ВЛАДИМИР ДРОНОВ

bhv®

# LARAVEL

## Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS

МИГРАЦИИ

КОМАНДЫ BLADE

ВАЛИДАЦИЯ ДАННЫХ

ВЫГРУЗКА ФАЙЛОВ

РАЗГРАНИЧЕНИЕ ДОСТУПА

ИСПОЛЬЗОВАНИЕ CAPTCHA

ПОДДЕРЖКА BBCode

ПУБЛИКАЦИЯ САЙТА

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

*Материалы на www.bhv.ru*

**Владимир Дронов**

# LARAVEL

## Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS

Санкт-Петербург
«БХВ-Петербург»
2017

УДК 004.738.5+004.43
ББК 32.973.26-018.2
Д75

**Дронов В. А.**

Д75 Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. — СПб.: БХВ-Петербург, 2017. — 768 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-3845-9

Книга посвящена быстрой разработке профессиональных динамических Web-сайтов с применением популярного PHP-фреймворка Laravel. Описаны технологии создания клиентской части сайта HTML 5, CSS 3 и JavaScript, а для серверной части сайта — язык PHP и сервер данных MySQL. Рассказано о применении миграций Laravel для создания в базе данных таблиц, полей, индексов и связей, о написании моделей, маршрутов, контроллеров и шаблонов. Описаны средства Laravel для ввода и правки данных, встроенные во фреймворк средства валидации с применением запросов форм и инструменты для выгрузки файлов на сайт. Рассказано о подсистеме разграничения доступа Laravel и ее настройке под конкретные нужды, а также об использовании CAPTCHA. Даны практические примеры по разработке дизайна страниц, интерактивных элементов — спойлера, лайтбокса и блокнота, создания универсального файлового хранилища, основанного на технологии AJAX, и реализации поддержки тегов BBCode для форматирования текста. Рассмотрен процесс разработки полнофункционального сайта и его публикации в Интернете. Все исходные коды доступны для загрузки с сайта издательства.

*Для Web-программистов*

УДК 004.738.5+004.43
ББК 32.973.26-018.2

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Евгений Рыбаков* |
| Зав. редакцией | *Екатерина Капалыгина* |
| Редактор | *Григорий Добин* |
| Компьютерная верстка | *Ольги Сергиенко* |
| Корректор | *Зинаида Дмитриева* |
| Дизайн серии | *Инны Тачиной* |
| Оформление обложки | *Марины Дамбиевой* |

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-3845-9

# Оглавление

**Введение в быструю разработку сайтов ..... 1**
Что придется сделать разработчику? ..... 1
Упрощение разработки серверной части Web-сайта. Фреймворки ..... 2
Фреймворк Laravel — номер один в Web-программировании! ..... 3
О чем эта книга? ..... 4
Типографские соглашения ..... 5
Что нас ждет в будущем? ..... 6

**ЧАСТЬ I. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ WEB-САЙТА ..... 9**

**РАЗДЕЛ 1. СОДЕРЖИМОЕ WEB-СТРАНИЦ. ЯЗЫК HTML 5 ..... 11**

**Глава 1. Современный Web-дизайн. Введение в язык HTML 5 ..... 13**
Клиентская часть Web-сайта ..... 13
Статические Web-страницы и Web-сайты ..... 14
Содержание, представление и поведение Web-страниц ..... 14
Интернет: как все это работает? ..... 15
Web-серверы ..... 15
Интернет-адреса ..... 16
Введение в язык HTML 5 ..... 18
Первая Web-страница ..... 18
Теги и атрибуты тегов ..... 19
Вложенность тегов ..... 21
Форматирование Web-страниц ..... 22
Секции Web-страницы ..... 22
Метаданные и метатеги ..... 23

**Глава 2. Структурирование и оформление текста. Литералы. Комментарии HTML ..... 25**
Структурирование текста ..... 25
Абзацы и заголовки ..... 25
Блочные элементы HTML ..... 27
Списки ..... 27
Блочные цитаты и адреса ..... 29
Текст фиксированного формата ..... 30
Блочные контейнеры ..... 31
Семантическая разметка текста ..... 31
Горизонтальные линии ..... 33
Оформление текста ..... 33
Выделение фрагментов текста ..... 33
Встроенные элементы HTML ..... 34
Встроенные контейнеры ..... 35
Разрыв строки ..... 35
Вставка специальных символов. Литералы ..... 36
Комментарии ..... 38

**Глава 3. Графика и мультимедиа ..... 39**
Внедренные элементы Web-страниц ..... 39
Графика ..... 39
Поддерживаемые форматы интернет-графики ..... 40
Вставка графических изображений ..... 41
Мультимедиа ..... 42
Поддерживаемые форматы интернет-мультимедиа ..... 42
Вставка аудиоролика ..... 43
Вставка видеоролика ..... 44

**Глава 4. Таблицы ..... 46**
Создание таблиц ..... 46
Дополнительные инструменты для создания таблиц ..... 49
Заголовок таблицы ..... 49
Секции таблицы ..... 50
Колонки и группы колонок ..... 51
Объединение ячеек таблиц ..... 53

**Глава 5. Средства навигации ..... 56**
Текстовые гиперссылки ..... 56
Создание гиперссылок ..... 56
Интернет-адреса в WWW ..... 58
Почтовые гиперссылки ..... 59
Дополнительные возможности гиперссылок ..... 59
Графические гиперссылки ..... 61
Изображения-гиперссылки ..... 61
Изображения-карты ..... 61
Панель навигации ..... 63
Якоря ..... 64

**Глава 6. Web-формы и элементы управления. Фреймы ..... 65**
Web-формы ..... 65
Что такое Web-форма и зачем она нужна? ..... 65
Создание Web-форм ..... 66
Элементы управления ..... 68
Общие вопросы создания элементов управления ..... 68
Поле ввода ..... 69
Поле ввода пароля ..... 70
Поле ввода числа ..... 70
Поле ввода интернет-адреса ..... 71
Поле ввода адреса электронной почты ..... 71
Флажок ..... 71
Переключатель ..... 72
Регулятор ..... 72
Область редактирования ..... 73
Список ..... 73
Поле ввода файла ..... 75
Скрытое поле ..... 76
Кнопки ..... 77

Элементы оформления ... 78
Надпись ... 78
Группа ... 78
Фреймы ... 79

**РАЗДЕЛ 2. ПРЕДСТАВЛЕНИЕ WEB-СТРАНИЦ. КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ CSS 3 ... 81**

**Глава 7. Введение в CSS ... 83**
Понятие о стилях CSS ... 83
Создание стилей CSS ... 84
Таблицы стилей. Встроенные стили ... 86
Правила каскадности и приоритет стилей ... 88
Наследование атрибутов стилей ... 90
Комментарии CSS ... 91

**Глава 8. Селекторы стилей. Единицы измерения CSS ... 92**
Селекторы стилей ... 92
Введение в селекторы стилей ... 92
Компоненты указателей ... 93
Основные указатели ... 93
Указатели на атрибуты тега ... 95
Псевдоклассы ... 96
Псевдоэлементы ... 100
Разделители ... 100
Единицы измерения и вычисления CSS ... 101
Важные атрибуты стилей ... 103

**Глава 9. Параметры текста ... 104**
Параметры шрифта ... 104
Параметры вывода текста ... 108
Параметры списков ... 109
Дополнительные параметры текста ... 111
Тень у текста ... 111
Вывод текста ... 112
Загружаемые шрифты. Директивы CSS ... 113

**Глава 10. Отображение и видимость элементов. Параметры курсора. Генерируемое содержание ... 116**
Параметры отображения ... 116
Параметры курсора ... 118
Генерируемое содержание ... 119
Статичное генерируемое содержание ... 119
Создание нумерации ... 120

**Глава 11. Параметры фона. Градиентные фоны CSS 3 ... 123**
Сплошной фон ... 123
Графический фон ... 124
Создание графического фона ... 124
Параметры графического фона ... 125

Задание сразу всех параметров фона ... 128
Градиентный фон ... 128
Введение в градиенты и градиентные фоны ... 128
Создание линейного градиента ... 129
Создание радиального градиента ... 130
Создание повторяющегося градиента ... 132

**Глава 12. Размеры, отступы, рамки, тени и выделение. Параметры таблиц ... 133**
Параметры отступов ... 133
Параметры рамки ... 135
Параметры размеров ... 139
Параметры переполнения. Элементы с прокруткой ... 140
Параметры тени у блочного элемента ... 142
Параметры выделения ... 142
Параметры таблиц ... 143
Параметры выравнивания ... 144
Параметры отступов и рамок ... 144
Параметры размеров ... 146
Прочие параметры ... 148

**Глава 13. Инструменты для создания разметки ... 149**
Плавающие элементы ... 149
Позиционируемые элементы ... 151
Понятие позиционируемого элемента ... 151
Создание позиционируемых элементов ... 152
Гибкая верстка ... 155
Реализация гибкой верстки ... 155
Выравнивание позиций ... 157
Выравнивание отдельных позиций ... 160
Управление размерами и порядком следования позиций ... 162
Многоколоночная верстка ... 164
Базовые средства многоколоночной верстки ... 164
Задание дополнительных параметров колонок ... 165
Разметка Web-страницы и ее создание ... 167
Табличная разметка на основе блоков ... 168
Табличная разметка с фиксированными «шапкой» и «поддоном» ... 169
Рамочная разметка ... 171

**Глава 14. Специальные эффекты CSS 3 ... 174**
Преобразования ... 174
Как задаются преобразования и их параметры? ... 174
Двухмерные преобразования ... 174
Смещение ... 175
Масштабирование ... 175
Наклон ... 176
Поворот ... 176
Трехмерные преобразования ... 177
Перспектива ... 177
Указание трехмерных преобразований ... 178

Точка зрения и ее местоположение ... 178
Скрытие обратной стороны элемента ... 180
Режим проецирования элементов-потомков ... 181
Позиционирование точки начала координат ... 182
Сложные преобразования ... 183
Анимация ... 183
Трансформационная анимация ... 183
Простейшая анимация ... 184
Обратная анимация ... 187
Сложная анимация ... 187
Покадровая анимация ... 188
Состояния анимации ... 189
Параметры анимации ... 190

**Глава 15. Медиазапросы. Управление выводом на печать ... 194**
Использование медиазапросов ... 194
Медиазапросы HTML ... 195
Введение в медиазапросы HTML ... 195
Указатели медиазапросов ... 196
Разделители медиазапросов ... 198
Медиазапросы CSS ... 199
Управление выводом на печать ... 199

**РАЗДЕЛ 3. ПОВЕДЕНИЕ WEB-СТРАНИЦ. WEB-СЦЕНАРИИ ... 201**

**Глава 16. Язык программирования JavaScript ... 203**
Основные понятия JavaScript ... 203
Типы данных JavaScript ... 205
Переменные ... 207
Именование переменных ... 207
Объявление переменных ... 207
Операторы ... 208
Арифметические операторы ... 208
Оператор объединения строк ... 209
Операторы присваивания ... 209
Операторы сравнения ... 210
Логические операторы ... 210
Оператор получения типа *typeof* ... 211
Преобразование типов данных ... 212
Приоритет операторов ... 213
Блоки ... 215
Управляющие конструкции ... 215
Ветвление ... 215
Оператор ветвления *?* ... 217
Переключение ... 217
Циклы ... 218
Цикл со счетчиком ... 218
Цикл с предусловием ... 219
Цикл с постусловием ... 220
Прерывание и перезапуск цикла ... 220

Функции....221
Объявление функций....221
Функции и переменные. Локальные переменные....222
Вызов функций....222
Присваивание функций. Функциональный тип данных....223
Массивы....224
Ссылки. Пустая ссылка *null*....226
Объекты и экземпляры объектов....227
Понятия объекта и экземпляра объекта....227
Создание экземпляра объекта....227
Работа с экземпляром объекта....228
Добавленные свойства и методы....229
Статические свойства и методы....229
Встроенные объекты языка JavaScript....229
Объект *Object* и использование его экземпляров....231
Оператор *instanceof*....232
Цикл по свойствам объекта....233
Обработка исключений....234
Комментарии JavaScript....236
Как Web-сценарии помещаются в код Web-страницы?....236

**Глава 17. Доступ к элементам страницы и управление ими....239**
Объектная модель документа....239
Доступ к странице и ее элементам....241
Доступ к странице....241
Доступ к элементам страницы....241
Прямой доступ к элементу страницы....241
Доступ по имени тега или стилевого класса. Коллекции....242
Доступ по селекторам CSS....243
Доступ к родителю, потомкам и соседним элементам. Узлы....244
Быстрый доступ к элементам страницы....246
Работа со страницей и ее элементами....247
Работа с параметрами страницы....247
Работа с параметрами элемента....247
Работа с основными параметрами....247
Работа с параметрами местоположения и размеров элемента страницы....248
Работа с атрибутами тега и их значениями....250
Работа со стилями....251
Работа с содержимым элемента....253
Добавление нового содержимого....254
Добавление и удаление элементов страницы....254

**Глава 18. Обработка событий....258**
Введение в события и их обработку....258
События....258
Обработчики событий и их привязка....259
Получение сведений о событии....260
События, поддерживаемые элементами страницы....261
События мыши....261
События клавиатуры....264

Событие прокрутки....265
События секции тела страницы....265
Особые случаи обработки событий....266
Туннелирование и всплытие событий....266
Обработчик события по умолчанию и его отмена....268

**Глава 19. Управление интерактивными и внедренными элементами....270**
Интерактивные элементы....270
Гиперссылки....270
Web-формы....271
Элементы управления....272
Внедренные элементы....279
Графические изображения....279
Аудио- и видеоролики....280

**Глава 20. Работа с Web-обозревателем....285**
Окна Web-обозревателя....285
Интернет-адрес текущей страницы....289
Список истории Web-обозревателя....290
Параметры экрана....291
Сведения о Web-обозревателе....292
Стандартные диалоговые окна и сообщения....293
Таймеры....294

**Глава 21. Работа с локальными файлами. Регулярные выражения....296**
Работа с локальными файлами....296
Получение выбранных файлов и сведений о них....296
Загрузка выбранных файлов....298
Регулярные выражения....301
Написание регулярных выражений. Литералы и группы....301
Работа с регулярными выражениями....304

**Глава 22. AJAX....308**
Введение в AJAX....308
Программная реализация AJAX....309
Объект *XMLHttpRequest*....309
Отправка запроса....309
Получение результата....312
Формат JSON....313
AJAX-навигация....315

**ЧАСТЬ II. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ WEB-САЙТА....319**

**РАЗДЕЛ 4. ВВЕДЕНИЕ В СЕРВЕРНОЕ ПРОГРАММИРОВАНИЕ. PHP. MYSQL....321**

**Глава 23. Серверные программы. Фреймворки....323**
Динамические страницы и сайты....323
Разработка серверных программ. Фреймворки....325
Введение во фреймворки. Модели, шаблоны, контроллеры....326

**Глава 24. Программная платформа PHP ..... 330**
Основные принципы, типы данных, переменные и операторы ..... 330
Управляющие конструкции ..... 332
Функции ..... 333
Массивы. Ассоциативные массивы ..... 334
Регулярные выражения ..... 335
Классы и объекты ..... 336
Доступ к свойствам и методам объекта ..... 336
Объявление классов ..... 337
Наследование классов ..... 338
Конструкторы и деструкторы ..... 340
Модификаторы доступа ..... 341
Константы класса ..... 342
Интерфейсы ..... 342
Трейты ..... 343
Пространства имен ..... 344
Определение пространств имен ..... 344
Работа с пространствами имен ..... 345
Принципы написания программного кода PHP ..... 346

**Глава 25. Базы данных. Сервер данных MySQL ..... 348**
Реляционные базы данных ..... 348
Введение в реляционные базы данных ..... 348
Поля ..... 349
Индексы и ключи ..... 350
Связи ..... 352
Язык SQL ..... 353
СУБД MySQL ..... 355
Типы данных, поддерживаемые MySQL ..... 355
Атрибуты полей и индексов ..... 357
Агрегатные функции ..... 357
Пользователи и их права ..... 358

**РАЗДЕЛ 5. ФРЕЙМВОРК LARAVEL ..... 361**

**Глава 26. Установка и настройка Laravel ..... 363**
Программные требования Laravel ..... 363
Создание нового проекта ..... 363
Структура папок Laravel-проекта ..... 365
Настройка сайта ..... 367
Настройка соединения с базой данных ..... 368
Настройки отправки электронной почты ..... 368
Настройки режима работы сайта ..... 369
Прочие настройки ..... 370
Обработка ошибок ..... 370

**Глава 27. Миграции ..... 372**
Преимущества миграций ..... 372
Создание миграций ..... 373
Прототипирование миграций ..... 373

Код новой миграции. Фасады Laravel ....374
Создание структур данных ....375
Создание таблиц ....375
Создание полей ....375
Создание индексов ....378
Создание связей ....379
Правка и переименование структур данных ....381
Добавление полей ....381
Правка и переименование полей ....382
Переименование таблиц ....382
Удаление структур данных ....383
Удаление связей ....383
Удаление индексов ....383
Удаление полей ....384
Удаление таблиц ....384
Выполнение и откат миграций ....385
Дополнительные возможности миграций ....385

**Глава 28. Модели ....387**
Модели Laravel: требования и соглашения ....387
Создание простых моделей ....388
Прототипирование моделей. Базовый класс модели ....388
Задание параметров модели ....389
Создание связей ....390
Связь «один-ко-многим» ....390
Связь «один-к-одному» ....393
Связь «многие-ко-многим» ....393
Сквозная связь ....395
Расширение функциональности модели ....397
Создание вычисляемых полей ....397
Указание другого поля для поиска при внедрении модели ....398
Создание обработчиков событий ....398
Произвольные свойства и методы модели ....401

**Глава 29. Маршрутизация ....403**
Введение в маршрутизацию ....403
Где хранятся настройки маршрутизации? ....404
Указание маршрутов ....405
Простые маршруты ....405
Параметризованные маршруты ....406
Правила для значений параметров в параметризованных маршрутах ....407
Именованные маршруты ....409
Указание посредников для маршрутов ....409
Массовое создание маршрутов ....410
Базовые средства для массового создания маршрутов ....410
Дополнительные параметры массово создаваемых маршрутов ....412
Внедрение модели в контроллер ....413
Неявное внедрение модели ....413
Явное внедрение модели ....414

Группы маршрутов ....416
Физические интернет-адреса ....417

**Глава 30. Контроллеры и действия ....419**
Контроллеры Laravel: требования и соглашения ....419
Создание контроллеров ....420
Получение данных от посетителя ....421
Работа с базой данных ....423
Простая выборка записей ....424
Поиск записей по их номерам ....424
Выборка всех записей ....425
Выборка первой записи ....425
Получение значений полей записи ....425
Получение связанных записей ....425
Создание запросов к базе данных ....427
Фильтрация записей ....427
Фильтрация по наличию или отсутствию связанных записей ....431
Сортировка записей ....432
Указание выбираемых полей ....433
Выборка уникальных записей ....434
Связывание таблиц ....434
Использование агрегатных функций. Группировка записей ....435
Получение количества связанных записей ....435
Использование агрегатных функций применительно ко всем записям ....436
Использование агрегатных функций применительно к сгруппированным записям ....437
Фильтрация и сортировка групп записей ....437
Ограничение количества выбираемых записей ....438
Специальные случаи выборки записей ....439
Использование пагинатора ....439
Упрощенный пагинатор ....440
Полнофункциональный пагинатор ....442
Получение сведений о запросе ....442
Получение путей к папкам фреймворка ....444
Вывод данных ....444
Вывод посредством шаблона ....445
Вывод в формате JSON ....446
Вывод пагинатора в формате JSON ....447
Отправка файлов ....447
Перенаправление ....448
Перенаправление с выводом всплывающих сообщений ....449
Указание посредников в контроллерах ....450
Особые разновидности контроллеров ....451
Контроллеры-функции ....451
Контроллеры-действия ....452

**Глава 31. Шаблоны ....453**
Шаблоны Laravel: требования и соглашения ....453
Создание шаблонов ....453

Команды языка Blade ....456
Команды вывода данных ....456
Ветвления ....456
Циклы ....457
Прерывание и перезапуск цикла ....457
Служебная переменная *loop* ....458
Комментарии Blade ....459
Вставка PHP-кода ....459
Особые случаи вывода данных ....459
Генерирование интернет-адресов ....459
Создание Web-форм и элементов управления ....461
Указание метода отправки данных ....461
Защита от сетевых атак ....461
Вывод введенных ранее данных ....462
Вывод сообщений об ошибках ....462
Вывод всплывающих сообщений ....463
Вывод пагинатора ....463
Вложенные шаблоны ....465
Наследование шаблонов ....467
Создание шаблонов-родителей ....467
Создание шаблонов-потомков ....468
Стеки ....469
Разделяемые данные и составители ....470
Разделяемые данные ....470
Составители ....471
Получение доступа к контроллеру ....473

**Глава 32. Ввод и правка данных ....474**
Создание, правка и удаление отдельных записей ....474
Создание и правка записей ....474
Создание Web-формы для ввода и правки записи ....474
Создание записи ....476
Правка записи ....479
Удаление записей ....481
Обработка связей между таблицами ....482
Связь «один-ко-многим» ....482
Связь "многие-ко-многим" ....483
Дополнительные инструменты для создания и правки отдельных записей ....485
Поиск или создание записей ....485
Исправление или создание записей ....486
Работа со связанными записями ....487
Проверка введенных в форму данных на корректность. Валидаторы ....489
Простейшие валидаторы ....489
Полностью автоматическая валидация ....489
Полуавтоматическая валидация ....490
Условные правила валидации ....492
Правила валидации ....492
Запросы форм ....495

Массовые создание, правка и удаление записей....497
Работа с выгруженными файлами....499
Файловое хранилище и диски Laravel....499
Особенности создания Web-формы для выгрузки файлов....501
Получение и сохранение выгруженных файлов....501
Получение выгруженного файла и сведений о нем....501
Сохранение выгруженного файла....502
Работа с выгруженными файлами....503

**Глава 33. Разграничение доступа. Использование CAPTCHA....505**
Встроенная подсистема разграничения доступа Laravel....505
Ограничение доступа к страницам....507
Простые случаи ограничения доступа....507
Ограничение доступа на основе более сложных условий....508
Шлюзы....509
Политики....510
Ограничение доступа на основе запросов форм....514
Ограничение доступа в шаблонах....514
Вывод сведений о текущем пользователе....515
Настройка встроенной подсистемы разграничения доступа....516
Базовые настройки....516
Модификация списка пользователей....521
Настройка писем, отправляемых действием восстановления пароля....522
Создание оповещения....522
Подготовка шаблона для оповещения....525
Отправка оповещений....525
Использование CAPTCHA. Библиотека Captcha for Laravel....526
Установка....526
Настройка....527
Использование....529

**Глава 34. Кэширование....530**
Настройки кэширования....530
Кэширование в папке....530
Кэширование в таблице базы данных....531
Кэширование в оперативной памяти....532
Занесение данных в кэш....532
Изменение данных, хранящихся в кэше....533
Получение данных из кэша....533
Простое получение данных....533
Получение данных из кэша с одновременной их записью....534
Получение данных из кэша с последующим их удалением....535
Удаление данных из кэша....535
Работа с другим хранилищем....536

**ЧАСТЬ III. ПРАКТИКА РАЗРАБОТКИ: СОЗДАНИЕ WEB-САЙТА ЭЛЕКТРОННЫХ ПУБЛИКАЦИЙ ..... 537**

**РАЗДЕЛ 6. РАЗРАБОТКА WEB-САЙТА — СВОДИМ ВСЕ ВОЕДИНО ..... 539**

**Глава 35. Планирование и предварительные действия ..... 541**
Планирование Web-сайта ..... 541
Основные этапы планирования Web-сайта ..... 541
Дизайн Web-сайта ..... 542
Логическая и физическая структуры Web-сайта ..... 543
«СЭП» — Web-сайт электронных публикаций ..... 544
Дизайн сайта ..... 544
Создание и настройка проекта ..... 545

**Глава 36. Создание дизайна Web-страниц ..... 548**
Особенности создания представления для Web-страниц ..... 548
Web-страницы для традиционных компьютеров ..... 549
Разметка ..... 549
Начальное представление ..... 550
«Шапка» ..... 551
Панель навигации ..... 553
Блок основного содержания ..... 555
Параметры самого блока основного содержания ..... 555
Параметры текста ..... 555
Параметры нумерации заголовков ..... 556
Параметры внедренных элементов ..... 557
Параметры Web-форм и элементов управления ..... 559
«Поддон» ..... 562
Web-страницы для мобильных устройств ..... 563
Разметка ..... 563
«Шапка» ..... 563
Блок основного содержания ..... 564
«Поддон» ..... 565
Печатная редакция Web-сайта ..... 566

**Глава 37. Интерактивные элементы ..... 568**
Спойлер ..... 568
Формирование спойлера ..... 568
Представление спойлера ..... 569
Программирование спойлера ..... 571
Лайтбокс ..... 573
Формирование лайтбокса ..... 573
Представление лайтбокса ..... 574
Программирование лайтбокса ..... 577
Блокнот ..... 579
Формирование блокнота ..... 579
Представление блокнота ..... 580
Программирование блокнота ..... 582

**Глава 38. Статические Web-страницы ..... 585**
Маршруты ..... 585
Контроллеры ..... 585
Базовый класс контроллера. Определение обращения с мобильного устройства ..... 586
Контроллер *MainController* ..... 588
Шаблоны ..... 588
Родительские шаблоны ..... 588
Шаблоны страниц ..... 591
Тестирование «мобильной» редакции Web-сайта ..... 592

**Глава 39. Разграничение доступа и список пользователей ..... 594**
Маршруты ..... 594
Миграция и модель ..... 595
Служебные страницы ..... 596
Регистрация нового пользователя ..... 596
Вход на Web-сайт ..... 598
Процедура сброса пароля ..... 599
Отправка письма со сведениями о сбросе пароля ..... 599
Собственно сброс пароля ..... 599
Электронное письмо со сведениями для сброса пароля ..... 600
Инструменты для работы со списком пользователей ..... 603
Список пользователей ..... 603
Правка пользователя ..... 606
Удаление пользователя ..... 608
Разграничение доступа ..... 610
Панель навигации ..... 611

**Глава 40. Категории и подкатегории ..... 613**
Маршруты ..... 613
Миграции и модели ..... 614
Миграция и модель списка категорий ..... 614
Миграция и модель списка подкатегорий ..... 616
Инструменты для работы со списками категорий и подкатегорий ..... 617
Список категорий ..... 617
Вывод списка категорий ..... 617
Автоматическое генерирование слагов ..... 619
Правка списка категорий ..... 620
Список подкатегорий ..... 623
Вывод списка подкатегорий ..... 623
Создание и правка подкатегорий ..... 624
Разграничение доступа ..... 624
Панель навигации ..... 625
Вывод списков категорий и подкатегорий ..... 626

**Глава 41. Статьи. Поддержка BBCode ..... 631**
Маршруты ..... 631
Миграции и модели ..... 632
Миграция и модель списка статей ..... 632
Модели списков подкатегорий и пользователей ..... 634

Разграничение доступа....634
Вывод списка статей....635
Вывод списка статей, относящихся к выбранной подкатегории....635
Вывод списка последних пяти статей, относящихся к выбранной категории....641
Поиск статей....642
Вывод статьи....645
Форматирование текста статей. BBCode....645
Набор тегов BBCode, поддерживаемых нашим сайтом....645
Собственно форматирование текстов статей....646
Собственно вывод статьи....649
Инструменты для работы над статьями....652

**Глава 42. Комментарии....657**
Маршруты....657
Миграция и модель....657
Комментарии, относящиеся к выбранной статье....659
Страницы для работы с комментариями....663

**Глава 43. Хранилище файлов....668**
Маршруты....668
Миграция и модель....668
Миграция....669
Модель....670
Контроллер....671
Визуальная часть....674
Шаблон....675
Представление....676
Web-сценарий....678

**РАЗДЕЛ 7. НАНЕСЕНИЕ ПОСЛЕДНИХ ШТРИХОВ И ПУБЛИКАЦИЯ WEB-САЙТА....685**

**Глава 44. Программируемая графика HTML 5....687**
Канва....687
Контекст рисования....687
Рисование прямоугольников....688
Задание цвета, уровня прозрачности и толщины линий....688
Рисование сложных фигур....690
Как рисуются сложные контуры?....690
Перо. Перемещение пера....690
Прямые линии....691
Дуги....691
Кривые Безье....692
Прямоугольники....693
Задание стиля линий....693
Вывод текста....695
Использование сложных цветов....696
Линейный градиент....696
Радиальный градиент....698
Графический цвет....699

Вывод внешних изображений....700
Создание тени у рисуемой графики....701
Преобразования системы координат....701
Сохранение и загрузка состояния....702
Перемещение начала координат канвы....702
Поворот системы координат....703
Изменение масштаба системы координат....704
Управление наложением графики....705
Создание маски....706

**Глава 45. Хранение данных на стороне клиента....707**
Хранилище HTML 5....707
Временное хранение текста статей на стороне клиента....708

**Глава 46. Публикация Web-сайта....711**
Подготовка Web-сайта к публикации....711
Указание окончательных настроек....711
Удаление ненужных данных....712
Создание страниц с сообщениями об ошибках....714
Публикация сайта....715

**Заключение....717**

**ПРИЛОЖЕНИЯ....719**

**Приложение 1. Установка и настройка пакета OpenServer....721**
Установка....721
Запуск, перезапуск и остановка....722
Настройка....723
Запуск консоли OpenServer....724

**Приложение 2. Работа с базами данных MySQL в программе phpMyAdmin....726**
Запуск и вход....726
Работа с пользователями....727
Создание пользователя и базы данных....727
Правка и удаление пользователей....729
Работа с записями таблиц....730
Перенос содержимого из одной базы данных в другую....732
Экспорт данных....732
Импорт данных....733
Выход....733

**Приложение 3. Перекодирование видеофайлов в формат MP4....734**

**Приложение 4. Файловый архив....738**

**Предметный указатель....739**

# Введение в быструю разработку сайтов

Рассмотрим типичную ситуацию. К разработчику сайтов приходит заказчик.

**Заказчик:** Мне нужен сайт. Только быстро!

**Разработчик:** Хорошо-хорошо! Какого рода сайт?

**Заказчик:** Система электронных публикаций. Чтобы посетители могли регистрироваться, писать статьи, а читатели оставлять на них комментарии. Только быстро!

**Разработчик:** Ясно. Сделаем следующим образом: все статьи будут храниться в базе данных, особая программа будет извлекать их из базы, преобразовывать к виду, понимаемому Web-обозревателями (то есть превращать в обычные Web-страницы), и выводить на экран.

**Заказчик:** Вам виднее — вы ведь разработчик… Делайте, как считаете нужным. Только быстро!

**Разработчик:** Нужно сделать еще разбиение статей на категории и подкатегории…

**Заказчик:** Да-да! Только быстро!

**Разработчик:** …и систему разграничения доступа, чтобы статьи могли оставлять только зарегистрированные пользователи…

**Заказчик:** Да-да! Только быстро!

**Разработчик:** …и не забыть про защиту от спамеров, например, CAPTCHA… Хорошо, сделаем!

**Заказчик:** Заплачу любые деньги!

**Разработчик:** Только быстро! Тьфу!.. прошу прощения…

Заказчик уходит. Разработчик погружается в тягостные раздумья…

## Что придется сделать разработчику?

Итак, что разработчик должен сделать, чтобы угодить заказчику?

- Нарисовать макеты всех страниц, что в совокупности составят заказанный сайт.

  Такой макет представляет собой обычное графическое изображение, представляющее готовую страницу. Обычно его делают в графических пакетах, позво-

ляющих располагать отдельные фрагменты на собственных слоях (например, в Adobe Photoshop), чтобы любой из этих фрагментов можно было скопировать, сохранить в отдельном изображении и использовать в верстке.

- ❒ Сверстать страницы на основе макетов.
- ❒ Написать Web-сценарии — программы, которые встраиваются непосредственно в страницы и выполняют какие-либо действия над их содержанием в ответ на манипуляции посетителя. Так, сценарии могут подсвечивать какие-либо элементы при наведении на них курсора мыши, скрывать одни элементы и показывать другие или подгружать дополнительные данные, чтобы вывести их на экран.
- ❒ Разработать структуру базы данных, в которой будут храниться все внутренние данные сайта: списки категорий, подкатегорий, статьи, комментарии, список зарегистрированных пользователей и проч. Создать эту базу.
- ❒ Превратить сверстанные ранее страницы в своего рода заготовки, или шаблоны. Они понадобятся на следующем шаге.
- ❒ Написать серверные программы, которые будут работать совместно с Web-сервером, извлекать данные из базы и преобразовывать их в Web-страницы на основе созданных ранее шаблонов.
- ❒ Написать серверные программы, которые будут реализовывать разграничение доступа, защиту от злоумышленников и выполнять всевозможные служебные операции.
- ❒ Проверить все на предмет устойчивой работы и отсутствия ошибок.
- ❒ Опубликовать сайт в Интернете.

Сами Web-страницы (или, как говорят Web-верстальщики, их содержание) вместе с их оформлением (представлением) и сценариями (поведением) составляют клиентскую часть сайта — ее-то и будет наблюдать на экране своего компьютера посетитель. А база данных и все серверные программы — суть серверная часть сайта, «невидимая» посетителям, но не менее (если не более) важная.

Разработка клиентской части сайта — процесс весьма трудоемкий. Только на нее уйдет несколько дней.

А разработка серверной части еще более трудоемка…

## Упрощение разработки серверной части Web-сайта. Фреймворки

Однако мы можем существенно упростить и, соответственно, ускорить процесс программирования серверной части. Как?

Все сайты, включающие, помимо страниц, еще и серверные программы (динамические сайты), содержат ряд программных модулей, которые выполняют одни и те же типовые задачи: работу с базой данных, вывод страниц на основе шаблонов, разграничение доступа, защиту от сетевых атак и т. п. А поскольку такие задачи оди-

наковы во всех сайтах, можно не писать эти модули каждый раз заново, а найти программный продукт, который их реализует, и сделать серверную часть сайта на его основе.

И такие программные продукты существуют! Это *фреймворки*. Уже в готовом виде, образно говоря, прямо из коробки, они готовы предоставить разработчику все, что необходимо, для:

- работы с базами данных;
- генерирования страниц на основе шаблонов;
- реализации разграничения доступа;
- защиты от сетевых атак;
- поддержки AJAX;
- и многого-многого другого.

Разработчику останется лишь установить фреймворк, сконфигурировать его (хотя бы указать сведения, необходимые для подключения к базе данных), написать и встроить в него программные модули, которые реализуют задачи, специфичные для разрабатываемого сайта: обработку конкретных данных и генерирование на их основе конкретных страниц.

Понятно, что, поскольку львиную долю работы сделали за Web-программиста разработчики фреймворка, сайт можно сделать очень быстро. Уж точно значительно быстрее, чем в очередной раз, что называется, изобретая велосипед…

## Фреймворк Laravel — номер один в Web-программировании!

Фреймворков в настоящее время очень много — десятки, а может быть, и сотни. Самым же популярным продуктом такого рода является Laravel. Почему? О, у него много преимуществ!

- Laravel написан на PHP — самом популярном языке программирования динамических сайтов.
- Laravel не имеет каких-либо специфических системных требований и работает на мощностях практически всех присутствующих на рынке хостинг-провайдеров.
- Он предоставляет практически всю базовую функциональность сайтов, включая мощную и полностью готовую к работе систему разграничения доступа, средства для отправки электронной почты и исключительно удобный механизм миграций.
- Все входящие в него инструменты уже сконфигурированы наилучшим для большинства случаев образом.
- Почти все необходимое можно реализовать небольшим количеством программного кода. То есть налицо дополнительное сокращение трудоемкости.

- Существует много дополнительных библиотек, написанных сторонними разработчиками и расширяющих функциональность Laravel.
- И — сладкая вишенка на пышном торте! Laravel включает средства прототипирования — создания заготовок для программных модулей разных типов, в которые разработчику сайта останется лишь добавить свой код.

Как можно пройти мимо такой находки? Да ее любой разработчик сайтов должен тут же взять на заметку!

Возьмем на заметку и мы, пока еще новички в мире Web-программирования.

# О чем эта книга?

Да-да, эта книга посвящена Laravel! И всем прочим Web-технологиям, без которых в трудном, но увлекательном, деле Web-программирования не обойтись. В их числе:

- HTML — язык для разработки самих Web-страниц, их содержания. Мы будем изучать самую актуальную его версию — HTML 5.
- CSS — язык для разработки оформления, или представления, страниц. Конкретно, CSS 3 — самую современную версию этого языка.
- JavaScript — язык программирования Web-сценариев, то есть поведения страниц.
- PHP — язык для написания серверных программ. На этом языке написан сам Laravel, следовательно, и встраиваемые в него дополнительные программные модули, реализующие специфическую для разрабатываемого сайта функциональность, пишутся также на нем.
- MySQL — самая популярная на данный момент программа, обрабатывающая базы данных.

Причем изучать эти технологии мы станем в том же порядке, в котором они здесь перечислены. А потом приступим и к самому Laravel.

А в завершение, в качестве практики, мы создадим полнофункциональный сайт — систему электронных публикаций «СЭП», предоставляющую площадку для размещения в Интернете статей на различные темы. Этот сайт можно свободно использовать как основу для разработки других, более сложных решений.

> ***Материалы полнофункционального сайта***
>
> Электронный архив с материалами сайта «СЭП: Система электронных публикаций» можно скачать с FTP-сервера издательства «БХВ-Петербург» по ссылке **ftp://ftp.bhv.ru/9785977538459.zip** или со страницы книги на сайте **www.bhv.ru** (см. *приложение 4*).

При работе над книгой автор использовал следующие версии ПО:

- Microsoft Windows 10, русскую 32-разрядную редакцию со всеми установленными обновлениями;

- OpenServer 5.2.2 — пакет хостинга, включающий в свой состав все программы, которые необходимы для запуска и отладки разрабатываемого сайта;
- Фреймворк Laravel 5.3.29.

Созданные Web-страницы проверялись в следующих Web-обозревателях:

- Microsoft Internet Explorer 11;
- Microsoft Edge 38.14393.0.0;
- Mozilla Firefox 47-50 (когда писалась книга, программа несколько раз обновлялась автоматически).

Для написания программного кода применялся текстовый редактор Notepad+, который можно найти по интернет-адресу **https://notepad-plus-plus.org/**.

# Типографские соглашения

В книге будут постоянно приводиться форматы написания различных конструкций, применяемых в языках HTML, CSS, JavaScript и PHP. В них использованы особые типографские соглашения, которые мы сейчас изучим.

> ***Внимание!***
> Все эти типографские соглашения применяются автором только в форматах написания языковых конструкций. В реальном программном коде они не имеют смысла.

- В угловые скобки (<>) заключаются и дополнительно выделяются курсивом наименования различных значений. В реальный код, разумеется, должны быть подставлены реальные значения. Например:

  ```
  <страница или элемент>.getElementsByClassName(<имя стилевого класса>)
  ```

  Здесь вместо подстроки *<страница или элемент>* должны быть подставлены реальная страница или реальный элемент страницы, а вместо подстроки *<имя стилевого класса>* — реальное имя стилевого класса.

- В квадратные скобки ([]) заключаются необязательные фрагменты кода. Например:

  ```
  background-position: <горизонтальная позиция> [<вертикальная позиция>]
  ```

  Здесь *вертикальная позиция* может присутствовать, а может и отсутствовать.

- Символом вертикальной черты (|) разделяются параметры или значения, из которых в коде должно присутствовать лишь одно. Например:

  ```
  font-style: normal|italic|oblique
  ```

  Здесь допускается указание лишь одного из перечисленных значений — либо `normal`, либо `italic`, либо `oblique`.

- Слишком длинные, не помещающиеся на одной строке языковые конструкции, автор разрывал на несколько строк и в местах разрывов ставил знаки ↳. Например:

```
var lstSubcategory = document.querySelector↳
("select[name=subcategory]");
```

  Приведенный код разбит на две строки, но должен быть набран в одну. Символ ↳ при этом нужно удалить.

- Многоточием (. . .) помечены пропущенные по тем или иным причинам фрагменты кода. Обычно такое можно встретить в исправленных впоследствии фрагментах кода — приведены лишь собственно исправленные выражения, а оставшиеся неизмененными пропущены.

  Также многоточие используется, чтобы показать, в какое место должен быть вставлен вновь написанный код, — в начало исходного фрагмента, в его конец или в середину, между уже присутствующими в нем выражениями.

> ***ЕЩЕ РАЗ ВНИМАНИЕ!***
>
> Все приведенные здесь типографские соглашения имеют смысл только в форматах описания конструкций языков HTML, CSS, JavaScript и PHP. В коде примеров используется лишь знак ↳ и многоточие.

# Что нас ждет в будущем?

Когда работа над этой книгой уже подходила к концу, была выпущена новая версия Laravel — 5.4. Она предлагает ряд полезных нововведений: средства для написания шаблонов оповещений, подобные командам Blade, применяемым в коде шаблонов; возможность разработки компонентов, аналогичную таковой у фреймворка Yii; расширенные средства обработки коллекций записей; усовершенствования в механизмах событий и фасадов; новые посредники и проч.

Как видим, Laravel пришел в мир Web-программирования всерьез и надолго. И в более отдаленном будущем мы увидим новые версии этого замечательного фреймворка, предлагающего новые и, безусловно, замечательные возможности.

Но что делать, если Laravel все же канет в Лету? Что, если его сменит на рынке другой фреймворк, молодой да резвый? Как в свое время сам Laravel сбросил с почетного первого места в рейтинге популярности аналогичный и, кстати, замечательный продукт Yii. (Автор когда-то написал о нем книгу…)

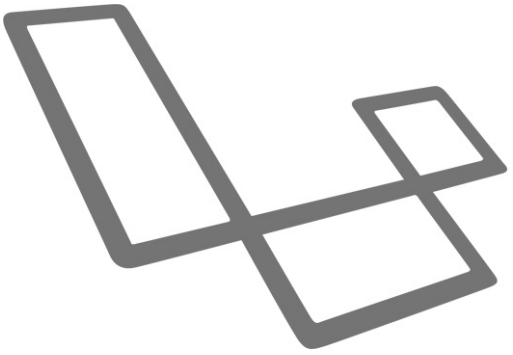
Все (по крайней мере, большинство) Web-фреймворков построены на единых принципах. Они включают модели, осуществляющие работу с таблицами баз данных, шаблоны, на основе которых генерируются страницы, контроллеры, выполняющие задачи по выборке данных и генерированию страниц на основе этих данных и того или иного шаблона, и маршрутизатор, анализирующий полученный в запросе интернет-адрес и на основе результатов анализа определяющий, какое

действие какого контроллера следует запустить. Разница только в реализации всего этого, и она зачастую не так уж и велика.

Так что мы, в случае чего, можем без проблем перейти на другой фреймворк и, после недолгого изучения документации, опять же, без особых проблем писать сайты на нем.

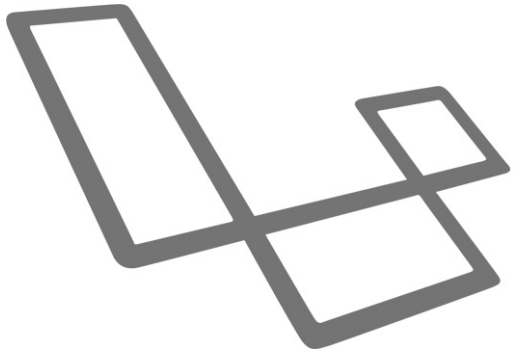
Но сначала нужно прочитать эту книгу. В ней есть все, что понадобится разработчику, чтобы сделать современный динамический сайт.

Быстро. Как и просил заказчик.



# ЧАСТЬ I

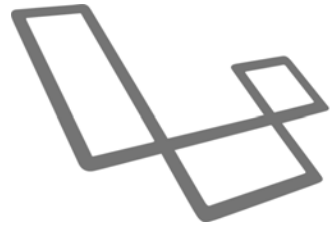
# РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ WEB-САЙТА



# I. РАЗДЕЛ 1

# Содержимое Web-страниц. Язык HTML 5


| | |
|---|---|
| **Глава 1.** | Современный Web-дизайн. Введение в язык HTML 5 |
| **Глава 2.** | Структурирование и оформление текста. Литералы. Комментарии HTML |
| **Глава 3.** | Графика и мультимедиа |
| **Глава 4.** | Таблицы |
| **Глава 5.** | Средства навигации |
| **Глава 6.** | Web-формы и элементы управления. Фреймы |



# ГЛАВА 1

# Современный Web-дизайн. Введение в язык HTML 5

Разработка любого приличного Web-сайта начинается с создания его клиентской части. Что это такое?

## Клиентская часть Web-сайта

*Клиентской* на жаргоне программистов называется та часть приложения, которую пользователь непосредственно наблюдает на экране своего компьютера. Это различные окна, выводящие какие-либо данные, что были получены из базы и обработаны серверной частью.

Применительно к нашему случаю — к сайту — клиентской частью являются всевозможные Web-страницы, которые видит на экране посетитель и которые, опять же, отображают данные, полученные и обработанные серверной частью сайта (речь о которой пойдет в *части II* этой книги).

Говоря поэтично, клиентская часть — лицо сайта, в то время как часть серверная — его душа. И, как всегда, лицо является зеркалом души.

Именно поэтому первый этап создания сайта — написание его клиентской части, всех входящих в его состав Web-страниц.

Но, постойте, спросит иной читатель, зачем же писать все эти страницы, если они все равно будут генерироваться программами, входящими в состав серверной части? Зачем выполнять двойную работу?

Во-первых, на начальном этапе разрабатываются образцы, «рыбы» страниц, содержащие какие-либо произвольные данные (все равно реальная информация потом будет формироваться серверной частью). Во-вторых, программировать серверные программы удобнее, имея перед глазами готовые образцы страниц, — с этим, думается, не поспорит никто.

И, в-третьих и в-главных, современные PHP-фреймворки (мы ведь еще не передумали использовать Laravel?) выполняют формирование страниц на основе так называемых *шаблонов* (речь о них пойдет в *главе 30*). Они представляют собой обычные Web-страницы, в нужных местах кода которых — там, где должны быть под-

ставлены выводимые серверной частью данные, — имеются особые пометки. Понятно, что эти шаблоны удобнее создавать на основе уже готовых страниц.

Итак, на первом этапе нам требуется создать набор страниц, которые впоследствии станут для нас образцами при программировании серверной части сайта. То есть создать статический сайт.

## Статические Web-страницы и Web-сайты

*Статические страницы* не генерируются серверными программами. Они верстаются Web-верстальщиком на основе подготовленного Web-дизайнером макета (представляющего собой обыкновенное графическое изображение), сохраняются в файлах, помещаются на диск серверного компьютера и все время существования сайта остаются неизменными (разумеется, до того момента, когда верстальщик не решит эти страницы переделать).

Набор взаимосвязанных статических страниц составляет *статический сайт*. Таких сайтов в Интернете довольно много — они представляют информацию, которая меняется крайне редко или не меняется вообще. Статические сайты просты и недороги в разработке и могут быть опубликованы у любого *хостинг-провайдера* (организации, предоставляющей услуги по публикации сайтов в Сети), вследствие чего популярны до сих пор.

В виде статических сайтов создается также клиентская часть полноценных динамических сайтов. Чем мы сейчас и займемся.

## Содержание, представление и поведение Web-страниц

Любая страница включает в себя содержание, представление и поведение.

- *Содержание* — это, собственно, сама страница, та информация, ради которой она создана. Эта информация структурирована, то есть разбита на отдельные абзацы, заголовки, представлена в виде списков, таблиц, сгруппирована в более крупные разделы: статьи, иллюстрации, «шапки», «поддоны» и проч.
- *Представление* — это оформление страницы и отдельных ее элементов. Представление описывает, в частности, каким шрифтом будет выводиться текст абзацев и заголовков, какой отступ слева будут иметь пункты списков, какую рамку получат таблицы и даже какой эффект будет иметь наведение курсора мыши на гиперссылки.
- *Поведение* — это интерактивность страницы, ее способность реагировать на действия посетителя. Это всевозможные интерактивные эффекты, наподобие замены одной картинки на другую при наведении курсора мыши, и интерактивные элементы страницы: слайдеры, лайтбоксы, «блокноты», таблицы с возможностью прокрутки и выбора элемента и проч.

Содержание, представление и поведение, согласно современным тенденциям Web-верстки, делаются независимыми друг от друга, их даже хранят в отдельных файлах. Это позволяет упростить сопровождение сайтов (для исправления какой-либо страницы в большинстве случаев достаточно изменить лишь содержание, не трогая представление и поведение, или, напротив, изменить представление, не трогая содержание и поведение) и использовать одно и то же представление и поведение (или отдельные их части) применительно к разному содержанию.

Более того, содержание, представление и поведение создаются с применением совершенно разных технологий (о которых мы, разумеется, поговорим, но позже).

# Интернет: как все это работает?

Прежде чем начать работу над клиентской частью сайта, следует хотя бы в общих чертах выяснить, как и по каким принципам работает Интернет. Мы не станем вдаваться в детали — в конце концов, мы собираемся разрабатывать сайты, а не администрировать серверы или настраивать сетевое оборудование.

Ранее упоминалось о статических сайтах, которые до сих пор часто встречаются в Интернете. Каждый такой сайт представляет собой набор файлов, хранящих входящие в его состав страницы и вспомогательные данные, и… и что дальше?

## Web-серверы

Понятно, что файлы сайта хранятся на дисках компьютера, подключенного к Интернету. Понятно, что это либо компьютер, принадлежащий автору или авторам сайта, либо компьютер хостинг-провайдера. Но каким образом мы можем извлекать эти расположенные неизвестно где файлы и наблюдать их содержимое — страницы сайта — у себя на экране?

Прежде всего, на компьютере, где хранится сайт, установлена и запущена особая программа — *Web-сервер*. Она относится к весьма специфической категории программ, называемых *серверами*. Она не выводит на экран никаких окон с информацией (максимум — сообщение о критической ошибке при запуске) и вообще никак не взаимодействует с пользователем.

Единственное назначение Web-сервера — обслуживание запросов Web-обозревателей. (Которые относятся к категории программ-*клиентов*, непосредственно взаимодействующих с пользователями.)

Как только пользователь набирает в строке адреса Web-обозревателя интернет-адрес нужной ему страницы, происходит целая цепочка примечательных событий.

1. Web-обозреватель из набранного интернет-адреса извлекает адрес компьютера, на котором опубликован сайт и на котором работает программа Web-сервера, и отправляет последнему запрос, в состав которого включен весь введенный интернет-адрес целиком.
2. Web-сервер получает отправленный Web-обозревателем запрос, извлекает из находящегося в нем интернет-адреса путь к запрашиваемому файлу, считывает этот файл и отправляет назад, Web-обозревателю.

Если файл с указанным путем отсутствует или по какой-то причине не может быть прочитан, Web-сервер отправляет Web-обозревателю страницу с сообщением об ошибке, которая либо генерируется программно, либо хранится в особом файле.

3. Web-обозреватель получает отправленный Web-сервером файл и либо выводит его содержимое на экран (если это страница), либо выполняет над ним какие-то другие действия (например, предлагает сохранить его на локальном диске).

Все это происходит, так сказать, за кулисами, и пользователь не уведомляется о том, что Web-обозреватель отправляет куда-то какие-то запросы и получает откуда-то какие-то ответы. Пользователь может даже не догадываться о существовании какого-то там Web-сервера.

## Интернет-адреса

А теперь подробнее рассмотрим интернет-адреса, которые набираются в строке адреса пользователем и отправляются Web-серверу Web-обозревателем.

Вообще, интернет-адрес записывается в формате:

```
[<обозначение протокола>://]<интернет-адрес сайта>[:<номер порта>]↵
[<путь к запрашиваемому файлу или папке>][?<GET-параметры>][#<имя якоря>]
```

Давайте рассмотрим все входящие в него составляющие.

- *Обозначение протокола* — указывает протокол и, опосредованно, серверную программу, к которой выполняется запрос.

  *Протоколом* называется система команд, посредством которых программа-клиент и программа-сервер взаимодействуют между собой. Будучи указанным в интернет-адресе, протокол неявно идентифицирует серверную программу, к которой отправляется запрос. Так, если указать в качестве обозначения протокола строку `http`, запрос к программе Web-сервера будет сформирован с применением протокола *HTTP* (HyperText Transfer Protocol, протокол передачи гипертекста), которым пользуются для «общения» Web-обозреватели и Web-серверы.

  Протокол HTTP будет использован и в том случае, когда обозначение протокола не указано вовсе.

- *Интернет-адрес сайта* — идентифицирует сам опубликованный сайт. Может представлять собой:

  - либо, как чаще всего случается, *доменное имя* — символьное наименование, удобное для запоминания и имеющее вид **www.yandex.ru**, **www.php.net** и т. п.;
  - либо низкоуровневый *IP-адрес* — четырехзначное шестнадцатеричное число вида **186.143.27.9**. Низкоуровневые IP-адреса применяются значительно реже доменных имен.

  Интернет-адрес сайта — единственная обязательная для указания часть интернет-адреса.

- *Номер порта* — указывает номер *TCP-порта*, одного из своего рода виртуальных каналов, по которым различные программы-клиенты общаются со «своими» программами-серверами, не мешая друг другу.

  Каждый тип программ-серверов по умолчанию использует один отведенный им TCP-порт — так, за Web-серверами закреплен порт 80. Однако по разным причинам сервер может быть настроен с тем, чтобы использовать другой порт, и тогда, чтобы обратиться к этому серверу, требуется явно указать номер порта в составе интернет-адреса.

  Если номер порта не указан, Web-обозреватель подразумевает, что должен применяться TCP-порт по умолчанию (для Web-сервера — порт 80).

- *Путь к запрашиваемому файлу или папке* — собственно указывает файл, который требуется получить.

  При публикации в Интернете сайта все составляющие его файлы на диске серверного компьютера помещаются в особой папке, носящей название *корневой*. Путь к этой папке указывается в настройках Web-сервера, чтобы последний смог найти ее.

  Все пути к запрашиваемым файлам, получаемые в составе интернет-адресов в запросах от Web-обозревателях, отсчитываются Web-сервером от корневой папки сайта.

  Может быть указан как путь непосредственно к запрашиваемому файлу, так и путь к папке. В последнем случае Web-сервер извлекает из этой папки и отправляет Web-обозревателю так называемый *файл по умолчанию*. Файл по умолчанию должен иметь имя index или default и расширение htm, html или php (впрочем, эти параметры можно изменить в настройках Web-сервера).

  Если путь к запрашиваемому файлу вообще не указан, будет считан и отправлен файл по умолчанию, хранящийся в корневой папке сайта. Обычно это *главная страница* сайта, с которой, собственно, и начинается «путешествие» по нему.

- *GET-параметры* — это набор данных, пересылаемых серверной программе. Мы рассмотрим их в *главе 6*.

- *Имя якоря* — указывает на фрагмент страницы. Если оно указано, содержимое страницы будет прокручено в окне Web-обозревателя таким образом, чтобы соответствующий якорю фрагмент появился в поле зрения посетителя. Если не указан, страница будет открыта как обычно. Более подробно о якорях мы поговорим в *главе 5*.

Как видим, интернет-адрес однозначно идентифицирует конкретный файл, входящий в состав конкретного сайта, и даже может указывать на конкретную часть страницы.

Вот несколько примеров интернет-адресов:

- **http://www.somesite.ru:8000/apps/app.php?page=1#chapter2**

  Доступ по протоколу HTTP к сайту **www.somesite.ru** по TCP-порту 8000. Здесь запрашивается файл серверной программы **app.php**, которая хранится в папке

**apps** корневой папки сайта и которой пересылается GET-параметр **page** со значением **1**, в полученной странице следует показать фрагмент, соответствующий якорю **chapter2**.

- **www.othersite.ru/folder/page1.html**

  Запрашивается файл **page1.html**, хранящийся в папке **folder** корневой папки сайта **www.othersite.ru**.

- **www.3dnews.ru**

  Запрашивается файл по умолчанию (главная страница), хранящийся в корневой папке сайта **www.3dnews.ru**.

На этом пока закончим разговор об интернет-адресах. Мы еще вернемся к ним в *главе 5*, когда будем рассматривать средства навигации по Web-сайту и, в частности, гиперссылки.

Давайте лучше приступим к изучению языка HTML, которым в дальнейшем будем пользоваться постоянно.

# Введение в язык HTML 5

Содержание Web-страниц создается с применением языка *HTML* (HyperText Markup Language, язык гипертекстовой разметки). В этом разделе мы будем иметь дело исключительно с этим языком.

На данный момент самой современной его версией является HTML 5, спецификация которого совсем недавно получила статус окончательной. Ее-то мы и станем использовать для написания наших страничек.

Язык HTML довольно прост и основан на нескольких совсем не сложных принципах. Сейчас мы в этом убедимся.

## Первая Web-страница

Изучать HTML лучше всего на примере. Так что давайте сразу же создадим нашу первую Web-страничку. Сделать это можно в любом простейшем текстовом редакторе — например, в Блокноте, входящем в комплект поставки Windows.

Откроем Блокнот и наберем в нем текст (или, как говорят бывалые программисты, *код*), приведенный в листинге 1.1.

**Листинг 1.1**

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Пример Web-страницы</title>
  </head>
```

```
  <body>
    <h1>Изучаем язык HTML</h1>
    <p>Язык <strong>HTML</strong> предназначен для создания содержимого
    Web-страниц.</p>
  </body>
</html>
```

Проверим набранный код на ошибки и сохраним в файл с именем 1.1.html. Только сделаем при этом две важные вещи.

1. Сохраним HTML-код в кодировке UTF-8. Для этого в диалоговом окне сохранения файла Блокнота найдем раскрывающийся список **Кодировка** и выберем в нем пункт **UTF-8**.
2. Заключим имя файла в кавычки. Иначе Блокнот добавит к нему расширение txt, и наш файл получит имя 1.1.html.txt.

Все, наша первая Web-страница готова! Теперь осталось открыть ее в любом Web-обозревателе (автор задействовал для этой цели Internet Explorer 11) и посмотреть на результат (рис. 1.1).

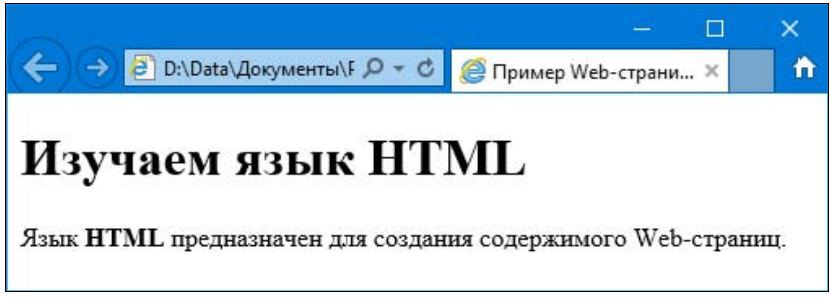



**Рис. 1.1.** Наша первая Web-страница

Мы только что создали Web-страницу, содержащую большой «кричащий» заголовок и абзац, включающий выделенную полужирным шрифтом аббревиатуру **HTML**. И все это — в «голом» тексте, набранном в Блокноте!

Настала пора уяснить несколько моментов, касающихся Web-страниц. Во-первых, все статические страницы представляют собой текстовые файлы. Во-вторых, файлы страниц должны иметь расширение html (используется чаще всего) или htm (на данный момент устаревшее).

## Теги и атрибуты тегов

Теперь посмотрим, что же мы такое написали в файле 1.1.html. Пока что ограничимся следующим небольшим фрагментом HTML-кода:

```
<h1>Изучаем язык HTML</h1>
<p>Язык <strong>HTML</strong> предназначен для создания содержимого
Web-страниц.</p>
```

Здесь мы видим тексты заголовка и абзаца. И еще странные слова, взятые в угловые скобки, — символы < и >. Что это такое?

Это *теги* HTML, особые команды, задающие назначение того или иного фрагмента содержимого, — будет он абзацем, заголовком или важным текстом, который следует выделить полужирным шрифтом.

Начнем с тегов `<h1>` и `</h1>`, поскольку они идут первыми. Эти теги превращают фрагмент текста, находящийся между ними, в заголовок. Тег `<h1>` помечает начало фрагмента, на который распространяется действие тега, и называется *открывающим*. А тег `</h1>` устанавливает конец «охватываемого» фрагмента и называется *закрывающим*. Что касается самого фрагмента, заключенного между открывающим и закрывающим тегами, то он называется *содержимым тега*. Именно к содержимому применяется действие тега.

Все теги HTML обозначаются символами < и >, внутри которых находится *имя тега*, определяющее его назначение (имена тегов можно набирать как строчными, так и прописными буквами, но в HTML 5 обычно используются строчные буквы). Закрывающий тег должен иметь то же имя, что и открывающий, единственное отличие закрывающего тега — символ /, который ставится между символом < и именем тега.

Рассмотренные нами теги `<h1>` и `</h1>` в HTML фактически считаются одним тегом `<h1>`. Такой тег называется *парным*.

Другой парный тег — `<p>` — создает на Web-странице абзац из текста, являющегося содержимым этого тега. Такой абзац будет отображаться с отступами сверху и снизу. Если он полностью помещается по ширине в окне Web-обозревателя, то отобразится в одну строку, в противном случае Web-обозреватель сам выполнит перенос строк по пробелам (это же справедливо и для заголовка).

Третий парный тег — `<strong>` — помечает свое содержимое как важный текст, на котором следует заострить внимание посетителя, для чего выводит его полужирным шрифтом. Видно, что этот тег вложен внутрь содержимого тега `<p>`. Это значит, что содержимое тега `<strong>` будет отображаться как часть абзаца (тега `<p>`).

А теперь рассмотрим вот такой тег:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Это *одинарный тег*, не имеющий закрывающей пары. Такие теги действует в той точке HTML-кода, где они находятся, и либо задают какие-либо сведения о самой странице (как приведенный здесь тег `<meta>`), либо помещают в соответствующее место страницы какой-либо элемент, не относящийся к тексту.

А еще мы видим, что в этом теге сразу после имени тега идут еще какие-то данные. Это *атрибуты тега*, задающие его параметры. В частности, атрибуты `http-equiv` и `content` тега `<meta>` указывают тип документа и его кодировку.

Каждый атрибут тега имеет *имя*, за которым ставится знак равенства и *значение* этого атрибута, взятое в двойные кавычки. Так, атрибут с именем `http-equiv` имеет значение `"Content-Type"`, указывающее, что этот тег задает тип документа. А атри-

бут с именем `content` имеет значение `"text/html; charset=utf-8"`, обозначает тип документа «Web-страница» и указывает, что он сохранен в кодировке UTF-8.

Атрибуты тегов бывают обязательными и необязательными. *Обязательные* атрибуты должны присутствовать в теге в обязательном порядке. *Необязательные* же атрибуты могут быть опущены — в таком случае тег ведет себя так, будто соответствующему атрибуту присвоено значение по умолчанию. Атрибуты `http-equiv` и `content` тега `<meta>` являются обязательными.

## Вложенность тегов

Если мы снова посмотрим на приведенный в листинге 1.1 HTML-код, то заметим, что одни теги вложены в другие. Так, тег `<strong>` вложен в тег `<p>`, являясь частью его содержимого. Тег `<p>`, в свою очередь, вложен в тег `<body>`, а тот — в тег `<html>`. (Теги `<body>` и `<html>` мы рассмотрим чуть позже.) Такая *вложенность тегов* в HTML — обычное явление.

Когда Web-обозреватель встречает тег, вложенный в другой тег, он как бы накладывает действие «внутреннего» тега на эффект «внешнего». Так, действие тега `<strong>` будет наложено на действие тега `<p>`, и фрагмент абзаца окажется выделенным полужирным шрифтом, при этом оставаясь частью этого абзаца.

Теперь — внимание! Порядок следования закрывающих тегов должен быть обратным тому, в котором следуют теги открывающие. Говоря иначе, теги со всем их содержимым должны полностью вкладываться в другие теги, не оставляя «хвостов» снаружи.

Осталось выучить несколько новых терминов. Тег, в который непосредственно вложен тот или иной тег, называется *родительским*, или *родителем*. В свою очередь, тег, вложенный в родительский тег, называется *дочерним*, или *потомком*. Так, для тега `<p>` в приведенном ранее листинге тег `<body>` — родительский, а тег `<strong>` — дочерний. Любой тег может иметь сколько угодно дочерних тегов, но только один родительский (что, впрочем, понятно — не может же он быть непосредственно вложен одновременно в два тега).

Аналогично, элемент Web-страницы, в который вложен элемент, создаваемый каким-либо тегом, называется *родительским*, или *родителем*. А элемент страницы, который вложен в этот элемент, — *дочерним*, или *потомком*.

*Уровень вложенности* того или иного тега показывает количество тегов, в которые он последовательно вложен. Так, если принять за точку отсчета тег `<html>`, то тег `<body>` будет иметь первый уровень вложенности, т. к. он вложен непосредственно в тег `<html>`. Тег `<p>` же будет иметь второй уровень вложенности, т. к. он вложен в тег `<body>`, а тот, в свою очередь, — в тег `<html>`. В сложных страницах уровень вложенности иных тегов может составлять несколько десятков.

# Форматирование Web-страниц

А теперь рассмотрим несколько важных моментов, касающихся структуры HTML-кода страницы и сведений о ней, что необходимы Web-обозревателю для успешного вывода ее содержания.

## Секции Web-страницы

Прежде всего, содержание любой страницы разделяется на две *секции*. Для этого применяются особые теги, которые часто называют *невидимыми*, поскольку они никак не отображаются на экране, по крайней мере, напрямую.

Давайте мысленно удалим из листинга 1.1 все содержимое, за исключением этих тегов, в результате чего получим листинг 1.2.

**Листинг 1.2**

```
<html>
  <head>
    . . .
  </head>
  <body>
    . . .
  </body>
</html>
```

Начнем с парного тега <body>, о котором уже упоминали ранее. Он формирует *секцию тела* страницы, которая описывает собственно содержание страницы, выводящееся на экран. Все теги, что формируют содержание, должны находиться в этой секции:

```
<body>
  <h1>Изучаем язык HTML</h1>
  <p>Язык <strong>HTML</strong> предназначен для создания содержимого
  Web-страниц.</p>
</body>
```

А в парном теге <head> находится *секция заголовка* Web-страницы. (Не путать с заголовком, который создается с помощью тега <h1>!) В эту секцию помещаются сведения о параметрах страницы, не отображаемые на экране и предназначенные исключительно для Web-обозревателя (например, знакомый нам тег <meta>):

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Пример Web-страницы</title>
</head>
```

И заголовок, и тело страницы находятся внутри парного тега <html>. Этот тег расположен на самом высшем (нулевом) уровне вложенности и не имеет родителя.

Любая страница должна быть правильно отформатирована: иметь секции заголовка и тела и необходимый набор метатегов (о которых мы также поговорим). Только в этом случае она будет считаться корректной с точки зрения стандартов HTML.

## Метаданные и метатеги

Мы только что узнали о секции заголовка страницы и о том, что она предназначена для задания параметров страницы. Эти параметры называются *метаданными*, т. е. данными, описывающими другие данные. А для их задания применяются *метатеги*, относящиеся к числу невидимых тегов.

Прежде всего, в состав метаданных входит *название* Web-страницы. Оно отображается в заголовке окна Web-обозревателя, где выводится содержание этой страницы, и хранится в «истории» (списке посещенных к настоящему времени страниц). Название Web-страницы помещается в парный тег <title>:

```
<head>
  . . .
  <title>Пример Web-страницы</title>
</head>
```

Далее, это уже знакомый нам тег <meta>, задающий тип документа и кодировку текста, которым он набран.

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  . . .
</head>
```

Приведенный тег указывает, что наш документ представляет собой Web-страницу, и задает для него кодировку текста UTF-8.

> ***На заметку***
>
> Кодировка *UTF-8* — это разновидность кодировки Unicode, предназначенная для написания Web-страниц. Кодировка Unicode (а значит, и UTF-8) может закодировать все символы всех языков, имеющихся на Земле.

> ***Внимание!***
>
> Для кодирования текста страниц, написанных на языке HTML 5, настоятельно рекомендуется применять кодировку UTF-8. Использование других кодировок хоть и допускается, но не приветствуется.

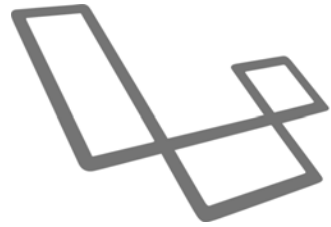
Теперь осталось рассмотреть последний тег, находящийся в самом начале HTML-кода нашей страницы — даже вне «всеобъемлющего» тега <html>. Это метатег <!doctype>, который задает, во-первых, версию языка HTML, на которой написана страница, а во-вторых, разновидность этой версии.

В нашем случае тег <!doctype> выглядит так:

```
<!doctype html>
```

Он указывает, что для создания страницы применяется версия 5 языка HTML.

> ***ВНИМАНИЕ!***
>
> Код любой страницы, написанной на HTML 5, должен включать метатег <!doctype html>, расположенный в самом его начале. В противном случае Web-обозреватель не сможет правильно обработать страницу.



# ГЛАВА 2

# Структурирование и оформление текста. Литералы. Комментарии HTML

В предыдущей главе мы изучили основы языка HTML, на котором создается содержание страниц. В этой главе мы познакомимся с тегами, с помощью которых содержание страниц структурируется и оформляется.

## Структурирование текста

Сначала мы изучим средства HTML по структурированию текста — разбиению его на абзацы, заголовки, списки и т. п. Это выполняется с помощью особых тегов.

И начнем мы с самых простых и самых «старых» средств, появившихся еще в первых версиях HTML.

### Абзацы и заголовки

Из чего состоит текст? Правильно, из отдельных абзацев, включающих логически законченные и относительно независимые фрагменты текста, и заголовков, дающих названия отдельным частям текста: разделам, главам и параграфам.

Как мы уже знаем из *главы 1*, для создания абзаца применяется парный тег `<p>`. Содержимое этого тега становится текстом абзаца:

```
<p>Я - совсем короткий абзац.</p>
<p>А я - уже более длинный абзац. Возможно, Web-обозреватель разобьет
меня на две строки.</p>
```

Теперь о заголовках. Скажем сразу, что в HTML есть такое понятие, как *уровень заголовка*. Он представляет собой целое число от 1 до 6, указывающее, насколько крупную часть текста открывает заголовок.

- Заголовок первого уровня (1) открывает самую крупную часть текста. Как правило, это заголовок всей Web-страницы. Web-обозреватель выводит заголовок первого уровня самым большим шрифтом.

- Заголовок второго уровня (2) открывает более мелкую часть текста. Обычно это большой раздел. Web-обозреватель выводит заголовок второго уровня несколько меньшим шрифтом, чем заголовок первого уровня.
- Заголовок третьего уровня (3) открывает еще более мелкую часть текста — обычно главу в разделе. Web-обозреватель выводит такой заголовок еще меньшим шрифтом.
- Заголовки четвертого, пятого и шестого уровней (4–6) открывают отдельные параграфы, крупные, более мелкие и самые мелкие соответственно. Web-обозреватель выводит заголовки четвертого уровня еще меньшим шрифтом, а шрифт заголовков пятого и шестого уровней — даже меньше, что шрифт, которым выводятся обычные абзацы.

Чтобы дополнительно выделить заголовки любого уровня, Web-обозреватель выводит их полужирным шрифтом.

Если мы откроем в Блокноте нашу первую страницу, сохраненную в файле 1.1.html, заменим в ней содержимое секции тела (тега `<body>`) на код, приведенный в листинге 2.1, сохраним под именем 2.1.html и откроем в Web-обозревателе, мы увидим то, что показано на рис. 2.1. Так мы можем примерно оценить размер шрифта, которым выводятся заголовки различных уровней.

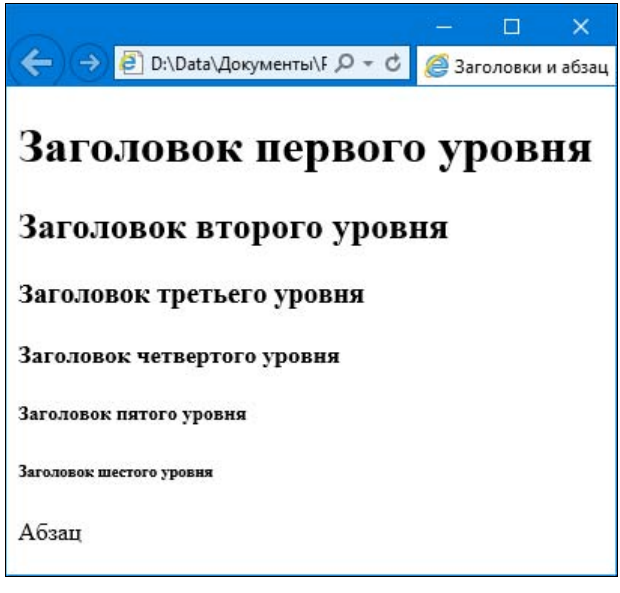



**Рис. 2.1.** Заголовки различных уровней и абзац

**Листинг 2.1**

```
<h1>Заголовок первого уровня</h1>
<h2>Заголовок второго уровня</h2>
<h3>Заголовок третьего уровня</h3>
```

```
<h4>Заголовок четвертого уровня</h4>
<h5>Заголовок пятого уровня</h5>
<h6>Заголовок шестого уровня</h6>
<p>Абзац</p>
```

## Блочные элементы HTML

А теперь рассмотрим один важный вопрос. Он касается типов элементов страниц, поддерживаемых HTML, и правил, согласно которым они выводятся на экран.

Уясним сразу, что абзацы и заголовки — это так называемые *блочные элементы*. Такие элементы:

- выводятся по вертикали в направлении сверху вниз;
- часто отделяются отступами от соседних блочных элементов;
- занимают все свободное пространство по ширине;
- принимают такую высоту, чтобы только вместить свое содержимое;
- могут содержать как обычный текст, так и другие блочные элементы;
- в случае необходимости Web-обозреватель сам выполняет перенос текста — содержимого блочных элементов.

И, если уж зашла речь о текстовом содержимом тегов, давайте приведем правила, согласно которым выполняется его вывод:

- два и более следующих друг за другом пробела считаются за один пробел;
- табуляция и перевод строки считается за пробел;
- пробелы, табуляции и переводы строки между тегами, создающие блочные элементы, никак не отображаются на Web-странице. (Благодаря этому мы можем форматировать HTML-код для более удобного чтения, в том числе, ставить отступы для обозначения вложенности тегов.)

> ***Внимание!***
>
> Все без исключения теги, предназначенные для структурирования текста, создают блочные элементы.

## Списки

Списки используются для того, чтобы представить читателю перечень каких-либо позиций, пронумерованных или непронумерованных, — пунктов списка. Список с пронумерованными пунктами так и называется — *нумерованным*, а с непронумерованными — *маркированным*. В маркированных списках пункты помечаются особым значком — *маркером*, который ставится левее пункта списка.

Маркированные списки обычно служат для простого последовательного вывода каких-либо позиций, порядок следования которых не важен. Если же требуется обратить внимание читателя на то, что позиции должны следовать друг за другом

именно в том порядке, в котором они перечислены, необходимо применить нумерованный список.

Любой список в HTML создается в два этапа. Сначала пишут строки, которые станут пунктами списка, и каждую из этих строк помещают внутрь парного тега <li>. Затем все эти пункты помещают внутрь парного тега <ul> (если создается маркированный список) или <ol> (в случае создания нумерованного списка) — эти теги определяют сам список (листинг 2.2).

**Листинг 2.2**

```
<ul>
  <li>Я - первый пункт маркированного списка.</li>
  <li>Я - второй пункт маркированного списка.</li>
  <li>Я - третий пункт маркированного списка.</li>
</ul>
<ol>
  <li>Я - первый пункт нумерованного списка.</li>
  <li>Я - второй пункт нумерованного списка.</li>
  <li>Я - третий пункт нумерованного списка.</li>
</ol>
```

Web-обозреватель сам расставляет в пунктах списков необходимые маркеры или нумерацию. Также он выводит пункты списка с отступом слева, а расстояние между ними делает меньшими, чем в случае абзацев или заголовков.

Списки можно помещать друг в друга, создавая *вложенные списки*. Делается это следующим образом. Сначала во «внешнем» списке (в который должен быть помещен вложенный) отыскивают пункт, после которого должен находиться вложенный список. Затем HTML-код, создающий вложенный список, помещают в разрыв между текстом этого пункта и его закрывающим тегом </li>. Если же вложенный список должен помещаться в начале «внешнего» списка, его следует вставить между открывающим тегом <li> первого пункта «внешнего» списка и его текстом. Что, впрочем, логично.

В листинге 2.3 представлен HTML-код, создающий два списка, один из которых вложен внутрь другого. Обратим внимание, где помещается HTML-код, создающий вложенный список.

**Листинг 2.3**

```
<ul>
  <li>Я - первый пункт внешнего списка.</li>
  <li>Я - второй пункт внешнего списка.
    <ul>
      <li>Я - первый пункт вложенного списка.</li>
      <li>Я - второй пункт вложенного списка.</li>
      <li>Я - третий пункт вложенного списка.</li>
```

```
    </ul>
  </li>
  <li>Я - третий пункт внешнего списка.</li>
</ul>
```

HTML позволяет вкладывать нумерованный список внутрь маркированного и наоборот. Количество последовательно вложенных друг в друга списков не ограничено.

HTML также позволяет создать так называемый *список определений*, представляющий собой перечень терминов и их разъяснений. Такой список создают с помощью парного тега <dl>. Внутри него помещают пары «термин — разъяснение». Термины заключают в парный тег <dt>, а разъяснения — в парный тег <dd> (листинг 2.4).

**Листинг 2.4**

```
<dl>
  <dt>Содержание</dt>
  <dd>Информация, отображаемая на Web-странице</dd>
  <dt>Представление</dt>
  <dd>Набор правил, определяющих формат отображения содержания</dd>
  <dt>Поведение</dt>
  <dd>Набор правил, определяющих реакцию Web-страницы или ее элементов на
  действия посетителя</dd>
</dl>
```

## Блочные цитаты и адреса

HTML предлагает нам еще два полезных тега. Первый позволяет создать большую цитату, включающую несколько абзацев, а второй — указать адрес, которым могут быть контактные данные разработчиков сайта, дата написания статьи или просто подпись ее автора.

*Блочная цитата* создается парным тегом <blockquote>. В нем помещается HTML-код, собственно формирующий блочную цитату (листинг 2.5).

**Листинг 2.5**

```
<blockquote>
  <p>Я - начало блочной цитаты.</p>
  <p>Я - продолжение блочной цитаты.</p>
</blockquote>
```

Как видим, в тег <blockquote> можно поместить несколько абзацев. Там также могут иметься заголовки и списки (если уж возникнет такая потребность).

Блочная цитата выводится на экран с отступом слева.

Что касается адреса, то он создается парным тегом <address>. Он ведет себя так же, как тег абзаца <p>, но его содержимое выводится курсивом:

```
<address>Я - адрес создателя этой Web-страницы: почтовый, электронный,
телефоны и факсы.</address>
```

## Текст фиксированного формата

Ранее мы ознакомились с правилами, согласно которым Web-обозреватель выполняет вывод текста. Эти правила, в частности, гласят, что несколько стоящих подряд пробелов считаются за один пробел, также за пробел считаются переводы строк и табуляция.

Однако часто бывает необходимо вывести какой-либо фрагмент текста как есть, со всеми множественными пробелами, стоящими подряд, и переводами строк. К таким фрагментам можно отнести, например, исходные тексты программ, — они набираются согласно правилам, определяемым языком программирования, на которых написаны, и, следовательно, выводиться они должны точно так же, как и набраны.

Специально для таких случаев HTML предусматривает парный тег <pre>. Внутри него помещается текст, который должен быть выведен так же, как он набран, или, как говорят Web-верстальщики, текст *фиксированного формата*.

Листинг 2.6 демонстрирует пример кода, выводящего текст фиксированного формата, а на рис. 2.2 показано, как этот текст выглядит на экране.

**Листинг 2.6**

```
<pre>Этот текст
  будет выведен
    на Web-страницу
      как есть,
        без всяких преобразований.</pre>
```

Правила отображения текста фиксированного формата:

- для вывода используется моноширинный, а не пропорциональный шрифт;
- все пробелы, табуляция и переносы строк сохраняются при выводе (это мы уже знаем);

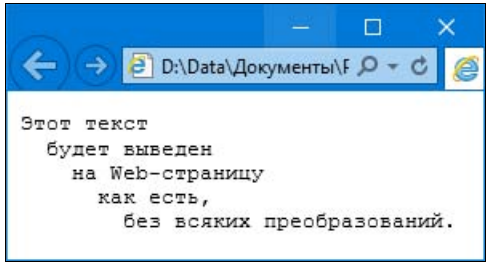



**Рис. 2.2.** Текст фиксированного формата

- если строка текста фиксированного формата не помещается в окне Web-обозревателя по ширине, она ни в коем случае не будет переноситься. Из-за этого может возникнуть потребность в горизонтальной прокрутке Web-страницы;
- допускается оформлять фрагмент текста фиксированного формата и создавать в нем гиперссылки, используя соответствующие теги (о которых будет рассказано далее в этой главе и в *главе 5*).

## Блочные контейнеры

Иногда бывает необходимо объединить несколько блочных элементов: абзацев, заголовков, списков — в один элемент. Это может потребоваться при создании разметки страницы, интерактивного элемента или просто чтобы привязать к этому элементу какой-либо стиль (о стилях речь пойдет в *разделе 2*).

Для таких случаев HTML предусматривает парный тег `<div>`. Внутри него помещается HTML-код, создающий содержимое, которое должно быть объединено в сущность, называемую *блочным контейнером*, или *блоком*:

```
<div>
  <p>Я вхожу в состав блока.</p>
  <p>Я тоже вхожу в состав блока.</p>
</div>
```

Web-обозреватели при выводе никак не выделяют блочные контейнеры. Однако мы можем, как было сказано ранее, привязать к ним стиль, который задаст для них нужное нам оформление.

Блоки очень пригодятся нам в дальнейшем, когда мы начнем работать со стилями, создавать разметку и интерактивные элементы страниц.

## Семантическая разметка текста

Описанных ранее простейших инструментов для структурирования текста, предлагаемых HTML, хватает во многих случаях. Но если страница содержит большой текст, разбитый на множество частей, их может оказаться недостаточно. Поэтому HTML 5 предоставляет в наше распоряжение набор новых тегов, выполняющих так называемую *семантическую разметку* текста.

Семантическая разметка заключается в разбиении текста на отдельные значащие блоки. Такими блоками могут быть сама статья, ее «шапка» или «поддон», отдельная часть статьи, примечание, иллюстрация в виде рисунка с подписью, набор гиперссылок для навигации и др.

Давайте рассмотрим все теги, используемые для семантической разметки, с указанием области их применения. Все они являются парными.

- `<article>` — независимый и самодостаточный фрагмент основного содержания страницы: статья, отдельный ее фрагмент, отдельная запись на форуме, в блоге или отдельный комментарий.