ГЛАВА 1

Введение в динамическое содержимое веб-страницы

Всемирная паутина — это непрерывно развивающаяся сеть, ушедшая далеко вперед от своей концепции начала 1990-х, когда ее создание было обусловлено решением конкретных задач. Высокотехнологичные эксперименты в ЦЕРНе (Европейском центре физики высоких энергий, известном в наши дни в качестве обладателя Большого адронного коллайдера) выдавали невероятно большой объем данных, который был слишком велик для распространения среди участвующих в экспериментах ученых, разбросанных по всему миру.

К тому времени интернет уже существовал и к нему было подключено несколько сотен тысяч компьютеров, поэтому Тим Бернерс-Ли (специалист ЦЕРНа) придумал способ навигации между ними с использованием среды гиперссылок — так называемого протокола передачи гиперссылок (Hyper Text Transfer Protocol — HTTP). Он также создал специальный язык разметки, названный языком гипертекстовой разметки (Hyper Text Markup Language — HTML). Для того чтобы собрать все это воедино, он создал первый браузер и веб-сервер.

Появление Web 1.0

Название Web 1.0 появилось, только когда был придуман термин Web 2.0. В эпоху Web 1.0 большинство пользователей были потребителями контента. Конечно, существовали отдельные персональные веб-страницы, но социальных сетей в ту пору еще не было. Вместо разделов для комментариев использовались гостевые книги. Некоторые сайты уже использовали базы данных, но ресурсы сервера и пропускная способность были очень ограниченны. Навигация между страницами в Web 1.0 осуществлялась с помощью простых кнопок, компоновка содержимого — с помощью графики, и перечень доступных взаимодействий был весьма ограничен.

Теперь эти средства воспринимаются нами как данность, но в то время концепция их применения носила революционный характер. До этого основной объем соединений приходился на пользователей домашних модемов, дозванивавшихся

и подключавшихся к электронным доскам объявлений, которые базировались на отдельном компьютере и позволяли общаться и обмениваться данными только с другими пользователями данной службы. Следовательно, для эффективного электронного общения с коллегами и друзьями нужно было становиться участником многих электронных досок объявлений.

Но Бернерс-Ли изменил все это одним махом, и к середине 1990-х годов уже существовали три основных конкурирующих друг с другом графических браузера, пользовавшихся вниманием 5 млн посетителей. Однако вскоре стало очевидно, что кое-что было упущено. Конечно, текстовые и графические страницы, имеющие гиперссылки для перехода на другие страницы, были блестящей концепцией, но результаты не отражали текущего потенциала компьютеров и интернета по удовлетворению насущных потребностей пользователей в динамическом изменении контекста. Всемирная паутина оставляла весьма невыразительное впечатление даже при наличии прокрутки текста и анимированных GIF-картинок.

Корзины покупателей, поисковые машины и социальные сети внесли существенные коррективы в порядок использования Всемирной паутины. В этой главе будет дан краткий обзор различных компонентов, формирующих ее облик, и программного обеспечения, способствующего обогащению и оживлению наших впечатлений от ее использования.

Пришло время воспользоваться аббревиатурами. Прежде чем делать это, я старался дать им четкое объяснение. Но если сразу не удастся разобраться, какое именно понятие они замещают или что означают, переживать не стоит, поскольку все подробности прояснятся по мере чтения книги.

HTTP и HTML: основы, заложенные Бернерсом-Ли

HTTP представляет собой стандарт взаимодействия, регулирующий порядок направления запросов и получения ответов — процесса, происходящего между браузером, запущенным на компьютере конечного пользователя, и веб-сервером. Задача сервера состоит в том, чтобы принять запрос от клиента и попытаться дать на него содержательный ответ, обычно передавая ему запрошенную веб-страницу. Именно поэтому и используется термин «сервер» («обслуживающий»). Партнером, взаимодействующим с сервером, является клиент, поэтому данное понятие применяется как к браузеру, так и к компьютеру, на котором он работает.

Между клиентом и сервером может располагаться ряд других устройств, например маршрутизаторы, прокси-серверы, шлюзы и т. д. Они выполняют различные задачи по обеспечению безошибочного перемещения запросов и ответов между клиентом и сервером. Как правило, для отправки этой информации используется

интернет. Некоторые из этих промежуточных устройств могут также ускорить интернет путем локального сохранения страниц или информации в так называемом кэше, обслуживая затем данное содержимое для клиентов непосредственно из кэша, без извлечения его из сервера-источника.

Обычно веб-сервер может обрабатывать сразу несколько подключений, а при отсутствии связи с клиентом он находится в режиме ожидания входящего запроса. Когда он приходит, сервер отправляет ответ.

Процедура «запрос — ответ»

В наиболее общем виде процесс «запрос — ответ» состоит из просьбы браузера или другой платформы к веб-серверу отправить ему веб-страницу, а сервер отправляет страницу обратно. После этого браузер занимается отображением или рендерингом страницы (рис. 1.1).

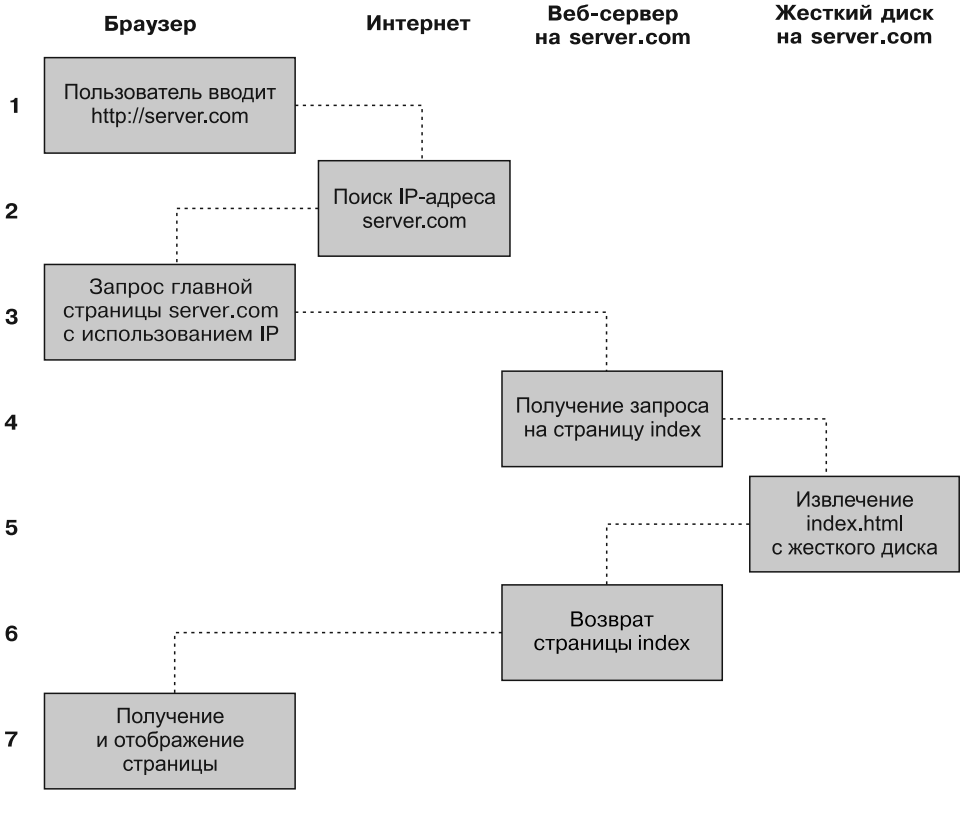

Рис. 1.1. Основная последовательность процесса «запрос — ответ» между клиентом и сервером

При этом соблюдается такая последовательность действий.

1. Вы вводите в адресную строку браузера `http://server.com`.
2. Ваш браузер ищет IP-адрес, соответствующий доменному имени `server.com`.
3. Браузер посылает запрос на отправку главной страницы `server.com`.
4. Запрос проходит по интернету и поступает на веб-сервер `server.com`.
5. Веб-сервер, получив запрос, ищет веб-страницу на своем жестком диске.
6. Сервер извлекает веб-страницу и отправляет ее по обратному маршруту браузеру.
7. Браузер отображает веб-страницу.

При передаче типовой веб-страницы этот процесс также осуществляется для каждого имеющегося на ней объекта: элемента графики, встроенного видеоролика или шаблона CSS.

Обратите внимание на то, что на шаге 2 браузер ищет IP-адрес, принадлежащий доменному имени `server.com`. У каждой машины, подключенной к интернету, включая и ваш компьютер, есть свой IP-адрес. Но, как правило, доступ к веб-серверам осуществляется по именам, таким как `google.com`. Браузер обращается к вспомогательной интернет-службе, так называемой системе доменных имен (Domain Name System — DNS), чтобы найти связанный с сервером IP-адрес, а затем воспользоваться им для связи с компьютером.

При передаче динамических веб-страниц процедура состоит из большего количества действий, поскольку к ней могут привлекаться как PHP, так и MySQL. Например, можно щелкнуть кнопкой мыши на картинке с изображением плаща. После этого PHP составит запрос, используя стандартный язык базы данных, SQL — множество используемых для этого команд будет рассмотрено в данной книге, — и отправит запрос в адрес MySQL-сервера. Этот сервер вернет информацию о выбранном вами плаще, и PHP-код заключит ее в некий код HTML, который будет отправлен сервером в адрес вашего браузера (рис. 1.2).

Выполняется такая последовательность действий.

1. Вы вводите в адресную строку браузера `http://server.com`.
2. Браузер ищет IP-адрес, соответствующий доменному имени `server.com`.
3. Браузер посылает запрос на отправку главной страницы server.com.
4. Запрос проходит по интернету и поступает на веб-сервер `server.com`.
5. Веб-сервер, получив запрос, ищет веб-страницу на своем жестком диске.
6. Теперь, когда главная страница размещена в его памяти, веб-сервер замечает, что она представлена файлом, включающим в себя PHP-сценарии, и передает страницу интерпретатору PHP.

7. Интерпретатор PHP выполняет PHP-код.
8. Кое-какие фрагменты кода PHP содержат SQL-инструкции, которые интерпретатор PHP, в свою очередь, передает процессору базы данных MySQL.
9. База данных MySQL возвращает результаты выполнения инструкций интерпретатору PHP.
10. Интерпретатор PHP возвращает веб-серверу результаты выполнения кода PHP, а также данные, полученные из базы данных MySQL.
11. Веб-сервер возвращает страницу выдавшему запрос клиенту, который отображает эту страницу на экране.

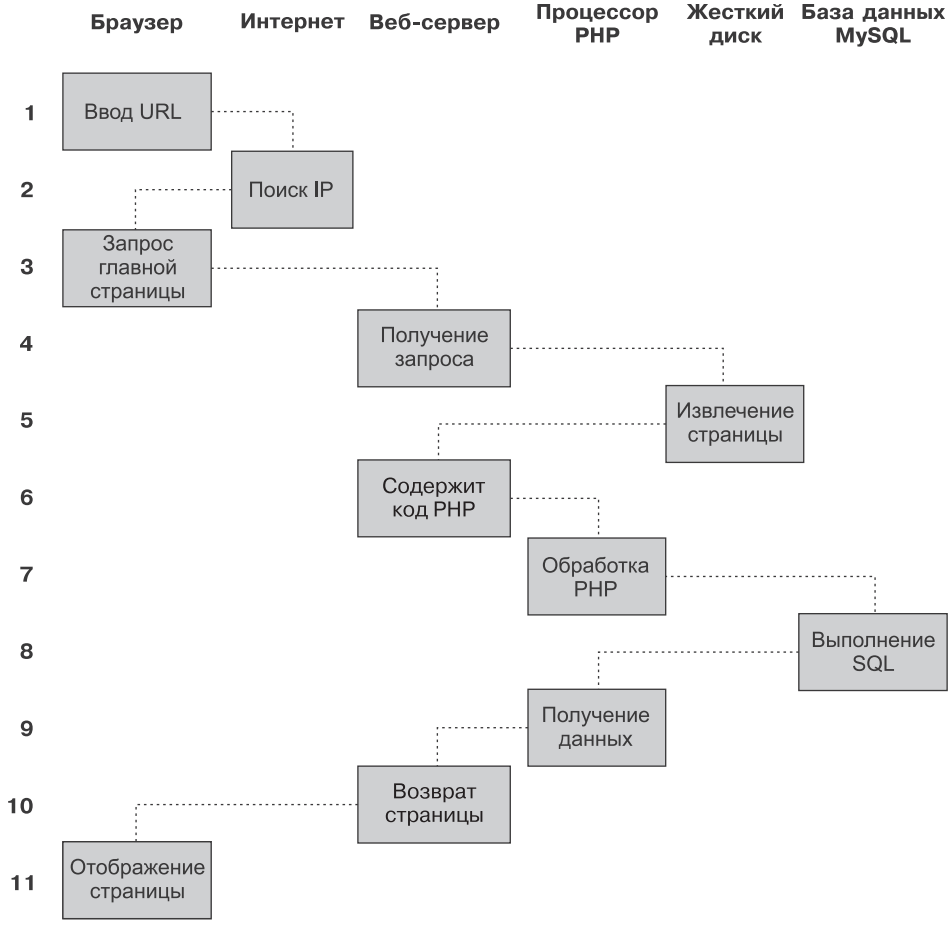

Рис. 1.2. Динамическая последовательность процесса «запрос — ответ», выполняемого клиентом и сервером

Конечно, полезно знать этапы этого процесса и как работают вместе три элемента, но на практике эти подробности не понадобятся, поскольку все происходит автоматически.

В каждом из примеров возвращенные браузеру HTML-страницы могут содержать также код JavaScript, интерпретируемый локально на машине клиента, который, в свою очередь, может инициировать еще один запрос.

Преимущества использования PHP, MySQL, JavaScript, CSS и HTML

В начале этой главы был представлен мир технологии Web 1.0, но рывок к созданию технологии Web 1.1, вместе с которой были разработаны такие браузерные расширения, как Java, JavaScript, Flash и ActiveX, не заставил себя долго ждать. На серверной стороне прогресс был обеспечен за счет общего шлюзового интерфейса (Common Gateway Interface, CGI), использования таких языков сценариев, как Perl (альтернатива языку PHP), и выполнения сценариев на стороне сервера — динамической вставки содержимого одного файла (или выходных данных выполняемой локальной программы) в другой файл.

Когда ситуация окончательно прояснилась, на передовых позициях остались три основные технологии. Несмотря на то что язык сценариев Perl силами своих стойких приверженцев сохранил популярность, простота PHP и допустимость использования в нем встроенных ссылок на программу базы данных MySQL обеспечили этому языку более чем двойное превосходство по количеству пользователей. А JavaScript, ставший важнейшей составной частью уравнения, используемого для динамического манипулирования HTML, в настоящее время берет на себя наиболее трудоемкие задачи осуществления асинхронного обмена данными (осуществляемого между клиентом и сервером после загрузки веб-страницы). Используя асинхронный обмен данными, веб-страницы обрабатывают данные и отправляют запросы веб-серверу в фоновом режиме, не оповещая пользователя о происходящем.

Несомненно, своеобразный симбиоз PHP и MySQL способствует их продвижению, но что привлекает к ним разработчиков в первую очередь? Ответ прост: та легкость, с какой эти технологии можно использовать для быстрого создания на сайтах динамических элементов. MySQL является быстродействующей и мощной, но при этом простой в использовании системой баз данных, предлагающей сайту практически все необходимое для поиска и обработки данных, которые предназначены для браузеров.

И когда вы так же соедините вместе JavaScript и CSS, у вас появится рецепт для создания высокодинамичных и интерактивных сайтов, особенно в современных условиях, когда доступно множество сложных функциональных фреймворков JavaScript, способных ускорить веб-разработку. К ним относится широко известный jQuery, который до недавнего времени был для программистов одним из наиболее распространенных средств доступа к функциям асинхронного обмена данными.

Кроме того, быстро набирает популярность более свежая JavaScript-библиотека React. В настоящее время это один из наиболее широко загружаемых и используемых фреймворков. Он распространен настолько, что на момент написания этих строк на сайте с предложениями работы Indeed числится намного больше вакансий для разработчиков React, чем для разработчиков jQuery.

React предлагает самые современные функциональные возможности для реализации сложных взаимодействий пользовательского интерфейса с сервером в реальном времени с помощью сценариев JavaScript на веб-страницах. Она позволяет создавать компоненты, являющиеся строительными блоками приложений React.

Компонент React (https://oreil.ly/iyLLS) может быть чем угодно в веб-приложении: кнопкой, текстом, меткой, таблицей и даже чем-то более сложным, например виджетом регистрации в системе или всплывающим модальным окном с кнопками управления. React также имеет возможность подготовки на стороне сервера информации, отображаемой компонентами, поддерживая такие инструменты, как Next.js (https://nextjs.org). React даже можно использовать в существующих приложениях (библиотека предусматривает такую возможность). Вы можете попробовать изменить небольшую часть существующего приложения, задействовав React, и если это изменение даст положительный результат, то вы сможете начать переводить на React.js все свое приложение. Однако для такой итеративной реализации можно использовать и другие фреймворки, такие как Vue.js.

MariaDB: клон MySQL

После того Oracle (корпорация, занимающаяся разработкой систем управления базами данных) приобрела Sun Microsystems (владельца MySQL), возникли опасения, что она закроет исходный код MySQL. Поэтому от этой СУБД отпочковалась MariaDB, дабы код оставался открытым в соответствии с положениями лицензии GNU GPL, которая гарантирует пользователям свободу запускать, изучать, изменять программное обеспечение и делиться им. Разработка MariaDB шла под руководством ряда первоначальных создателей MySQL, и эта СУБД

сохранила максимальную совместимость с MySQL. Поэтому вероятность встречи на некоторых серверах MariaDB вместо MySQL весьма высока, что не вызывает никаких опасений, поскольку все продемонстрированное в данной книге одинаково успешно работает и с MySQL, и с MariaDB. Для любых целей одна СУБД может заменяться другой, и вы при этом не заметите никакой разницы.

Впрочем, вышло так, что многие возникшие поначалу опасения были напрасны, поскольку код MySQL остался открытым, а Oracle просто сделала платными приобретение и поддержку тех выпусков, которые предоставляют дополнительные функциональные возможности, включающие георепликацию и автоматическое масштабирование. Тем не менее, в отличие от MariaDB, MySQL больше не поддерживается сообществом, но осознание того, что MariaDB никогда не лишится этой поддержки, позволит многим разработчикам спать спокойно и, вероятно, даст гарантии того, что код самой MySQL останется открытым.

PHP

Использование PHP существенно упрощает встраивание средств, придающих веб-страницам динамические свойства. Когда страницам присваивается расширение `.php`, у них появляется прямой доступ к языку сценариев. Разработчику нужно лишь написать код, похожий на этот:

```
<?php
  echo " Сегодня " . date('T') . ". ";
?>

Последние новости.
```

Открывающий тег `<?php` дает веб-серверу разрешение на интерпретацию всего последующего кода вплоть до тега `?>`. Все, что находится за пределами этой конструкции, отправляется клиенту в виде простого HTML. Поэтому текст `Последние новости` просто выводится в браузер. А внутри PHP-тегов встроенная функция `date()` отображает текущий день недели, соответствующий системному времени сервера.

В итоге на выходе из этих двух частей получается примерно следующее:

```
Сегодня Wednesday. Последние новости.
```

PHP — довольно гибкий язык, и некоторые разработчики предпочитают помещать PHP-конструкцию непосредственно рядом с кодом PHP, как в этом примере:

```
Сегодня <?php echo date("l"); ?>. Последние новости.
```

Существуют также другие способы форматирования и вывода информации, которые будут рассмотрены в главах, посвященных PHP. Важно усвоить, что, используя PHP, веб-разработчики получают язык сценариев, который хотя и не обладает быстротой скомпилированного кода на C или ему подобных языках, но все же работает невероятно быстро и к тому же очень хорошо вписывается в разметку HTML.

Если вы собираетесь набирать встречающиеся в этой книге примеры на PHP в программе-редакторе, чтобы работать параллельно с моим повествованием, не забывайте предварять их тегом `<?php`, а в конце ставить тег `?>`, чтобы обеспечить их обработку интерпретатором PHP. Для упрощения этой задачи можно заранее подготовить файл `example.php`, содержащий эти теги.

Используя PHP, вы получаете средство управления своим веб-сервером с неограниченными возможностями. Если понадобится на лету внести изменения в HTML, обработать данные кредитной карты, добавить сведения о пользователе в базу данных или извлечь информацию из стороннего сайта, все это можно будет сделать из тех же самых PHP-файлов, в которых находится и сам код HTML.

MySQL

Разумеется, без средств отслеживания информации, предоставляемой пользователем в ходе работы с вашим сайтом, нельзя в полной мере говорить о возможностях динамического изменения выходного кода HTML. На заре создания Всемирной паутины многие сайты использовали неструктурированные текстовые файлы для хранения таких данных, как имена пользователей и пароли. Но такой подход мог вызвать ряд проблем, если файл не был надежно заблокирован от повреждений, возникающих при одновременном доступе к нему множества пользователей. К тому же неструктурированный файл мог разрастаться до таких размеров, что с ним непросто было работать, не говоря уже о трудностях, связанных с попытками объединения файлов и осуществления в них сложных поисковых операций за какое-нибудь мало-мальски приемлемое время.

Именно в таких случаях большое значение приобретает использование реляционных баз данных со структурированной системой запросов. И MySQL, будучи совершенно бесплатной и установленной на огромном количестве веб-серверов системой, оказывается как нельзя кстати. Она представляет собой надежную и исключительно быстродействующую систему управления базами данных, использующую команды, похожие на простые английские слова.

Высшим уровнем структуры MySQL является база данных, внутри которой можно иметь одну или несколько таблиц, содержащих ваши данные. Это похоже, скажем, на файл электронной таблицы Excel, состоящей из нескольких листов: файл электронной таблицы можно рассматривать как базу данных, а отдельные листы — как таблицы.

Предположим, вы работаете с таблицей с именем `users` (пользователи), внутри которой созданы графы для фамилий — `surname`, имен — `firstname` и адресов электронной почты — `email`, и теперь нужно добавить еще одного пользователя. Вот одна из команд, с помощью которой это можно сделать:

```
INSERT INTO users VALUES('Smith', 'John', 'jsmith@mysite.com');
```

Для создания базы данных и таблицы, а также настройки всех нужных полей понадобится сначала выдать и другие команды, но используемая здесь SQL-команда `INSERT` демонстрирует простоту добавления в базу данных новой информации.

Так же просто выполняется и поиск данных. Предположим, что имеется адрес электронной почты пользователя и нужно найти имя его владельца. Для этого можно ввести следующий SQL-запрос:

```
SELECT surname,firstname FROM users WHERE email='jsmith@mysite.com';
```

После этого MySQL вернет `Smith, John` и любые другие пары имен, которые могут быть связаны в базе данных с указанным адресом электронной почты.

Нетрудно предположить, что возможности MySQL простираются значительно дальше выполнения простых команд вставки и выбора — `INSERT` и `SELECT`. Например, можно скомбинировать родственные наборы данных, чтобы собрать вместе взаимосвязанные части информации, запросить результаты, выбрав порядок их выдачи из множества вариантов, найти частичные совпадения, если известна только часть искомой строки, вернуть конкретно заданное количество результатов и сделать многое другое.

При использовании PHP все эти вызовы можно направлять непосредственно к MySQL без использования ее интерфейса командной строки напрямую. То есть, чтобы докопаться до нужного элемента данных, вы можете сохранять результаты в массивах для их обработки и выполнять множество поисковых запросов, каждый из которых зависит от результатов, полученных ранее.

Далее будет показано, что для придания еще большей мощности прямо в MySQL встроено несколько дополнительных функций, которые позволяют эффективно выполнять наиболее часто встречающиеся в MySQL операции, не составляя их из нескольких PHP-вызовов к MySQL.

JavaScript

JavaScript был создан для получения доступа из сценариев ко всем элементам HTML-документа. Иными словами, он предоставляет средства для динамического взаимодействия с пользователем, например для проверки приемлемости адресов электронной почты в формах ввода данных, отображения подсказок наподобие «Вы действительно подразумевали именно это?» и т. д. (хотя с точки зрения безопасности, которая всегда должна реализовываться на веб-сервере, на эту технологию положиться нельзя).

В сочетании с CSS JavaScript закладывает основу мощности динамических веб-страниц, которые изменяются буквально на глазах, в отличие от новой страницы, возвращаемой сервером.

Однако ранее использование JavaScript вызывало сложности, обусловленные некоторыми существенными различиями в способах реализации этого языка, выбранных разными разработчиками браузеров. В основном эти различия возникали, когда некоторые производители пытались придать своим браузерам дополнительные функциональные возможности, не обращая внимания на совместимость с продуктами своих конкурентов.

К счастью, разработчики в большинстве своем уже взялись за ум, и теперь оптимизация вашего кода для различных браузеров утратила прежнюю актуальность.

А сейчас взглянем, как можно воспользоваться обычным JavaScript-кодом, воспринимаемым всеми браузерами:

```
<script>
  document.write("Сегодня " + Date() );
</script>
```

Этот фрагмент кода предписывает браузеру интерпретировать все, что находится внутри тегов `<script>`, как код JavaScript, что браузер и сделает, записав в текущий документ текст `"Сегодня"`, а также дату, полученную за счет использования JavaScript-функции `Date()`. В результате получится нечто подобное следующему:

```
Сегодня wed Jan 01 2025 01:23:45
```

Ходьба перед бегом

Функция `document.write` намеренно используется здесь так, как изначально предназначалось, ради простоты в очень маленьких фрагментах кода. Однако существуют другие, более удачные способы вывода информации на веб-страницы и сообщений для целей отладки. Все они будут рассматриваться далее в книге и сопровождаться объяснениями, когда и почему те или иные варианты лучше.

Ранее было упомянуто, что изначально JavaScript разрабатывался, чтобы получить возможность динамически управлять различными элементами внутри HTML-документа, и это его предназначение по-прежнему является основным. Но все чаще JavaScript применяется для *Ajax* — процесса доступа к веб-серверу в фоновом режиме.

Асинхронный обмен данными позволил веб-страницам стать похожими на автономные программы, поскольку для отображения нового содержимого его не нужно загружать целиком. Вместо этого с помощью асинхронного вызова можно извлекать и обновлять отдельно взятые элементы веб-страницы, например, изменить вашу фотографию на сайте социальной сети или заменить кнопку, на которой нужно щелкнуть, отвечая на вопрос. Полностью эта тема будет рассмотрена в главе 17.

CSS

CSS является ключевым дополнением к HTML, которое обеспечивает соответствующую разметку HTML-текста и встроенных изображений, сообразуясь с параметрами применяемого пользователем экрана. После появления третьего стандарта (CSS3) CSS предлагает уровень динамической интерактивности, которая прежде поддерживалась только с помощью JavaScript. Например, можно не только придать стиль любому элементу HTML, чтобы изменить его размеры, цвета, границы, интервалы, но и, используя всего лишь несколько строк CSS, добавить в свои веб-страницы анимированные переходы и преобразования.

Кстати, стандартная нумерация выпусков CSS (например, CSS2 или CSS3) теперь отменена, поэтому сегодня каскадные таблицы стилей называются просто CSS, но различные подмодули имеют свою нумерацию, например CSS Selectors Level 4 и CSS Images Level 3.

Чтобы использовать CSS, достаточно просто вставить правила между тегами `<style>` и `</style>` в заголовке веб-страницы:

```
<style>
  p{
    text-align:justify;
    font-family:Helvetica;
  }
</style>
```

Эти правила будут изменять исходное выравнивание текста в теге `<p>`, чтобы содержащиеся в нем абзацы были выровнены по левому и правому краям и для их отображения использовался шрифт Helvetica.