WEB-программирование

WEB-программирование

Web-программирование – раздел программирования, ориентированный на разработку web-приложений

WEB-приложение

Веб-приложение – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

- Логика веб-приложения распределена между сервером и клиентом.
- Хранение данных осуществляется, преимущественно, на сервере.
- Обмен информацией происходит по сети

Преимущество такого подхода является факт, что клиенты не зависят от ОС пользователя.

Однако различная реализация HTML, CSS, DOM и других спецификаций в браузере может вызвать проблемы при разработке веб-приложения и последующей поддержке.

Так чем на самом деле занимаются веб-разработчики?

Короткий ответ: они создают и поддерживают веб-сайты.

Во многом зависит от типа выполняемой разработчиком работы. Хорошая новость заключается в том, что веб-разработчики высоко востребованы и, хотя они работают очень усердно, они могут рассчитывать на комфортную оплату труда и здоровое соотношение между работой и жизнью. Это профессия, вознаграждающая людей, которые решают реальные проблемы и получают удовольствие от созидания.

Выносим ключевые моменты

- Разработчики часто работают на клиентов, которые хотят представить свой продукт или услугу в сети.
- Работа обычно сильно сфокусирована на проекте и включает в себя сотрудничество с командой людей, которые помогают сопоставлять требования клиента с конечным продуктом.
- Не все разработчики работают на внешних клиентов... "Клиент" может быть компанией, где вы работаете, организацией, государственным учреждением, которому нужен веб-сайт или веб-приложение.
- Это довольно весело, вы реально создаете вещи, которые используются людьми, и вы можете играть со множеством новых игрушек.

Важные различия

"Фронтенд" обычно означает те вещи, которые вы непосредственно видите на сайте в браузере.

"Бэкенд", как правило, обозначает ту часть приложения, которая живет на сервере.

Google может быть довольно простым поисковиком с нашей точки зрения, но они нанимают армию инженеров, чтобы шестеренка, которую вы не видите, работала правильно.

Фуллстак (full stack)

"Фуллстак"-разработчики работают одновременно с обеими сторонами. Хотя каждый из "енд"-ов включает в себя изучение большого количества информации, фуллстак разработчики могут с комфортом "общаться" одновременно с базой данных и с браузером. В наши дни довольно популярно искать разработчиков с большим опытом работы с обеими частями веб-приложения

На стороне сервера (back-end)

Название	Лицензия	Веб-сервер
ASP.NET	проприетарная	специализированный
Java	свободная	множество, в том числе свободных
PHP	свободная	практически любой
Python	свободная	практически любой
Ryby	свободная	практически любой
NodeJS	свободная	практически любой

Веб-разработчик против веб-дизайнера

Итак, если у нас есть бэкенд-разработчики и фронтенд-разработчики, то кто же делает веб-сайт действительно красивым? Кто объединяет изображения, логотипы и цветовые схемы? Это работа веб-дизайнера.

Работа и карьера веб-разработчика

Работа в стабильной технической компании. Компании-гиганты, такие как Google, Facebook, Twitter и прочие подобные

Маленький технологический стартап

Небольшие технологические стартапы ценят способность писать работоспособный код превыше всего, поэтому иногда они могут оказаться жестким местом для начала работы с начальным набором навыков

Фрилансеры

Фрилансеры сами назначают стоимость часа своей работы и вольны уделять время на работу над собственными проектами.

Рабочие инструменты

Компьютер

Браузер

Текстовый редактор: Sublime Text, Atom, Visual Studio или Notepad++

IDE: PHPStorm, WebStorm

Google

Stack Overflow

Git - система контроля версий

Github - это место, где будут храниться копии файлов с вашим кодом.

Основные этапы

- веб-дизайн
- вёрстка страниц
- программирование на стороне клиента и сервера
- тестирование
- публикация приложения

Фронтенд

Фронтенд технологии "живут" в браузере. После того, как вы вбиваете URL в адресную строку и нажимаете Enter, ваш браузер получает HTML-файл с веб-сервера. Этот файл также предложит браузеру запросить у сервера дополнительные CSS и Javascript файлы.

Технологии разработки web-приложений

Основы HTML и CSS

HTML является языком разметки, который размечает, как располагается все содержимое веб-страницы, которое вы видите в браузере. Весь текст на этой странице сейчас находится внутри **HTML-тегов**, которые говорят вашему браузеру, как он (текст) должен быть расположен.

CSS говорит вашему браузеру, как конкретно отображать любой из тегов. **Например**, сделать фон тега голубым и сдвинуть его немного влево. В ваших инструментах разработчика вы можете видеть css-стили на другой панели, обычно показывающей конкретные свойства, унаследованные от каждой строки CSS-кода.

На стороне клиента (front-end)

- Для реализации User interface(UI): HTML, CSS
- Для создания интерактивных страниц: JavaScript
- Для выполнения асинхронных запросов: Ajax
- Для манипуляции с DOM: jQuery
- Для реализации SPA: js-frameworks(Angular, React и т. п.)

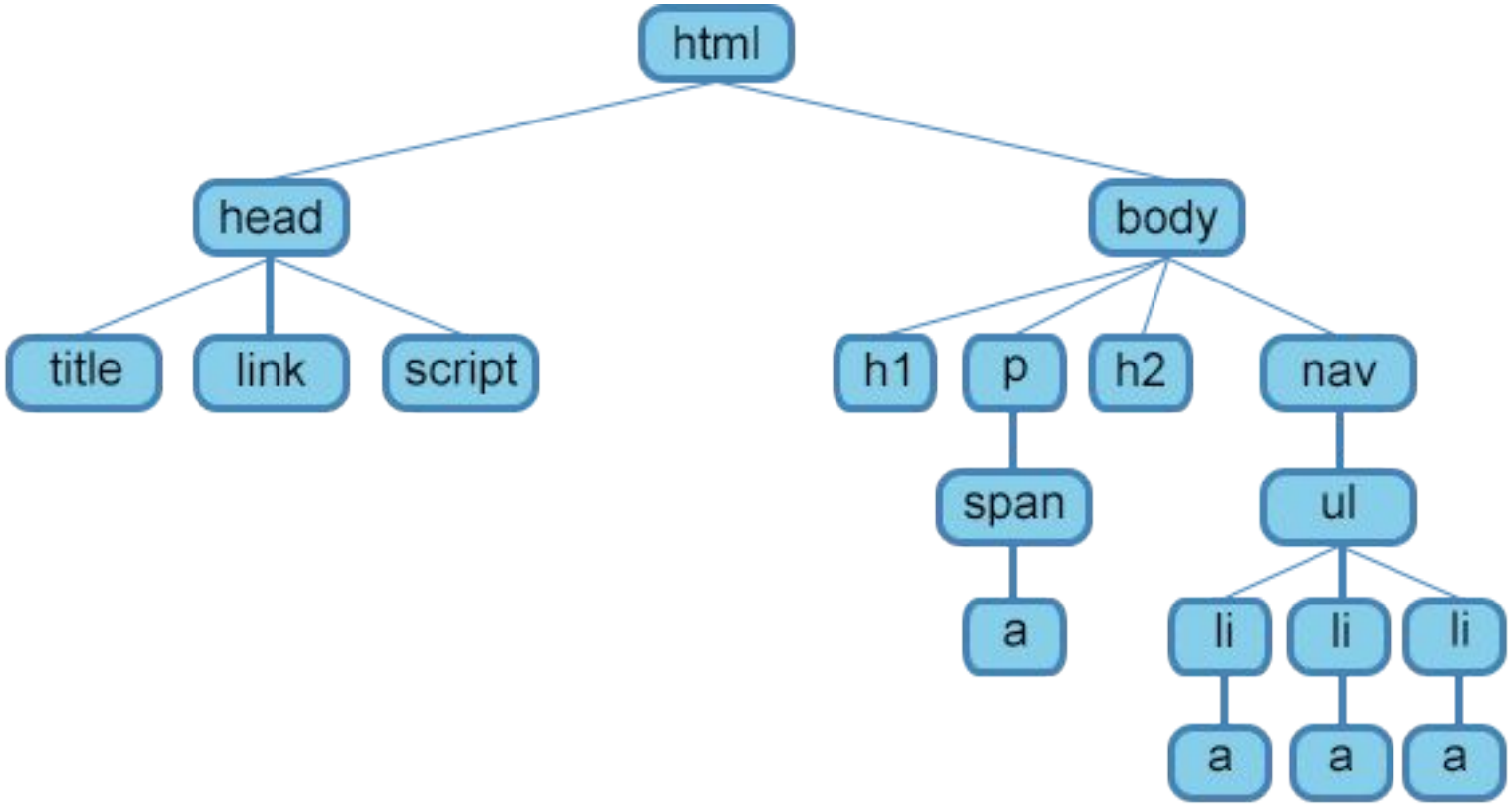
Структура HTML-документа

```
<!DOCTYPE html>
<!-- Объявление формата документа -->
<html>
  <head>
    <!-- Техническая информация о документе -->
    <meta charset="UTF-8">
    <!-- Определяем кодировку символов документа -->
    <title>...</title>
    <!-- Задаем заголовок документа -->
    <link rel="stylesheet" type="text/css" href="style.css">
    <!-- Подключаем внешнюю таблицу стилей -->
    <script src="script.js"></script>
    <!-- Подключаем сценарии -->
</head>
<body>
    <!-- Основная часть документа -->
</body>
</html>
```

HTML-теги

- **HTML-тег** — основа языка HTML. Теги используются для разграничения начала и конца элементов в разметке.
- Каждый HTML-документ состоит из **дерева HTML-элементов** и текста.
- Каждый HTML-элемент обозначается начальным (открывающим) и конечным (закрывающим) тегом.
- Открывающий и закрывающий теги содержат имя тега.

Объектная модель документа. DOM (document object model).



Структура веб-страницы

Предок — элемент, который заключает в себе другие элементы. Предком для всех элементов является <html>. В то же время элемент <body> является предком для всех содержащихся в нем тегов: <h1>, <p>, , <nav> и т.д.

Потомок — элемент, расположенный внутри одного или более типов элементов. Например, <body> является потомком <html>, а элемент <p> является потомком одновременно для <body> и <html>.

Родительский элемент — элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их. <html> является родительским только для <head> и <body>. Тег <p> является родительским только для .

Дочерний элемент — элемент, непосредственно подчиненный другому элементу более высокого уровня. Элементы <h1>, <h2>, <p> и <nav> являются дочерними по отношению к <body>.

Сестринский элемент — элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня. <head> и <body> — элементы одного уровня, так же как и элементы <h1>, <h2> и <p> являются между

Теги HTML

- html
- head
- title
- meta
- link
- script
- body

HTML-атрибуты

- **HTML-атрибуты** сообщают браузеру, каким образом должен отображаться тот или иной элемент страницы. Атрибуты позволяют сделать более разнообразными внешний вид информации, добавляемой с помощью **одинаковых** тегов.

- Значение атрибута заключается в кавычки "". Названия и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.

Основные атрибуты

- **class** - определяет имя класса для элемента (используется для определения класса в таблице стилей).
- **id** - определяет уникальный идентификатор элемента.
- hidden - указывает на то, что элемент должен быть скрыт. Значения: true/false
- style - указывает на код CSS, применяемую для оформления элемента. Значения: код CSS
- title- Определяет дополнительную информацию об элементе, задавая всплывающую подсказку для страницы. Значения: текст

```
<div id="wrapper">
  <h1 class="title">Title</h1>
  <p hidden="true">Скрыты текст</p>
  <p style="font-size: 24px;">Размер шрифта</p>
</div>
```

HTML-текст

- **HTML- текст** представлен в спецификации тегами для форматирования и группировки текста. Теги представляют собой контейнеры для текста и не имеют визуального отображения.

- Теги для форматирования текста несут смысловую нагрузку и обычно задают для текста, заключенного внутрь, стилевое оформление, например, выделяют текст жирным начертанием или отображают его шрифтом другого семейства (свойство font-family).

- Грамотно отформатированный текст дает понять поисковым системам, какие слова несут важную смысловую нагрузку, по каким из них предпочтительно ранжировать веб-страницу в поисковой выдаче. Вся текстовая информация, отображаемая на сайте, размещается внутри тега <body>.

Теги **ДЛЯ** HTML текста

- Теги заголовков: <h1...h6>
- Теги для форматирования текста: <b>, , <i>, <small>, , <sub>, <sup>, <ins>, <del>, <mark>
- Теги для ввода «компьютерного» текста: <code>, <kbd>, <samp>, <var>, <pre>
- Теги для оформления цитат и определений: <abbr>, <bdo>, <blockquote>, <q>, <cite>, <dfn>
- Абзацы, средства переноса текста: <p>,
, <hr/>

HTML-ССЫЛКИ

- **HTML-ссылки** создаются с помощью элементов <a> и <link>. Ссылки

  представляют собой связь между двумя ресурсами, одним из которых является текущий документ.

- Ссылки можно поделить на две категории:

  ссылки на внешние ресурсы — создаются с помощью тега <link> и используются для расширения возможностей текущего документа при обработке браузером;
  гиперссылки — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Структура ссылки

- Гиперссылки создаются с помощью парного тега <a>. Внутрь тега помещается текст, который будет отображаться на веб-странице.
- Обязательным параметром тега <a> является атрибут href, который задает URI-адрес веб-страницы.

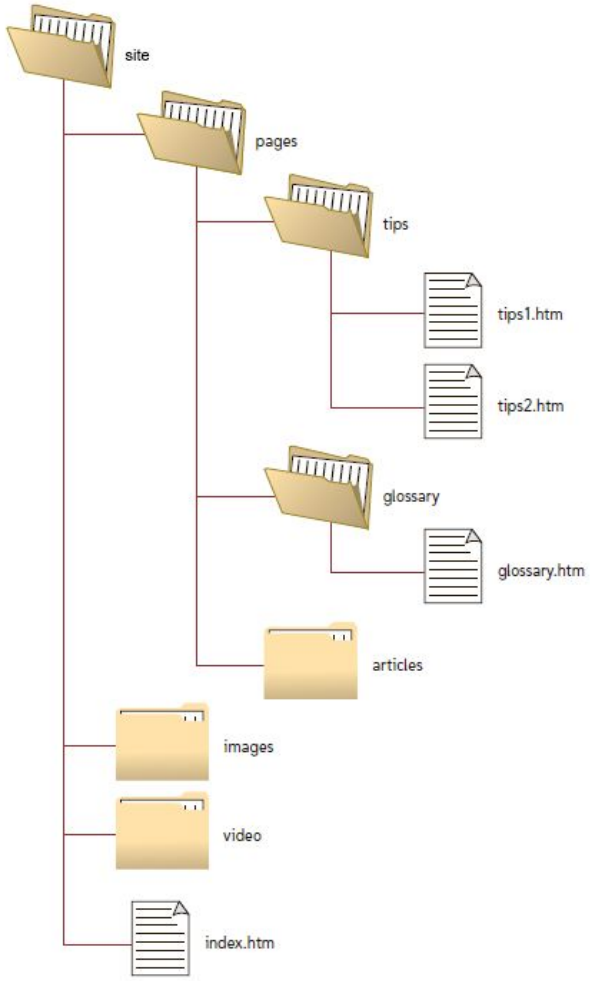
```
<a href="http://site.ru">указатель ссылки</a>
```

```
<протокол>://хост[:порт]/путь/имя ресурса[?get-параметры-запроса][#якорь]
```

Абсолютный и относительный путь

Чтобы создать ссылку на файл, находящийся вне папки, содержащей текущий документ, необходимо указать расположение файла или путь. HTML поддерживает два вида пути: абсолютный и относительный.

Пример структуры папок



Абсолютный путь

Абсолютный путь указывает *точное* местоположение файла в пределах всей структуры папок на компьютере (сервере). Абсолютный путь к файлу даёт доступ к файлу со сторонних ресурсов и содержит следующие компоненты:
1) протокол, например, http (опционально);
2) домен (доменное имя или IP-адрес компьютера);
3) папка (имя папки, указывающей путь к файлу);
4) файл (имя файла).

```
http://site.ru/pages/tips/tips1.html
//site.ru/pages/tips/tips1.html
```

Относительный путь

Относительный путь описывает путь к указанному документу относительно текущего.

Путь определяется с учётом местоположения веб-страницы, на которой находится ссылка.

Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте.

Относительный путь содержит следующие компоненты:

- папка (имя папки, указывающей путь к файлу);
- файл (имя файла).

Путь для относительных ссылок имеет три специальных обозначения:

1. / - указывает на корневую директорию и говорит о том, что нужно начать путь от корневого каталога документов и идти вниз до следующей папки
2. ./ - указывает на текущую папку
3. ../ -подняться на одну папку (директорию) выше

Якоря

Якоря, или внутренние ссылки, создают переходы на различные разделы текущей веб-страницы, позволяя быстро перемещаться между разделами. Внутренние ссылки также создаются при помощи тега <a> с разницей в том, что атрибут href содержит имя указателя — так называемый **якорь**, а не URI-адрес. Перед именем указателя всегда ставится знак #.

Примеры использования

```
<h1>Времена года</h1>
<h2>Оглавление</h2>
<a href="#p1">Лето</a>
<!--создаём якорь, указав #id элемента-->
<a href="#p2">Осень</a>
<a href="#p3">Зима</a>
<a href="#p4">Весна</a>
<p id="p1">...</p>
<!--добавляем соответствующий id элементу-->
<p id="p2">...</p>
<p id="p3">...</p>
<p id="p4">...</p>
```

Ссылка на телефонный номер, скайп или адрес электронной почты

```
ссылка на телефонный номер
<a href="tel:+74951234567">+7 (495) 123-45-67</a>
ссылка на адрес электронной почты
<a href="mailto:example@mail.ru">example@mail.ru</a>
ссылка на скайп (позвонить)
<a href="skype:имя-пользователя?call">Skype</a>
ссылка на скайп (открыть чат)
<a href="skype:имя-пользователя?chat">Skype</a>
ссылка на скайп (добавить в список контактов)
<a href="skype:имя-пользователя?add">Skype</a>
ссылка на скайп (отправить файл)
<a href="skype:имя-пользователя?sendfile">Skype</a>
```

Атрибуты ссылок

Атрибут	Описание, принимаемое значение
download	Дополняет атрибут href и сообщает браузеру, что ресурс должен быть загружен в момент, когда пользователь щелкает по ссылке, вместо того, чтобы, например, предварительно открыть его (как PDF-файл). Задавая имя для атрибута, мы таким образом задаем имя загружаемому объекту. Разрешается использовать атрибут без указания его значения: <img src="/images/logo.png" alt="logo">
href	Значением атрибута является URL-адрес документа, на который указывается ссылка.
rel	Дополняет атрибут href информацией об отношении между текущим и связанным документом.
target	Указывает на то, в каком окне должен открываться документ, к которому ведет ссылка. Принимает следующие значения: _self — страница загружается в текущее окно; _blank — страница открывается в новом окне браузера; _parent — страница загружается во фрейм-родитель; _top — страница загружается в полное окно браузера.
type	Указывает MIME-тип файлов ссылки, т.е. расширение файла. На данный момент носит больше справочную информацию.

HTML-изображения

HTML-изображения добавляются на веб-страницы с помощью тега <img>.
Использование графики делает веб-страницы визуально привлекательнее.
Изображения помогают лучше передать суть и содержание веб-документа.

Тег <img>

Элемент <img> представляет изображение и его резервный контент, который добавляется с помощью атрибута **alt**. Так как элемент <img> является строчным, то рекомендуется располагать его внутри блочного элемента, например, <p> или .

- Тег <img> имеет обязательный атрибут **src**, значением которого является абсолютный или относительный путь к изображению:

```
<img src="image.png" alt="Пример кода">
```

Атрибут	Описание, принимаемое значение
alt	Атрибут alt добавляет альтернативный текст для изображения. Выводится на месте появления изображения до его загрузки или при отключенной графике, а также выводится всплывающей подсказкой при наведении курсора мыши на изображение. Синтаксис: alt="описание изображения".
height	Атрибут height задает высоту изображения в px. Синтаксис: height="300".
src	Атрибут src задает путь к изображению. Синтаксис: src="flower.jpg".
width	Атрибут width задает ширину изображения в px.

HTML-таблицы

HTML-таблицы упорядочивают и выводят на экран данные с помощью строк или столбцов. Таблицы состоят из ячеек, образующихся при пересечении строк и столбцов. **Ячейки таблиц** могут содержать любые HTML-элементы, такие как заголовки, списки, текст, изображения, элементы форм, а также другие таблицы. Каждой таблице можно добавить связанный с ней заголовок, расположив его перед таблицей или после неё.

Таблицы больше не используются для вёрстки веб-страниц и компоновки отдельных элементов, потому что такой приём не обеспечивает гибкость структуры и адаптивность сайта, существенно увеличивая HTML-разметку.

Для всех элементов таблицы доступны глобальные атрибуты, а также собственные атрибуты.

Как создать таблицу

Таблица создаётся при помощи парного тега <table></table>. Данный тег является контейнером для элементов таблицы и все элементы должны находиться внутри него. Например, с помощью данной разметки можно создать таблицу, состоящую из **двух столбцов и двух строк**:

```
<table>
  <tr>
    <th>текст заголовка</th>
    <th>текст заголовка</th>
  </tr>
  <!--ряд с ячейками заголовков-->
  <tr>
    <td>данные</td>
    <td>данные</td>
  </tr>
  <!--ряд с ячейками тела таблицы-->
</table>
```

Группировка разделов таблицы

Элемент <thead> создает группу заголовков для строк таблицы с целью задания единого оформления. Используется в сочетании с элементами <tbody> и <tfoot> для указания каждой части таблицы.

Элемент должен быть использован в следующем порядке: как дочерний элемент <table>, после <caption> и <colgroup>, и перед <tbody>, <tfoot> и <tr> элементами. В пределах одной таблицы можно использовать один раз.

Элемент *<tbody>* группирует основное содержимое таблицы. Используется в сочетании с элементами <thead> и <tfoot>.

Элемент <tfoot> создает группу строк для представления информации о суммах или итогах, расположенную в нижней части таблицы. Используется в таблице один раз. Располагается после тега <thead>, перед тегами <tbody> и <tr>.

```
<table>
  <thead>
    <tr>
      <th>№ п/п</th>
      <th>Наименование товара</th>
      <th>Ед. изм.</th>
      <th>Количество</th>
      <th>Цена за ед. изм., руб.</th>
      <th>Стоимость, руб.</th>
   </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="5" style="text-align:right">ИТОГО:</td>
      <td>1168,80</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>1.</td>
      <td>Томаты свежие</td>
      <td>кг</td>
      <td>15,20</td>
      <td>69,00</td>
      <td>1048,80</td>
    </tr>
    <tr>
      <td>2.</td>
      <td>Огурцы свежие</td>
      <td>кг</td>
      <td>2,50</td>
      <td>48,00</td>
      <td>120,00</td>
   </tr>
  </tbody>
</table>
```

Как объединить ячейки таблицы

Атрибуты *colspan* и *rowspan* объединяют ячейки таблицы.
Атрибут *colspan* задает количество ячеек, объединенных по горизонтали, а *rowspan* - по вертикали.

```
<table>
    <tr>
        <th>№ п/п</th>
        <th>Наименование товара</th>
        <th>Ед. изм.</th>
        <th>Количество</th>
        <th>Цена за ед. изм., руб.</th>
        <th>Стоимость, руб.</th>
    </tr>
    <tr>
        <td>1.</td>
        <td>Томаты свежие</td>
        <td>кг</td>
        <td>15,20</td>
        <td>69,00</td>
        <td>1048,80</td>
    </tr>
    <tr>
        <td>2.</td>
        <td>Огурцы свежие</td>
        <td>кг</td>
        <td>2,50</td>
        <td>48,00</td>
        <td>120,00</td>
    </tr>
    <tr>
        <td colspan="5" style="text-align:right">ИТОГО:</td>
        <td>1168,80</td>

<!-- Задаем количество ячеек по горизонтали для объединения-->
    </tr>
</table>
```

№ п/п	Наименование товара	Ед. изм.	Количество	Цена за ед. изм., руб.	Стоимость, руб.
1.	Томаты свежие	кг	15,20	69,00	1048,80
2.	Огурцы свежие	кг	2,50	48,00	120,00
				ИТОГО:	1168,80

HTML-списки

HTML-списки используются для группировки связанных между собой фрагментов информации. Существует три вида списков:

- **маркированный список** - – каждый элемент списка <li> отмечается маркером,
- **нумерованный список** - - каждый элемент списка <li> отмечается цифрой,

  список определений - <dl> - состоит из пар термин <dt> - <dd> определение.

Каждый список представляет собой контейнер, внутри которого располагаются элементы списка или пары термин-определение. Элементы списка ведут себя как блочные элементы, располагаясь друг под другом и занимая всю ширину блока-контейнера. Каждый элемент списка имеет дополнительный блок, расположенный сбоку, который не участвует в компоновке.

Маркированный список

Маркированный список представляет собой неупорядоченный список *(от англ. Unordered List)*.

Создаётся с помощью парного тега *
*. В качестве маркера элемента списка выступает метка, например, закрашенный кружок.

Браузеры по умолчанию добавляют следующее форматирование блоку списка:

```
<ul>
 <li>Microsoft</li>
 <li>Google</li>
 <li>Apple</li>
 <li>IBM</li>
</ul>
```

- Microsoft
- Google
- Apple
- IBM

Нумерованный список

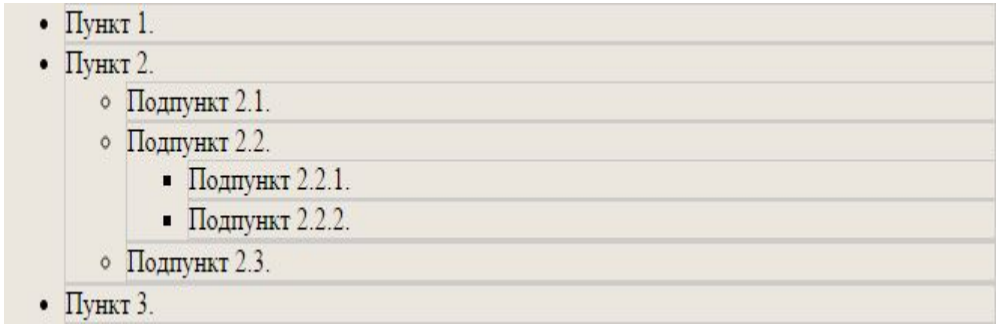
Нумерованный список создаётся с помощью парного тега . Каждый пункт списка также создаётся с помощью элемента <li>. Браузер нумерует элементы по порядку автоматически и если удалить один или несколько элементов такого списка, то остальные номера будут автоматически пересчитаны.

```
<ol>
 <li>Microsoft</li>
 <li>Google</li>
 <li>Apple</li>
 <li>IBM</li>
</ol>
```

1. Microsoft
2. Google
3. Apple
4. IBM

Вложенный список

```
<ul>
 <li>Пункт 1.</li>
 <li>Пункт 2.
  <ul>
   <li>Подпункт 2.1.</li>
   <li>Подпункт 2.2.
     <ul>
       <li>Подпункт 2.2.1.</li>
       <li>Подпункт 2.2.2.</li>
     </ul>
   </li>
   <li>Подпункт 2.3.</li>
  </ul>
 </li>
 <li>Пункт 3.</li>
</ul>
```



HTML-формы

HTML-формы являются элементами управления, которые применяются для сбора информации от посетителей веб-сайта.

Для получения и обработки данных форм используются серверные ЯП, такие как **PHP**, **Perl и др**.

Веб-формы состоят из набора текстовых полей, кнопок, списков и других элементов управления, которые активизируются щелчком мыши. Технически формы передают данные от пользователя удаленному серверу.

Элемент <form>

Основу любой формы составляет элемент *<form>...</form>*. Он не предусматривает ввод данных, так как является контейнером, удерживая вместе все элементы управления формы – **поля**. Атрибуты этого элемента содержат информацию, общую для всех полей формы, поэтому в одну форму нужно включать поля, объединенные логически.

Атрибуты <form>

- *accept-charset* - список кодировок символов
- *action* - **обязательный атрибут**, который указывает url обработчика формы на сервере, которому передаются данные.
- *enctype* - Используется для указания **MIME**-типа данных, отправляемых вместе с формой, например, enctype="multipart/form-data".
- *method* - задает способ передачи данных формы. Метод **get** передает данные на сервер через адресную строку браузера.
  Метод **post** применяется для пересылки данных больших объемов, а также конфиденциальной информации и паролей.
- name - задает **имя формы**, которое будет использоваться для доступа к элементам формы через сценарии, например, name="opros".

Создание полей формы

Элемент *<input>* создает большинство полей формы. Атрибуты элемента отличаются в зависимости от типа поля, для создания которого используется этот элемент.

Основные атрибуты <input>

- *name* – задает **имя поля.**
- *required* - выводит сообщение о том, что данное поле является обязательным для заполнения.
- *value* - определяет текст, отображаемый на кнопке, в поле или связанный текст. Не указывается для полей типа file.
- *width* - ширина поля.
- *height* - высота поля.
- *max/min* - позволяет ограничить допустимый ввод числовых данных максимальным и минимальным значением, значение атрибута может содержать целое или дробное число.

<input> атрибут type

- *button* - создает кнопку.
- *checkbox* - превращает поле ввода во флажок.
- *color* - генерирует палитры цветов в поддерживающих браузерах, давая пользователям возможность выбирать значения цветов в шестнадцатеричном формате.
- *date* - позволяет вводить дату в формате **дд.мм.гггг**.
- *datetime-local* - позволяет вводить дату и время, по шаблону **дд.мм.гггг чч:мм**.
- *email* - браузеры, будут ожидать, что пользователь введет данные, адрес электронной почты
- *file* - позволяет загружать файлы с компьютера пользователя.
- *hidden* - скрывает элемент управления, который не отображается браузером и не дает пользователю изменять значения по умолчанию.

<input> атрибут type

- *number* - предназначено для ввода целочисленных значений. Его атрибуты min, max и step задают верхний, нижний пределы и шаг между значениями соответственно.
- *radio* - создает переключатель - элемент управления в виде небольшого кружка, который можно включить или выключить.
- *range* - позволит создать такой элемент интерфейса, как ползунок, min / max.
- *reset* - создает кнопку, которая очищает поля формы от введенных пользователем данных.
- *submit* - создает стандартную кнопку, активизируемую щелчком мыши. Кнопка собирает информацию с формы и отправляет ее для обработки.
- *text* - создает текстовые поля в форме, выводя однострочное текстовое поле для ввода текста.
- *password* - создает текстовые поля в форме, при этом вводимые пользователем символы заменяются на звездочки.

Текстовые поля ввода

Элемент *<textarea>...</textarea>* используется вместо элемента *<input type="text">*, когда нужно создать большие текстовые поля. Текст, отображаемый как исходное значение, помещается внутрь тега. Размеры поля устанавливаются при помощи атрибутов *cols* - размеры по горизонтали, *rows* - размеры по вертикали. Высоту поля можно задать свойством *height*. Все размеры считаются исходя из размера одного символа моноширинного шрифта.

Раскрывающийся список

Списки дают возможность расположить большое количество пунктов компактно. Раскрывающиеся списки создаются при помощи элемента *<select>...</select>*. Они позволяют выбрать одно или несколько значений из предложенного множества. По умолчанию в поле списка отображается его первый элемент.

Для добавления в список пунктов используются элементы *<option>...</option>*, которые располагаются внутри *<select>*.

Надписи к полям формы

Надписи к элементам формы создаются с помощью элемента *<label>...</label>*. Существует два способа группировки надписи и поля. Если поле находится внутри элемента *<label>*, то атрибут *for* указывать не нужно.

HTML

```
<!-- с указанием атрибута for -->
<label for="comments">Когда вы последний раз летали на самолете?</label>
<textarea id="comments"></textarea>

<!-- без атрибута for -->
<p><label>Кошка<input id="cat" type="checkbox"></label></p>
```

Кнопки

Элемент *<button>...</button>* создает кликабельные кнопки. В отличие от кнопок, созданных *<input>*, внутрь элемента *<button>* можно поместить контент - текст или изображение.

Для корректного отображения элемента *<button>* разными браузерами нужно указывать атрибут *type*, например,

<button type="submit">Отправить</button>.

Флажки и переключатели в формах

Флажки в формах задаются с помощью конструкции *<input type="checkbox">*, а переключатель - при помощи *<input type="radio">*.

Флажков, в отличие от переключателей, в одной форме может быть установлено несколько. Если для флажков указан атрибут *checked*, то при загрузке станицы на соответствующих полях формы флажки уже будут установлены.

HTML5-аудио

HTML5-аудио предоставляет улучшенные возможности работы с аудио контентом. До недавнего времени единственным способом добавления звуковых файлов на веб-страницы было интегрирование фонового звука с помощью тега *<bgsound>*, который проигрывался во время просмотра пользователем страницы без возможности выключения.

Элемент <audio>

HTML5-элемент *<audio>* используется для внедрения звукового контента в веб-страницы. В общем виде HTML-разметка имеет следующий вид:

```HTML
<audio src="name.ogg" controls></audio>
```

Атрибут *controls* добавляет отображение браузерами интерфейса управления аудио плеера - кнопки воспроизведения, паузы, громкости.

Внешний вид аудио плеера в разных браузерах

В настоящий момент не существует аудио формата, который бы работал во всех браузерах, поэтому для обеспечения доступности контента максимально широкой аудитории рекомендуется включать несколько источников звука, представленных с использованием атрибута *src* элемента *<source>*. Одновременно можно добавить резервный контент для браузеров, которые не

HTML

```
<audio controls>
  <source src="name.ogg" type="audio/ogg">
  <source src="name.mp3" type="audio/mpeg">
  <a href="sounds/name.mp3">Скачать name.mp3</a>
</audio>
```

HTML5-видео

HTML5-видео — новый стандарт для размещения мультимедийных файлов в сети с оригинальным программным интерфейсом без привлечения подключаемых модулей. С помощью элемента *<video>* появилась возможность добавлять видеосодержимое на веб-страницы, а также стилизовать внешний вид видеоплеера при помощи css-стилей.

Внешний вид видеоплеера в основных браузерах

Firefox

Google, Opera

Internet Explorer

Элемент <video>

В простом варианте HTML-разметка для размещения видеофайла на странице имеет следующий вид:

```HTML
<video src="video.ogv" controls></video>
```

Атрибут *controls* отвечает за появление элементов управления видеоплеером. Вы можете добавить изображение с помощью атрибута *poster*, которое браузер будет использовать, пока загружается видео или пока пользователь не нажмет на кнопку воспроизведения, а также задать высоту и ширину видео.

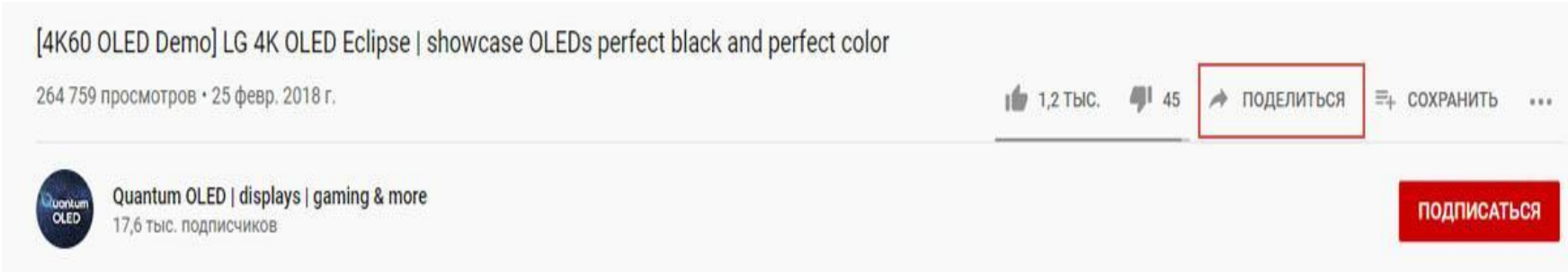
Как и в случае с аудиофайлами, рекомендуется перечислять в *<source>* все форматы, начиная с более предпочтительного. Также нужно указывать MIME-тип для каждого видеофайла.

HTML

```
<video controls width="400" height="300">
  <source src="video.mp4" type="video/mp4"><!-- MP4 для Safari, IE9, iPhone, iPad, Android, и Windows Phone 7 -->
  <source src="video.webm" type="video/webm"><!-- WebM/VP8 для Firefox4, Opera, и Chrome -->
  <source src="video.ogv" type="video/ogg"><!-- Ogg/Vorbis для старых версий браузеров Firefox и Opera -->
  <object data="video.swf" type="application/x-shockwave-flash"><!-- добавляем видеоконтент для устаревших браузеров, в которых нет поддержки элемента video -->
    <param name="movie" value="video.swf">
  </object>
</video>
```

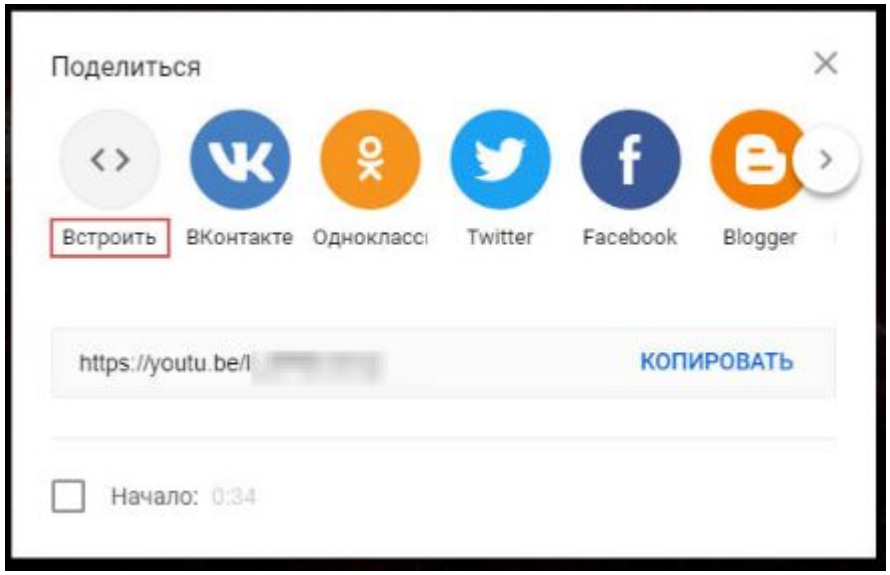
Видео с YouTube

Для того, чтобы добавить видео с YouTube на сайт, откройте страницу и найдите кнопку **Поделиться** под видеоплеером:



Видео с YouTube

Самой первой кнопкой в списке будет **Встроить**



Видео с YouTube

YouTube сгенерирует код для вставки автоматически. Тег *<iframe>* будет иметь URL исходного видео, высоту и ширину плеера и еще несколько атрибутов

```
<iframe width="330" height="200"
    src="https://www.youtube.com/embed/li_9PBrcOcQ"
    frameborder="0"
    allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
    allowfullscreen>
</iframe>
```

Доменное имя

Домен - это онлайн-адрес сайта, место его размещения в интернете. С технической позиции доменный адрес — запись в базе данных. Когда пользователь указывает в поисковой строке доменное имя, компьютер понимает, какой сайт необходимо показать и по какому адресу отправить запрос.

Домен состоит уникальный адрес IPv4:port. Расположены домены по иерархии справа налево: третьего уровня, второго уровня, первого уровня. Иерархический порядок позволяет браузеру оперативно определить искомый сервер и соответствующий запросу сайт. Благодаря иерархии функционирует DNS (система доменных имён).

Первый уровень

Домены первого уровня разделяют на два основных типа:

- Географические — национальные домены верхнего уровня (ccTLD). Для них используют лишь две буквы с учётом международного кода страны. Например, **.eu** — Евросоюз, **.ru** — Россия, **.ua** — Украина. Такие домены чаще всего обозначают гео-принадлежность сайта. Однако не существует запрета на применение несоответствующего расширения. Иногда для экономии выбирают недорогое доменное имя какой-нибудь небольшой страны.

- Тематические — общие домены верхнего уровня (gTLD). Теоретически, они обозначают тематику сайта. К примеру, **.com** — коммерция, .military — военный, **.gov** — правительственный. Но никто не проверяет, соответствует ли gTLD каким-либо критериям. Хотя существуют и домены ограниченного использования. Так расширение .int применяют исключительно для доменов международных организаций.

Второй уровень

Слева от последней точки — домен второго уровня, он же — основной (материнский).

site.com

Основной домен — уникальная комбинация из букв и цифр, придуманная как название сайта. Выбранный домен второго уровня сначала проверяют на уникальность и лишь потом регистрируют. Второй и все последующие уровни подведомственны регистратору доменов. Это организация, которая аккредитована ICANN. Она может регистрировать новые домены и продлевать уже существующие.

Третий уровень

Слева от основного домена (от предпоследней точки) расположен домен третьего уровня — поддомен или субдомен. Поддомены делят сайт на обособленные разделы. Например, чтобы открыть блог, люди могут ввести поддомен и сразу перейти в соответствующий раздел, минуя главную страницу.

blog.site.com

Поддомены делают структуру ресурса более понятной. С ними не нужно создавать дополнительные адреса для разделов на сайте. К тому же, поисковые системы по отдельности индексируют разделы — их можно по отдельности продвигать в выдаче.

DNS (Domain Name System)

Люди не любят запоминать наборы чисел: далеко не все из нас помнят больше двух телефонным номеров.

Представим, что мы написали веб-приложение, которое отображает среднюю температуру воздуха во всех странах в режиме реального времени. Мы развернули его на сервере с адресом 226.69.237.119 и на порту 8080.

Система доменных имен - “псевдоним” нашего адреса, например **blog.site.ru**

Для сопоставления доменных имен и реальных адресов существуют DNS-сервера. Когда пользователь в браузере вводит, например, **blog.site.ru**, запрос отправляется на DNS-сервер, где он превращается в реальный адрес.

Как работает DNS?

Принцип работы DNS похож на поиск и вызов контактов из телефонной книги смартфона. Ищем имя, нажимаем «позвонить», и телефон соединяет нас с нужным абонентом. Понятно, что смартфон в ходе звонка не использует само имя человека, вызов возможен только по номеру телефона. Если вы внесете имя без номера телефона, позвонить человеку не сможете.

Так и с сайтом. Каждому имени сайта соответствует набор цифр формата 000.000.000.000. Когда пользователь вводит в адресной строке браузера имя сайта, например google.com, компьютер запрашивает IP-адрес этого сайта на специальном DNS-сервере и после получения корректного ответа открывает сам сайт.

Что такое DNS-сервер?

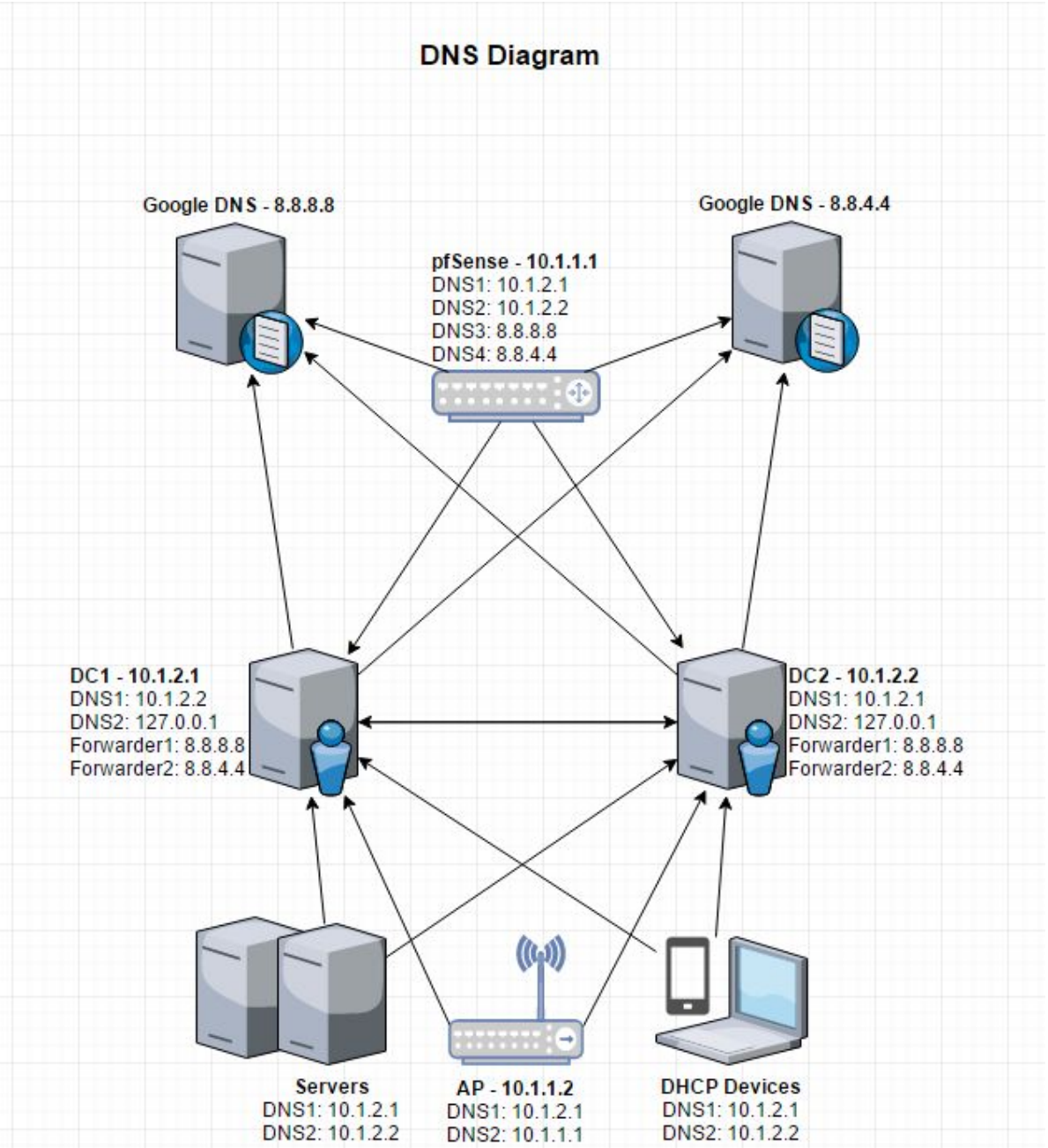
Это как раз и есть «книга контактов» интернета. DNS-сервер — это специализированный компьютер (или группа), который хранит IP-адреса сайтов. Последние, в свою очередь, привязаны к именам сайтов и обрабатывает запросы пользователя. В интернете много DNS-серверов, они есть у каждого провайдера и обслуживают их пользователей.

Зачем нужны DNS-серверы и какие они бывают?

Основное предназначение DNS-серверов — хранение информации о доменах и ее предоставление по запросу пользователей, а также кэширование DNS-записей других серверов. Это как раз «книга контактов».

Локальный DNS-сервер в большинстве случаев взаимодействует с другими DNS-серверами из региона, в котором находится запрошенный сайт. После нескольких обращений к таким серверам локальный DNS-сервер получает искомое и отправляет эти данные в браузер — запрошенный сайт открывается. Полученные данные сохраняются на локальном сервере, что значительно ускоряет его работу. Поскольку, единожды «узнав» IP-адрес сайта, запрошенного пользователем, локальный DNS сохраняет эту информацию. Процесс сохранения полученных ранее данных и называется кэшированием.

DNS Diagram


Google DNS - 8.8.8.8
Google DNS - 8.8.4.4
pfSense - 10.1.1.1
DNS1: 10.1.2.1
DNS2: 10.1.2.2
DNS3: 8.8.8.8
DNS4: 8.8.4.4
DC1 - 10.1.2.1
DNS1: 10.1.2.2
DNS2: 127.0.0.1
Forwarder1: 8.8.8.8
Forwarder2: 8.8.4.4
DC2 - 10.1.2.2
DNS1: 10.1.2.1
DNS2: 127.0.0.1
Forwarder1: 8.8.8.8
Forwarder2: 8.8.4.4
Servers
DNS1: 10.1.2.1
DNS2: 10.1.2.2
AP - 10.1.1.2
DNS1: 10.1.2.1
DNS2: 10.1.1.1
DHCP Devices
DNS1: 10.1.2.1
DNS2: 10.1.2.2

Сетевая модель OSI

- Физический уровень
- Канальный уровень
- Сетевой уровень
- Транспортный уровень
- Сеансовый уровень
- Представления уровень
- Прикладной уровень

Физический уровень

Определяет метод передачи (например, через оптоволокно) или без проводов (например, через Bluetooth или IRDA, Wi-Fi, GSM, 4G и так далее).

Канальный уровень

У канального уровня есть два подуровня — это MAC и LLC. MAC (Media Access Control, контроль доступа к среде) отвечает за присвоение физических MAC-адресов, а LLC (Logical Link Control, контроль логической связи) занимается проверкой и исправлением данных, управляет их передачей.

На втором уровне OSI работают коммутаторы, их задача - передать сформированные кадры от одного устройства к другому, используя в качестве адресов только физические MAC-адреса.

Сетевой уровень

Объединения участков сети и выбор оптимального пути (т.е. маршрутизация). Каждое сетевое устройство должно иметь уникальный сетевой адрес в сети. Многие слышали про протоколы IPv4 и IPv6. Эти протоколы работают на данном уровне.

Транспортный уровень

Этот уровень берет на себя функцию транспорта. К примеру, когда вы скачиваете файл с Интернета, файл в виде сегментов отправляется на Ваш компьютер. Также здесь вводятся понятия портов, которые нужны для указания назначения к конкретной службе. На этом уровне работают протоколы TCP (с установлением соединения) и UDP (без установления соединения).

Сеансовый уровень

Роль этого уровня в установлении, управлении и разрыве соединения между двумя хостами. К примеру, когда открываете страницу на веб-сервере, то Вы не единственный посетитель на нем. И вот для того, чтобы поддерживать сеансы со всеми пользователями, нужен сеансовый уровень.

Уровень представления

О задачах уровня представления вновь говорит его название. Шестой уровень занимается тем, что представляет данные в понятном человеку и машине виде. Например, когда одно устройство умеет отображать текст только в кодировке ASCII, а другое только в UTF-8, перевод текста из одной кодировки в другую происходит на шестом уровне.

Шестой уровень также занимается представлением картинок (в JPEG, GIF и т.д.), а также видео-аудио (в MPEG, QuickTime). Помимо перечисленного, шестой уровень занимается шифрованием данных, когда при передаче их необходимо защитить.

Прикладной уровень

Самый понятный для всех уровень. Как раз на этом уроне работают привычные для нас приложения — e-mail, браузеры по протоколу HTTP, FTP и остальное.

Он обеспечивает взаимодействие приложений с сетью. На этом уровне мы будем принимать, отправлять сообщения, делать запросы к сервисам и удаленным базам данных.

Существует множество протоколов, которые используются на этом уровне: POP3, FTP, SMTP, XMPP, RDP, SIP, TELNET и, конечно же, HTTP/HTTPS.

Пример

Когда мы переходим, в интернет-магазин, мы указываем его адрес нахождения и порт. При первом посещении создается сессия, в которую магазин может записывать информацию. Например, о товарах, которые мы оставили в корзине. Если мы закроем вкладку с интернет-магазином, а потом опять зайдем в него, наши товары останутся в корзине, потому что они сохранены в сессии. Всю информацию, которую мы получаем от магазина, мы получаем по протоколу HTTP/HTTPS, а наш браузер умеет его обрабатывать.

Принципы работы элементов

- IP-адрес — адрес абонента в сети;
- Порт — адрес приложения конкретного абонента;
- Сессия — сущность которая существует на протяжении всего общения двух абонентов;
- Прикладные протоколы (HTTP/HTTPS) — правила, которыми мы будем руководствоваться при составлении и отправке сообщений.

Протоколы HTTP/HTTPS

Протокол - это всеобщее соглашение, которого мы придерживаемся, составляя сообщения.

HTTP/HTTPS - протоколы передачи данных по сети на прикладном уровне модели OSI.

Структура HTTP

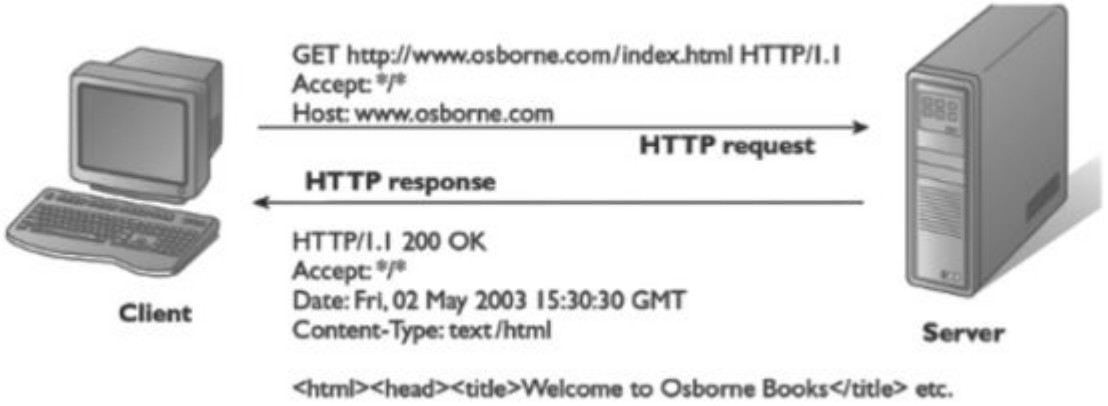
HTTP-протокол состоит только из текста. Нас больше всего интересует структура, в которой расположен этот текст.

Каждое сообщение состоит из трех частей:

- Стартовая строка (Starting line) — определяет служебные данные.
- Заголовки (Headers) — описание параметров сообщения.
- Тело сообщения (Body) — данные сообщения. Должны отделяться от заголовков пустой строкой.

По HTTP-протоколу можно отправить запрос на сервер (request) и получить ответ от сервера (response). Запросы и ответы немного отличаются параметрами.

Как выглядит простой HTTP-запрос



В стартовой строке указаны:

- GET - метод запроса;
- http://www.osborne.com/index.html - путь запроса (path);
- HTTP/1.1 - версия протокола передачи данных.

Затем следуют заголовки:

- Host - хост, которому адресован запрос;
- User-Agent - клиент, который отправляет запрос.

Методы HTTP-запросов

Всего их девять: GET, POST, PUT, OPTIONS, HEAD, PATCH, DELETE, TRACE, CONNECT.

Самые распространенные — GET и POST. Этих двух методов на первых порах будет достаточно.

GET

GET — запрашивает контент из сервера. Поэтому у запросов с методом GET нет тела сообщения. Но при необходимости можно отправить параметры через path в таком формате:

https://cdn.javarush.ru/images/article/155cea79-acfd-4968-9361-ad585e939b82/original.pngsend?name1=value1&name2=value2

Здесь:

- javarush.ru — хост,
- /send — путь запроса,
- ? — разделитель, обозначающий, что дальше следуют параметры запроса.

В конце перечисляются параметры в формате ключ=значение, разделенные амперсандом.

POST

POST — публикует информацию на сервере. POST-запрос может передавать разную информацию: параметры в формате ключ=значение, JSON, HTML-код или даже файлы. Вся информация передается в теле сообщения.

```
POST /user/create/json HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 28
Host: javarush.ru

{
  "Id": 12345,
   "User": "John"
}
```

Запрос отправляется по адресу javarush.ru/user/create/json, версия протокола — HTTP/1.1. Accept указывает, какой формат ответа клиент ожидает получить, Content-Type — в каком формате отправляется тело сообщения. Content-Length — количество символов в теле.

HTTP-запрос может содержать много разных заголовков. Подробнее с ними можно ознакомиться в спецификации протокола.

HTTP-ответы

После получения запроса, сервер его обрабатывает и отправляет ответ клиенту

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 98

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

Стартовая строка в респонсе содержит версию протокола (HTTP/1.1), Код статуса (200), Описание статуса (OK). В заголовках — тип и длина контента. В теле ответа — HTML-код, который браузер прорисует в HTML-страницу.

Response Status Codes

Response Status Codes всегда трехзначные, и первая цифра кода указывает категорию ответа:

- 1xx — информационный. Запрос получен, сервер готов к продолжению;
- 2xx — успешный. Запрос получен, понятен и обработан;
- 3xx — перенаправление. Следующие действия нужно выполнить для обработки запроса;
- 4xx — ошибка клиента. Запрос содержит ошибки или не отвечает протоколу;
- 5xx — ошибка сервера. Сервер не смог обработать запрос, хотя был составлен верно;

Вторая и третья цифры в коде детализируют ответ.

Например

- 200 OK — реквест получен и успешно обработан;
- 201 Created — реквест получен и успешно обработан, в результате чего создан новый ресурс или его экземпляр;
- 301 Moved Permanently — запрашиваемый ресурс был перемещен навсегда, и последующие запросы к нему должны происходить по новому адресу;
- 307 Temporary Redirect — ресурс перемещен временно. Пока к нему можно обращаться, используя автоматическую переадресацию;
- 403 Forbidden — запрос понятен, но нужна авторизация;
- 404 Not Found — сервер не нашел ресурс по этому адресу;
- 501 Not Implemented — сервер не поддерживает функциональность для ответа на этот запрос;
- 505 HTTP Version Not Supported — сервер не поддерживает указанную версию HTTP-протокола.

В чем отличие между HTTPS и HTTP

HTTPS синтаксически идентичен протоколу HTTP, то есть использует те же стартовые строки и заголовки. Единственные отличия — дополнительное шифрование и порт по умолчанию (443).

HTTPS шифруется между HTTP и TCP, то есть между прикладным и транспортным уровнями.

Современный стандарт шифрования — по протоколу TLS. В HTTPS шифруется вся информация, кроме хоста и порта, куда отправлен запрос.