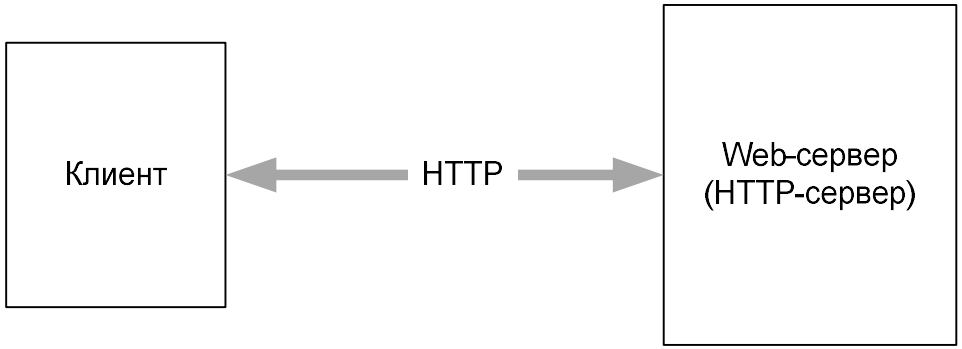
ПРОГРАММИРОВАНИЕ СЕРВЕРНЫХ КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ

ОСНОВЫ WEB-ПРОГРАММИРОВАНИЯ

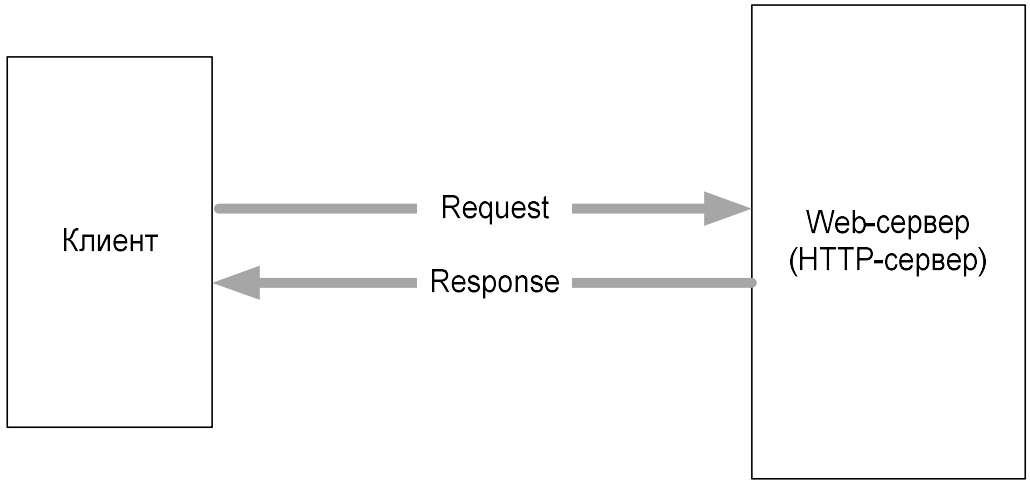
Web-программирование = раздел программирования, ориентированный на разработку web-приложений.

Архитектура web-приложения

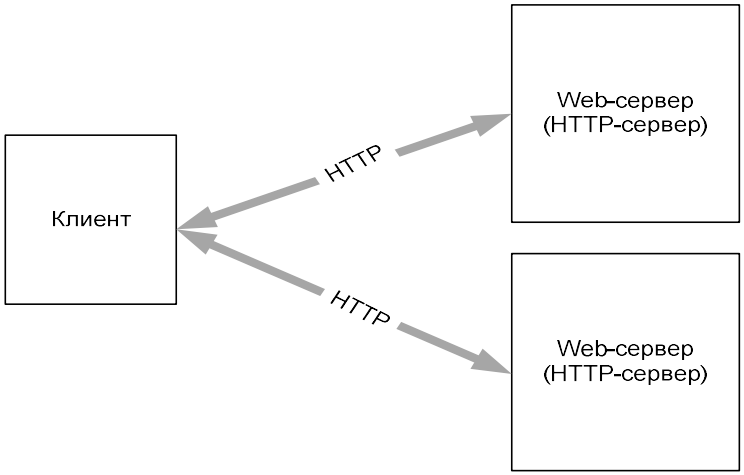


Web-приложение состоит из клиента и сервера, взаимодействующих по протоколу HTTP.

Взаимодействие по протоколу HTTP происходит посредством запросов и ответов.

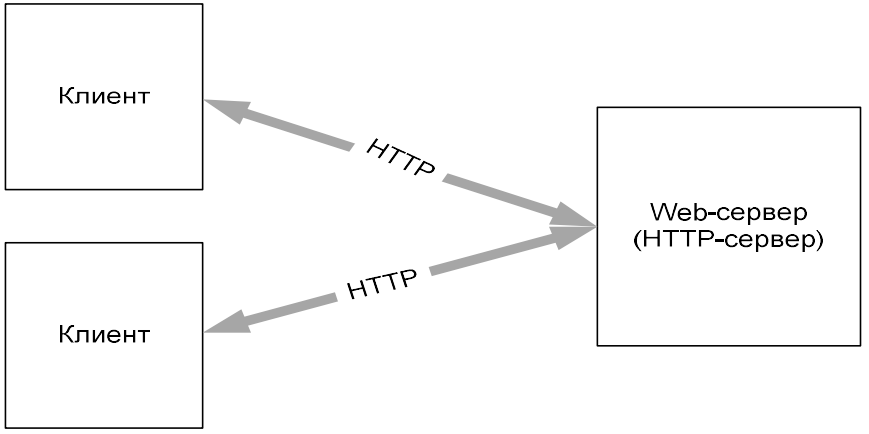


Архитектура web-приложения

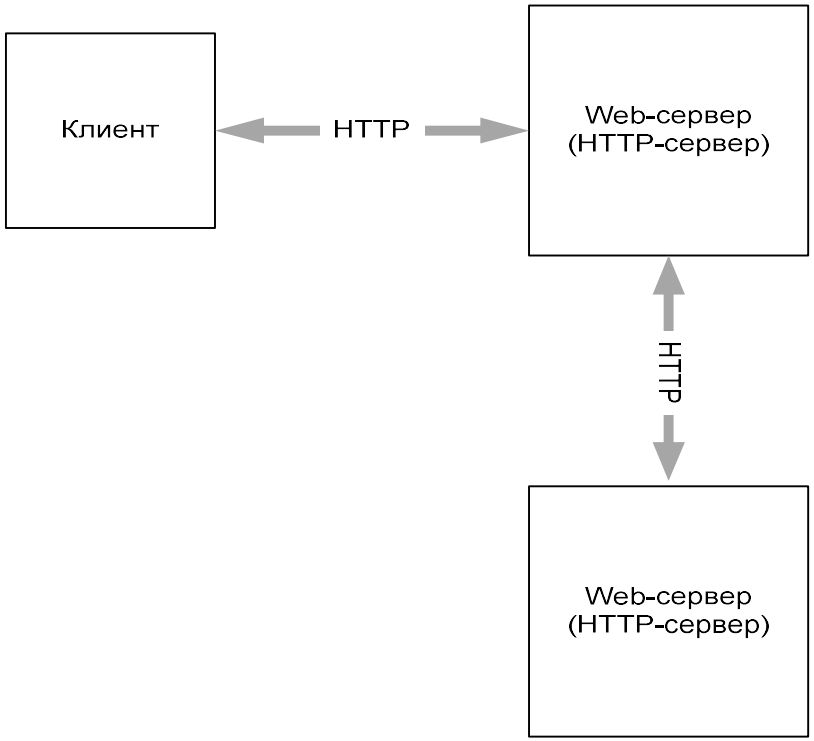


Обрабатывать запросы клиента могут различные серверы из кластера серверов.

Логично, что сервер может принимать запросы от множества клиентов.



Архитектура web-приложения



Сервер, в свою очередь, может выступать клиентом по отношению к другому серверу.

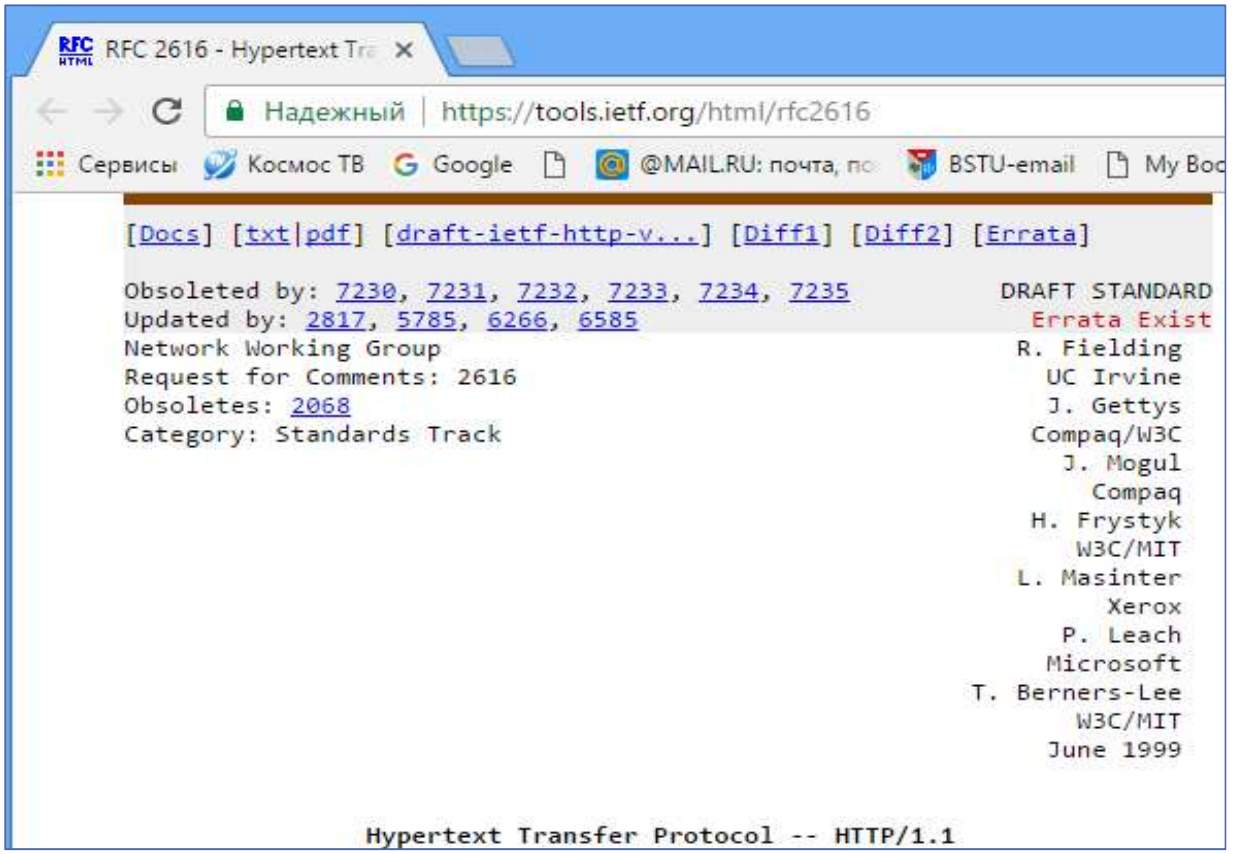
HTTP

протокол прикладного уровня модели OSI

Основные свойства

- версии HTTP/1.1 – действующий (текстовый), HTTP/2 – действующий (бинарный), HTTP/3 (раньше HTTP-over-QUIC, стадия черновика);
- два типа абонентов: клиент и сервер;
- два типа сообщений: request и response;
- от клиента к серверу – request;
- от сервера к клиенту – response;
- на один request всегда один response, иначе ошибка;
- одному response всегда один request, иначе ошибка;
- stateless;
- TCP-порты: 80, 443;
- для адресации используется URI или URL;
- поддерживается W3C, описан в нескольких RFC.

RFC2616 (стандарт HTTP)



Request

- метод;
- URI;
- версия протокола (HTTP/1.1);
- заголовки (пары: имя/значение);
- тело.

```
POST /cgi/process HTTP/1.1
User-Agent: Mozilla/4.0 (compatible)
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string
```

Response

- версия протокола (HTTP/1.1);
- код состояния (1xx, 2xx, 3xx, 4xx, 5xx);
- пояснение к коду состояния;
- заголовки (пары: имя/заголовок);
- тело.

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Методы HTTP

- GET – получение представления ресурса (идемпотентный);
- POST – передача сущности заданному ресурсу;
- PUT – замена представления указанного ресурса данными запроса (идемпотентный);
- DELETE – удаление указанного ресурса (идемпотентный);
- OPTIONS – определение возможностей веб-сервера;
- HEAD – аналогичен GET, но в ответе нет тела (идемпотентный);
- TRACE – предоставление информации о добавлении или изменении данных в запрос промежуточными серверами (идемпотентный);
- CONNECT - устанавливает двустороннюю связь с указанным ресурсом.

Заголовки HTTP

General: общие заголовки, используются в запросах и ответах.

```
general-header = Cache-Control            ; Section 14.9
               | Connection               ; Section 14.10
               | Date                     ; Section 14.18
               | Pragma                   ; Section 14.32
               | Trailer                  ; Section 14.40
               | Transfer-Encoding        ; Section 14.41
               | Upgrade                  ; Section 14.42
               | Via                      ; Section 14.45
               | Warning                  ; Section 14.46
```

Request: используются только в запросах.

```
request-header = Accept                   ; Section 14.1
               | Accept-Charset           ; Section 14.2
               | Accept-Encoding          ; Section 14.3
               | Accept-Language          ; Section 14.4
               | Authorization            ; Section 14.8
               | Expect                   ; Section 14.20
               | From                     ; Section 14.22
               | Host                     ; Section 14.23
               | If-Match                 ; Section 14.24

               | If-Modified-Since        ; Section 14.25
               | If-None-Match           ; Section 14.26
               | If-Range                 ; Section 14.27
               | If-Unmodified-Since      ; Section 14.28
               | Max-Forwards             ; Section 14.31
               | Proxy-Authorization      ; Section 14.34
               | Range                    ; Section 14.35
               | Referer                  ; Section 14.36
               | TE                       ; Section 14.39
               | User-Agent               ; Section 14.43
```

Заголовки HTTP

Response: используются только в ответах

```
response-header = Accept-Ranges           ; Section 14.5
                | Age                     ; Section 14.6
                | ETag                    ; Section 14.19
                | Location                ; Section 14.30
                | Proxy-Authenticate      ; Section 14.33
                | Retry-After             ; Section 14.37
                | Server                  ; Section 14.38
                | Vary                    ; Section 14.44
                | WWW-Authenticate        ; Section 14.47
```

Entity: для сущности в ответах и запросах

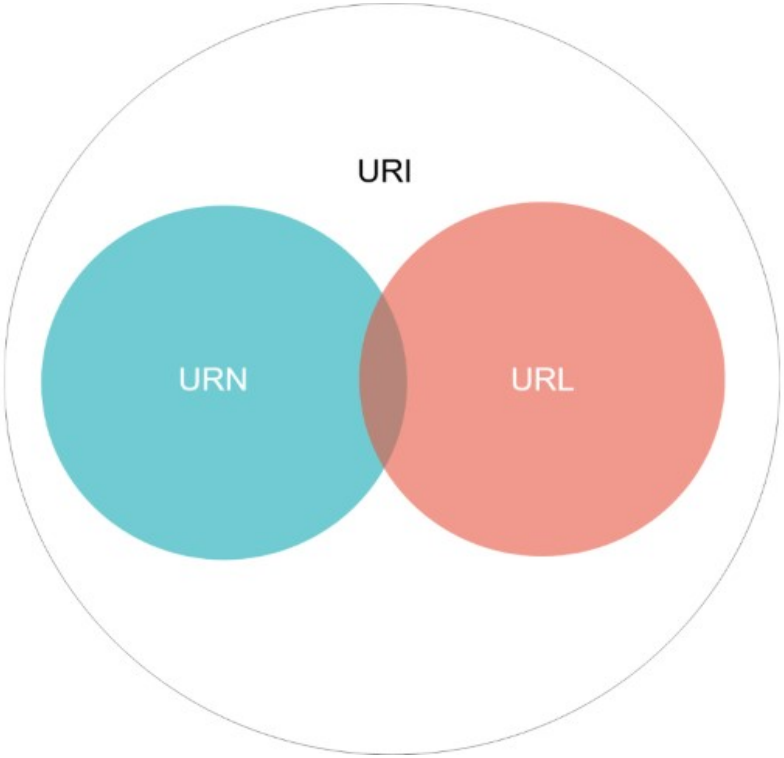
```
entity-header  = Allow                    ; Section 14.7
               | Content-Encoding          ; Section 14.11
               | Content-Language          ; Section 14.12
               | Content-Length            ; Section 14.13
               | Content-Location          ; Section 14.14
               | Content-MD5               ; Section 14.15
               | Content-Range             ; Section 14.16
               | Content-Type              ; Section 14.17
               | Expires                   ; Section 14.21
               | Last-Modified             ; Section 14.29
               | extension-header
```

Коды состояния ответа

- 1xx: информационные сообщения;
- 2xx: успешный ответ;
- 3xx: переадресация;
- 4xx: ошибка клиента;
- 5xx: ошибка сервера.

```
Status-Code    =
      "100"  ; Section 10.1.1: Continue
    | "101"  ; Section 10.1.2: Switching Protocols
    | "200"  ; Section 10.2.1: OK
    | "201"  ; Section 10.2.2: Created
    | "202"  ; Section 10.2.3: Accepted
    | "203"  ; Section 10.2.4: Non-Authoritative Information
    | "204"  ; Section 10.2.5: No Content
    | "205"  ; Section 10.2.6: Reset Content
    | "206"  ; Section 10.2.7: Partial Content
    | "300"  ; Section 10.3.1: Multiple Choices
    | "301"  ; Section 10.3.2: Moved Permanently
    | "302"  ; Section 10.3.3: Found
    | "303"  ; Section 10.3.4: See Other
    | "304"  ; Section 10.3.5: Not Modified
    | "305"  ; Section 10.3.6: Use Proxy
    | "307"  ; Section 10.3.8: Temporary Redirect
    | "400"  ; Section 10.4.1: Bad Request
    | "401"  ; Section 10.4.2: Unauthorized
    | "402"  ; Section 10.4.3: Payment Required
    | "403"  ; Section 10.4.4: Forbidden
    | "404"  ; Section 10.4.5: Not Found
    | "405"  ; Section 10.4.6: Method Not Allowed
    | "406"  ; Section 10.4.7: Not Acceptable
    | "407"  ; Section 10.4.8: Proxy Authentication Required
    | "408"  ; Section 10.4.9: Request Time-out
    | "409"  ; Section 10.4.10: Conflict
    | "410"  ; Section 10.4.11: Gone
    | "411"  ; Section 10.4.12: Length Required
    | "412"  ; Section 10.4.13: Precondition Failed
    | "413"  ; Section 10.4.14: Request Entity Too Large
    | "414"  ; Section 10.4.15: Request-URI Too Large
    | "415"  ; Section 10.4.16: Unsupported Media Type
    | "416"  ; Section 10.4.17: Requested range not satisfiable
    | "417"  ; Section 10.4.18: Expectation Failed
    | "500"  ; Section 10.5.1: Internal Server Error
    | "501"  ; Section 10.5.2: Not Implemented
    | "502"  ; Section 10.5.3: Bad Gateway
    | "503"  ; Section 10.5.4: Service Unavailable
    | "504"  ; Section 10.5.5: Gateway Time-out
    | "505"  ; Section 10.5.6: HTTP Version not supported
    | extension-code
```

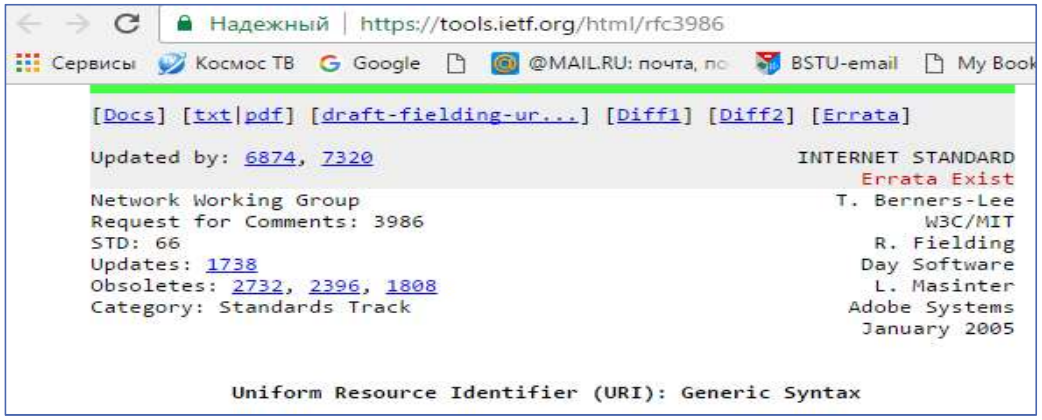
URI, URL, URN



URI, URL, URN – рекомендуется использовать термин URI

- URI: Uniform Resource Identifier – унифицированный **идентификатор** ресурса (документ, изображение, файл, электронная почта, …). Идентифицирует ресурс по имени, местоположению или тому и другому.
- URL: Uniform Resource Location – унифицированный **локатор** ресурса, содержащий местонахождение ресурса и способ обращения (протокол) к ресурсу, описывает множество URI.
- URN: Uniform Resource Name – унифицированное **имя** ресурса – имя ресурса, не содержащее месторасположение и способ доступа к ресурсу. Позволяет ресурсам перемещаться с одного места на другое. В контексте веба URN практически не используется.

STD 66



Надежный | https://tools.ietf.org/html/rfc3986

Сервисы | Космос ТВ | Google | @MAIL.RU: почта, по | BSTU-email | My Book

[Docs] [txt|pdf] [draft-fielding-ur...] [Diff1] [Diff2] [Errata]

Updated by: 6874, 7320 — INTERNET STANDARD

Errata Exist

Network Working Group — T. Berners-Lee

Request for Comments: 3986 — W3C/MIT

STD: 66 — R. Fielding

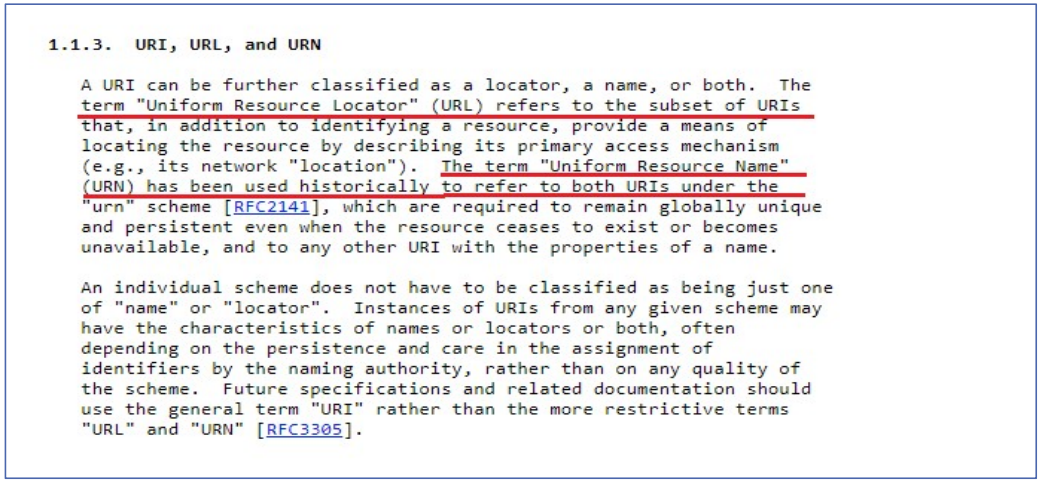
Updates: 1738 — Day Software

Obsoletes: 2732, 2396, 1808 — L. Masinter

Category: Standards Track — Adobe Systems

January 2005

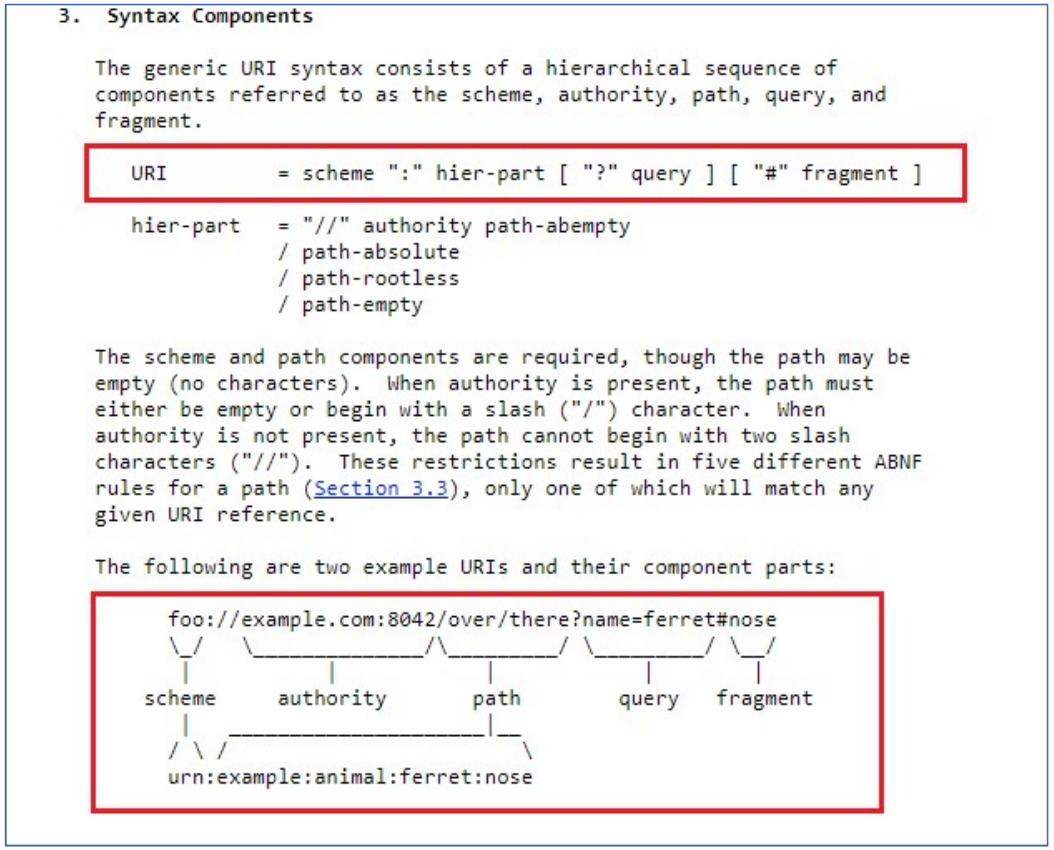
Uniform Resource Identifier (URI): Generic Syntax



1.1.3. URI, URL, and URN

A URI can be further classified as a locator, a name, or both. The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). The term "Uniform Resource Name" (URN) has been used historically to refer to both URIs under the "urn" scheme [RFC2141], which are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable, and to any other URI with the properties of a name.

An individual scheme does not have to be classified as being just one of "name" or "locator". Instances of URIs from any given scheme may have the characteristics of names or locators or both, often depending on the persistence and care in the assignment of identifiers by the naming authority, rather than on any quality of the scheme. Future specifications and related documentation should use the general term "URI" rather than the more restrictive terms "URL" and "URN" [RFC3305].



3. Syntax Components

The generic URI syntax consists of a hierarchical sequence of components referred to as the scheme, authority, path, query, and fragment.

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty
```

The scheme and path components are required, though the path may be empty (no characters). When authority is present, the path must either be empty or begin with a slash ("/") character. When authority is not present, the path cannot begin with two slash characters ("//"). These restrictions result in five different ABNF rules for a path (Section 3.3), only one of which will match any given URI reference.

The following are two example URIs and their component parts:

```
      foo://example.com:8042/over/there?name=ferret#nose
      \_/   \______________/\_________/ \_________/ \__/
       |           |            |            |        |
    scheme     authority       path        query   fragment
       |   _____________________|__
      / \ /                        \
      urn:example:animal:ferret:nose
```

STD 66

3.2. Authority

Many URI schemes include a hierarchical element for a naming authority so that governance of the name space defined by the remainder of the URI is delegated to that authority (which may, in turn, delegate it further). The generic syntax provides a common means for distinguishing an authority based on a registered name or server address, along with optional port and user information.

The authority component is preceded by a double slash ("//") and is terminated by the next slash ("/"), question mark ("?"), or number sign ("#") character, or by the end of the URI.

```
authority   = [ userinfo "@" ] host [ ":" port ]
```

URI producers and normalizers should omit the ":" delimiter that separates host from port if the port component is empty. Some schemes do not allow the userinfo and/or port subcomponents.

If a URI contains an authority component, then the path component must either be empty or begin with a slash ("/") character. Non-validating parsers (those that merely separate a URI reference into its major components) will often ignore the subcomponent structure of authority, treating it as an opaque string from the double-slash to the first terminating delimiter, until such time as the URI is dereferenced.

3.2.1. User Information

The userinfo subcomponent may consist of a user name and, optionally, scheme-specific information about how to gain authorization to access the resource. The user information, if present, is followed by a commercial at-sign ("@") that delimits it from the host.

```
userinfo    = *( unreserved / pct-encoded / sub-delims / ":" )
```

Use of the format "user:password" in the userinfo field is deprecated. Applications should not render as clear text any data after the first colon (":") character found within a userinfo subcomponent unless the data after the colon is the empty string (indicating no password). Applications may choose to ignore or reject such data when it is received as part of a reference and should reject the storage of such data in unencrypted form. The passing of authentication information in clear text has proven to be a security risk in almost every case where it has been used.

Applications that render a URI for the sake of user feedback, such as in graphical hypertext browsing, should render userinfo in a way that is distinguished from the rest of a URI, when feasible. Such rendering will assist the user in cases where the userinfo has been misleadingly crafted to look like a trusted domain name (Section 7.6).

RFC 2141 (URN)

2. Syntax

All URNs have the following syntax (phrases enclosed in quotes are REQUIRED):

<URN> ::= "urn:" <NID> ":" <NSS>

where <NID> is the Namespace Identifier, and <NSS> is the Namespace Specific String. The leading "urn:" sequence is case-insensitive. The Namespace ID determines the _syntactic_ interpretation of the Namespace Specific String (as discussed in [1]).

PURL

Persistent Uniform Resource Locator – постоянный унифицированный локатор ресурса.

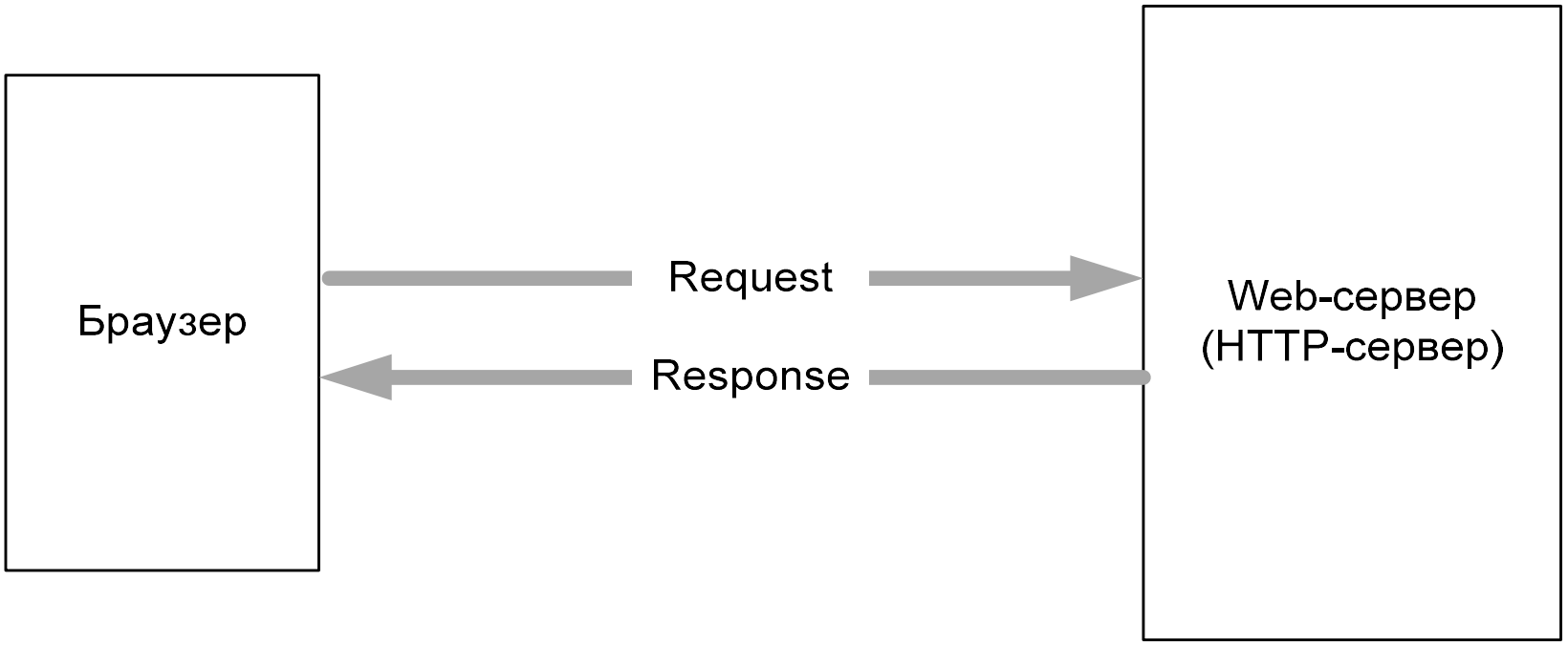
Например: http://purpl.by/bstu/faculties/lh/lv.html,

http://purpl.by/bstu – указывает на БД, в которой по имени bstu можно найти URL http://www.bstu.by и получить искомый URL:

http://www.bstu.by/faculties/lh/lv.html.

Доступ к конечному ресурсу через redirect.

Web-браузер – это клиент

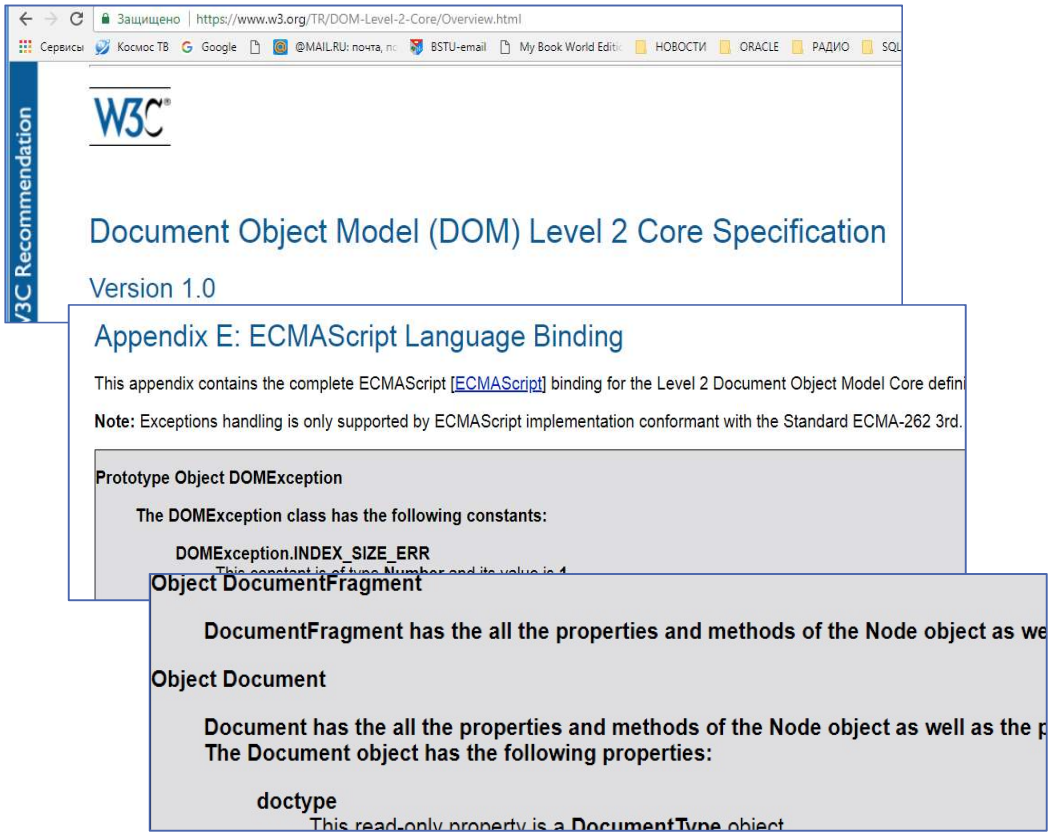


Способы генерации запросов web-браузером

- адресная URI-строка (GET);
- HTML-тег: **<form>** (POST, GET);
- HTML-тег: **<a>** (GET);
- HTML-тег: **<img>** (GET);
- HTML-тег: **<script>** (GET);
- HTML-тег: **<link>** (GET);
- HTML-тег: **<audio>** (GET);
- HTML-тег: **<video>** (GET);
- HTML-тег: **<frame>** (GET), не поддерживается в HTML5;
- Объект web-браузера: **XMLHTTPRequest**;
- JavaScript API: **web-сокеты**;
- Fetch API: fetch;
- JQuery;
- Axios.

Модель DOM (Document Object Model)

- Объектная модель документа, представление HTML-документа web-браузером в виде объектов, интерфейс JavaScript для доступа к содержимому HTML-документа.
- Три уровня DOM0, DOM1, DOM2, DOM3.
- Современные браузеры уровня 2 с элементами уровня 3.
- Содержит описание JS (ECMAScript).



CSS (Cascading Style Sheets)

- Каскадные таблицы стилей – код, используемый для стилизации вашей страницы.
- Уровни: CSS1, CSS2, CSS3 (текущий), CSS4 (в разработке с 2011).
- CSS-фреймворки: Bootstrap, Kube, Foundation, Semantic UI, PureCSS.
- CSS-расширения: Sass, SCSS, LESS.

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

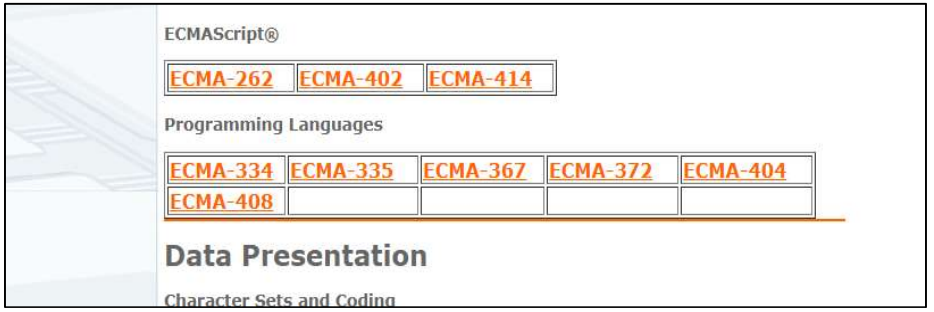
Рекомендуют использовать HTML5, CSS3, SVG, JavaScript (стандарты: ECMAScript5, ECMA-262-6, ECMA-262-7).

ECMAScript

ECMAScript — это язык программирования, используемый в качестве основы для построения других скриптовых языков. Стандартизирован международной организацией ECMA (European Computer Manufacturers Association) в спецификации ECMA-262. Расширения языка: JavaScript, JScript и ActionScript.

ECMAScript – это стандарт, а JavaScript его реализация.

У стандарта есть много версий: ES1, ES2, ES3, ES5, ES2015 (ES6), ES2016 (ES7) и т.д. каждый год.



Web-engine (web-движок)

Это программа, преобразующая содержимое html-страниц во внутреннее представление web-браузера в соответствии с моделью DOM.

Движок	Разработчик(и)	Лицензия	Браузер	Язык программирования
Blink	Google, Opera, Samsung, Intel, others [1]	GNU LGPL, BSD-style	Google Chrome & Opera >15.0	C++
EdgeHTML	Microsoft	Проприетарная	Microsoft Edge	C++
Gecko	Netscape/Mozilla Foundation	MPL / LGPL / GPL	Mozilla Firefox	C++
GtkHTML	GNOME	GNU LGPL	Novell Evolution	C
KHTML	KDE	GNU LGPL	Konqueror	C++
Presto	Opera Software	Проприетарная	Opera	C++[2]
Prince XML	YesLogic Pty Ltd	Проприетарная	Prince XML	Mercury
Robin	Ritlabs	Проприетарная	The Bat!	Delphi
Tasman	Microsoft	Проприетарная	Microsoft Entourage	—
Trident	Microsoft	Проприетарная	Internet Explorer	C++[3]
WebKit	KDE, Apple, Nokia, Google, RIM, Palm и другие.	GNU LGPL, BSD	Google Chrome, Safari	C++
XEP	RenderX	Проприетарная	XEP	Java

Web-серверы

IIS (Microsoft), Apache (Apache Software Foundation), WebSphere (IBM), WebLogic (Oracle), Glassfish (Oracle), Apache Tomcat (Apache Software Foundation), nginx (И.Сысоев),…