# Технические требования к HTML-вёрстке

## 1. Общие требования

a. HTML5
b. CSS3, допустимо применение наборов css-инструментов bootstrap и аналогов
c. Любая HTML-страница должна быть создана в соответствии со стандартами, предписанными W3C (всемирный Интернет консорциум), и как следствие проходить тест на валидность, не выдавать ошибок (errors) и, по возможности, замечаний (warnings).
d. Проверять валидность следует с помощью онлайн сервиса http://validator.w3.org
e. Необходимо обязательно объявлять HTML5 DOCTYPE в первой строке HTML документа: <!DOCTYPE html>.
f. HTML-страница и сопровождающие её файлы должны быть оптимизированы по физическому размеру

## 2. Оптимизация графики

a. Каждый графический файл должен быть реализован в том графическом формате (*.gif, *.jpeg, *.png), который обеспечивает его минимальный размер без видимой потери качества отображения.
b. Графические файлы, обеспечивающие отображение элементов пользовательского интерфейса в различном состоянии (при наведении, при клике), должны быть выполнены спрайтами – логическими объединенными графическими файлами.

## 3. Оптимизация технического кода

a. Классы обязательно должны иметь приставки (“префиксы”), чтобы их названия не совпадали с классами, которые присваивает CMS по умолчанию. Например:
Вместо search использовать ngl-search
b. Необходимо минимизировать определения классов в HTML коде, главным образом за счет иерархического доступа и предельного сокращения имён, но не в ущерб читаемости.

```
<!--
Не правильно:
- не используется возможность иерархического доступа
- классы названы избыточно
 -->

<style>
div.superDiv {...}
ul.ulSuperDiv {...}
li.itemUlSuperDiv {...}
</style>

<div class="superDiv">
<ul class="ulSuperDiv">
<li class="itemUlSuperDiv">item 1</li>
<li class="itemUlSuperDiv">item 2</li>
<li class="itemUlSuperDiv">item 3</li>
</ul>
</div>

<!--
Правильно:
- используется возможность иерархического доступа
- классы названы минимально допустимо
 -->

<style>
div.sd {...}
div.ds ul {...}
div.ds ul li {...}
</style>

<div class="sd">
<ul>
<li>item 1</li>
<li>item 2</li>
<li>item 3</li>
</ul>
</div>
```

c. Все элементы, имеющие порядковую нумерацию должны быть выполнены с помощью тегов <ul><ol></ol></ul> либо нумерация должна задаваться с помощью правил css
d. Следуйте правилу DRY (do not repeat yourself) - одинаковые блоки на разных страницах должны иметь одинаковую структуры и одинаковые классы тэгов. Идеальный вариант - вынесенные в отдельные файлы.
e. Максимально использовать однотипные блоки, если их структура одинаковая. Различия определять одним максимум родительским тегом с определенным классом.
f. Если отображение элемента зависит от четного или нечетного порядкового номера, не использовать дополнительных классов img-left, text-right и подобные. Различие задавать псевдоклассами стилей.
g. В контент-зонах не ипользховать div с классами, использовать только теги для структуры текстового контента без классов - em, strong, h1,h2,h3, blockquote и т.д.
h. Не использовать загрузку js-скриптов с помощью другого скрипта - это задача cms и, возможно, CDN
i. Оптимизация под https://developers.google.com/speed/pagespeed/insights/

## 4. Кодировка

a. HTML-страницы, а также сопровождающие их текстовые файлы (*.html,

*.css, *.js и др.) должны представляться в кодировке UTF-8 без BOM, который является международным стандартом и гарантирует правильное отображение HTML страниц и их текстового содержания на любом конечном устройстве.

b. В файлах HTML необходимо указывать мета тег с кодировкой страницы: <meta charset="utf8">
c. Разметка HTML должна быть семантической.

5. **Контент**
   a. Текст должен оставаться в рамках родительского блока при переполнении.
   b. Текст не должен обрезаться или вываливаться из родительского блока при переполнении.
   c. Текст не должен смещать другие блоки.
   d. Родительский блок должен сохранять минимальные размеры при неполном наполнении.
   e. Слова, длиннее минимальной ширины, должны переноситься.
   f. Контент больше ширины родителя не должен выходить за его пределы, ломать сетку или смещать другие блоки.
   g. При увеличении количества элементов они должны оставаться в рамках родительского блока.
   h. Элементы могут переноситься на следующую строку при уменьшении размера родителя.
   i. При минимальном количестве элементов или их отсутствии родительский блок должен сохранять минимальные размеры.
   j. Последние блоки в сетках должны выравниваться по направлению текста.
6. **Стайлгайд**
   a. Все состояния элементов должны соответствовать стайлгайду, предусмотренному в макете. Стайлгайд — это часть дизайна проекта, он имеет наивысший приоритет. При наличии стайлгайда верстальщик должен ему следовать.
   b. Если в стайлгайде не предусмотрены состояния элемента, то их нужно реализовать на своё усмотрение.
   c. Для элементов форм, кнопок и ссылок:`hover`, `focus`, `active`.
   d. Для элементов форм: `disabled`.
   e. При взаимодействии с элементами (наведение, нажатие) ни сам элемент, ни окружающие его блоки не меняют своего положения, если такое поведение не предусмотрено макетом.
7. **Используемые вспомогательные технологии**
   a. В случае возникновения необходимости в использовании JavaScript, в качестве основной технологии следует использовать библиотеку jQuery (http://www.jquery.com).
   b. Не использовать Flash
   c. В случае возникновения необходимости в использовании нестандартных шрифтовых наборов, следует использовать технологию Google Fonts или сервис fontsquirrel.com

## 8. Файловая организация

a. Все пути к ресурсам верстки должны быть относительными
b. В корневой директории размещаются файлы *.html и директория ресурсов верстки assets
c. Директория "assets/img" — все графические файлы
d. Директория "assets/css" — все файлы CSS
e. Директория "assets/js" — файлы JavaScript
f. Все шрифты, за исключением шрифтов google fonts, необходимо размещать в папке с css файлами.
g. В папках не должно быть картинок, шрифтов и других файлов, не использующихся в верстке. Если что-то исключили из верстки или переделали — не забывайте удалять уже ненужные картинки.
h. Для картинок временного контента использовать папку tmp

## 9. Совместимость с браузерами и их версиями

a. Поддерживаемые версии броузеров: Mozilla Firefox, Opera, Safari, Google Chrome, MS IE, Edge - актуальные версии, поддерживаемые разработчиком.
b. Сайт адаптивный, рабочие области по макету.

## 10. Оформление вывода контента

a. Необходимо задавать стили для стандартных тегов контент зоны: P, STRONG, B, EM, UL, OL, LI, H1-H6, BLOCKQUOTE
b. Ожидаемыми являются отступы, в том числе и относительные (например, при вложенных списках)
c. Ожидаемыми являются идентичные шрифтовые наборы, если это подразумевается дизайном. По умолчанию это подразумевается. Таким образом, если не предусмотрено дизайн-макетом, то в контентной области одним и тем же должны быть свойства шрифта и текстового набора у всех перечисленных выше тегов.
d. Стили изображения должны предусматривать то что возможно будет использована разная геометрия картинок для thumbnail preview - обрезать или масштабировать так чтобы при любой геометрии картинка соответствовала размерам из дизайна.
e. Теги заголовков, абзацев, списков и других текстовых блоков не должны иметь классов и оборачиваться в другие теги.

## 11. Форматирование HTML-кода

a. Каждая новая строка должна отбиваться двумя пробелами. Все вложенные друг в друга элементы должны визуально выделяться при просмотре HTML-кода в браузере.
b. Комментарии к коду необязательны, а избыточные не приветствуются и должны удаляться.
c. Названия классов и id должны по смыслу соответствовать применению например: header, menu, footer, news
d. Разделять блоки страницы комментариями. <!-—header begin --><!-—- header end -->.
e. Если графический элемент имеет разное отображение в “спокойном”

состоянии и “под курсором” (:hover), этот элемент должен иметь одно “склеенное” изображение с изменяемой видимой частью в различных состояниях. Например, фиксированная высота (height) и разное значение css-атрибута background-position или margin. Если не придерживаться этого правила и использовать разные картинки для “спокойного” состояния и “под курсором”, при наведении курсора будет пропадать основной элемент, а новый будет появляться только после загрузки.

f. В коде разместить html-комментарий, содержащий словосочетание Facta, non verba
g. Навигация (меню) – использовать тэги ul, li. Не использовать атрибут класса для тега ul, для li, использовать только active, current или подобный, если нужно его выделить среди остальных.
h. Использовать таблицы (table) только для табличных данных.
i. Кнопки являются кнопками, поля в форме – полями в форме, и не забываем про label. Форма обязательно должна быть оформлена с внешним тегом <form></form>
j. Использовать атрибут стиля whitespace: nowrap для текстовых элементов, которые ни в коем случае не должны переноситься на новую строку. Например, для элементов навигации (меню).
k. Сквозные элементы (шапка, подвал, боковые панели) и повторяющиеся элементы (сокращенные варианты записей, отображение товаров и т.д.) имеют одинаковый код и, либо выносятся в отдельный файл и включаются (методами gulp.js-плагинов), либо выделяются одинаковыми комментариями html (см. пункт про комментарии html-кода)
l. Формат постраничной навигации:

```
<nav class="navigation pagination" role="navigation">
        <div class="nav-links">
        <span class='page-numbers current'>1</span>
        <a class='page-numbers' href='#'>2</a>
        <a class='page-numbers' href='#'>3</a>
        <a class='page-numbers' href='#'>4</a>
        <a class='page-numbers' href='#'>5</a>
</div>
</nav>
```

m. Верстка будет использована для темы CMS, файлы темы будут лежать в папке domain.com/wp-content/themes/themename/, при этом возможно также размещение самого сайта на domain.com/demo/site1/ Таким образом нужно продумать чтобы все ссылки до скриптов и до картинок в html и js можно было задать непосредственно в html в абсолютном виде. Можно использовать переменные скриптом в html, либо атрибуты тегов ( data-themepath=”domain.com/wp-content/themes/themename”).
n. Текстовый контент и картинки, для которых возможна замена (например, слайдеры) должны брать картинки из html-кода
o. Сниппет для переключения языков:

```
<ul class=”languages”>
```

```
        <li><a href="#">Русский</a></li>
        <li><a href="#">English</a></li>
    </ul>
```

## 12. JavaScript

a. Сопровождающий js-код (находящийся в теге SCRIPT) по возможности должен быть вынесен во внешние файлы – один или несколько в зависимости от логики и объёма.
b. Все что можно сделать без Javascript, делать без него.
c. Комментировать, для чего служит Javascript (можно не комментировать применение стандартных библиотек jquery, jquery-ui и других популярных)
d. Должен осуществляться вывод в консоль браузера (console.log) текста quos ego!
e. Не использовать без согласования никаких обработчиков событий (например, отправки форм).

## 13. Gulp.js (если применяется)

a. В исходном несобранном коде не применяется минимизация и объединение файлов css/js
b. Собранная верстка располагается в папке dist.
c. В собранном варианте (dist) применяется минификация и объединение файлов css/js
d. Все плагины gulp.js – публично доступные или передаются отдельно.

## 14. Разршенения и броузеры

a. Оптимизировать под следующие экраны:
    i. 320-768 пикселей - мобильные телефоны
    ii. 992-1199 пикселей - планшеты и средние ноутбуки
    iii. 1200 пикселей и более - большие экраны
b. Оптимизировать под следующие броузеры последних версий старше одного года:
    i. Mozilla Firefox
    ii. Google Chrome
    iii. Safari
    iv. MS Edge