Современные веб-технологии

История развития

- **1958 год.** В США при Министерстве обороны создано Агентство Передовых Исследовательских Проектов – ARPA
- **29.10.1969 года.** Между двумя первыми узлами сети ARPANET осуществлена связь
- **1976 год.** Создана Ethernet
- **1983 год.** ARPA перешла на TCP/IP
- **1984 год.** Разработана система доменных имен DNS
- **1991 год.** CERN создала протокол World Wide Web (WWW)

История развития

- **1993 год.** Создан первый общедоступный графический браузер Mosaic
- **1994 год.** Образовался консорциум W3C
- **2008 год.** Число пользователей, регулярно использующих Интернет, составило около 1,5 млрд человек
- **2010 год.** Прямой доступ в Интернет получил экипаж Международной космической станции

Протоколы TCP/IP в модели OSI

7	**Прикладной**	**HTTP, SMTP, SNMP, FTP, Telnet, scp, SMB,NFS, RTSP, BGP**
6	Представительский	XDR, ASN.1, AFP
5	Сеансовый	TLS, SSL, ISO 8327 / CCITT X.225, RPC, NetBIOS, ASP
4	Транспортный	TCP, UDP, RTP, SCTP, SPX, ATP, DCCP, GRE
3	Сетевой	IP, ICMP, IGMP, CLNP, OSPF, RIP, IPX, DDP
2	Канальный	Ethernet, Token ring, PPP, HDLC, X.25, Frame relay, ISDN, ATM, MPLS, Wi-Fi, ARP, RARP
1	Физический	электрические провода, радиосвязь, оптоволоконные провода

Физическая модель Интернета

OSI		*TCP/IP*
7 6	WWW, Gopher, WAIS, SNMP, FTP telnet, SMTP, TFTP	I
5 4	TCP, UDP	II
3	IP, ICMP, RIP, OSPF, ARP	III
2 1	Не регламентируется Ethernet, Token Ring, FDDI, X.25, SLIP, PPP	IV

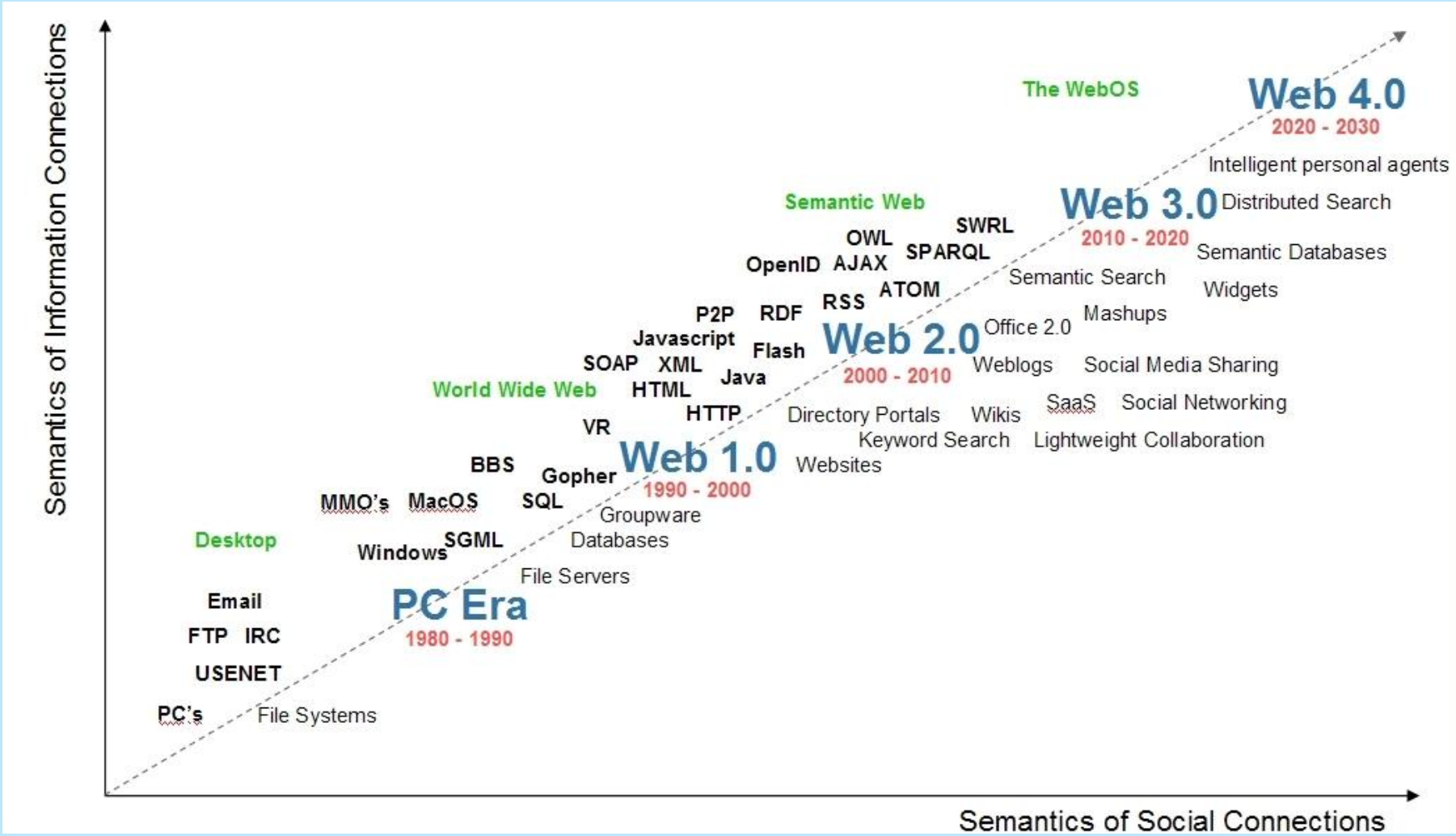
Логическая модель Интернета

- **Всемирная паутина** (World Wide Web, Веб) – распределенная система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к Интернету
- Основные составляющие:
  - язык гипертекстовой разметки документов HTML
  - универсальный способ адресации ресурсов в сети URL
  - протокол обмена гипертекстовой информацией HTTP

Концепции Веб

Концепция / Свойство	Web 1.0	Web 2.0	Web 3.0
Концепция	Веб только для чтения	Веб для бурного чтения-записи	Портативный индивидуальный Веб
Количество пользователей	45 миллионов пользователей (1996)	Больше 1 миллиарда пользователей (2006)	Еще больше
Ориентация	Ориентация на компании	Ориентация на сообщества	Ориентация на индивидуальностях
Структура данных	Домашние страницы	Блоги	Lifestreaming-функции
Концепция данных	Владение контентом	Обмен контентом	Объединение динамического контента
Управление знаниями	Britannica Online	Wikipedia	Интернет
Технологии	HTML, порталы	XML, RSS	Технологии «drag and drop» и mashups
Представление	Веб формы	Веб-приложения	Виджеты и гаджеты
Классификация	Директории	Тэги	Поведение пользователей
Поиск	Netscape	Google	iGoogle, NetVibes
Стоимость рекламы	Просмотр страниц	Цена за клик	Активность пользователей
Продвижение	реклама	«из уст в уста»	advertainment

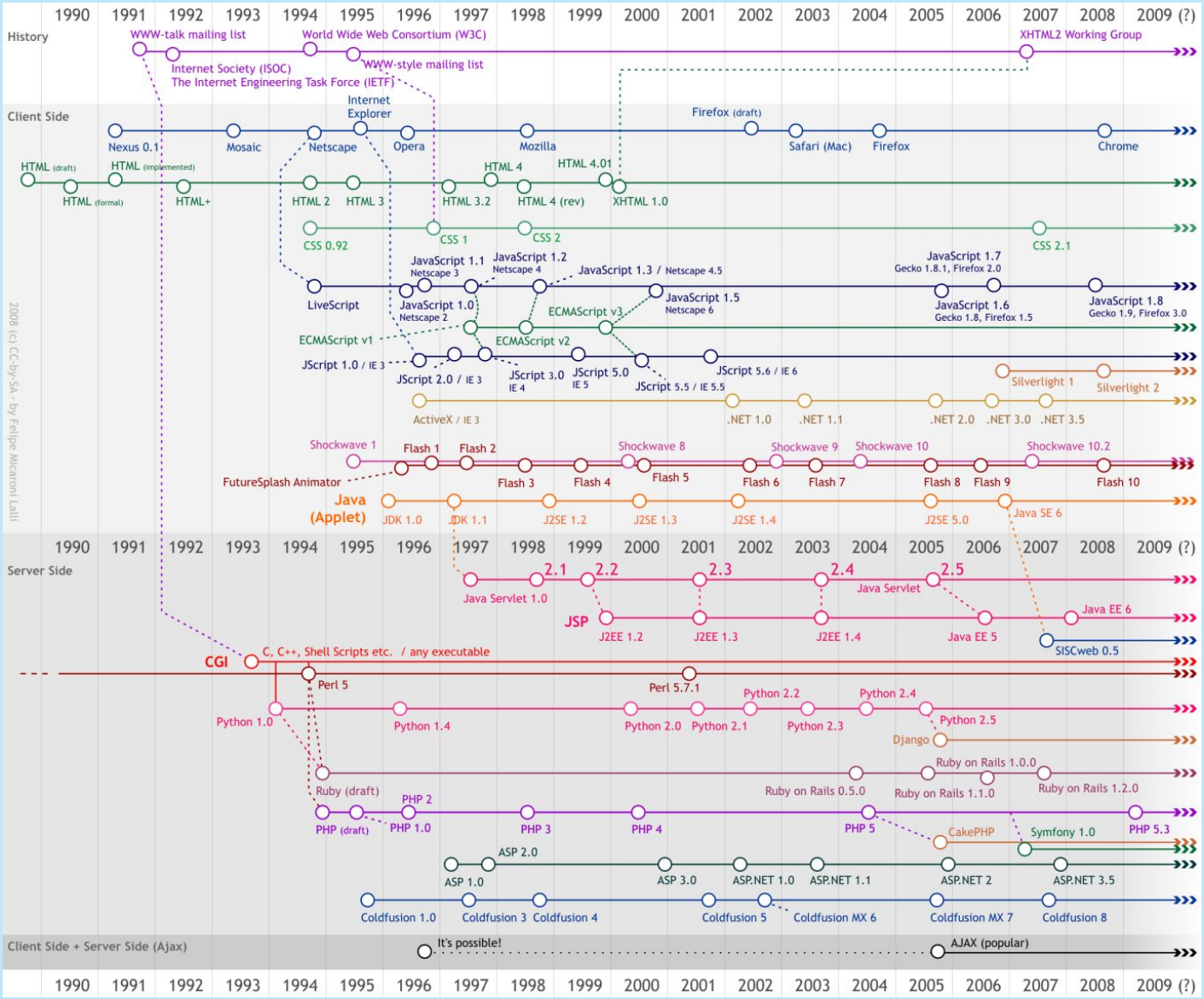
Технологическая карта концепций Веб



Веб-разработка

- **Веб-программирование** – раздел программирования, ориентированный на разработку динамических Internet-приложений
- Языки программирования
  - Клиентские
  - Серверные

Хронология развития веб-технологий



Веб-технологии

- HTML – стандартный язык разметки документов во Всемирной паутин
- XHTML – язык разметки веб-страниц, по возможностям сопоставимый с HTML, созданный на базе XML
- CSS – технология описания внешнего вида документа, написанного языком разметки
- XML – текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки

Веб-технологии

- JavaScript – это объектно-ориентированный скриптовый язык программирования
- PHP – скриптовый язык программирования общего назначения, интенсивно применяющийся для разработки веб-приложений
- Perl – высокоуровневый интерпретируемый динамический язык программирования общего назначения
- AJAX – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером

Веб-технологии

- Adobe Flash и Microsoft Silverlight – мультимедийные платформы, используемые для создания RIA-приложений, а также для интеграции видеороликов в веб-страницы.
- ASP.NET – технология создания веб-приложений и веб-сервисов от компании Microsoft

Клиент-серверные технологии Веб

- Основой протокола *HTTP* является взаимодействие «*клиент-сервер*», то есть предполагается, что:
  - потребитель-клиент инициировав соединение с поставщиком-сервером посылает ему запрос;
  - Поставщик-сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.
- *Тонкий клиент* — это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с *браузером*, использующийся для работы с *веб-приложениями*.
- *Толстый клиент*, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Протокол HTTP

- Всё программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:
  - Серверы - поставщики услуг хранения и обработки информации (обработка запросов).
  - Клиенты — конечные потребители услуг сервера (отправка запросов).
  - Прокси-серверы для поддержки работы транспортных служб.

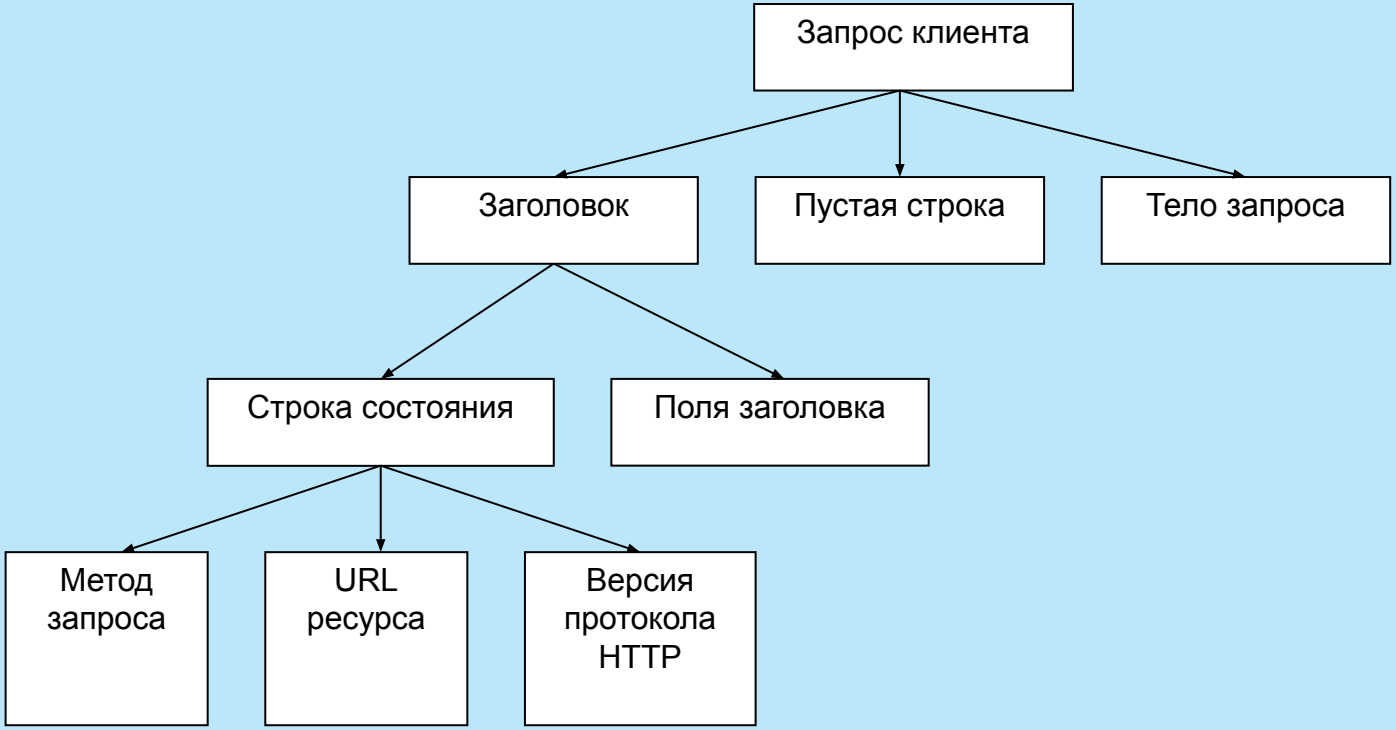
Схема HTTP-сеанса

1. Установление TCP-соединения.
2. Запрос клиента.
3. Ответ сервера.
4. Разрыв TCP-соединения.

Структура протокола HTTP

- Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:
  - *Заголовок сообщения*, который начинается со *строки состояния*, определяющей тип сообщения, и *полей заголовка*, характеризующих тело сообщения, описывающих параметры передачи и прочие сведения;
  - *Пустая строка*;
  - *Тело сообщения* — непосредственно данные сообщения.
- *Поля заголовка* и *тело* сообщения могут отсутствовать, но *строка состояния* является обязательным элементом, так как указывает на тип запроса/ответа.

Структура запроса клиента HTTP



Методы запроса клиента

Метод, указанный в строке состояния, определяет способ воздействия на ресурс, URL которого задан в той же строке.

- Метод может принимать значения *GET*, *POST*, *HEAD*, *PUT*, *DELETE* и др.
- Несмотря на обилие методов, для Web-программиста по-настоящему важны лишь два из них: *GET* и *POST*.

Методы запроса клиента

- GET. Согласно формальному определению, метод GET предназначается для получения ресурса с указанным URL.

Получив запрос GET, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс,

Несмотря на то что, по определению, метод GET предназначен для получения информации, он вполне подходит для передачи небольших фрагментов данных на сервер.

Методы запроса клиента

- POST. Согласно тому же формальному определению, основное назначение метода POST - передача данных на сервер.

Однако, подобно методу GET, метод POST может применяться по-разному и нередко используется для получения информации с сервера. Как и в случае с методом GET, URL, заданный в строке состояния, указывает на конкретный ресурс.

Поля заголовка запроса клиента

- Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос, т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

  Имя_поля: значение

- Назначение поля определяется его именем, которое отделяется от значения двоеточием.

Поля заголовка запроса клиента

Поля заголовка HTTP-запроса	Значение
Host	Доменное имя или *IP*-адрес узла, к которому обращается клиент
Referer	*URL* документа, который ссылается на ресурс, указанный в строке состояния
From	Адрес электронной почты пользователя, работающего с клиентом
Accept	*MIME*-типы данных, обрабатываемых клиентом. Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент
Accept-Language	Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом
Accept-Charset	Перечень поддерживаемых наборов символов
Content-Type	*MIME*-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Content-Length	Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Range	Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть
Connection	Используется для управления *TCP*-соединением. Если в поле содержится *Close*, это означает, что после обработки запроса сервер должен закрыть соединение. Значение *Keep-Alive* предлагает не закрывать *TCP*-соединение, чтобы оно могло быть использовано для последующих запросов
User-Agent	Информация о клиенте

Пример запроса

GET http://oak.oakland.edu/ HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.04 [en] (Win95; I)

Host: oak.oakland.edu

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*

Accept-Language: en

Accept-Charset: iso-8859-l,*,utf-8

Структура ответа сервера HTTP

- Ответ сервера также состоит из четырех перечисленных ниже компонентов.
  - Строка состояния.
  - Поля заголовка.
  - Пустая строка.
  - Тело ответа.

Структура ответа сервера HTTP

- Ответ сервера клиенту начинается со строки состояния, которая имеет следующий формат:

  Версия_протокола Код_ответа Пояснительное_сообщение

- **Версия_протокола** задается в том же формате, что и в запросе клиента, и имеет тот же смысл.
- **Код_ответа** - это трехзначное десятичное число, представляющее в закодированном виде результат обслуживания запроса сервером.
- **Пояснительное_сообщение** дублирует код ответа в символьном виде. Это строка символов, которая не обрабатывается клиентом. Она предназначена для системного администратора или оператора, занимающегося обслуживанием системы, и является расшифровкой кода ответа.

Тело ответа веб-сервера

- В используемых в настоящее время реализациях протокола HTTP первая цифра не может быть больше 5 и определяет следующие классы ответов.
  - 1 - специальный класс сообщений, называемых *информационными*. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса. При обмене данными между HTTP-клиентом и HTTP-сервером сообщения этого класса используются достаточно редко.
  - 2 - успешная обработка запроса клиента.
  - 3 - перенаправление запроса. Чтобы запрос был обслужен, необходимо пред-принять дополнительные действия.
  - 4 - ошибка клиента. Как правило, код ответа, начинающийся с цифры 4, возвра-щается в том случае, если в запросе клиента встретилась синтаксическая ошибка.
  - 5 - ошибка сервера. По тем или иным причинам сервер не в состоянии выполнить запрос.

Классы кодов ответа сервера

Код	Расшифровка	Интерпретация
100	Continue	Часть запроса принята, и сервер ожидает от клиента продолжения запроса
200	OK	Запрос успешно обработан, и в ответе клиента передаются данные, указанные в запросе
201	Created	В результате обработки запроса был создан новый ресурс
202	Accepted	Запрос принят сервером, но обработка его не окончена. Данный код ответа не гарантирует, что запрос будет обработан без ошибок.
206	Partial Content	Сервер возвращает часть ресурса в ответ на запрос, содержавший поле заголовка Range
301	Multiple Choice	Запрос указывает более чем на один ресурс. В теле ответа могут содержаться указания на то, как правильно идентифицировать запрашиваемый ресурс
302	Moved Permanently	Затребованный ресурс больше не располагается на сервере
302	Moved Temporarily	Затребованный ресурс временно изменил свой адрес
400	Bad Request	В запросе клиента обнаружена синтаксическая ошибка
403	Forbidden	Имеющийся на сервере ресурс недоступен для данного пользователя
404	Not Found	Ресурс, указанный клиентом, на сервере отсутствует
405	Method Not Allowed	Сервер не поддерживает метод, указанный в запросе
500	Internal Server Error	Один из компонентов сервера работает некорректно
501	Not Implemented	Функциональных возможностей сервера недостаточно, чтобы выполнить запрос клиента
503	Service Unavailable	Служба временно недоступна
505	HTTP Version not Supported	Версия HTTP, указанная в запросе, не поддерживается сервером

Поля заголовка ответа веб-сервера

Имя поля	Описание содержимого
Server	Имя и номер версии сервера
Age	Время в секундах, прошедшее с момента создания ресурса
Allow	Список методов, допустимых для данного ресурса
Content-Language	Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс
Content-Type	*MIME*-тип данных, содержащихся в теле ответа сервера
Content-Length	Число символов, содержащихся в теле ответа сервера
Last-Modified	Дата и время последнего изменения ресурса
Date	Дата и время, определяющие момент генерации ответа
Expires	Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей
Location	В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса
Cache-Control	Директивы управления кэшированием. Например, *no-cache* означает, что данные не должны кэшироваться

Тело ответа веб-сервера

- В *теле ответа* содержится код ресурса, передаваемого клиенту в ответ на запрос.
- Это не обязательно должен быть HTML-текст веб-страницы. В составе ответа могут передаваться *изображение*, *аудио-файл*, фрагмент *видеоинформации*, а также любой *другой тип данных*, поддерживаемых клиентом.
- О том, как следует обрабатывать полученный ресурс, клиенту сообщает содержимое поля заголовка *Content-type*.

Пример ответа веб-сервера

HTTP/1.1 200 OK
Date: Thu, Об Apr 2000 23:39:01 GMT
Server: Apache/1.3.11 (Unix)
Last-Modified: Fri, 03 Mar 2000 22:17:57 GMT
Content-Length: 4685
Connection: close
Content-Type: text/html

<HTML>
<head>
<meta name="GENERATOR" content="Mozilla/4.7 (Macintosh; I; PPC) [Netscape]">
<title>OAK Software Repository</title>
<link REV="made" HREF="mailto:archives@oakland.edu">
</head>
<body text="#000000" bgcolor="#D8CA87" link="#0000C0" vlink="#E00000" alink="#0000FF" background="/images/oak.jpg">
<CENTER><A HREF=http://www.oakland.edu><img SRC="/images/ousail.jpg" BORDER=0 height=87 width=72 align=CENTER></A>
</body>
</HTML>

Спецификация MIME

- Поле с именем Content-type может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается MIME-тип содержимого запроса или ответа.
- *MIME-тип* также передается в поле заголовка *Accept*, присутствующего в запросе.
- Спецификация *MIME* (*Multipurpose Internet Mail Extension*) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем.
- Однако применение MIME не исчерпывается электронной почтой. Средства MIME успешно используются в WWW и, по сути, стали неотъемлемой частью этой системы.

MIME типы данных

Тип/подтип	Расширение файла	Описание
application/pdf	.pdf	Документ, предназначенный для обработки Acrobat Reader
application/msexcel	.xls	Документ в формате Microsoft Excel
application/postscript	.ps, .eps	Документ в формате PostScript
application/x-tex	.tex	Документ в формате TeX
application/msword	.doc	Документ в формате Microsoft Word
application/rtf	.rtf	Документ в формате RTF, отображаемый с помощью Microsoft Word
image/gif	.gif	Изображение в формате GIF
image/ jpeg	.jpeg, .jpg,	Изображение в формате JPEG
image/tiff	.tiff, .tif	Изображение в формате TIFF
image/x-xbitmap	.xbm	Изображение в формате XBitmap
text/plain	.txt	ASCII-текст
text/html	. html , . htm	Документ в формате HTML
audio/midi	.midi, .mid	Аудиофайл в формате MIDI
audio/x-wav	.wav	Аудиофайл в формате WAV
message/rfc822		Почтовое сообщение
message/news		Сообщение в группы новостей
video /mpeg	.mpeg, .mpg, .mpe	Видеофрагмент в формате MPEG
video/avi	.avi	Видеофрагмент в формате AVI

URI, URL, URN

- *URI* (*Uniform Resource Identifier*) — единообразный идентификатор ресурса, представляющий собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс.
- Самые известные примеры *URI* — это *URL* и *URN*.
- *URL* (*Uniform Resource Locator*) - это *URI*, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.
- *URN* (*Uniform Resource Name*) — это *URI*, который идентифицирует ресурс в определённом пространстве имён, но, в отличие от *URL*, *URN* не указывает на местонахождение этого ресурса.
- *URI* не указывает на то, как получить ресурс, а только идентифицирует его. Что даёт возможность описывать с помощью *RDF* (*Resource Description Framework*) ресурсы, которые не могут быть получены через Интернет (имена, названия и т.п.)

Структура URL

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>

где:

- *схема* - схема обращения к ресурсу (обычно сетевой протокол);
- *логин* - имя пользователя, используемое для доступа к ресурсу;
- *пароль* - пароль, ассоциированный с указанным именем пользователя;
- *хост* - полностью прописанное доменное имя хоста в системе *DNS* или *IP-адрес* хоста;
- *порт* - порт хоста для подключения;
- *URL-путь* - уточняющая информация о месте нахождения ресурса.

Структура URL

- Общепринятые схемы (протоколы) URL включают протоколы: *ftp*, *http*, *https*, *telnet*, а также:
  - *gopher* — протокол *Gopher*;
  - *mailto* — адрес электронной почты;
  - *news* — новости *Usenet*;
  - *nntp* — новости *Usenet* через протокол *NNTP*;
  - *irc* — протокол *IRC*;
  - *prospero* — служба каталогов *Prospero Directory Service*;
  - *wais* — база данных системы *WAIS*;
  - *xmpp* — протокол *XMPP* (часть *Jabber*);
  - *file* — имя локального файла;
  - *data* — непосредственные данные (*Data*: *URL*);

Порт TCP/IP

- TCP/IP *порт* — целое число от 1 до 65535, позволяющие различным программам, выполняемым на одном хосте, получать данные независимо друг от друга. Каждая программа обрабатывает данные, поступающие на определённый порт («слушает» этот порт).
- Самые распространенные сетевые протоколы имеют стандартные номера портов, хотя в большинстве случаев программа может использовать любой порт.
- Для наиболее распространенных протоколов стандартные номера портов следующие:
  - HTTP: 80
  - FTP: 21 (для команд), 20 (для данных)
  - telnet: 23
  - POP3: 110
  - IMAP: 143
  - SMTP: 25
  - SSH: 22

HTTPS

- HTTPS — расширение протокола *HTTP*, поддерживающее шифрование. Данные, передаваемые по протоколу *HTTP*, «упаковываются» в криптографический протокол *SSL* или *TLS*, тем самым обеспечивается защита этих данных. В отличие от *HTTP*, для *HTTPS* по умолчанию используется TCP-порт 443.
- Чтобы подготовить веб-сервер для обработки *HTTPS* соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера.

SSL И TLS

- *SSL* (Secure Sockets Layer) — криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет.
- При его использовании создаётся защищённое соединение между клиентом и сервером. *SSL* изначально разработан компанией *Netscape Communications*. Впоследствии на основании протокола *SSL 3.0* был разработан и принят стандарт *RFC*, получивший название TLS.
- Протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и безопасных хэш-функций.

SSL И TLS

- На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (например POP3, IMAP, SMTP или HTTP).
- Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнет передавать и получать данные.
- Для доступа к веб-страницам, защищённым протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https — 443.

Методы аутентификации в WWW

- *Basic* — базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках *http-пакетов*. Пароль при этом не шифруется и присутствует в чистом виде в кодировке *base64*. Для данного типа аутентификации использование *SSL* является обязательным.
- *Digest* — дайджест-аутентификация, при которой пароль пользователя передается в хешированном виде. По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только *хеш* от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование *SSL* является обязательным.

Методы аутентификации в WWW

- *Integrated* — интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов *NTLM* или *Kerberos*. Этот тип аутентификации защищен от перехвата удостоверений пользователей, поэтому для него не требуется протокол *SSL*. Только при использовании данного типа аутентификации можно работать по схеме *http*, во всех остальных случаях необходимо использовать схему *https*.

Современные веб-технологии

HTML5

HTML5

Что именно привнес HTML5?

- HTML5 определяет новый алгоритм парсинга для создания структуры DOM.
- Добавление новых элементов и тегов, как например, элементы video, audio и ряд других.
- Переопределение правил и семантики уже существовавших элементов HTML.

HTML5

В итоге, как правило, HTML 5 применяется преимущественно в двух значениях:

- HTML 5 как обновленный язык разметки гипертекста, некоторое развитие предыдущей версии HTML 4.
- HTML 5 как мощная платформа для создания веб-приложений, которая включает не только непосредственно язык разметки гипертекста, обновленный HTML, но и язык программирования JavaScript и каскадные таблицы стилей CSS 3.

HTML5

Документ HTML5, как и любой документ HTML, состоит из элементов, а элементы состоят из тегов. Как правило, элементы имеют открывающий и закрывающий тег, которые заключаются в угловые скобки.

Например:

* Текст элемента div *

HTML5

Элементы также могут состоять из одного тега, например, элемент
, функция которого - перенос строки.

* Текст
 элемента div *

Такие элементы еще называют пустыми элементами (void elements).

HTML5

Каждый элемент внутри открывающего тега может иметь атрибуты. Например:

```
<div style="color:red;">Кнопка</div>
```

```
<input type="button" value="Нажать">
```

Существуют глобальные или общие для всех элементов атрибуты, как например, style, а есть специфические, применяемые к определенным элементам, как например, type.

HTML5

Кроме обычных атрибутов существуют еще булевые или логические атрибуты (boolean attributes). Подобные атрибуты могут не иметь значения. Например, у кнопки можно задать атрибут disabled:

```
<input type="button" value="Нажать" disabled>
```

Атрибут disabled указывает, что данный элемент отключен.

HTML5

Глобальные атрибуты

- **class**: задает класс CSS, который будет применяться к элементу
- **contenteditable**: определяет, можно ли редактировать содержимое элемента
- **contextmenu**: определяет контекстное меню для элемента, которое отображается при нажатии на элемент правой кнопкой мыши
- **draggable**: определяет, можно ли перетаскивать элемент
- **dropzone**: определяет, можно ли копировать переносимые данные при переносе на элемент
- **hidden**: скрывает элемент

HTML5

Глобальные атрибуты

- **id**: уникальный идентификатор элемента. На веб-странице элементы не должны иметь повторяющихся идентификаторов
- **lang**: определяет язык элемента
- **spellcheck**: указывает, будет ли для данного элемента использоваться проверка правописания
- **style**: задает стиль элемента
- **tabindex**: определяет порядок, в котором по элементам можно переключаться с помощью клавиши TAB
- **title**: устанавливает дополнительное описание для элемента
- **translate**: определяет, должно ли переводиться содержимое элемента

Пользовательские атрибуты

В отличие от предыдущей версии языка разметки в HTML5 были добавлены пользовательские атрибуты (custom attributes). Теперь разработчик или создатель веб-страницы сам может определить любой атрибут, предваряя его префиксом data-.

Например:

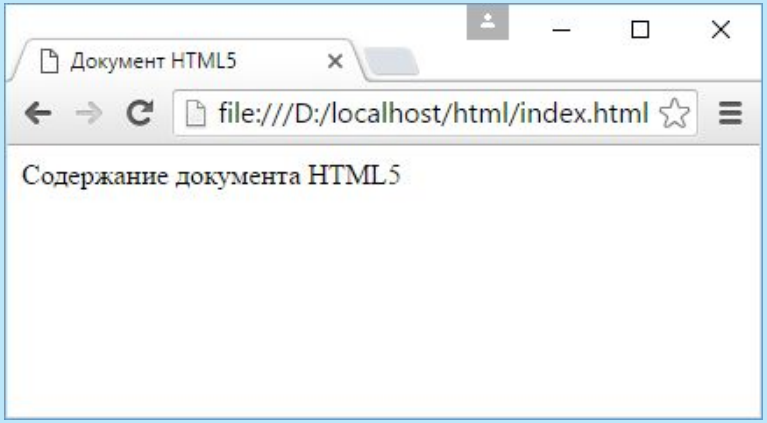
<input type="button" value="Нажать" data-color="red" >

Здесь определен атрибут data-color, который имеет значение "red".

HTML5

Для создания документа HTML5 нужны в первую очередь два элемента: DOCTYPE и html.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Документ HTML5</title>
   </head>
   <body>
      <div>Содержание документа HTML5</div>
   </body>
</html>
```



HTML5

При создании документа HTML5 мы можем использовать два различных стиля: HTML и XML.

Стиль HTML предполагает следующие моменты:

- Начальные открывающие теги могут отсутствовать у элементов
- Конечные закрывающие теги могут отсутствовать у элементов
- Только пустые элементы (void elements) (например, br, img, link) могут закрываться с помощью слеша />
- Регистр названий тегов и атрибутов не имеет значения
- Можно не заключать значения атрибутов в кавычки
- Некоторые атрибуты могут не иметь значений (checked и disabled)
- Специальные символы не экранируются
- Документ должен иметь элемент DOCTYPE

HTML5

Документ HTML5 также может быть описан с помощью синтаксиса XML. Такой стиль еще называют XHTML:

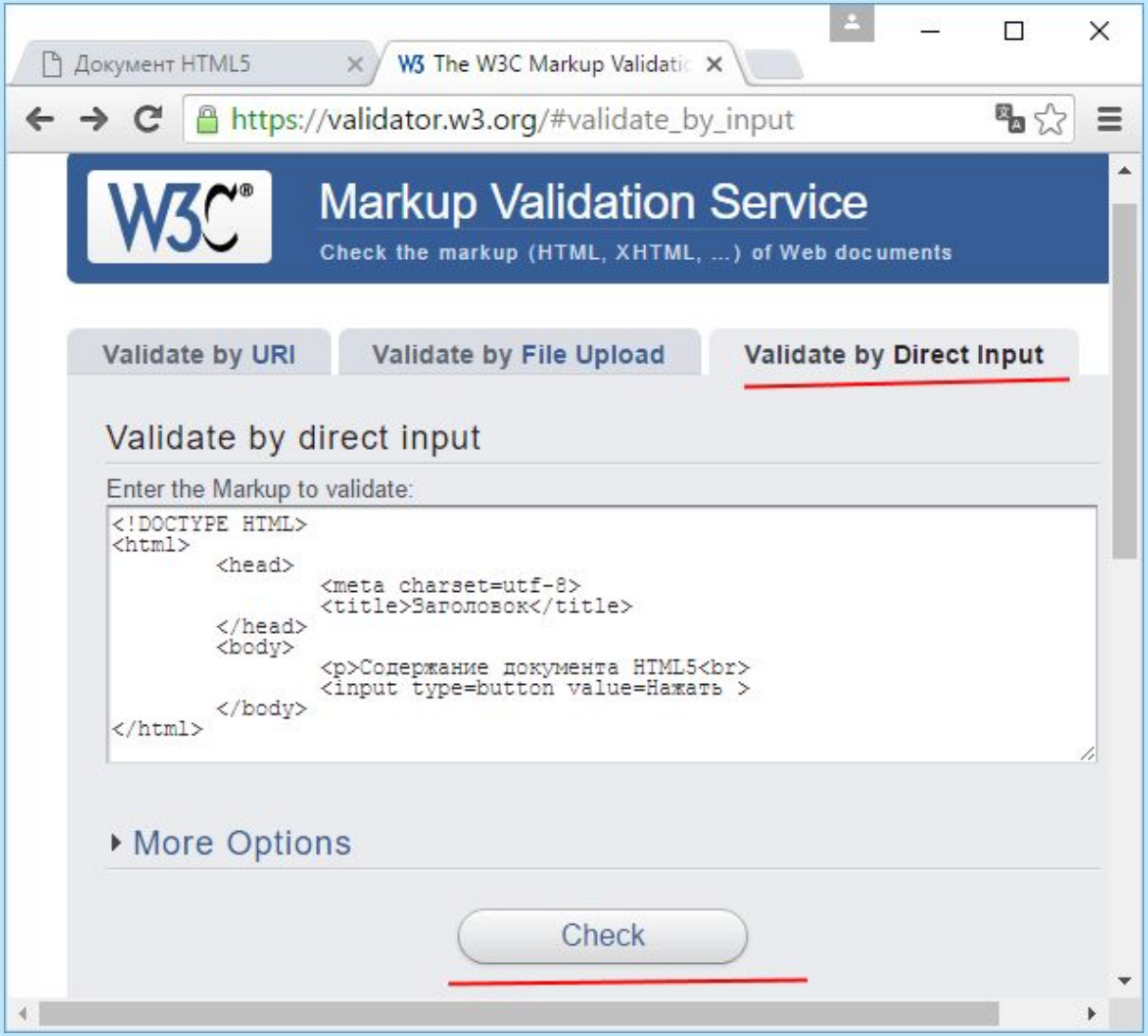
- Каждый элемент должен иметь начальный открывающий тег
- Непустые элементы (non-void elements) с начальным открывающим тегом также должны иметь конечный закрывающий тег
- Любой элемент может закрываться с помощью слеша />
- Названия тегов и атрибутов регистрозависимы, как правило, используются в нижнем регистре
- Значения атрибутов должны быть заключены в кавычки
- Атрибуты без значений не допускаются (checked="checked" вместо просто checked)
- Специальные символы должны быть экранированы

HTML5

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=utf-8>
        <title>Заголовок</title>
    </head>
    <body>
        <p>Содержание документа HTML5<br>
        <input type=button value=Нажать >
    </body>
</html>
```

```
<!doctype html>
<html
xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8">
        <title>Заголовок</title>
    </head>
    <body>
  <p>Содержание документа HTML5<br />
   <input type="button" value="Нажать" /></p>
    </body>
</html>
```

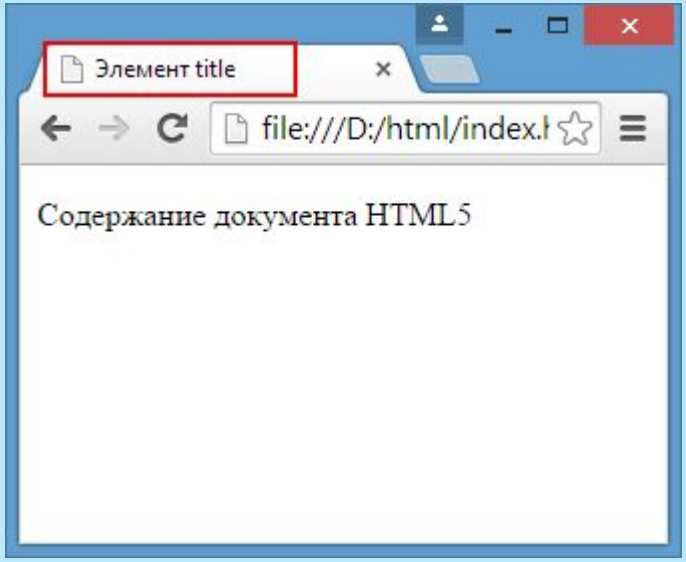
HTML5


Документ HTML5
The W3C Markup Validati
https://validator.w3.org/#validate_by_input
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Validate by URI
Validate by File Upload
Validate by Direct Input
Validate by direct input
Enter the Markup to validate:
<!DOCTYPE HTML>
<html>
<head>
<meta charset=utf-8>
<title>Заголовок</title>
</head>
<body>
<p>Содержание документа HTML5
<input type=button value=Нажать >
</body>
</html>
More Options
Check

HTML5. Элемент head и метаданные веб-страницы

Заголовок

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Элемент title</title>
    </head>
    <body>
        <p>Содержание документа HTML5</p>
    </body>
</html>
```



Элемент base

Позволяет указать базовый адрес, относительно которого устанавливаются другие адреса:

```
<!DOCTYPE html>
<html>
   <head>
      <base href="content/">
      <meta charset="utf-8">
      <title>Элемент base</title>
   </head>
   <body>
      <a href="newpage.html">Перейти</a>
   </body>
</html>
```

Можно также указывать полный адрес:
<base href="http://www.microsoft.com/">
В это случае ссылка будет вести по адресу http://www.microsoft.com/newpage.html

HTML5. Элемент head и метаданные веб-страницы

Элемент **meta** определяет метаданные документа.

Чтобы документ корректно отображал текст, необходимо задать кодировку с помощью атрибута charset.

Рекомендуемой кодировкой является utf-8:

<meta charset="utf-8">

HTML5. Элемент head и метаданные веб-страницы

По умолчанию в HTML определены пять типов метаданных:

- **application name**: название веб-приложения, частью которого является данный документ
- **author**: автор документа
- **description**: краткое описание документа
- **generator**: название программы, которая сгенерировала данный документ
- **keywords**: ключевые слова документа

HTML5. Элемент head и метаданные веб-страницы

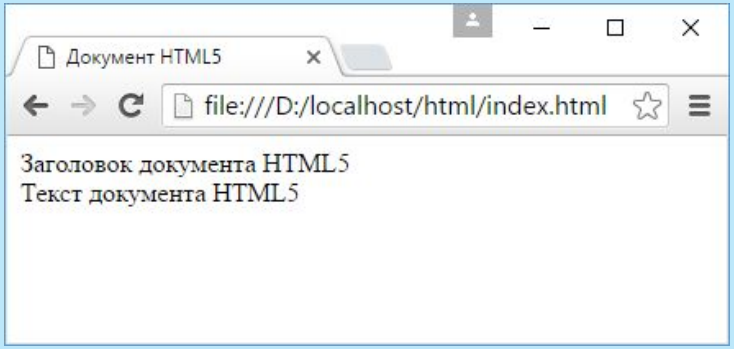
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <base href="content/">
        <title>Элемент title</title>
        <meta name="description" content="Первый документ HTML5">
        <meta name="author" content="Student">
    </head>
    <body>
        <a href="newpage.html">Содержание документа HTML5</a>
    </body>
</html>
```

HTML5. Элементы группировки

Элемент div

Служит для структуризации контента на веб-странице, для заключения содержимого в отдельные блоки. Div создает блок, который по умолчанию растягивается по всей ширине браузера, а следующий после div элемент переносится на новую строку.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <div>Заголовок документа HTML5</div>
        <div>Текст документа HTML5</div>
    </body>
</html>
```

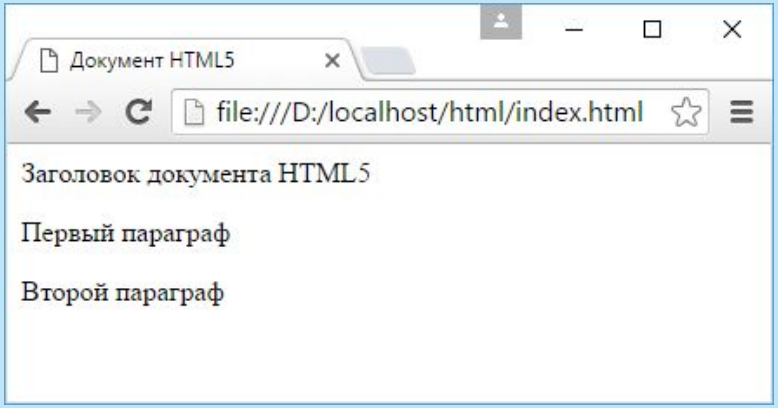


HTML5. Элементы группировки

Параграфы

Параграфы создаются с помощью тегов ** ** и ** **, которые заключают некоторое содержимое. Каждый новый параграф располагается на новой строке.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <div>Заголовок документа HTML5</div>
        <div>
            <p>Первый параграф</p>
            <p>Второй параграф</p>
        </div>
    </body>
</html>
```

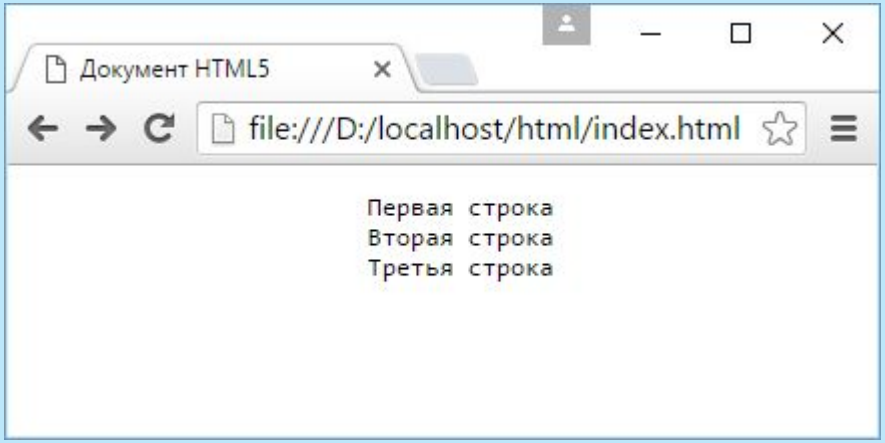


HTML5. Элементы группировки

Элемент pre

Элемент **pre** выводит предварительно отформатированный текст так, как он определен

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <pre>
            Первая строка
            Вторая строка
            Третья строка
        </pre>
    </body>
</html>
```

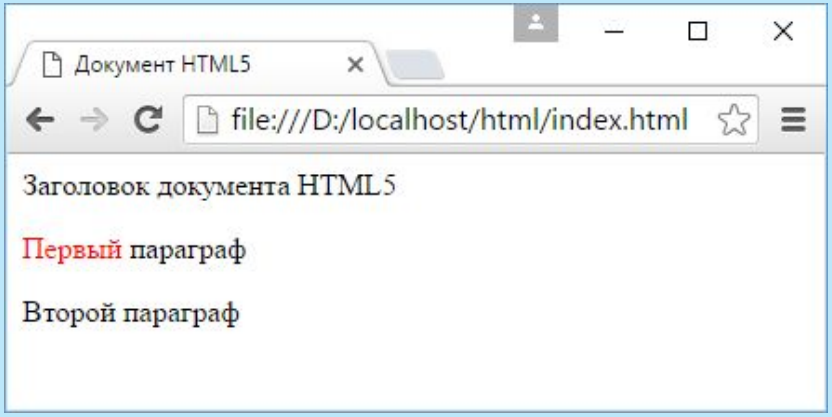


HTML5. Элементы группировки

Элемент span

Предназначен для определения строчных элементов документа. И служит преимущественно для стилизации заключенного в него текстового содержимого.

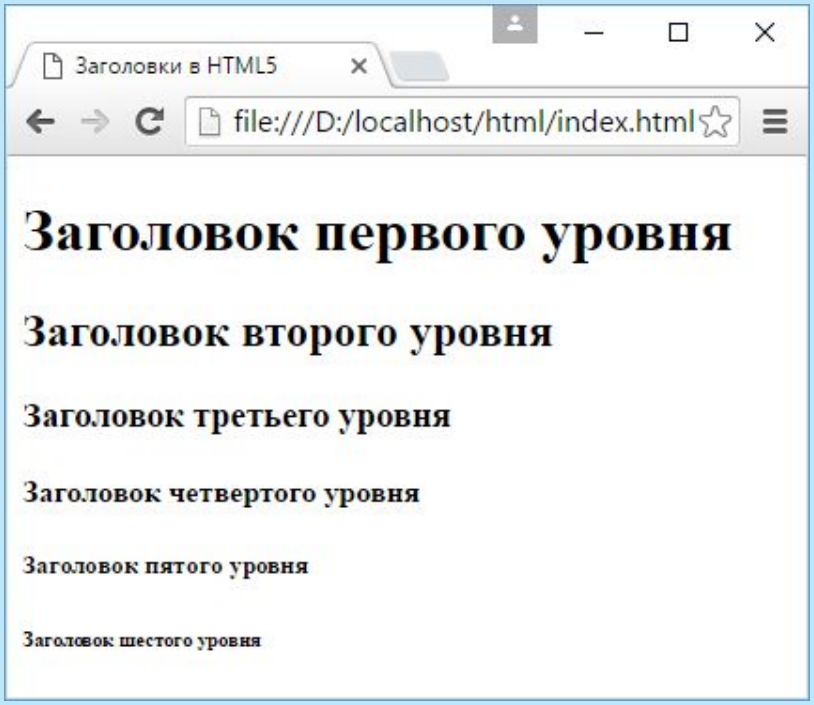
```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Документ HTML5</title>
   </head>
   <body>
      <div>Заголовок документа HTML5</div>
      <div>
         <p><span style="color:red;">Первый</span> параграф</p>
         <p><span>Второй</span> параграф</p>
      </div>
   </body>
</html>
```



HTML5. Заголовки

Элементы **<h1>, <h2>, <h3>, <h4>, <h5>** и **<h6>** служат для создания заголовков различного уровня.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Заголовки в HTML5</title>
   </head>
   <body>
      <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>
   </body>
</html>
```



HTML5. Форматирование текста

<b>: выделяет текст жирным

<del>: зачеркивает текст

<i>: выделяет текст курсивом

****: выделяет текст курсивом, в отличие от тега <i> носит логическое значение, придает выделяемому тексту оттенок важности

<s>: зачеркивает текст

<small>: делает текст чуть меньше размером, чем окружающий

****: выделяет текст жирным. В отличие от тега <b> предназначен для логического выделения, чтобы показать важность текста. А <b> не носит характера логического выделения, выполняет функции только форматирования

<sub>: помещает текст под строкой

<sup>: помещает текст над строкой

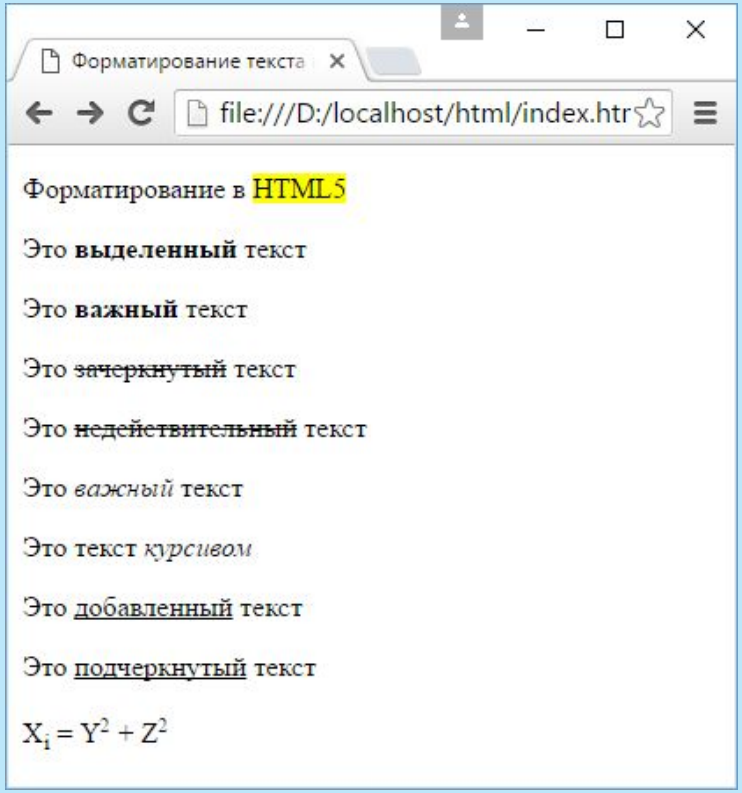
<u>: подчеркивает текст

<ins>: определяет вставленный (или добавленный) текст

<mark>: выделяет текст цветом, придавая ему оттенок важности

HTML5. Форматирование текста

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Форматирование текста в HTML5</title>
   </head>
   <body>
      <p>Форматирование в <mark>HTML5</mark></p>
      <p>Это <b>выделенный</b> текст</p>
      <p>Это <strong>важный</strong> текст</p>
      <p>Это <del>зачеркнутый</del> текст</p>
      <p>Это <s>недействительный</s> текст</p>
      <p>Это <em>важный</em> текст</p>
      <p>Это текст <i>курсивом</i> </p>
      <p>Это <ins>добавленный</ins> текст</p>
      <p>Это <u>подчеркнутый</u> текст</p>
      <p>X<sub>i</sub> = Y<sup><small>2</small></sup> + Z<sup><small>2</small></sup></p>
   </body>
</html>
```



HTML5. Работа с изображениями

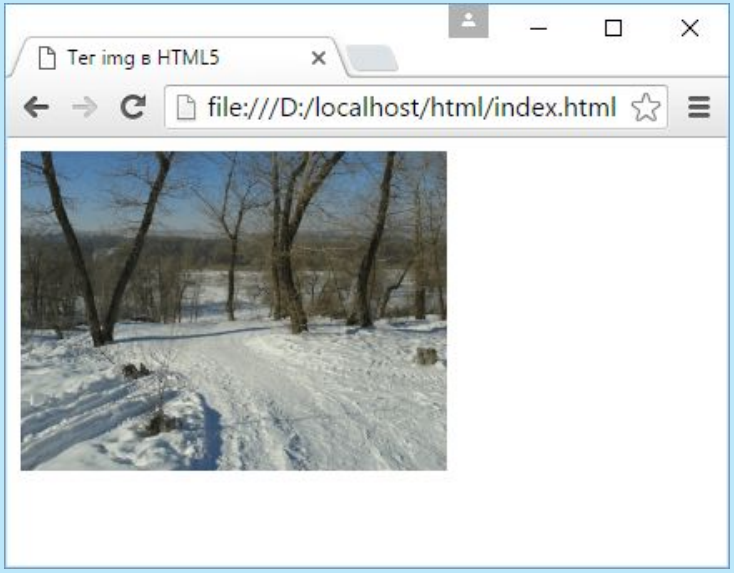
Для вывода изображений в HTML используется элемент **img**. Этот элемент представляет нам два важных атрибута:

- **src**: путь к изображению. Это может быть относительный или абсолютный путь в файловой системе или адрес в интернете

- **alt**: текстовое описание изображения. Если браузер по каким-то причинам не может отобразить изображение (например, если у атрибута src некорректно задан путь), то браузер показывает вместо самой картинки данное текстовое описание.

HTML5. Работа с изображениями

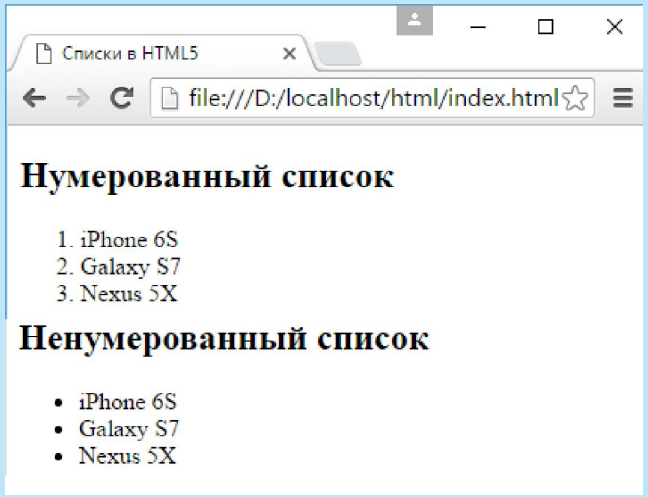
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Тег img в HTML5</title>
    </head>
    <body>
        <img src=«forest.png" alt="Зимняя равнина" />
    </body>
</html>
```



HTML5. Списки

Для создания списков в HTML5 применяются элементы <ol> (нумерованный список) и <ul> (ненумерованный список).

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Списки в HTML5</title>
   </head>
   <body>
      <h2>Нумерованный список</h2>
      <ol>
         <li>iPhone 6S</li>
         <li>Galaxy S7</li>
         <li>Nexus 5X</li>
      </ol>
      <h2>Ненумерованный список</h2>
      <ul>
         <li>iPhone 6S</li>
         <li>Galaxy S7</li>
         <li>Nexus 5X</li>
      </ul>
   </body>
</html>
```



HTML5. Списки

При необходимости можно настроить нумерацию или отражаемый рядом с элементом символ с помощью стиля **list-style-type**:

decimal: десятичные числа, отсчет идет от 1

decimal-leading-zero: десятичные числа, которые предваряются нулем, например, 01, 02, 03, … 98, 99

lower-roman: строчные римские цифры, например, i, ii, iii, iv, v

upper-roman: заглавные римские цифры, например, I, II, III, IV

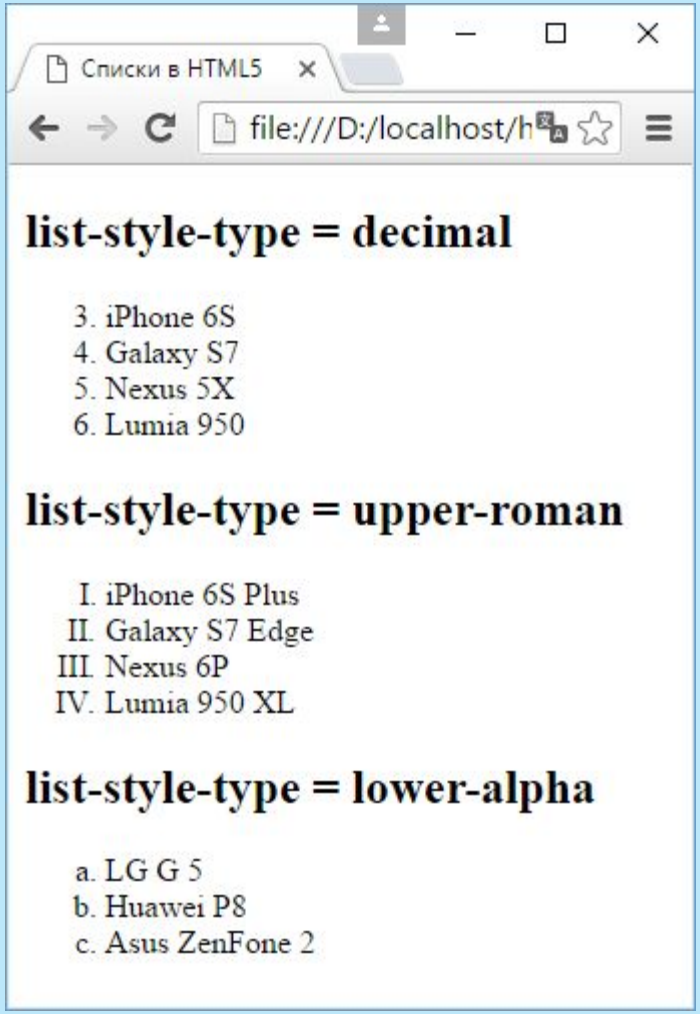
lower-alpha: строчные римские буквы, например, a, b, c..., z

upper-alpha: заглавные римские буквы, например, A, B, C, … Z

Для нумерованных список с помощью атрибута **start** можно дополнительно задать символ, с которого будет начинаться нумерация.

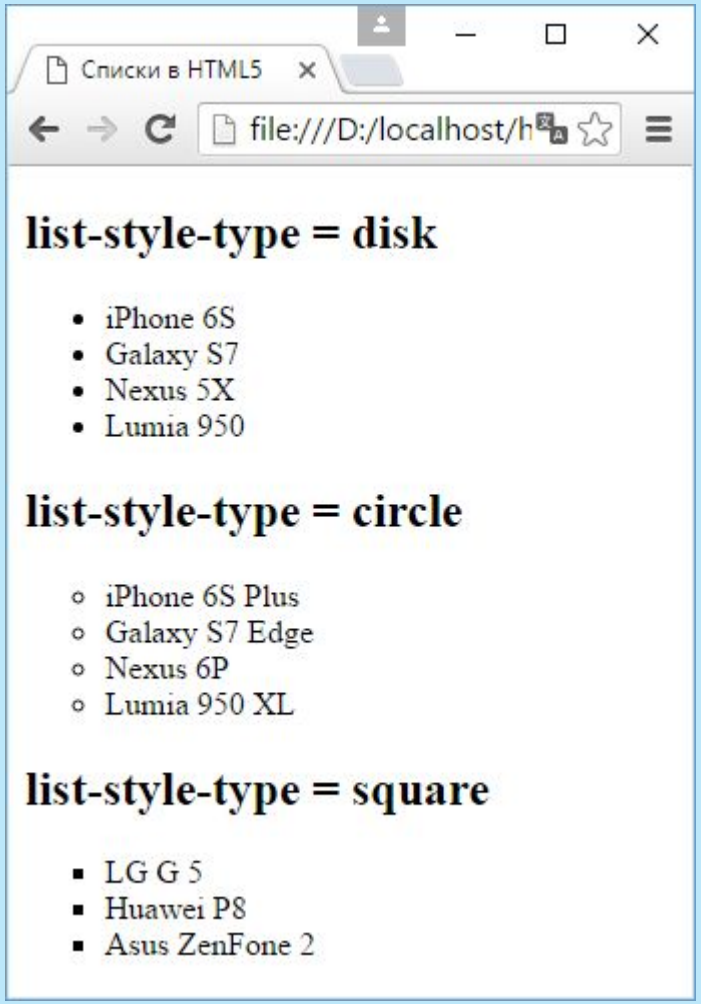
HTML5. Списки

```
<h2>list-style-type = decimal</h2>
<ol style="list-style-type:decimal;" start="3">
   <li>iPhone 6S</li>
   <li>Galaxy S7</li>
   <li>Nexus 5X</li>
   <li>Lumia 950</li>
</ol>
<h2>list-style-type = upper-roman</h2>
<ul style="list-style-type:upper-roman;">
   <li>iPhone 6S Plus</li>
   <li>Galaxy S7 Edge</li>
   <li>Nexus 6P</li>
   <li>Lumia 950 XL</li>
</ul>
<h2>list-style-type = lower-alpha</h2>
<ul style="list-style-type:lower-alpha;">
   <li>LG G 5</li>
   <li>Huawei P8</li>
   <li>Asus ZenFone 2</li>
</ul>
```



HTML5. Списки

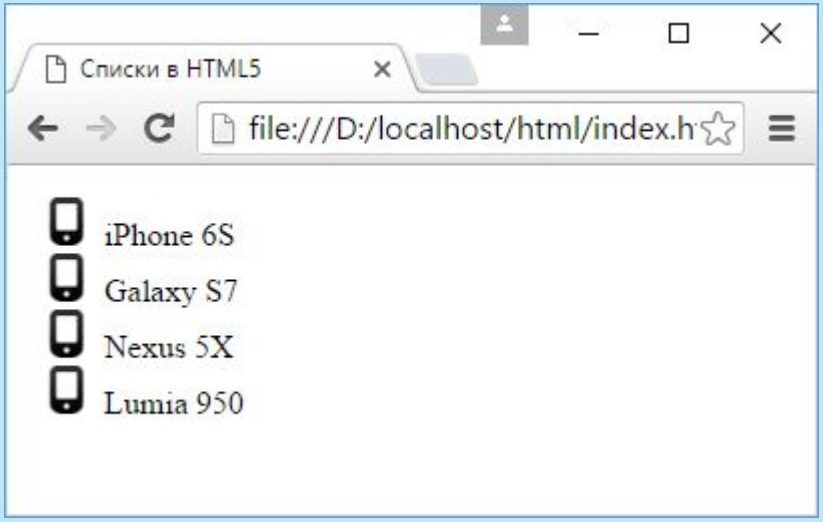
```
<h2>list-style-type = disc</h2>
<ul style="list-style-type:disc;">
   <li>iPhone 6S</li>
   <li>Galaxy S7</li>
   <li>Nexus 5X</li>
   <li>Lumia 950</li>
</ul>
<h2>list-style-type = circle</h2>
<ul style="list-style-type:circle;">
   <li>iPhone 6S Plus</li>
   <li>Galaxy S7 Edge</li>
   <li>Nexus 6P</li>
   <li>Lumia 950 XL</li>
</ul>
<h2>list-style-type = square</h2>
<ul style="list-style-type:square;">
   <li>LG G 5</li>
   <li>Huawei P8</li>
   <li>Asus ZenFone 2</li>
</ul>
```



HTML5. Списки

Еще одну интересную возможность по настройке списков предоставляет стиль **list-style-image**. Он задает изображение, которое будет отображаться рядом с элементом списка:

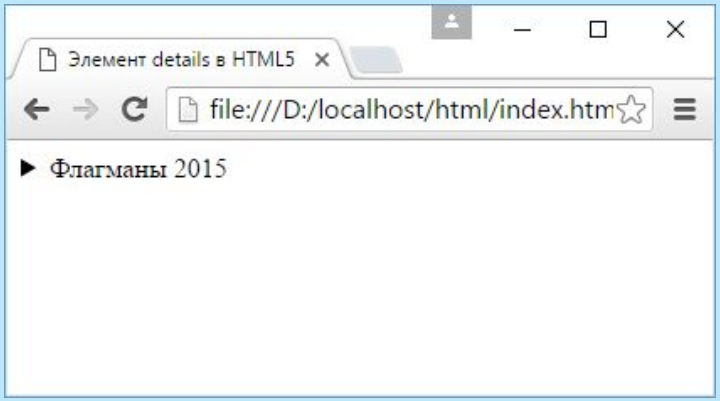
```
<ul style="list-style-image:url(phone_touch.png);">
    <li>iPhone 6S</li>
    <li>Galaxy S7</li>
    <li>Nexus 5X</li>
    <li>Lumia 950</li>
</ul>
```

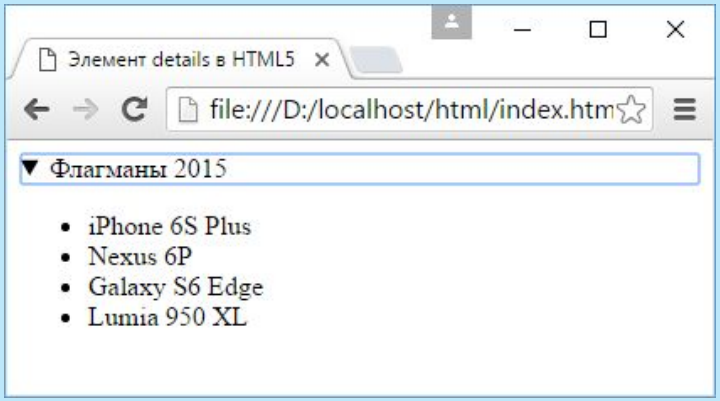


HTML5. Элемент details

Элемент **details** позволяет создавать раскрываемый блок, который по умолчанию скрыт. Данный элемент содержит элемент **summary**, который представляет заголовок для блока, и этот заголовок отображается в скрытом режиме.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Элемент details в HTML5</title>
   </head>
   <body>
      <details>
         <summary>Флагманы 2015</summary>
         <ul>
            <li>iPhone 6S Plus</li>
            <li>Nexus 6P</li>
            <li>Galaxy S6 Edge</li>
            <li>Lumia 950 XL</li>
         </ul>
      </details>
   </body>
</html>
```





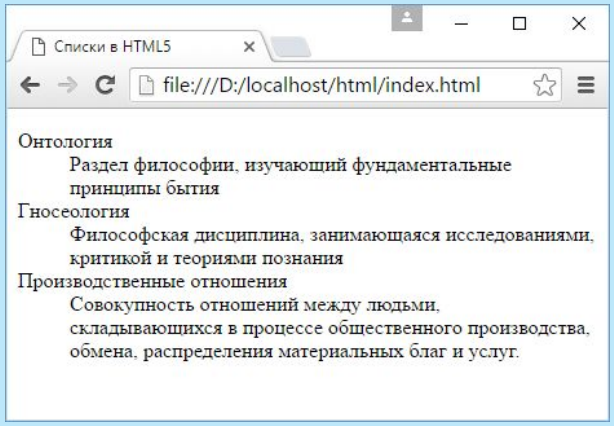
HTML5. Список определений

Для создания списка определений применяются теги **<dl>** и **</dl>** (definition list). Внутрь этих тегов помещаются элементы списка.

Каждый элемент списка состоит из термина и определения. Термин помещается в теги **<dt>** и **</dt>** (dt - сокращение от "definition term"), а определение - в теги **<dd>** и **</dd>** (dd - сокращение от "definition description").

HTML5. Список определений

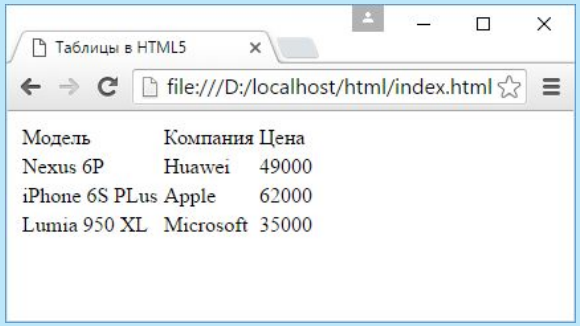
```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Списки в HTML5</title>
   </head>
   <body>
      <dl>
         <dt>Онтология</dt>
         <dd>Раздел философии, изучающий фундаментальные принципы бытия</dd>
         <dt>Гносеология</dt>
         <dd>Философская дисциплина, занимающаяся исследованиями, критикой и теориями познания</dd>
         <dt>Производственные отношения</dt>
         <dd>Совокупность отношений между людьми, складывающихся в процессе общественного производства, обмена, распределения материальных благ и услуг.</dd>
      </dl>
   </body>
</html>
```



HTML5. Таблицы

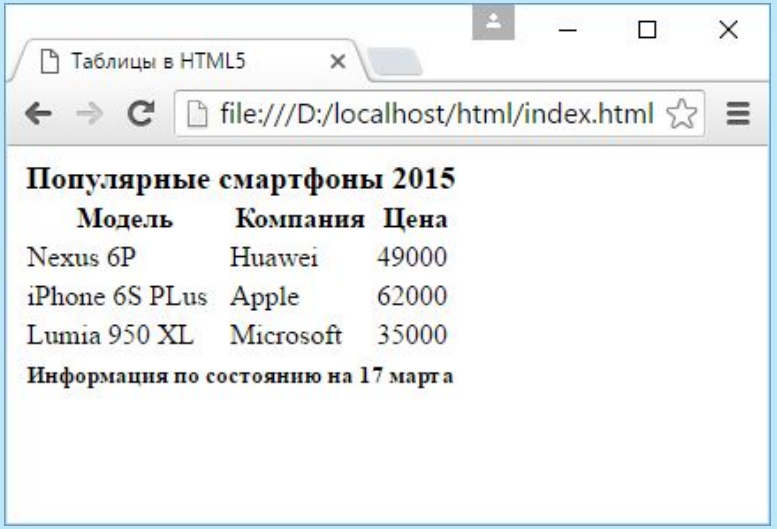
Для создания таблиц в html используется элемент **table**.
Каждая таблица между тегами <table> и </table> содержит строки, который представлены элементом **tr**.
А каждая строка между тегами <tr> и </tr> содержит ячейки в виде элементов **td**.

```
<table>
    <tr>
        <td>Модель</td> <td>Компания</td> <td>Цена</td>
    </tr>
    <tr>
        <td>Nexus 6P</td> <td>Huawei</td> <td>49000</td>
    </tr>
    <tr>
        <td>iPhone 6S PLus</td> <td>Apple</td> <td>62000</td>
    </tr>
    <tr>
        <td>Lumia 950 XL</td> <td>Microsoft</td> <td>35000</td>
    </tr>
</table>
```



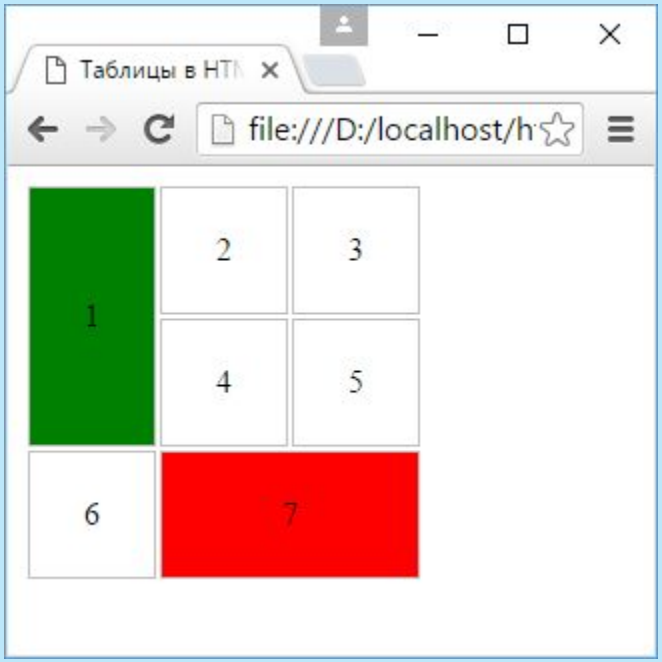
HTML5. Таблицы

```
<table>
   <caption><b>Популярные смартфоны 2015</b></caption>
   <thead>
      <tr>
         <th>Модель</th> <th>Компания</th> <th>Цена</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Nexus 6P</td> <td>Huawei</td> <td>49000</td>
      </tr>
      <tr>
         <td>iPhone 6S PLus</td> <td>Apple</td> <td>62000</td>
      </tr>
      <tr>
         <td>Lumia 950 XL</td> <td>Microsoft</td> <td>35000</td>
      </tr>
   <tbody>
   <tfoot>
      <tr>
         <th colspan="3">Информация по состоянию на 17 марта</th>
      </tr>
   </tfoot>
</table>
```



HTML5. Таблицы

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Таблицы в HTML5</title>
      <style>
         td{
            width: 60px;
            height:60px;
            border: solid 1px silver;
            text-align:center;
         }
      </style>
   </head>
   <body>
      <table>
         <tr>
            <td rowspan="2" style="background-color:green;">1</td>
            <td>2</td>
            <td>3</td>
         </tr>
         <tr>
            <td>4</td>
            <td>5</td>
         </tr>
         <tr>
            <td>6</td>
            <td colspan="2" style="background-color:red;">7</td>
         </tr>
      </table>
   </body>
</html>
```



HTML5. Ссылки

Ссылки, которые представлены элементом **<a>**, играют важную роль - они обеспечивают навигацию между отдельными документами.

Этот элемент имеет следующие атрибуты:

href: определяет адрес ссылки

hreflang: указывает на язык документа, на который ведет данная ссылка

media: определяет устройство, для которого предназначена ссылка

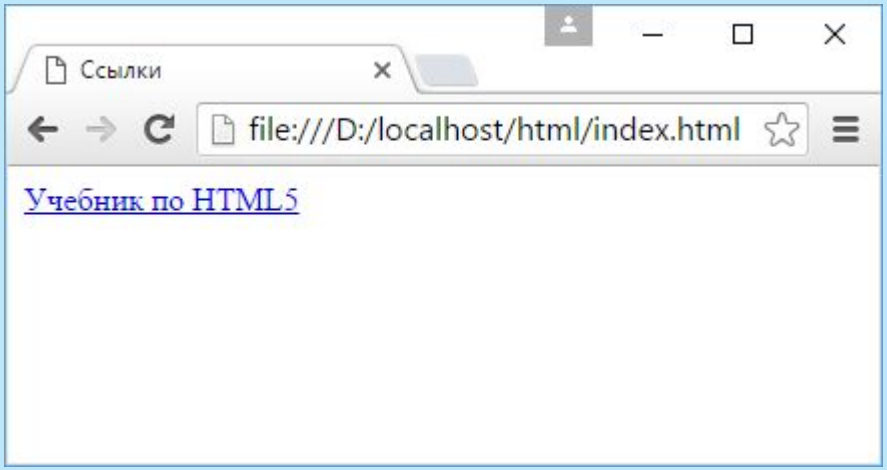
rel: определяет отношение между данным документом и ресурсом, на который ведет ссылка

target: определяет, как документ по ссылке должен открываться

type: указывает на mime-тип ресурса по ссылке

HTML5. Ссылки

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Ссылки</title>
    </head>
    <body>
        <a href="content.html">Учебник по HTML5</a>
    </body>
</html>
```



Навигация внутри документа

Можно задать внутренние ссылки, которые будут переходить к определенным блокам внутри элементов:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Внутренние ссылки</title>
    </head>
    <body>
        <a href="#paragraph1">Параграф 1</a> | <a href="#paragraph2">Параграф 2</a> | <a href="#paragraph3">Параграф 3</a>
        <h2 id="paragraph1">Параграф 1</h2>
        <p>Содержание параграфа 1</p>
        <h2 id="paragraph2">Параграф 2</h2>
        <p>Содержание параграфа 2</p>
        <h2 id="paragraph3">Параграф 3</h2>
        <p>Содержание параграфа 3</p>
    </body>
</html>
```

HTML5. Ссылки

Атрибут target

По умолчанию ресурсы, на которые ведут ссылке, открываются в том же окне. С помощью атрибута **target** можно переопределить это действие.

Атрибут **target** может принимать следующие значения:

- _blank: открытие html-документа в новом окне или вкладке браузера
- _self: открытие html-документа в том же фрейме (или окне)
- _parent: открытие документа в родительском фрейме, если ссылка расположена во внутреннем фрейме
- _top: открытие html-документа на все окно браузера
- framename: открытие html-документа во фрейме, который называется framename (В данном случае framename - только пример, название фрейма может быть произвольным)

HTML5. Ссылки

Стилизация ссылок

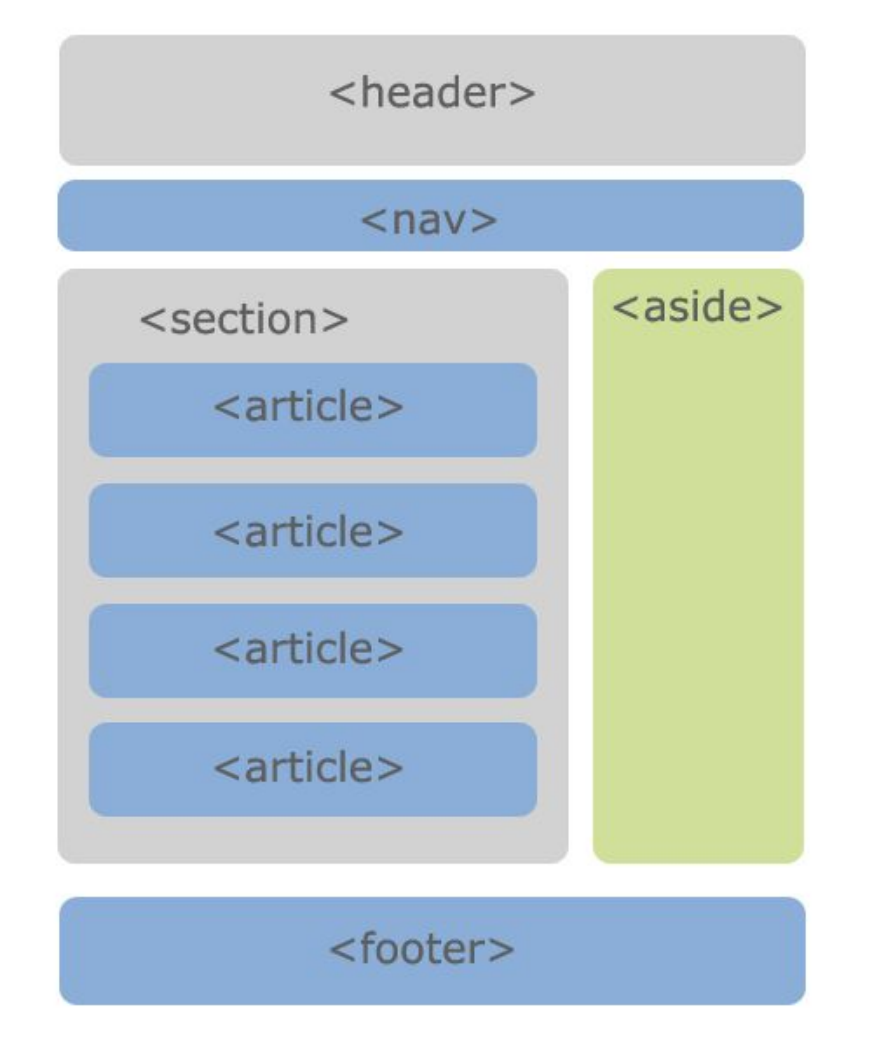
```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Ссылки</title>
      <style>
         a:link    {color:blue; text-decoration:none}
         a:visited {color:pink; text-decoration:none}
         a:hover   {color:red; text-decoration:underline}
         a:active  {color:yellow; text-decoration:underline}
      </style>
   </head>
   <body>
      <a href="index.html">Учебник по HTML5</a>
   </body>
</html>
```

Стиль color устанавливает цвет ссылки. А стиль text-decoration устанавливает подчеркивание: если значение underline, то ссылка поддчеркнута, если none, то подчеркивание отсутствует.

Фреймы позволяют встраивать на веб-страницу еще какую-нибудь другую веб-страницу. Фреймы представлены элементом iframe.


```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Фреймы в HTML5</title>
    </head>
    <body>
        <h2>Элемент iframe</h2>
        <iframe src="http://wikipedia.com" width="400" height="200">
        </iframe>
    </body>
</html>
```

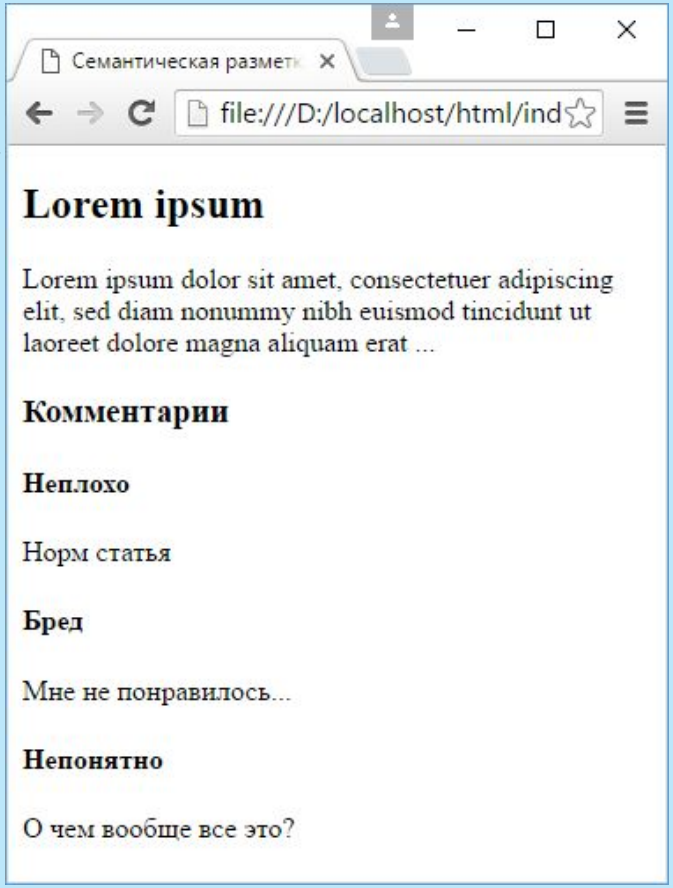
HTML5. Семантическая структура страницы



HTML5. Семантическая структура страницы

Элемент **article** представляет целостный блок информации на странице, который может рассматриваться отдельно и использоваться независимо от других блоков. Один элемент article может включать несколько элементов article.

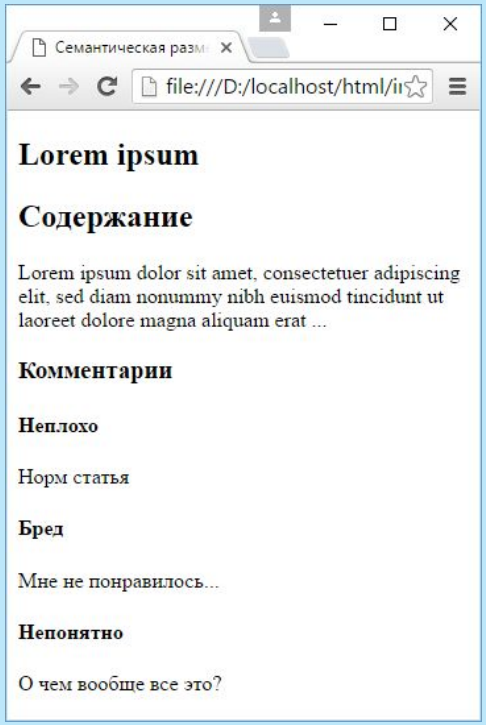
```
<body>
  <article>
    <h2>Lorem ipsum</h2>
    <div>
      Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy
      euismod tincidunt ut laoreet dolore magna aliquam erat ...
    </div>
    <div>
      <h3>Комментарии</h3>
      <article>
        <h4>Неплохо</h4>
        <p>Норм статья</p>
      </article>
      <article>
        <h4>Бред</h4>
        <p>Мне не понравилось...</p>
      </article>
      <article>
        <h4>Непонятно</h4>
        <p>О чем вообще все это?</p>
      </article>
    </div>
  </article>
</body>
```



HTML5. Семантическая структура страницы

Элемент **section** объединяет связанные между собой куски информации html-документа, выполняя их группировку. Например, section может включать набор вкладок на странице, новости, объединенные по категории и т.д.

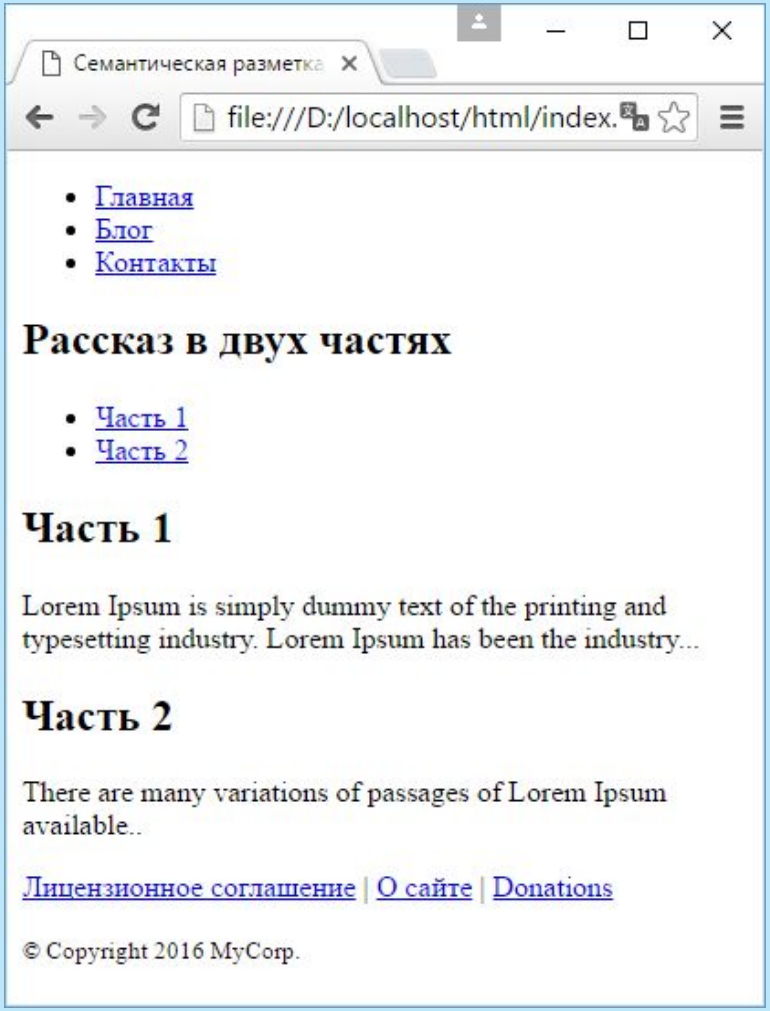
```
<body>
  <article>
    <h1>Lorem ipsum</h1>
    <section>
      <h2>Содержание</h2>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
      euismod tincidunt ut laoreet dolore magna aliquam erat ...</p>
    </section>
    <section>
      <h3>Комментарии</h3>
      <article>
        <h4>Неплохо</h4>
        <p>Норм статья</p>
      </article>
      <article>
        <h4>Бред</h4>
        <p>Мне не понравилось...</p>
      </article>
      <article>
        <h4>Непонятно</h4>
        <p>О чем вообще все это?</p>
      </article>
    </section>
  </article>
</body>
```



HTML5. Семантическая структура страницы

Элемент **nav** призван содержать элементы навигации по сайту. Как правило, это ненумерованный список с набором ссылок.

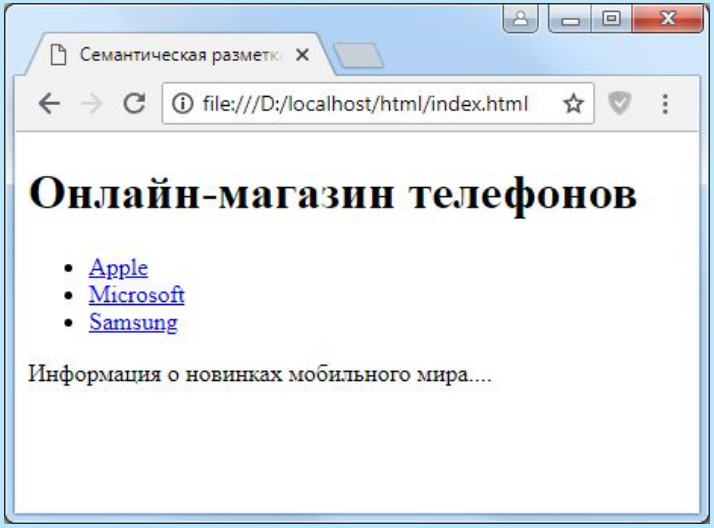
```
<body>
    <nav>
      <ul>
        <li><a href="/">Главная</a></li>
        <li><a href="/blog">Блог</a></li>
        <li><a href="/contacts">Контакты</a></li>
      </ul>
    </nav>
    <article>
      <header>
        <h2>Рассказ в двух частях</h2>
      </header>
      <nav>
        <ul>
          <li><a href="#part1">Часть 1</a></li>
          <li><a href="#part2">Часть 2</a></li>
        </ul>
      </nav>
      <div>
        <section id="part1">
          <h2>Часть 1</h2>
          <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.
          Lorem Ipsum has been the industry...</p>
        </section>
        <section id="part2">
          <h2>Часть 2</h2>
          <p>There are many variations of passages of Lorem Ipsum available..</p>
        </section>
      </div>
    </article>
    <footer>
      <p><a href="/license">Лицензионное соглашение</a> |
      <a href="/about">О сайте</a> |
      <a href="/donation">Donations</a></p>
      <p><small>© Copyright 2016 MyCorp.</small></p>
    </footer>
  </body>
```



HTML5. Семантическая структура страницы

Элемент **header** является как бы вводным элементом, предваряющим основное содержимое. Здесь могут быть заголовки, элементы навигации или какие-либо другие вспомогательные элементы, например, логотип, форма поиска и т.п.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Семантическая разметка в HTML5</title>
  </head>
  <body>
    <header>
      <h1>Онлайн-магазин телефонов</h1>
        <nav>
          <ul>
            <li><a href="/apple">Apple</a>
            <li><a href="/microsoft">Microsoft</a>
            <li><a href="/samsung">Samsung</a>
          </ul>
        </nav>
    </header>
    <div>
      Информация о новинках мобильного мира....
    </div>
  </body>
</html>
```

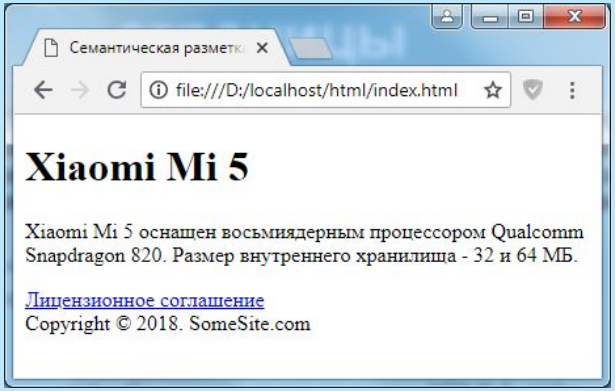


Элемент header нельзя помещать в такие элементы как address, footer или другой header.

HTML5. Семантическая структура страницы

Элемент **footer** обычно содержит информацию о том, кто автор контента на веб-странице, копирайт, дата публикации, блок ссылок на похожие ресурсы и т.д. Как правило, подобная информация располагается в конце веб-страницы или основного содержимого, однако, footer не имеет четкой привязки к позиции и может использоваться в различных местах веб-страницы.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Семантическая разметка в HTML5</title>
   </head>
   <body>
      <h1>Xiaomi Mi 5</h1>
      <div>
         Xiaomi Mi 5 оснащен восьмиядерным процессором Qualcomm Snapdragon 820.
         Размер внутреннего хранилища - 32 и 64 МБ.
      </div>
      <footer>
         <p><a href="/license">Лицензионное соглашение</a><br/>
         Copyright © 2018. SomeSite.com</p>
      </footer>
   </body>
</html>
```



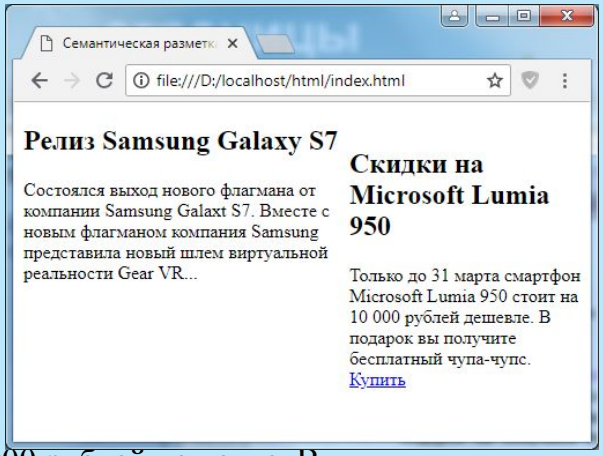
HTML5. Семантическая структура страницы

Элемент **address** предназначен для отображения контактной информации, которая связана с ближайшим элементом article или body. Нередко данный элемент размещается в футере:

```
<footer>
    <address>
        Контакты для связи <a href="mailto:js@example.com">Том Смит</a>.
    </address>
    <p>© copyright 2018 Example Corp.</p>
</footer>
```

HTML5. Семантическая структура страницы

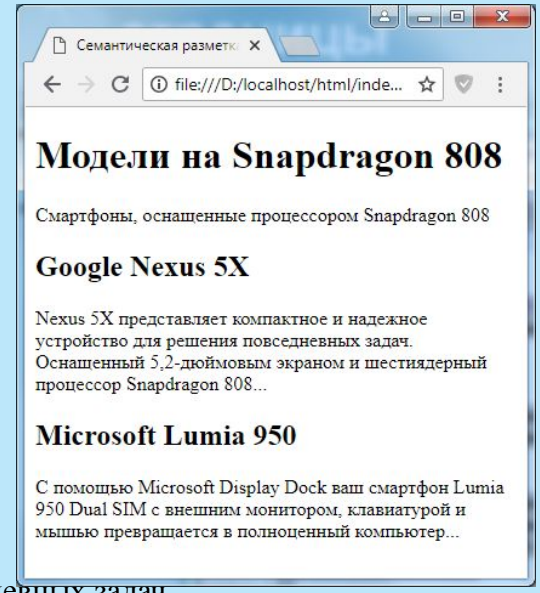
Элемент **aside** представляет содержимое, которое косвенно связано с остальным контентом веб-станицы и которое может рассматриваться независимо от него. Данный элемент можно использовать, например, для сайдбаров, для рекламных блоков, блоков навигационных элементов, различных плагинов типа твиттера или фейсбука и т.д.




```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Семантическая разметка в HTML5</title>
   </head>
   <body>
      <aside style="float:right; width:200px;">
         <h2>Скидки на Microsoft Lumia 950</h2>
         <p>Только до 31 марта смартфон Microsoft Lumia 950 стоит на 10 000 рублей дешевле. В
            подарок вы получите бесплатный чупа-чупс. <a href="buy/id=3">Купить</a></p>
      </aside>
      <article>
         <h2>Релиз Samsung Galaxy S7</h2>
         <p>Состоялся выход нового флагмана от компании Samsung Galaxt S7. Вместе с новым флагманом
компания
         Samsung представила новый шлем виртуальной реальности Gear VR...</p>
      </article>
   </body>
</html>
```

HTML5. Семантическая структура страницы

Элемент **main** представляет основное содержимое веб-страницы. Он представляет уникальный контент, в который не следует включать повторяющиеся на разных веб-страницах элементы сайдбаров, навигационные ссылки, информацию о копирайте, логотипы и тому подобное.




```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Семантическая разметка в HTML5</title>
  </head>
  <body>
    <main>
      <h1>Модели на Snapdragon 808</h1>
      <p>Смартфоны, оснащенные процессором Snapdragon 808</p>
      <article>
        <h2>Google Nexus 5X</h2>
        <p>Nexus 5X представляет компактное и надежное устройство для решения повседневных задач.
        Оснащенный 5,2-дюймовым экраном и шестиядерный процессор Snapdragon 808...</p>
      </article>
      <article>
        <h2>Microsoft Lumia 950</h2>
        <p>С помощью Microsoft Display Dock ваш смартфон Lumia 950 Dual SIM с внешним монитором,
          клавиатурой и мышью превращается в полноценный компьютер...</p>
      </article>
    </main>
  </body>
</html>
```