HPACE
IT-Development For Business

Регламент-чеклист по верстке, frontend и SEO

от команды HPACE.dev

HPACE
IT-Development For Business

Общие рекомендации

Кроссбраузерность

Вёрстка должна корректно отображаться на последних версиях следующих браузеров:

- Google Chrome
- Firefox
- Android Browser
- Яндекс.Браузер
- Safari
- Microsoft Edge
- IE
- Opera (Desktop & Mobile)

Фоны (в чеклисте)

Фон для body и инпутов описывается всегда, даже если он белый.

Заголовки H1 (в чеклисте)

H1 на странице должен быть один. H1 должен отличаться от тега <title>.

Выносим js и css в отдельные файлы (в чеклисте)

Вроде бы само собой разумеющаяся вещь, но некоторые всё же допускают стили и скрипты прямо в html коде.

Выносим js и css в подвал сайта (в чеклисте)

По возможности выносим все js и css файлы в конец страницы и/или подключаем скрипты асинхронно, исключением являются, например, системные файлы битрикса, такие как **kernel_main.css** и **kernel_main.js**. Если стили страницы весят больше 1Мб, то выносим самые необходимые в отдельный файл и подключаем первыми, дабы страница не выглядела ужасно при загрузки сайта.

Минифицируем js и css файлы (в чеклисте)

Коммитим файлы вида style.css и style.min.css. Работаем в style.css, на страницах подключаем style.min.css. Аналогично с js файлами.

Удаляем лишний код (в чеклисте)

На странице не должно быть закомментированных кусков html, css и js. Если вам необходимо будет в дальнейшем вернуться к какому-либо закомментированному коду, это можно сделать по истории гита.
Неиспользуемых байтов в CSS и JS не должно быть больше 50% - Открываем консоль в Google Chrome, смотрим вкладку Coverage.

Идентификаторы не используются в CSS (в чеклисте)

ID используются исключительно для работы JS скриптов, в стилях мы их не используем.

Оптимизация изображений (в чеклисте)

Изображения необходимо оптимизировать после распаковки из макета. Делается это любыми десктопными и веб-приложениями по типу http://kraken.io

Разделение изображений (в чеклисте)

Если изображение не относится к стилю сайта, т.е. оно контентное (какая-нибудь диаграмма, фото человека, оставившего отзыв), то данное изображение должно быть в html как <img> и наоборот.
Стилевые мелкие изображения (иконки) выводим в .svg.

Адаптивные изображения (в чеклисте)

Если на сайте присутствует адаптивная вёрстка, то необходимо использовать плагины для подгрузки оптимизированных изображений под конкретные устройства. Как пример, HISRC https://github.com/teleject/hisrc или blazy - в итоге подключаем различные изображения:

```
<img src="//placekitten.com/200/100" data-1x="http://placekitten.com/400/200" data-2x="http://placekitten.com/800/400" class="hisrc" />
```

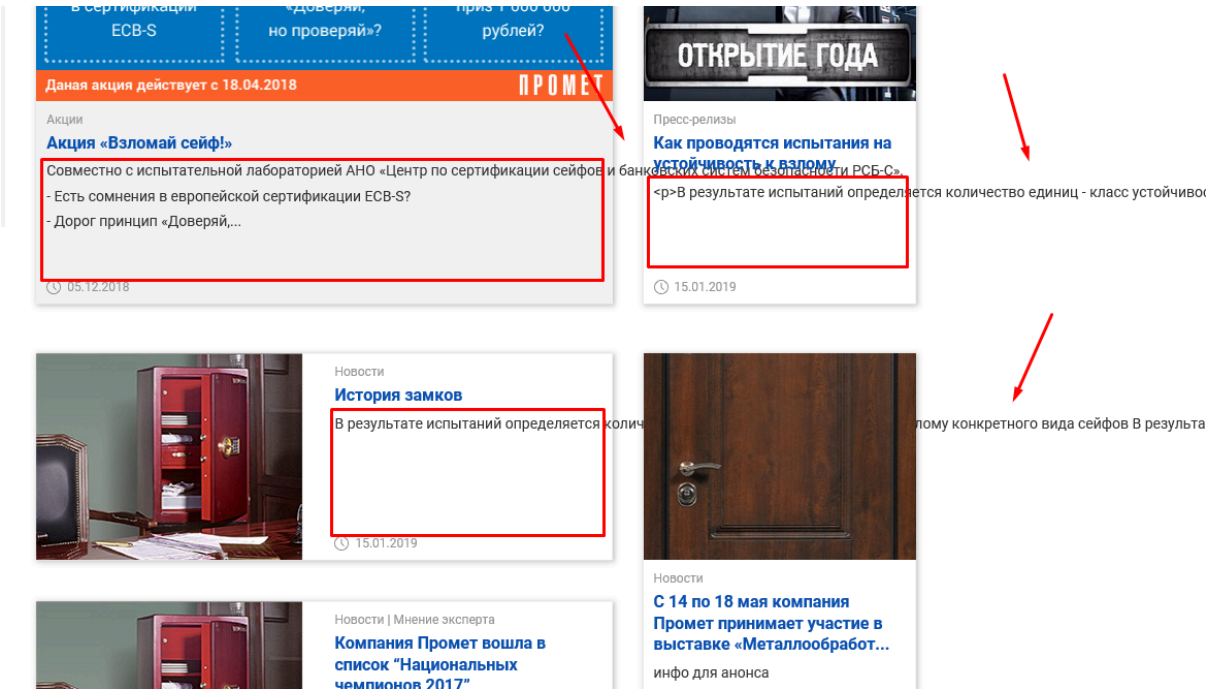
Скрипт проверяет пропускную способность клиента и поддержку ретины его браузером и исходя из этих данных подтягивает необходимое изображение.

Допускается разница максимально в ~400px.

Использовать в мобильном слайдере картинки от десктопа размером 1920px - не хорошо.

Динамические размеры контентных блоков

При изменении длины текста в блоках, блоки должны растягиваться. Текст не должен выходить за рамки блоков, если конечно такое отображение не задумывалось по дизайну. Такого быть не должно:



Контролируемые размеры блоков с изображениями (в чеклисте)

Блоки с контентными изображениями без фиксированного размера должны иметь свои максимальные размеры, дабы блок с контентом не ломал всю вёрстку при загрузке в контент изображений, которые отличаются от размеров изображений в макете.

Отступы (в чеклисте)

Делать отступы именно через стили, а не

.
Вертикальные отступы делаем сверху вниз, не сверху и снизу одновременно.

Ссылки (в чеклисте)

Используем <a href=””> только если это действительно ссылка. Если это кнопка с обработчиком формы, делаем через <button>.
Если это ссылка на другой сайт, ставит target=”_blank”.

Цикличность ссылок (в чеклисте)

Избегаем цикличности ссылок в меню. Вместо <a> с классом “active” ставим <p> с текстом активной ссылки.

Валидация (в чеклисте)

Стопроцентной валидации добиваться не требуется, но стоит учитывать, что использование невалидного кода должно быть оправдано.
Инструменты для проверки валидности html: https://validator.w3.org
И CSS: https://jigsaw.w3.org/css-validator/

Требования к формам (в чеклисте)

1. Одно модальное окно.
2. Одна форма - один <form>, не дублируем одну и ту же форму в разные места по всему сайту. Если необходимо размножить одну форму по нескольким местам на странице, делаем это через JS. Если сайт на битриксе - подключаем форму из компонента.
3. Должна быть минимальная валидация у формы, не должно быть возможности отправить пустую форму.
4. Форма должна корректно обрабатывать ответ, если не 200, то ошибка.
5. Все label элементы в форме должны иметь свою пару из input или select элемента.

Набор плагинов

Данный набор плагинов мы используем на сайтах. Если у вас есть гораздо лучшие решения, либо какой-либо из плагинов не подходит для вашей вёрстки - пишите тимлиду (пока что это kea@completo) и будем обновлять список:

1. FancyBox - отображение модальных окон.
2. SwiperJS - слайдер.
3. FormValidation - валидация форм.
4. MaskedInput - маска полей.
5. LazyLoad или blazy - ленивая подгрузка изображений.
6. ChosenJquery - стилизация списков.
7. mCustomScrollbar - скролл

Требования к проекту с вёрсткой

Организация файлов проекта

Проект с вёрсткой может находится на мастер ветке.

Иерархия папок проекта:

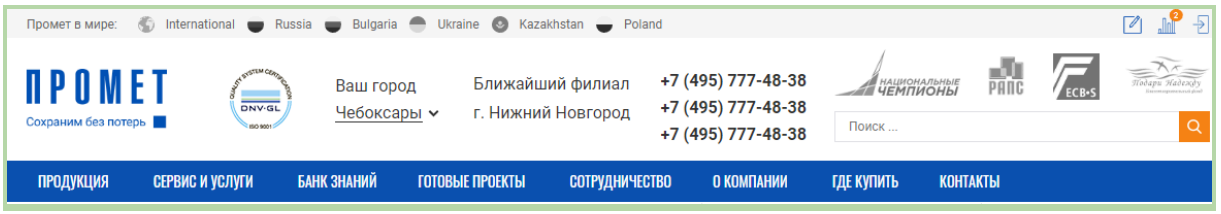
- node_modules - папка в которой хранятся зависимости проекта **(модули в игнор)**
- html - папка в которой собираем проект, содержит:
    - build - здесь находиться собранный проект для prod;
    - src - проект;
    - static_libs - подключаемые библиотеки.

- .gitignore - исключения для git:
    - /node_modules/
    - /bitrix/
    - .DS_Store
    - Thumbs.db
    - .idea
    - *.log
    - *.psd
    - *.ai
- Остальные папки CMS.

Собирать зависимости необходимо с помощью галпа.

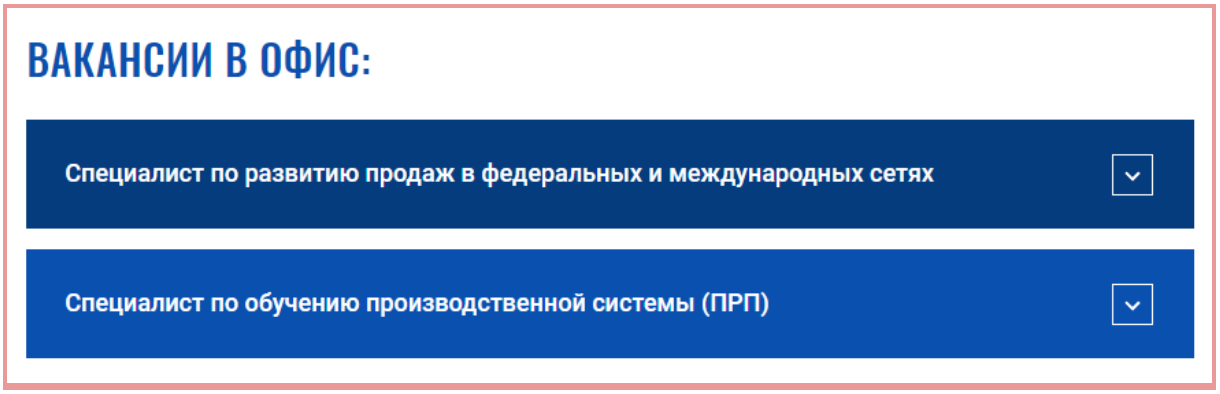
Блоки страниц, не относящиеся к шаблону сайта должны быть вынесены в модули (/build/modules/…).
Например:

Шапка сайта



Очевидно, что она используется на всех страницах и её стили должны находится в общем файле (/build/css/main.css).

Контентный блок



Данный блок используется далеко не на всех страницах и поэтому тянуть его стили на все страницы не имеет смысла и необходимо выносить его стили в отдельный файл (/build/modules/__modulesname__/style.css).

Тоже самое делаем с html и js. Таким образом, на бэкенде мы можем подключать эти стили и скрипты по надобности, а битрикс в свою очередь компанует единый файл стилей и скриптов для каждой страницы.

Графика

SVG иконки упаковываем в шрифты с помощью **iconfontcss**, за некоторыми исключениями, например, когда svg выходит из контента, из динамической части, тогда вставляем просто в html.
Прочие изображения, типа логотипы, фоновые вставки и прочее - упаковываем в спрайт с помощью **spritesmith**.

UI-kit (БЭМ)

Если кит уже создан - мы не перетираем стили кита, просто пишем модификаторы.

Для вёрстки кита в контексте бэма можно использовать bem-kit:
https://github.com/mazipan/bem-kit
Подробные примеры вёрстки вы можете найти в доках bem-kit
https://mazipan.github.io/bem-kit/demo/

UI-kit (вне БЭМа)

Можно использовать фреймворк UIkit:
https://github.com/uikit/uikit
Подробная инструкция и примеры:
https://getuikit.com/docs/

В командной работе киты не перетираем, если надо что-то добавить - пишем модификатор.

HPACE
IT-Development For Business

Подробнее по БЭМу

Именование классов по БЭМу

Схема именования стандартная:

- Имена записываются латиницей в нижнем регистре;
- Для разделения слов в именах используется дефис (-);
- Имя блока задает пространство имен для его элементов и модификаторов;
- Имя элемента отделяется от имени блока двумя подчеркиваниями (__);
- Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (_);
- Значение модификатора отделяется от имени модификатора одним подчеркиванием (_);
- Значение булевых модификаторов в имени не указывается;
- Селектор - только класс;
- Именовать классы коротко, понятно, без сокращений;
- Избегаем каскадов;
- Избегаем !important;
- Избегаем тегов в селекторах;
- Избегаем классов-помощников (.hide , .clearfix и т.п.);

Элемент — часть БЭМ-блока

- Название класса должно отвечать на вопрос «Что это?».

Элемент можно использовать вне его блока только в исключительных случаях

- Элемент — часть блока, но, поскольку, БЭМ-дерево независимо от HTML-дерева, элемент можно использовать и вне его блока в некоторых исключительных условиях (сам блок должен быть на странице).

Пример: расположение попапа-элемента не внутри блока-родителя, а в самом конце DOM (дабы показывать попап независимо от ограничения видимости родителей).

Элементов может не быть

- Не у всех блоков должны быть элементы: кнопка — всегда БЭМ-блок, но БЭМ-элементы у неё внутри встречаются относительно редко.

Как отличить БЭМ-блок и БЭМ-элемент

- Просто задайте себе вопрос: «Эта сущность может потребоваться мне отдельно, сама по себе? Или она нужна только внутри её родителя?» Если нужна отдельно — это БЭМ-блок, если мыслима только внутри родителя — это БЭМ-элемент.
- В действительно сомнительных случаях делайте выбор в пользу БЭМ-блока.

Модификатор — дополнительный класс для смены оформления или поведения

- Название класса может отвечать на вопросы «Что это?», «Что меняется?», «Чем отличается от прочих?».

Модификатор нельзя использовать самостоятельно

- Класс модификатор никогда не должен использоваться самостоятельно, но всегда только с тем классом, который он модифицирует.

Один БЭМ-блок = один файл

- В файловой системе при работе с CSS-препроцессорами каждый БЭМ-блок должен быть описан в своём отдельном файле. Тоже самое относится к JS скриптам.

БЭМ-дерево плоское, в отличие от DOM

- В классах БЭМ-элементов нельзя прописывать иерархию, два и более сегмента __ недопустимы.

SEO

Теги (в чеклисте)

Необходимо использовать семантические теги на стадии вёрстки сайта.

В первую очередь:

1. <header> - Шапка вместо ;
2. <footer> - Подвал вместо ;
3. <menu> - Главное меню вместо ;
4. <nav> - Навигационная цепочка сайта (в единственном экземпляре) вместо ;
5. <h1> - Используется на странице единожды.

Также, по возможности использовать html5 теги для вторичных элементов:

1. <aside> - Сайдбар;
2. <article> - Основной контент страницы (Новость, пост, статья);
3. <adress> - В данный тег заключаем контактные данные, обычно они в подвале и на странице контактов.

Head страницы (в чеклисте)

В шапке сайта обязательно должны быть:

- Doctype - <!DOCTYPE html>
- Кодировка документа - <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
- Viewport - <meta name="viewport" content="width=device-width">
- Title - <title>Заголовок</title>
- Description - <meta name="description" content="description">
- favicon - <link rel="shortcut icon" href=".../favicon.ico" type="image/x-icon">
- Язык страницы - <html lang="ru">

Командная вёрстка (для проджектов)

1. В проекте должен быть один главный верстальщик, который выполняет большую часть работы и может оперативно дать справку по проекту, он же может быть и бэкендщиком, то бишь фуллстак;
2. Изначально необходимо верстать кит, вероятнее всего, делает это главный верстальщик;
3. Также, по началу работы необходима типовая страница, на которую будут ориентироваться в дальнейшем.

В остальном, всё по регламенту и не будет никаких конфликтов и косяков в проекте. По желанию команды / заказчика для объемных проектов по верстке, в которых сжатые сроки может быть составлена таблица общих элементов макетов и определены эталонные элементы, которые далее будут использованы в последующих работах по верстке данного проекта.

Прочее

Кэш (в чеклисте)

На сайте должен быть включен gzip и прописано кэширование в .htaccess.

Чеклист по вёрстке

Смотреть здесь.