

Вёрстка текстового содержимого страниц

Выполнен на 0%

Обзор основных тем и элементов

Углублённая теория

Шрифты, типографика, инлайн

Шрифты: специфика подключения

Инструменты для работы со шрифтами

Типографика

Инлайн и инлайн-блок

Дополнительная теория

Вёрстка для CMS

Вариативные шрифты

Вариативная типографика

Методология вёрстки текстовых элементов

Кейс 1, лёгкий уровень

Кейс 2, лёгкий уровень

Кейс 3, лёгкий уровень

Кейс 4, средний уровень

Кейс 5, средний уровень

Кейс 6, средний уровень

Кейс 7, сложный уровень

Кейс 8, сложный уровень

Кейс 9, сложный уровень

Главная / Моё обучение / Вёрстка текстового содержимого страниц / Углублённая теория / Дополнительная теория /

Вёрстка для CMS

Нужно начать с того, что такое CMS и с чем это едят. *Content Management System*, или система управления контентом, — это программное обеспечение, которое позволяет управлять наполнением сайта извне, обычно — совместно (многопользовательски) и в режиме реального времени. Основной функционал CMS — это упрощение процесса работы с содержимым и контроль версий и деятельности пользователей. Иногда CMS — это настолько масштабная программа, что она позволяет также управлять дизайном сайта, интегрировать сервисы сайта с другими системами (например, оплаты через сайт — с бухгалтерскими данными).

Наш курс не подразумевает подробного разбора видов, особенностей и механики CMS, поэтому разберём только то, что нужно знать верстальщику, который верстает для CMS стили для статей или постов.

Основная задача CMS — упростить работу с контентом. То есть подразумевается, что с сайтом работают разные люди, и все они публикуют информацию, и это не должно даваться им слишком сложно. Можно представить себе тех, кто будет публиковать контент, как пользователей «дневниковых сервисов»: правил и ограничений должно быть не слишком много, и кодить никто при этом не будет.

Нужно учесть, что публикация в CMS почти всегда происходит через «визуальный редактор», который позволяет сразу видеть, что именно свёрстано и будет опубликовано, но не подразумевает, что пользователь будет писать код: классы, обёртки и прочее. Многие редакторы и CMS «чистят» код, который в них попадает, удаляя `<div>`, `<span>` и «лишние» символы. При этом некоторые CMS имеют встроенный инструмент просмотра, как будет выглядеть пост на сайте, а некоторые — нет. Некоторые CMS также позволяют редактировать или добавлять материалы «с лица» сайта, не заходя в админку.

Как это всё влияет на вёрстку? А так, что те решения, которые мы зачастую используем для вёрстки, потому что они семантические и прозрачные, здесь просто не будут работать. Пользователи скорее всего не будут вставлять «распорки» и «обёртки», каждый раз прописывать классы элементам и тем более править стили.

Для CMS подход стилизации по классам практически не работает. Придётся стилизовать по тегам или учитывать возможности визуального редактора, а потом писать инструкцию о том, какая его возможность что означает на сайте.

Порядок работы с CMS — это сначала настройка самой системы, как она будет работать, какие возможности у неё будут, какие плагины или расширения к системе будут подключены, потом — «натягивание дизайна», то есть собственно работа с тем, чтобы привести обычно очень неприглядный стандартный вид CMS в состояние, близкое к макету. А потом, когда всё готово, приходит человек (не разработчик) и постит текст. И, что важно, к этому тексту должны применяться те стили, которые для него предусмотрены. Раз вёрстка текстов на сайтах с CMS обычно происходит через визуальный редактор, то привязывать стили нужно к тем характеристикам, которые текстовые элементы получают через визуальный редактор. Обычно это теги, иногда в неожиданных сочетаниях.

Итак, основные особенности, отличающие вёрстку для CMS от вёрстки вообще:

- Хедер и футер — неприкосновенны, идентичны для всех страниц сайта, классы для различения шапки одной страницы от шапки другой страницы не сработают. Классы на `body` тоже потеряются, стилизуйте без классов.
- Никаких классов для фотографий, форм и контентных элементов! Если без классов вообще никак — они должны быть простыми и утилитарными (про функционал, а не про смысл содержимого).
- Любой блок должен нормально заработать на любой другой странице. Например, если кто-то захочет взять с главной страницы слайдер и поместить его на любую другую страницу — это должно получиться без переписывания кода. Значит, стилизуем не от конкретной страницы, а от самого блока.
- Нельзя делать классы в зависимости от порядка/положения элемента. Например, порядок карточек должно быть можно менять как угодно, количество — тоже.

**Важное правило, которые обязательно нужно выполнять при публикации:**

Обязательно посмотреть на сайте, что у вас получилось после нажатия кнопки «Опубликовать». Даже если вы редактировали что-то и просто сохраняете изменения, не поленитесь посмотреть на результат. Это спасает от многих глупых ситуаций, вроде неправильной картинки, дважды повторённого абзаца и нелепых опечаток.

Ознакомились со статьёй?

Сохранить прогресс

Дополнительная теория

Вариативные шрифты



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Профессии

Фронтенд-разработчик

React-разработчик

Фулстек-разработчик

Бэкенд-разработчик

Услуги

Работа наставником

Для учителей

Стать автором

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

PHP. Профессиональная веб-разработка

PHP и Yii. Архитектура сложных веб-сервисов

Node.js. Разработка серверов приложений и API

Анимация для фронтендеров

Вёрстка email-рассылок

Vue.js для опытных разработчиков

Регулярные выражения для фронтендеров

Шаблонизаторы HTML

Алгоритмы и структуры данных

Анатомия CSS-каскада

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Информация

Об Академии

О центре карьеры

Остальное

Написать нам

Мероприятия

Форум