Редакция от Февраля 2018

Общие требования и рекомендации по верстке

**0. Общие положения**

* Адаптивная верстка типовых страниц интернет-ресурса в формате языка гипертекстовой разметки (HTML), включая каскадные таблицы стилей (CSS) и базовый функционал манипуляций объектной модели документа (DOM) средствами JavaScript (JS). Определение базового функционала - см. пункт 13
* Верстка должна соответствовать макетам. При этом верстка не выполняется методом Pixel Perfect (если не обсуждается отдельно) и при наложении прозрачного слоя возможны отличия с относительной погрешностью до 5% (размеры элементов, рендеринг шрифтов и межстрочных/межбуквенных расстояний, а также внутренних и внешних отступов). Например в дизайне отступ от секции 100px, в верстке ставим 100px как измерялось в макетах, а на практике получилось 105px из-за того что сыграл line-height заголовка
* Все текстовые элементы макетов должны верстаться текстом, за исключением случаев, где подобная реализация невозможна технически
* Все изменения :hover состояний, и :focus эффекты полей вида input сопровождаются микро анимациями и transition эффектами

**1. Перед стартом / проверка макетов**

* Если макеты в PSD, Проверьте настройки вашего рабочего профиля. Должен быть установлен sRGB. Если по какой то причине дизайнер прислал файлы в другом профиле, все равно открывайте его в рабочем sRGB. [как установить профиль](https://www.color-management-guide.com/color-settings-photoshop.html)
* Заранее проверьте макеты на предмет возможности экспорта векторных файлов, режимов наложения и технический возможности реализации задумок дизайнера.
* Проверяйте скрытые слои если таковые присутствуют
* В процессе работы проверяйте стили текстов в плане line-height, letter-spacing и др. Часто это упускается из виду
* Проверьте сетку макета и спланируйте работы. Иногда сетки имеют gutter space в 20px или 50px, а не стандартные 30px. Это поможет избежать необходимости в костылях в дальнейшем
* Пробегитесь [по этому списку](https://github.com/nicothin/web-design) (гайд для дизайнеров по верстке, но полезно и в обратном направлении)

**2. Поддержка браузеров**

* HTML-шаблоны должны корректно отображаться в следующих версиях браузеров
* Internet Explorer версии 10 и выше, платформа — операционные системы семейства Windows;
* Mozilla Firefox версии 28 и выше — Windows версии XP и выше и Mac OS X версии 10.8 и выше;
* Safari версии 6.1 и выше — Mac OS X версии 10.8 и выше;
* Google Chrome версии 21 и выше — Windows версии XP и выше и Mac OS X версии 10.8 и выше;
* Opera версии 15 и выше — Windows версии XP и выше и Mac OS X версии 10.8 и выше;
* Браузеры мобильных устройств iOS 7 и выше
* Браузеры мобильных устройств Android 5 и выше, за исключением UC Browser
* При верстке учитываются особенности браузеров, их ограничения внешнего вида и поведения объектов. Для обеспечения правильного отображения элементов в разных браузерах может применяться грациозная деградация.

**3. Стиль программирования**

* Табуляция кода через двойной пробел. Можно использовать плагины в котором tab превращается в двойной пробел (так например делает редактор atom по умолчанию). Требование связано с единостью стиля чтобы компиляция не падала на ошибках вложенности
* Обязательно комментирование крупных смысловых блоков в целях читаемости
* Верстка может быть невалидной по валидатору W3C
* Осмысленное наименование классов по [BEM методологии](https://ru.bem.info/method/naming-convention/) (классический стиль block\_\_element--modifier). Обязательно загляните [сюда](http://nicothin.github.io/idiomatic-pre-CSS/#bem)

**4. Сборка**

* Новый проект нужно начинать на основе [этой Gulp.js сборки](https://github.com/dpmango/gulp-starter-pack) (доработанная версия сборки Артюха). Есть индексная страница, все препроцессоры, результат собирается в dist который игнорируется в git и не конфликтует при пуше. Удобно настроена работа с svg и png спрайтами. По умолчанию установлены все необходимые js плагины и barba.js
* Зависимости которые собираются в js и css продакшен-кода должны быть установлены через bower

**5. Адаптивность**

* Под адаптивной версткой подразумевается отзывчивая (англ. - “responsive”) верстка, при которой содержимое страницы подстраивается под фактические размеры окна браузера, без рывков по ширине глобальных контейнеров на ключевых точках (“брейкпойнтах”). Другими словами, глобальные контейнеры всегда должны быть 100% ширины
* При размерах более 320px не должно возникать горизонтальной прокрутки страницы (за исключением отдельных блоков верстки, где подобная реализация предусмотрена дизайн-макетами).
* Допускается использование desktop first методов в подходе
* Ключевыми “брейкпойнтами” принято считать - 1920(hd+), 1440 (hd), 1200px (wide), 992px (desktop), 768px (tablet), 568px(mobile), 414px(mobile-s). Применяются для адекватного, пропорционального изменение отступов и размеров шрифтов. Либо, при наличии макетов мобильной версии, соответствии таковым
* Допускается использование других брейкпойнтов при необходимости с произвольным интервалом @media выражений

**6. Каскадные стили таблиц (CSS)**

* Для создания CSS должны использоваться пре/пост процессоры. В порядке приоритетности, исполнитель в праве выбрать: sass, postcss, less, stylus
* Крайне желательно использование “сахарного” синтаксиса, без ; и {} (прим. sass против scss). Либо исполнитель может разрабатывать промежуточные версии в scss синтаксисе и [конвертировать](https://www.sassmeister.com) в sass при готовности компонента
* Обязательное использование autoprefixer и flexbugs. При желании любые другие postcss плагины
* Недопустимо использование миксинов (@include) и экстендов(@extend) в целях общего понимания происходящего, пожалуйста без головоломок. Разработчики которые будут поддерживать проект не поймут что вы имеете ввиду под +cr, +bf, +padb, +flexjcsb
* Нельзя использовать вендорные префиксы в препроцессорах которые могут быть добавлены автопрефиксером. Исключениями могут быть -webkit-appearance, -webkit-overflow-scrolling и прочие подобные
* Предпочтительно использование собственной сетки на базе flexbox. В отдельных случаях допускается bootstrap 4 (обсуждается отдельно)
* Не рекомендуется использование !important без крайней необходимости. Исключениями могут быть перебивание стилей от вендорных библиотек (например .slick-slide - display: flex !important }
* Использовать normalize.css или reset.css

**7. Плагины**

* Предпочтительно, но не обязательно, использование следующих js/css плагинов (в случае необходимости использования их функционала)
* Карусели - slick, [swiper](http://idangero.us/swiper/) (поддержка flex и очень гибкая настройка), owl.carousel
* Модальные окна - magnific popup, fancybox
* Валидация форм - jQuery Validate Plugin.
* Валидация “маской” (прим. - поля ввода телефон) - [jQuery Mask Plugin](https://igorescobar.github.io/jQuery-Mask-Plugin/)
* Календарь [air-datepicker](http://t1m0n.name/air-datepicker/docs/index-ru.html)
* svg4everybody (поддережка svg для IE10), viewport-units-buggyfill (фикс для вьюпорта мобильных устройств с учетом адрес-бара. особенно актуально для iOS)
* scrollMonitor.js - в сборке настроено для эмуляции wow.js, только с гибкой настройкой и задержкой выполнения события. либо AOS
* anime.js / пакет GSAP - для сложных анимаций
* rellax.js - для parallax анимаций (плавность за счет использования requestAnimationFrame)
* barba.js - опциональный плагин. В существенной мере усложняет разработку JS из-за необходимости байдинга от $(document) и переинициализации всех скриптов при загрузке новой страницы
* [Lazy](http://jquery.eisbehr.de/lazy/) загрузка изображений, тоже опционально

**8. Javascript**

* Подробное комментирование в javascript коде, если функции имеют запутанный, нестандартный подход
* Функции должны группироваться смысловыми блоками (прим. сначала - общие функции, затем все карусели, затем все модальные окна и так далее). Простое дописывание кода в конец файла в ходе верстки крайне не рекомендуется
* При использовании ES6 синтаксиса, обязательна компиляция через babel в пределах /src
* Хорошей практикой считается использование в JS байдингах атрибутов либо классов с префиксом js- . В наименовании использовать синтаксис camelCase. Необходимо для разграничения разметки, чтобы при интеграции разработчику было понятно в каких элементов привязан функционал. Например \_document.on(‘click’, ‘[js-openTab]’, func.. ).
* Классы на изменения состояния должны быть с префиксами .is- . Например .is-opened, .is-active, .is-hovered и т.д. Но не модификаторами --active
* Кеширование jQuery объектов если используется более 2х раз внутри функции
* Использование debounce/throttle функций на .scroll, .resize и прочие ресурсоемкие задачи (есть в сборке)
* Округление рассчитываемых css значений через Math.floor() или Math.round (трансформы, размеры, марджины и т.д)

**9. Иконки и маленькие изображения**

* Первый приоритет - использование svg спрайтов.
* Второй приоритет - png спрайты через элемент <i>, <span>
* Третий приоритет - html теги img
* Все иконки должны быть оптимизированы для retina экранов. Экспорты слоев из photoshop/sketch происходит в разрешении 1x и 2x (1x без постфикса, 2x - filename@2x.png)
* Допускается использование только 2x изображений по умолчанию

**10. Изображения**

* Все изображения должны быть оптимизированы через сервис tinypng в целях уменьшения суммарного веса картинок. Стандартные gulp оптимизаторы не проявили достаточной эффективности
* При использовании inline png в css, картинка должна быть оптимизирована До перевода в base64
* Обязательно использование srcset параметров в случае если макеты позволяют экспортировать изображения в большем размере для 2x экранов с высокой плотностью пикселей src=’img/name.png srcset=’img/name@2x.png 2x
* Непрозрачные картинки необходимо экспортировать в .jpg в целях экономии размера файла

**11. Шрифты**

* Шрифты, доступные через сервис google fonts должны подключаться с CDN google.fonts
* В случае, если шрифты не находятся в бесплатном публичном доступе, исполнитель имеет право:
  + Запросить шрифт у заказчика
  + Использовать любой другой шрифт на свое усмотрение
* Используются колбеки типа шрифтов sans-serif, serif, monospace и т.д.
* Шрифты необходимо подключать в форматах eot, svg, ttf, woff, woff2. Используйте <https://transfonter.org>
* При подключении шрифтов одного семейства, перепишите font-weight в @font-face правиле чтобы в css можно было управлять жирностью и стилем через font-weight: 300 вместо font-family: ‘SuperFont-Light’, sans-serif

**12. Верстка элементов**

* Верстка должна проходить тесты на переполнения и незаполнения. То есть также быть отзывчивой по высоте элементов. В большинстве случаев недопустимы фиксированные значения height. Используйте min-height вместо него, чтобы при наличии больших текстовых блоков текст не вылезал из блоков или обрезался
* Текстовые блоки статей и user generated content (UGC) должны верстаться чистой разметкой пренебрегая правилами БЭМ. например блок с контентом новости/статьи или комментария пользователя который будет выводиться из админки должен содержать чистые теги p, ul, li, blockquote, а не p.article-text\_\_paragraph (наследование от родителя .article-content или .comment\_\_body)
* Избегайте наследования от глобального класса-модификатора от body, .global-wrapper и подобных для индивидуальных страниц. Для этого есть модификторы. Такая разметка неудобна при интеграции и конфликтует с barba.js
* Приоритетом динамичного изменения DOM являются css методы реализации. Например добавление класса .is-active с opacity: 0 -> opacity: 1 вместо .hide() и .show()
* Используйте pointer-events: none для объектов скрытых подобным образом чтобы не было ghost кликов по невидимкам

**13. Базовый функционал**

* Под базовым функционал манипуляций объектной модели документа (DOM) средствами JavaScript (JS) принять следующий перечень функций:
* Тогглеры состояний. Кнопки показать еще, скрыть должны раскрыть контент
* Открытие панелей мобильного меню по иконке-гамбургеру
* Показ/скрытие любых элементов, предусмотренных и отрисованных в макетах
* Поля отмеченные как иконка-календарь должна содержать интерактивный компоненты выбора даты (datepicker). Выбор компонента (плагина) осуществляется исполнителем, или в соответствии с заказчиком по предварительной договоренности
* Поля вида input могут иметь маски ввода, например телефон может иметь маску +7 999 999-99-99
* Валидация форм на базовые параметры (email, телефон, заполненность имя)
* Интеграция API решений заказчика и реальный функционал готового сайта не входит в понятие базового функционала верстки и обсуждается отдельно

**14. Минификация кода**

* Перед сдачей должен иметь версию минифицированного кода css/js.
* Допустимо, но не обязательно, использование минифицированного кода в каждой итерации
* Допустимо использование мастер-файла (bundle). Проект имеет один общий js файл и общий .css. Другими словами, если проект имеет 20 типовых шаблонов, не создается 20 отдельных файлов .min.js и .min.css, уникальных для каждой страницы

**15. Структура проекта**

* При использовании сборщиков, должны быть глобальные папки /src и /dist
* css файлы лежат в поддиректории /css
* javascript файлы лежат в поддиректории /js
* Файлы препроцессоров css должны быть разбиты на компоненты
* Приветствуется разбитие на pug компоненты элементов которые повторяются более 2х раз на разных страницах. Но не нужно переусердствовать с разбитем чтобы не держать одновременно 20 файлов перед глазами чтобы понять что происходит на странице
* media выражения должны лежать в том же файле компонента. Недопустимо использование общего \_media.sass файла в котором содержатся все @media

**16. Организационные моменты:**

* Хорошей практикой является предоставление ежедневной отчетности и информирование о ходе работ.
* Допускаются перерывы в работе до 2х дней. Работа в выходные дни и государственные праздники остается на усмотрение исполнителя
* Работа ограничена 3мя итерациями - основная и 2 правок. В первой исполнитель делает как считает нужным, но допускает незначительное вмешивание заказчика в рамках корректировки глобальных моментов на которые нужно обратить внимание, чтобы потом не переделывать двойную работу. Далее собирается пакет правок и отправляется на исправление. После этого может быть еще одна такая итерация правок.
* Обязательно ведение git реопзитория и домена на surge.sh

**17. Примеры проектов**

* <https://axur.surge.sh> (senior)
* [https://avalon2.surge.sh](http://avalon.surge.sh) (senior)
* <https://sancom.co> (senior)
* <https://dpmango.github.io/draw/src> (senior)
* [https://triskk.surge.sh](http://triskk.surge.sh) (senior, middle)
* <https://mashe.surge.sh> (middle)
* <http://carggo.surge.sh> (middle)
* <https://wanderbuild.surge.sh> (middle)
* <https://itboat-journal.surge.sh> (middle)
* <http://roof.surge.sh> (middle)
* <https://ortis.surge.sh> (middle)
* [https://aliworld.surge.sh](http://aliworld.surge.sh) (junior)
* <https://pm-logistics.surge.sh> (junior)

**Хмелевской Сергей**

<https://t.me/khmelevskoy_s>

<https://khmelevskoy.co>