | index.shtml |
| Английский язык | index.shtml | index.shtml | index.shtml |
| Немецкий язык | index.shtml | index.shtml | index.shtml |

Отображаем эту концепцию в виде структуры папок сайта. Поскольку файловая структура папок может быть только иерархическая, отображаем нашу таблицу в виде двухуровневого дерева.

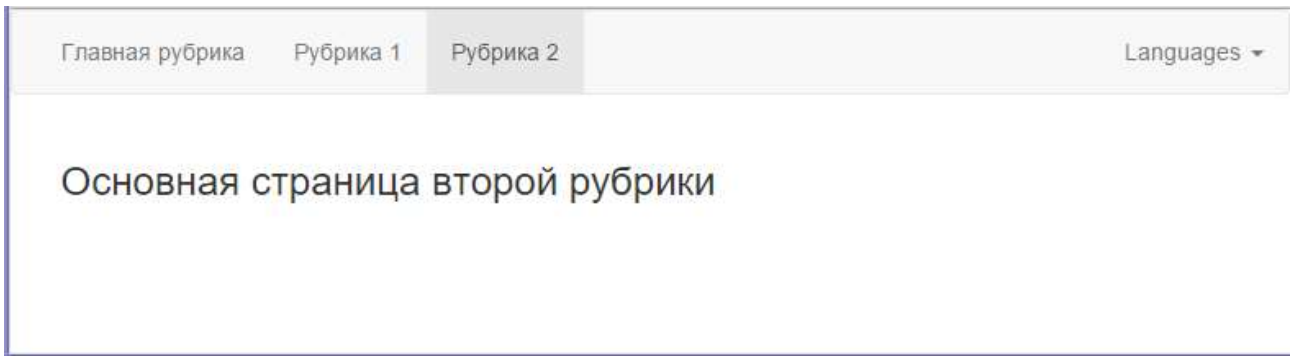



Все страницы, показывающие меню, должны как-то уметь его загружать и показывать. Вопрос - каковы будут затраты, если через какое-то время вы захотите что-то изменить в этом общем меню? Теоретически мыслимы две альтернативы помещения меню на страницу: статический и динамический методы.

В случае статического метода информация о меню будет просто "вписана" на каждую страницу как часть её кода. Если надо изменить меню, все страницы надо переписать. Это реально, если ваши страницы всё равно генерируются какой-то CMS (Content Management System) или у вас под руками есть хороший репарсер, способный интеллигентно изменять содержание файлов (лежащих, кстати, на сервере). Если у вас таких возможностей нет, лучше пользоваться динамическим методом.

В этом случае мыслимы опять таки два варианта. При первом варианте остов страницы загружается в браузер. Страница содержит Javascript функцию, которая загружает файл с информацией о меню с сервера и конвертирует эту информацию в DOM  - элементы страницы. Информация меню может лежать на сервере в виде HTML фрагмента.

Очевидно, что загружаемый фрагмент должен быть один (не зависеть от выбранного языка). При этом страница, загруженный фрагмент или они вместе должны суметь настроить меню на язык браузера. В моих экспериментах я ne смог этого добиться на всех современных браузерах только с помощью JavaScript. И потому я выбрал комбинированный метод из JavaScript и SSI (Server Side Include). Суть SSI - технологии в том, что на основании URL зарошеннй страницы и установкам в файле htacess сервер "знает" что страница содержит "закладки", которые надо заменить на текст из файла, указанного в закладке.




Моя тестовая страница, выглядящая так как показано на скриншоте вверху, имеет такой HTML код:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Path and parameter using example</title>
<!--#include virtual="../../navbar/navbar1.htm" -->
</head>
<body>
<!--#include virtual="../../navbar/navbar2.htm" -->
<div class="container">
    <h3>Основная страница второй рубрики</h3>
</div>
</body>
```

Мы видим здесь две закладки, которые одинаковы для всех страниц сайта:
`<!--#include virtual="../../navbar/navbar1.htm" -->`
и
`<!--#include virtual="../../navbar/navbar2.htm" -->`

Первая содержит ссылки на используемые файлы  с JavaScript библиотеками и стилями.

Вторая закладка побольше. Текст её приведен внизу.

```
<nav class="navbar navbar-default" id="nav_bar">
    <div class="container-fluid">
        <ul class="nav navbar-nav" >
            <li id="rub_main"><a href="../../rub_main/lang_en/index.shtml">Main
Rubric</a></li>
        </ul>
        <ul class="nav navbar-nav">
            <li  id="rub_1" ><a
href=href="../../rub_1/lang_en/index.shtml">Rubric 1</a></li>
        </ul>
        <ul class="nav navbar-nav">
            <li id="rub_2" ><a
href=href="../../rub_2/lang_en/index.shtml">Rubric 2</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right" >
            <li class="dropdown"><a class="dropdown-toggle" data-
toggle="dropdown" href="#">Languages <span class="caret"></span></a>
                <ul class="dropdown-menu" >
                    <li id="lang_en"><a
href="../../rub_main/lang_en/index.shtml">English</a></li>
                    <li id="lang_de"><a
href="../../rub_main/lang_de/index.shtml">Deutsch</a></li>
                    <li id="lang_ru"><a
```

```
href="../../rub_main/lang_ru/index.shtml">Русский</a></li>
                </ul>
              </li>
          </ul>
      </div>
</nav>

<script>
    const RUBRIC_NAMES_TABLE = '{' +
            '"en" : {"rub_main" : "Main Rubric",     "rub_1" : "Rubric 1",
"rub_2" : "Rubric 2"},' +
            '"de" : {"rub_main" : "Hauptrubrik",     "rub_1" : "Rubrik 1",
"rub_2" : "Rubrik 2"},' +
            '"ru" : {"rub_main" : "Главная рубрика", "rub_1" : "Рубрика 1",
"rub_2" : "Рубрика 2"}' +
             '}';

    const RUBRICS =["rub_main", "rub_1", "rub_2"];

    $(document).ready(process_href());
    function process_href(){
        var href =  window.location.href;
        var posLang = href.indexOf("/lang_");
        if(posLang > 0){
            var lang = href.substring(posLang + 6, posLang + 8);
            var names = JSON.parse(RUBRIC_NAMES_TABLE)[lang];
            var posRub = href.indexOf("/rub_");
            if(posRub > 0){
                 var activeRubric = href.substring(posRub + 1, posLang);
                 resetRubrics(lang, names, activeRubric);
            }
        }
    }
    function resetRubrics(lang, names, activeRubric){
          for(i = 0; i < RUBRICS.length; i++){
            var rub  =RUBRICS[i];
            resetRubric(rub, names, lang, activeRubric);
          }
     }

     function resetRubric(rub, names, lang, activeRubric){
        var el = $( "#" + rub);
        var className = "";
        if(activeRubric == rub){
            className = "active";
        }
        el.toggleClass(className);
        var inHTML = "<a href='../../" + rub + "/lang_" + lang +
"/index.shtml'>" + names[rub] + "</a>"
        el.html(inHTML);
    }
</script>
```

HTML скрипт внутри элемента <**nav …**</**nav**> задаёт элеметы меню. Пункты меню сделаны так, что по клику происходит переход к соответствующей странице. Она начнет загружаться и тогда заработает JavaScript внутри <**script**>…</**script**>.

Главную работу выполняет функция process_href() Она разбирает URL загружаемой страницы на составные части и подстраивает пункты меню под новый язык пользователя.

Ну вот и готово. В итоге все получилось настолько просто, что даже кажется, в приличном программистском обществе упоминать об таком решении стыдно. 60 строк JavaScript кода

Вот и хорошо. Именно такие простые до противного решения долго живут и надежно работают..
Хотя разумеется далеко не факт, что решение действительно надежно работает. Для этого его надо опробовать не только на моем сайте. Ктати, его адрес – www.sirotin.eu Так что если у вас есть интерес - пользуйтесь. Улучшайте. Давайте делать это вместе в проекте https://github.com/vsirotin/Matrix-Navigator