HTTP, the Internet, and the Web

Outline

HTTP, the Internet, and the Web

It explains the protocol that defines how Web browsers communicate with Web servers, the mechanisms that keep that communication secure from counterfeits and eavesdroppers, and the tech- nologies that accelerate our Web experience.
It was in March 1989 that Tim Berners-Lee first outlined the advantages of a hypertext-based, linked infor- mation system. And by the end of 1990, Berners-Lee, along with Robert Cailliau, created the first Web browsers and servers. Those browsers needed a protocol to regulate their communications; for that Berners-Lee and Cailliau designed the first version of http.

Internet architecture

closer look at network structure

1. Network edge:
   applications and hosts
2. Network core:
   routers
   network of networks
3. Access networks physical media:
   communication links

Program

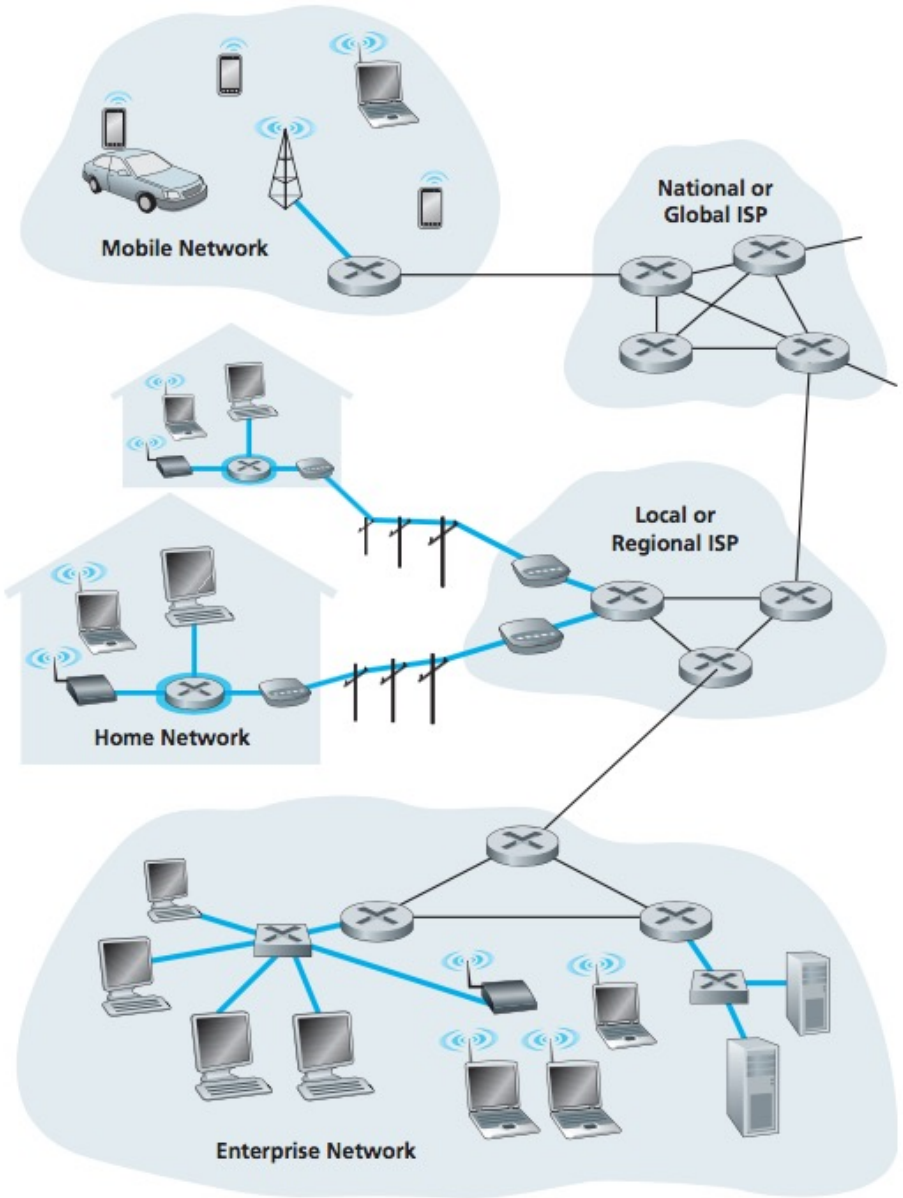


Figure 1.4 ♦ Access networks

Protocol stack

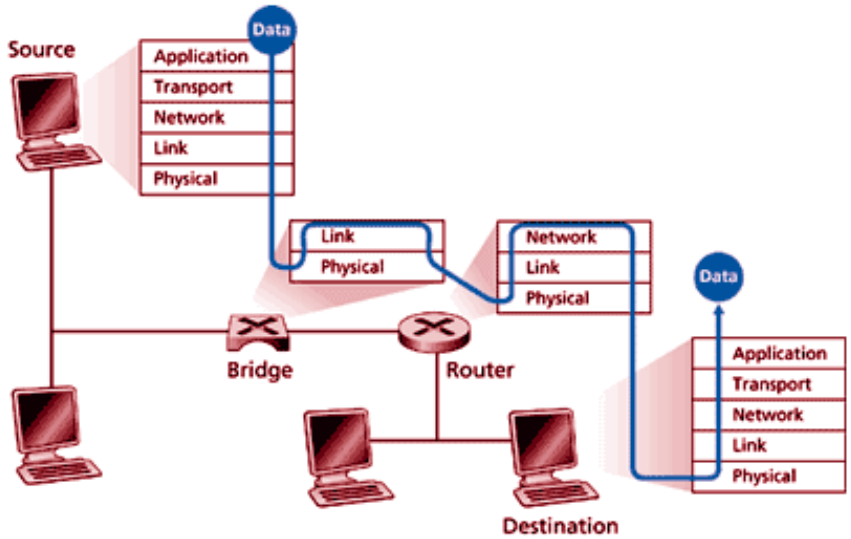


Figure 2: Internet Protocol stack

Communication system

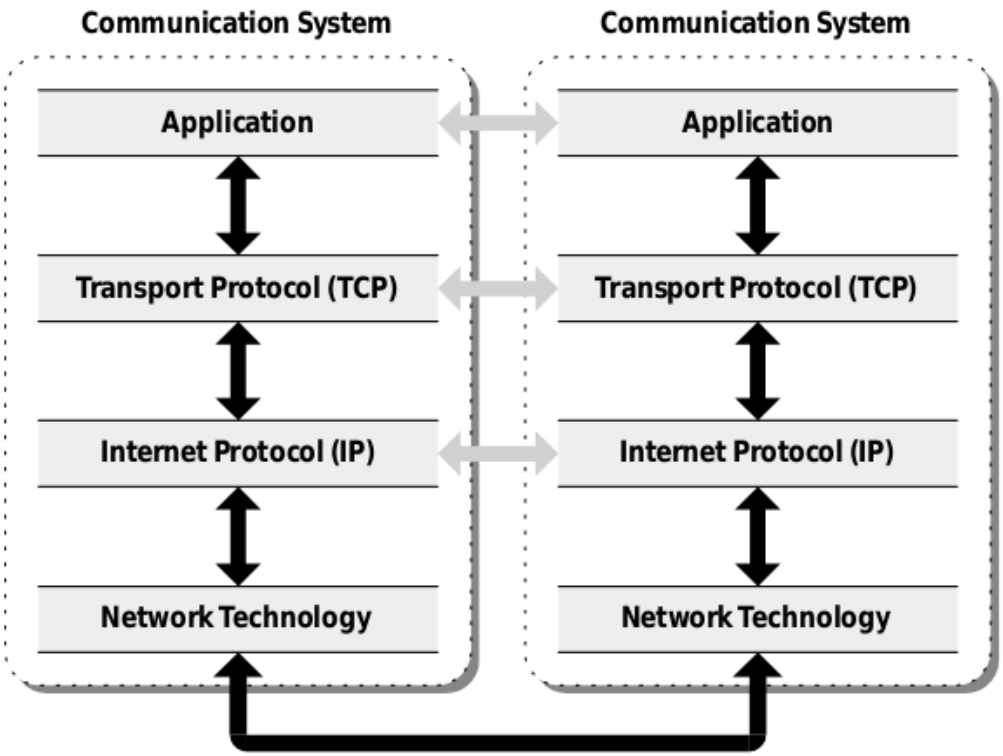


Figure 3: TCP/IP

Protocols

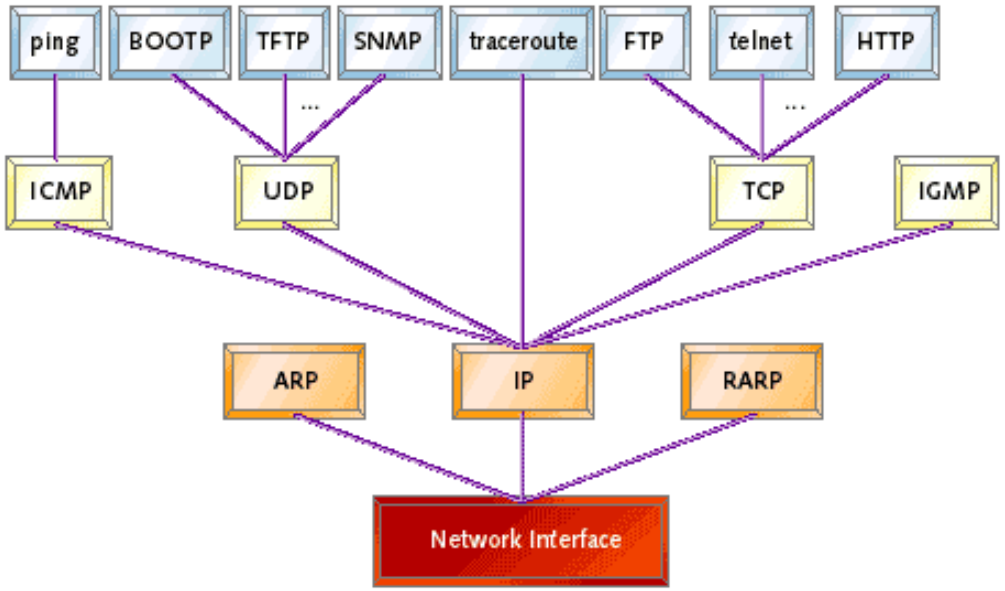
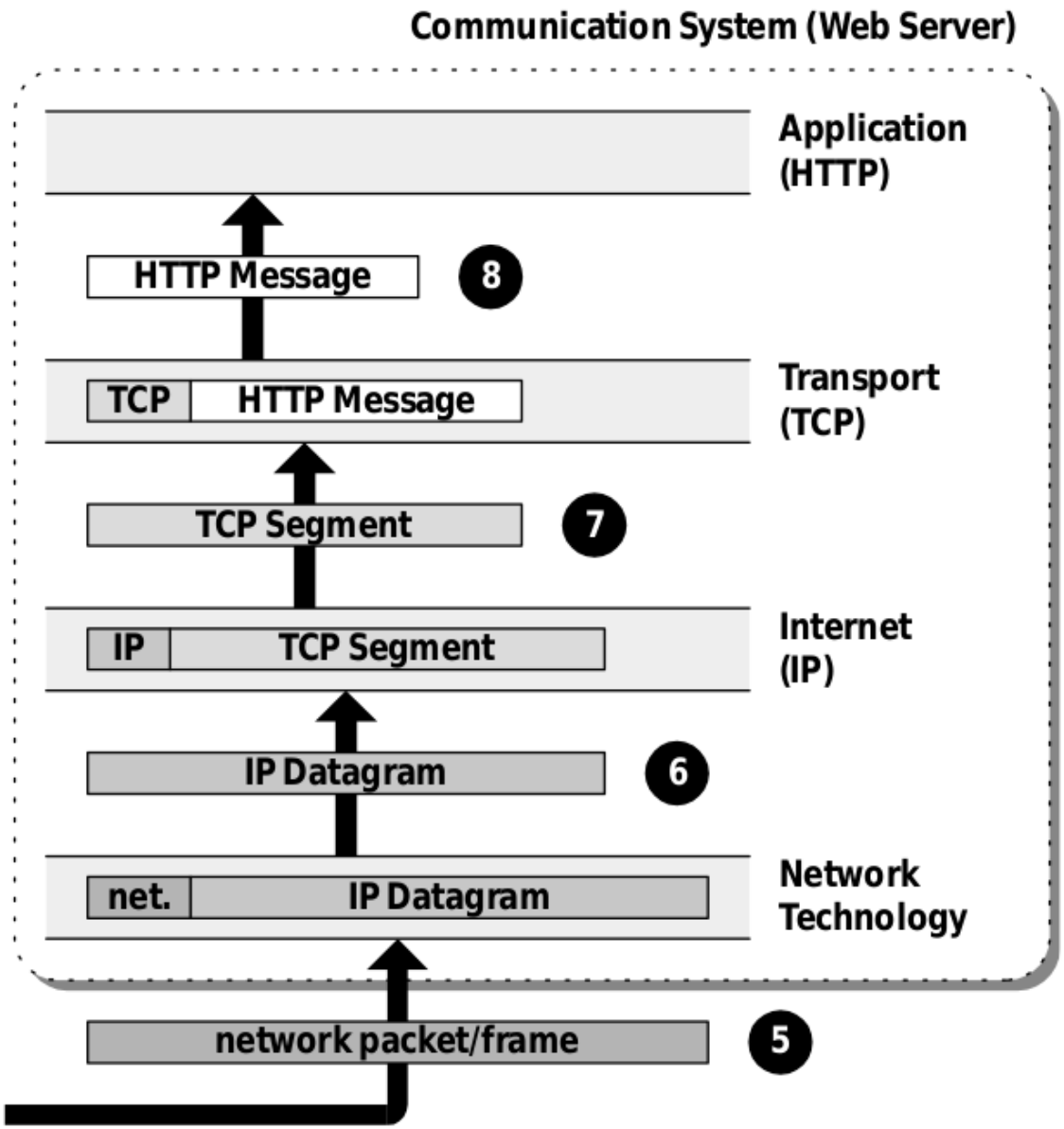


Figure 4: Protocols

webserver


Communication System (Web Server)
Application
(HTTP)
HTTP Message
8
Transport
(TCP)
TCP
HTTP Message
TCP Segment
7
Internet
(IP)
IP
TCP Segment
IP Datagram
6
Network
Technology
net.
IP Datagram
network packet/frame
5

Uniform Resource Identifiers

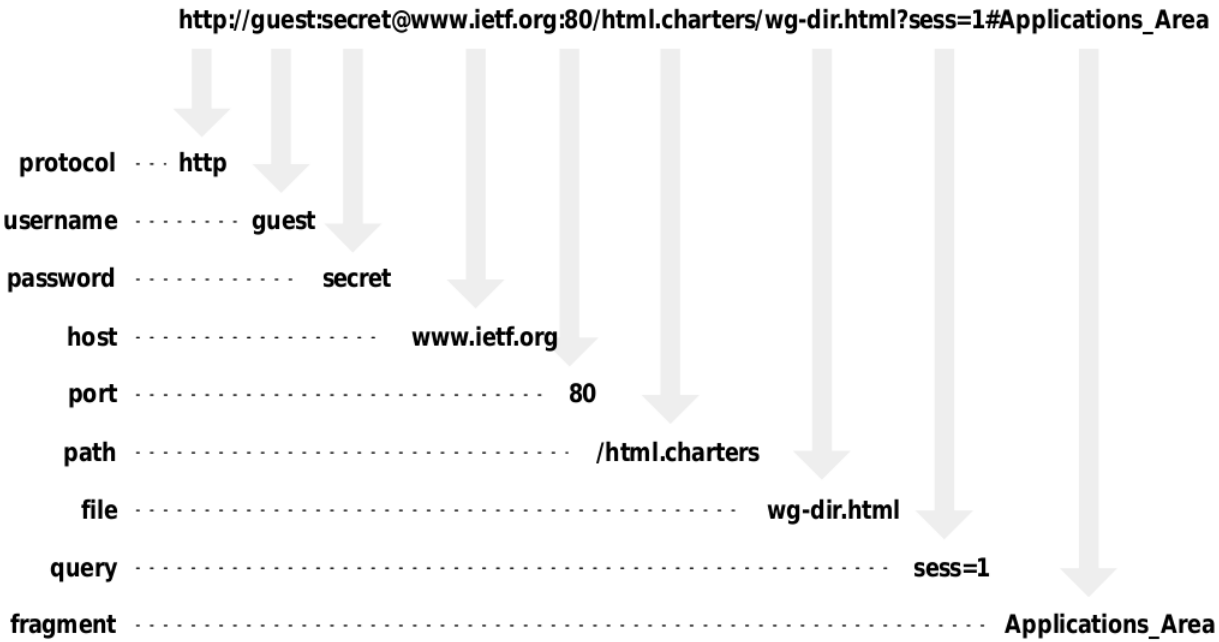


Figure 6: URL

HTTP Operation How Clients and Servers Use HTTP

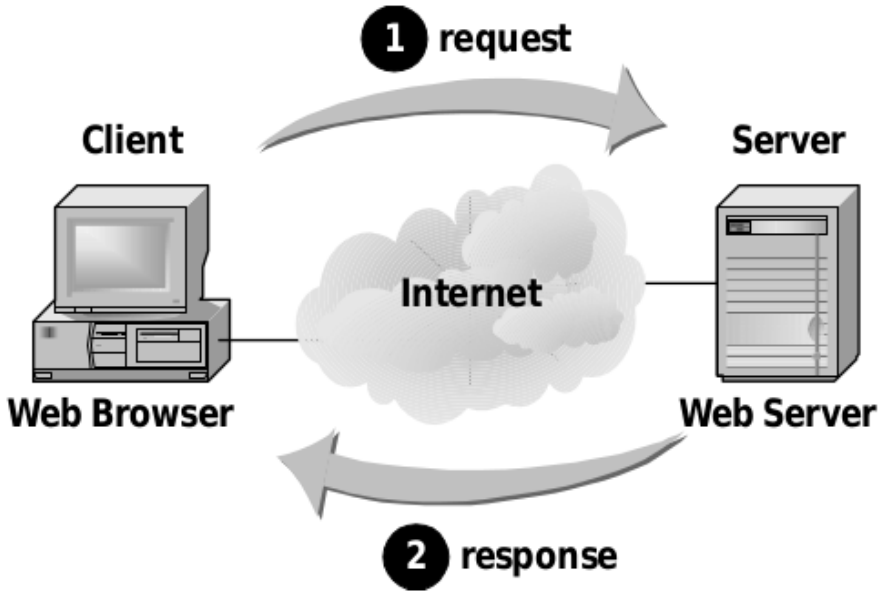


Figure 7: clientserver

HTTP Operation : How Clients and Servers Use HTTP

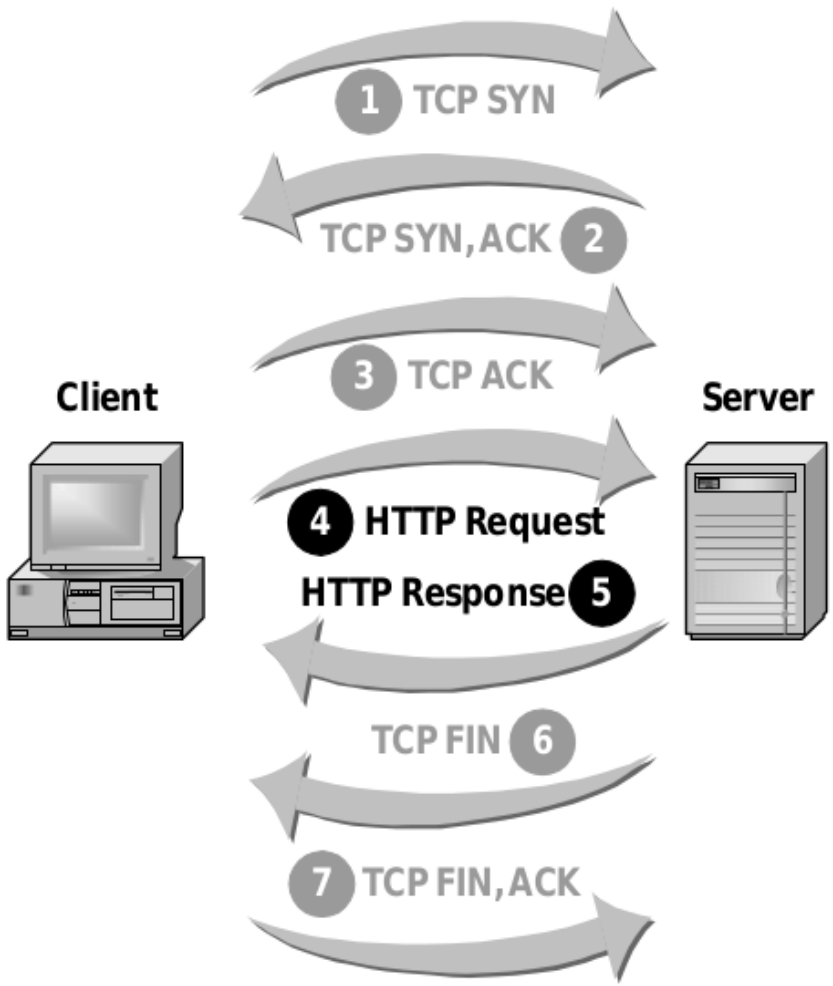


Figure 8: Handshake

HTTP Operation How Clients and Servers Use HTTP

TCP Connections

Figure 2.2 highlights key characteristics of TCP messages. The first message that the client sends has a SYN, for "synchronize," flag. The SYN indicates that the client wishes to establish a connection. The server responds by setting the SYN and ACK (for "acknowledge") flags, indicating its willingness to accept the connection. The client completes the connection establishment by sending a TCP message with only the ACK flag. These three messages are usually called the "three-way handshake." Closing the connection requires only two messages. The first has the FIN (for "finished") flag, and the second has both the FIN and ACK flags set.

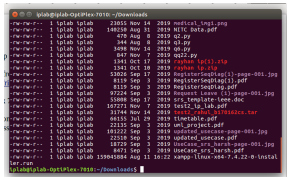
Figure 9: 3 way handshake

Steps for installation of XAMPP on Ubuntu 18.04

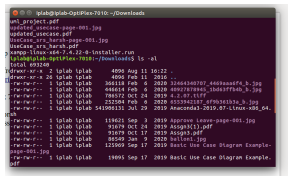
1. Click the XAMPP for Linux link and Save the file.
   `https://www.apachefriends.org/download.html`
2. Make Installation Package Executable To run the installation process, the file permissions require modification. To execute a program, you need to make the file executable. Open the terminal (Ctrl+Alt+T) and follow these instructions to do so:
   1. Move into the folder where your installation package is located. By default, the system stores it in the Downloads folder. Navigate to Downloads using the command:
3. Now make the file executable by running a chmod command:
   sudo chmod +x [package_name]
4. Launch Setup Wizard 1. Now, you can run the installer and launch the graphical setup wizard with the following command:
   sudo ./[package_name]
   2. Install XAMPP 3. Launch XAMPP

Steps for installation of XAMPP on Ubuntu 18.04 (contd..)

1. Verify XAMPP is Running
   Verify localhost is working by entering the following URL in a browser:
   `http://localhost/dashboard`
2. Next, verify whether the MariaDB service is working. To do so, open the URL:
   `http://localhost/phpmyadmin`
3. Next, verify whether the MariaDB service is working. To do so, open the URL:
   `http://localhost/phpmyadmin`



(a) A subfigure



(b) A subfigure

Figure 10: A figure with two subfigures

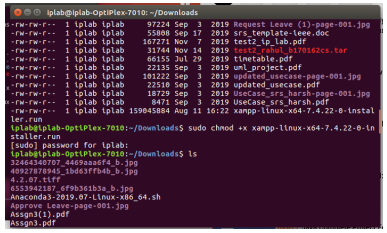


Figure 11: A figure

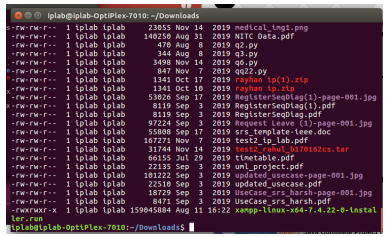
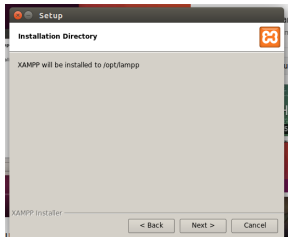
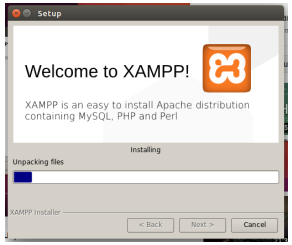


Figure 12: Another figure



(a) A subfigure



(b) A subfigure

Figure 13: A figure with two subfigures

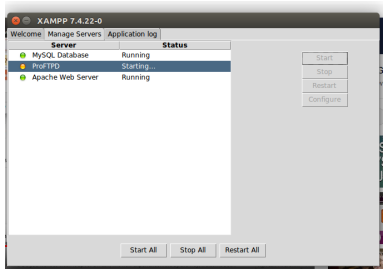


Figure 14: A figure

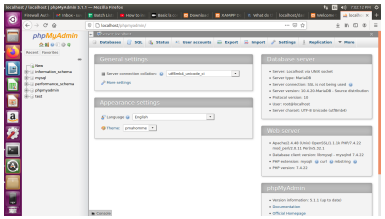
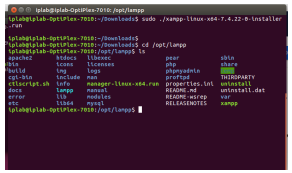
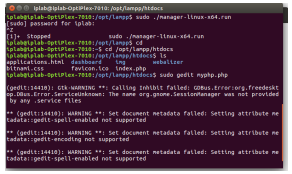


Figure 15: Another figure



(a) A subfigure



(b) A subfigure

Figure 16: A figure with two subfigures

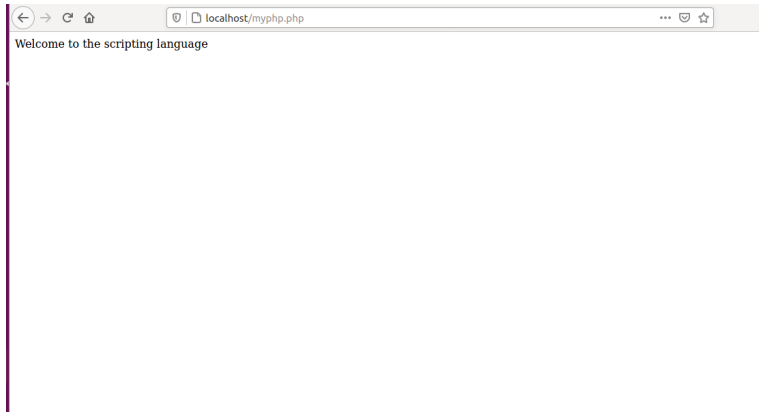
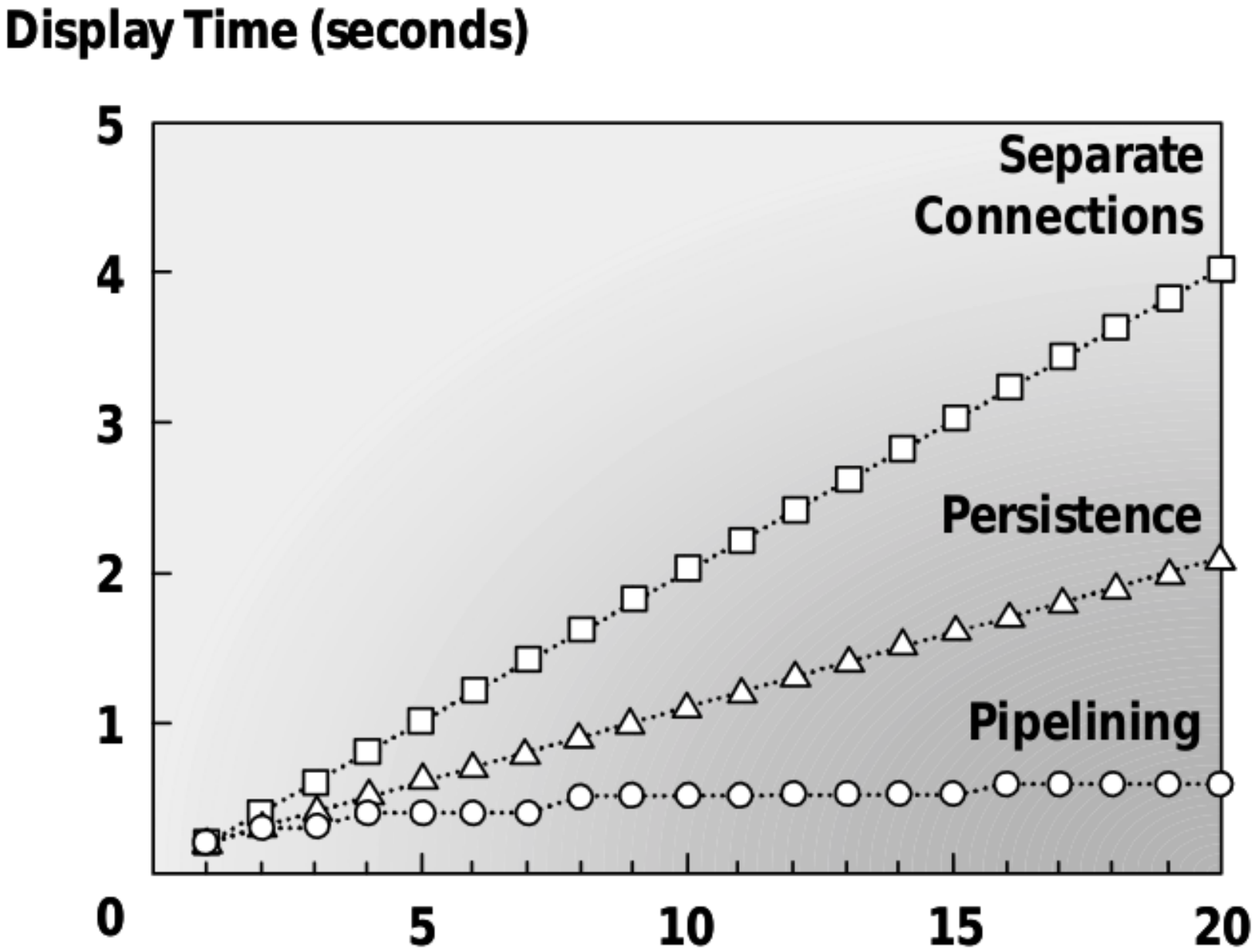
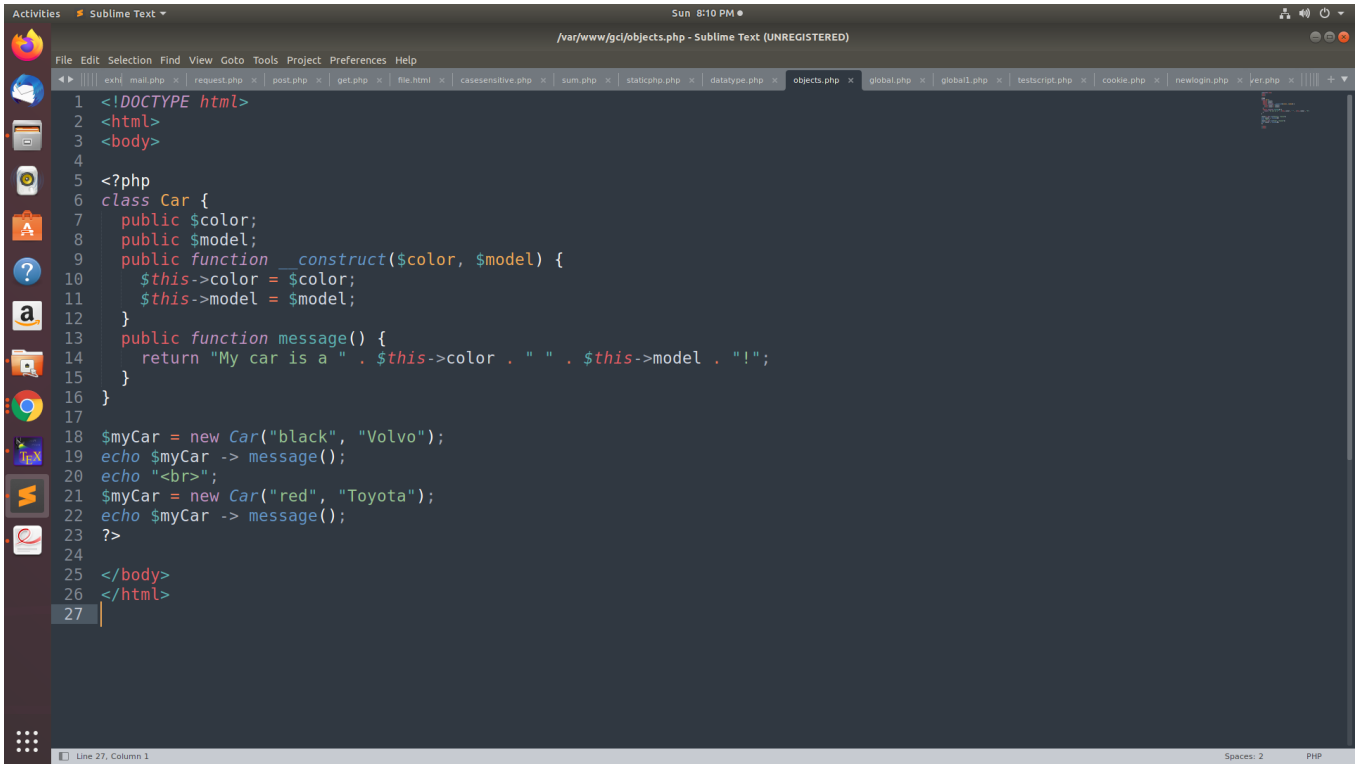


Figure 17: A figure

The first versions of http required clients to establish a separate tcp connection with each request. For simple Web pages, this requirement did not present much of a problem. As Web sites grew more complex and graphic, however, tcp connection establishment began to have a noticeable effect on Web performance. That?s because complex Web pages consist of many separate objects, and the client must issue a separate http request to retrieve each of those objects.

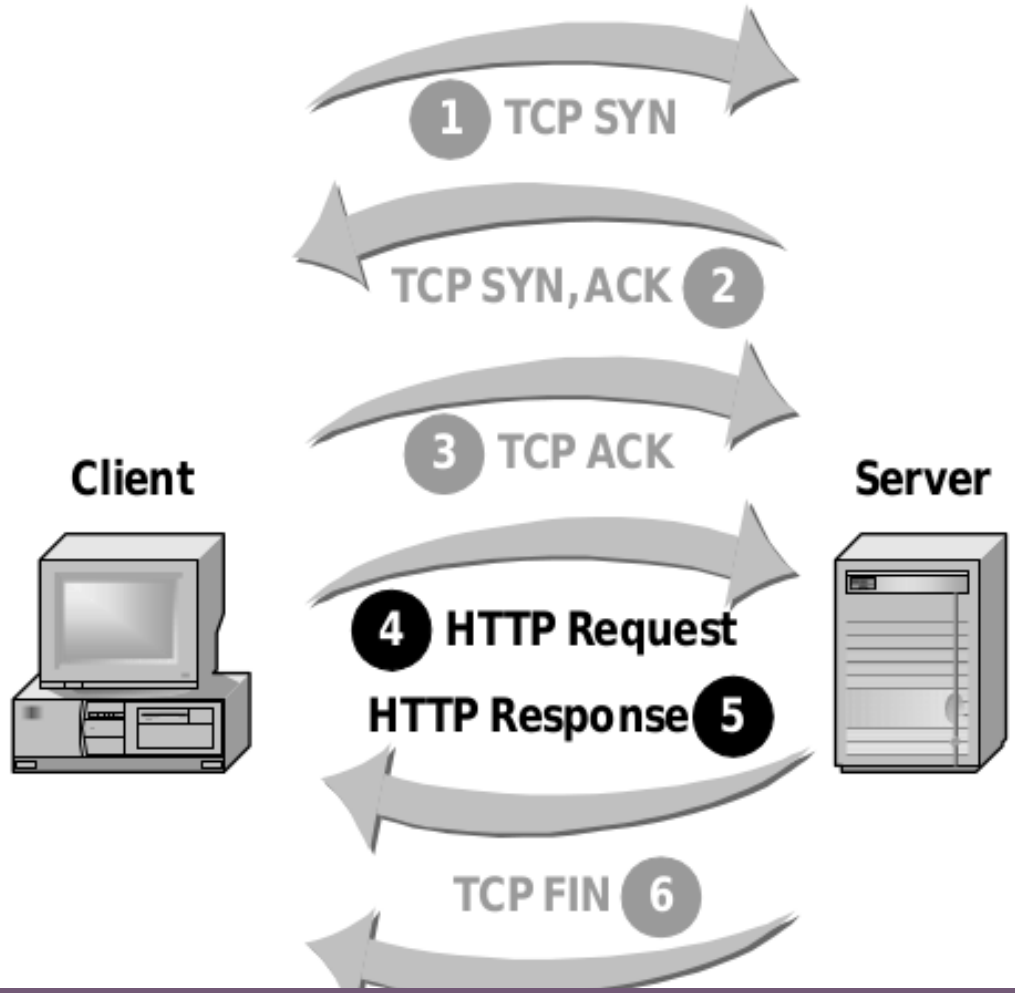

Display Time (seconds)
5
4
3
2
1
0
5
10
15
20
Separate Connections
Persistence
Pipelining

Pipelining


Activities
Sublime Text
Sun 8:10 PM
/var/www/gci/objects.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
mail.php
request.php
post.php
get.php
file.html
casesensitive.php
sum.php
staticphp.php
datatype.php
objects.php
global.php
global1.php
testscript.php
cookie.php
newlogin.php
<!DOCTYPE html>
<html>
<body>
<?php
class Car {
public $color;
public $model;
public function __construct($color, $model) {
$this->color = $color;
$this->model = $model;
}
public function message() {
return "My car is a " . $this->color . " " . $this->model . "!";
}
}
$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "
";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
</body>
</html>
Line 27, Column 1
Spaces: 2
PHP

Pipelining

With pipelining, a client does not have to wait for a response to one request before issuing a new request on the connection. It can follow the first request immediately with a second request.



User Operations

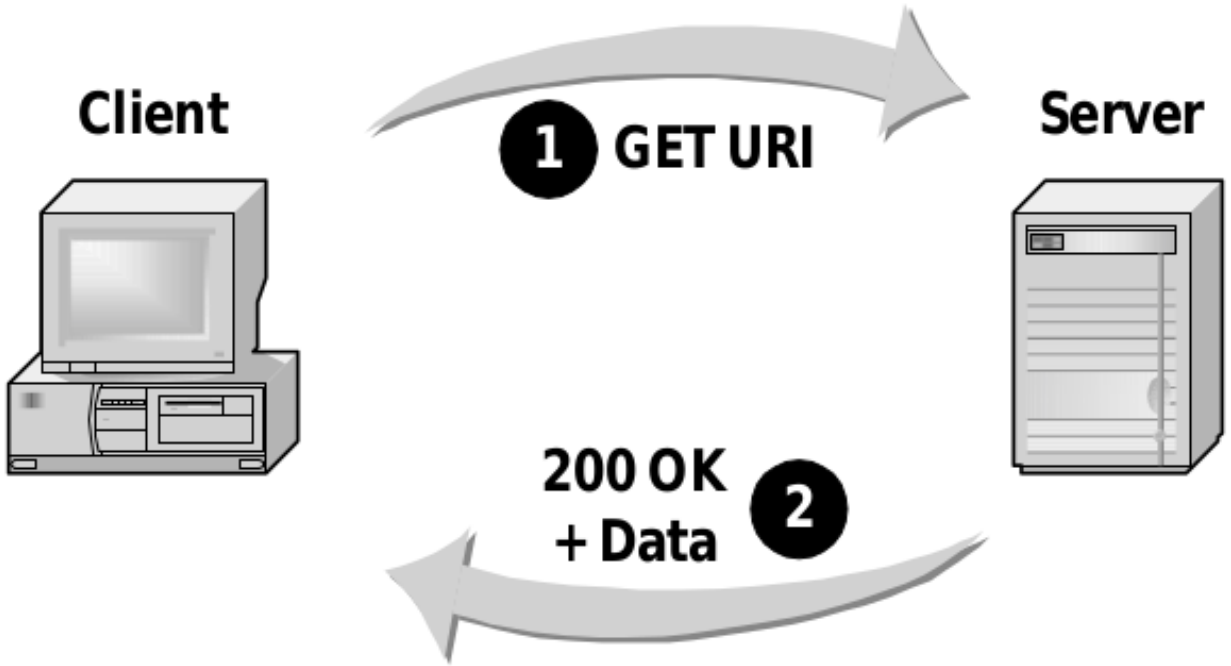
The http protocol defines four basic operations

1. GET
2. POST
3. PUT
4. DELETE

, We consider these to be user operations because, at least in the context of Web browsing, they are each the direct result of user actions.

Web Page Retrieval: GET

GET is a simple two-message exchange. The client initiates it by sending a GET message to the server. The message identifies the object the client is requesting with a Uniform Resource Identifier (uri). If the server can return the requested object, it does so in its response. As the figure shows, the server indicates success with an appropriate status; 200 OK is the status code for a successful response.



Capabilities - OPTIONS

Clients can use an OPTIONS message to discover what capabilities a server supports. If the client includes a uri, the server responds with the options relevant to that object. If the client sends an asterisk $(*)$ as the uri, the server returns the general options that apply to all objects it maintains.

A client might use the OPTIONS message to determine the version of http that the server supports or, in the case of a specific uri, which encoding methods the server can provide for the object.

Status - HEAD

Clients can use a HEAD message when they want to verify that an object exists, but they don't need to actually retrieve the object. Programs that verify links in Web pages, for example, can use the HEAD message to ensure that a link refers to a valid object without consuming the network bandwidth and server resources that a full retrieval would require

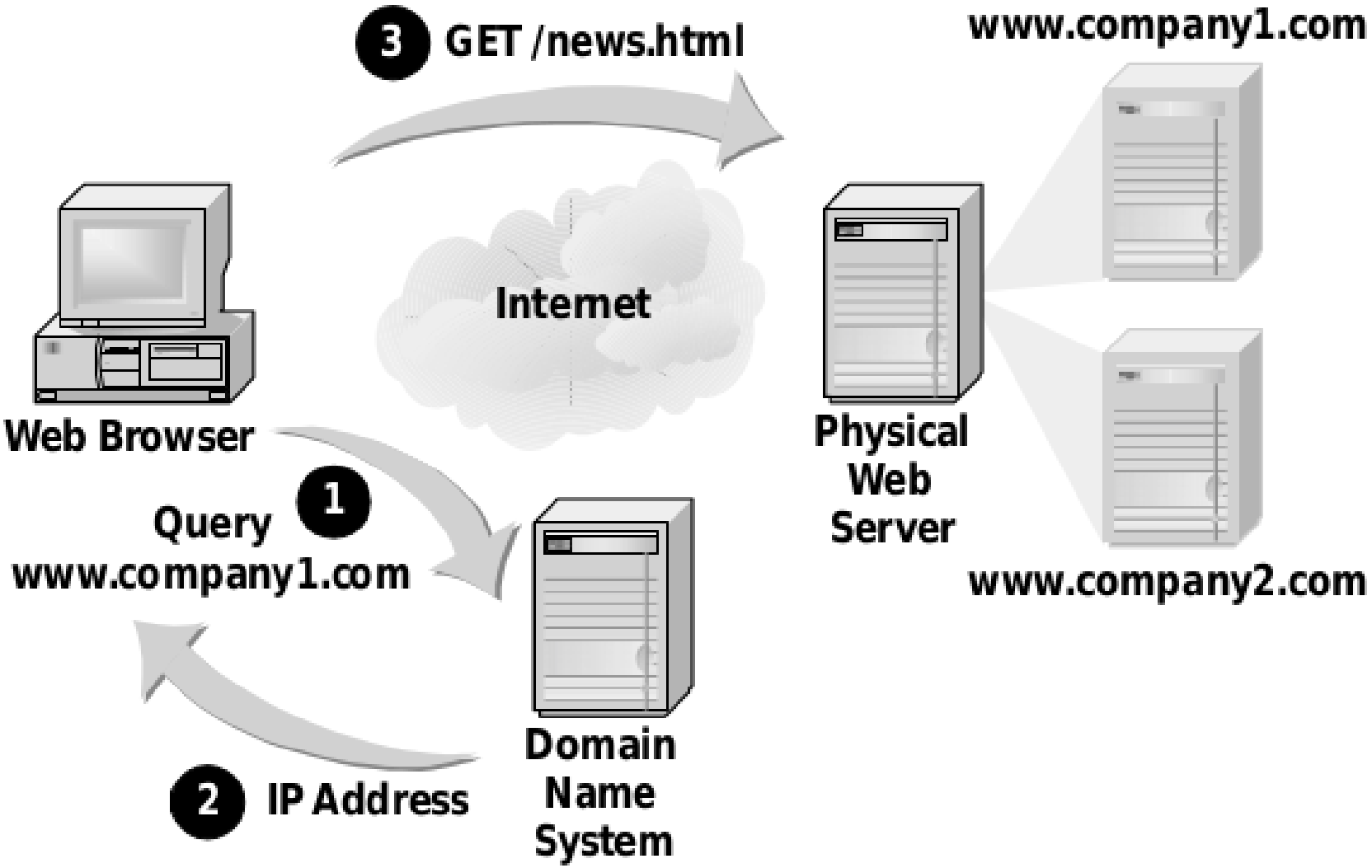
Path-Trace

The TRACE message gives clients a way to check the network path to a server. When a server receives a TRACE, it responds simply by copying the TRACE message itself into the data for the response. TRACE messages are more useful when multiple servers are involved in responding to a request An intermediate server, for example, may accept requests from clients but turn around and forward those requests onto additional servers (Proxies and cache servers).

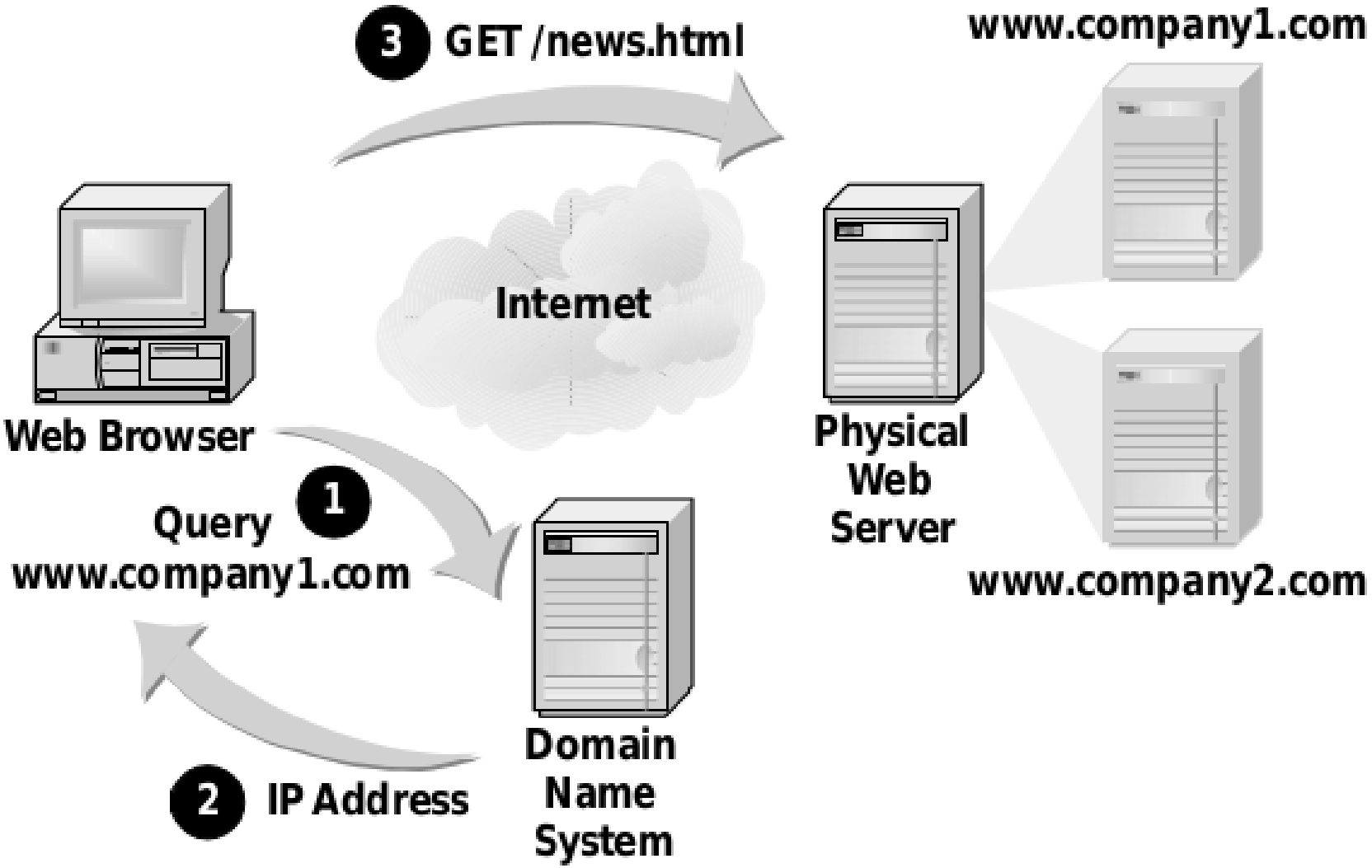
Cooperating Servers

The http protocol defines more complex interactions, however, that frequently involve multiple servers cooperating on a client's behalf.

Virtual Host

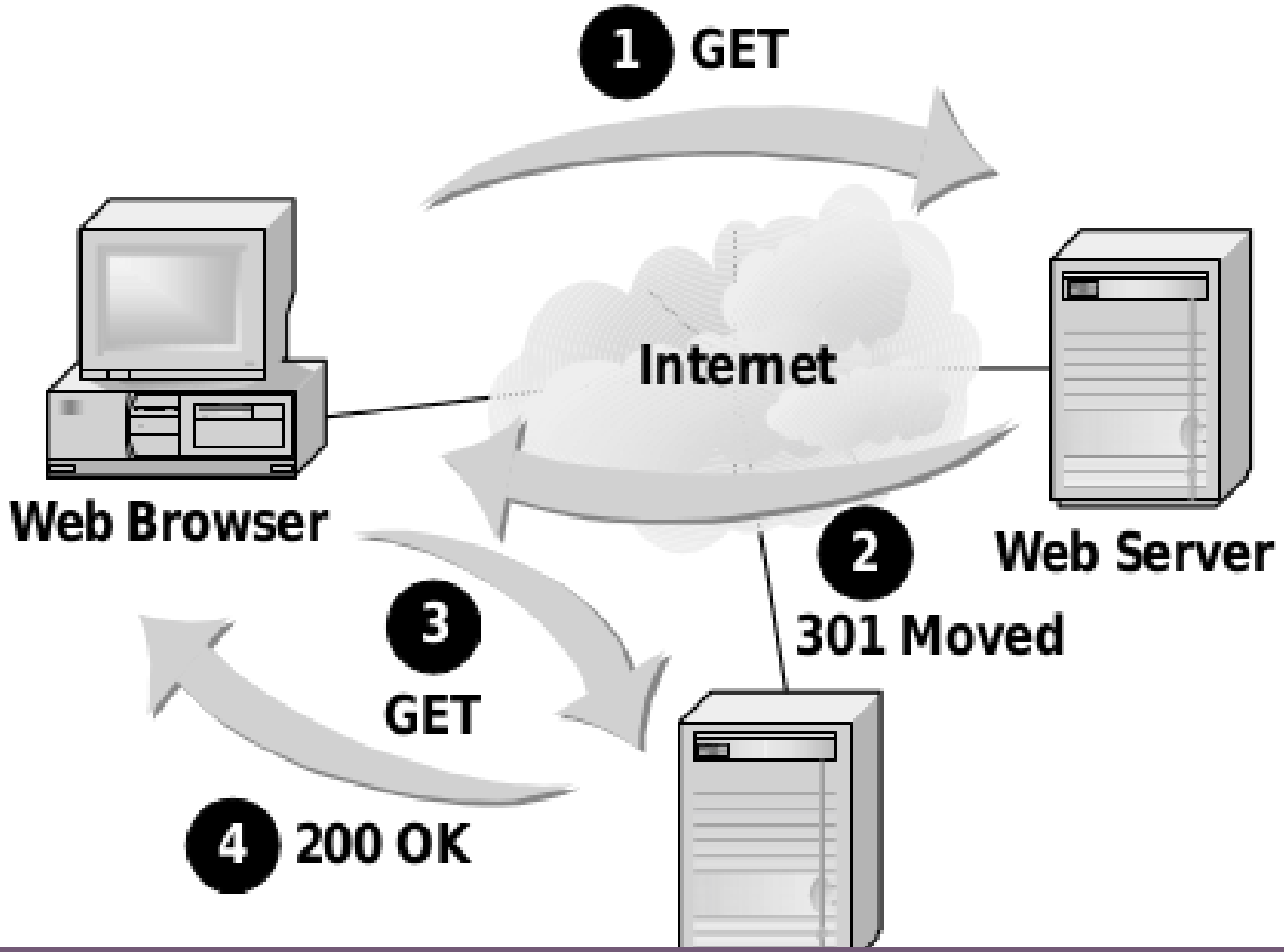


Virtual Host



Redirection

While virtual host support allows a single server to support multiple Web sites easily, redirection offers a way to support a single site to use multiple servers. Redirection lets a server redirect a client to another uri for an object.



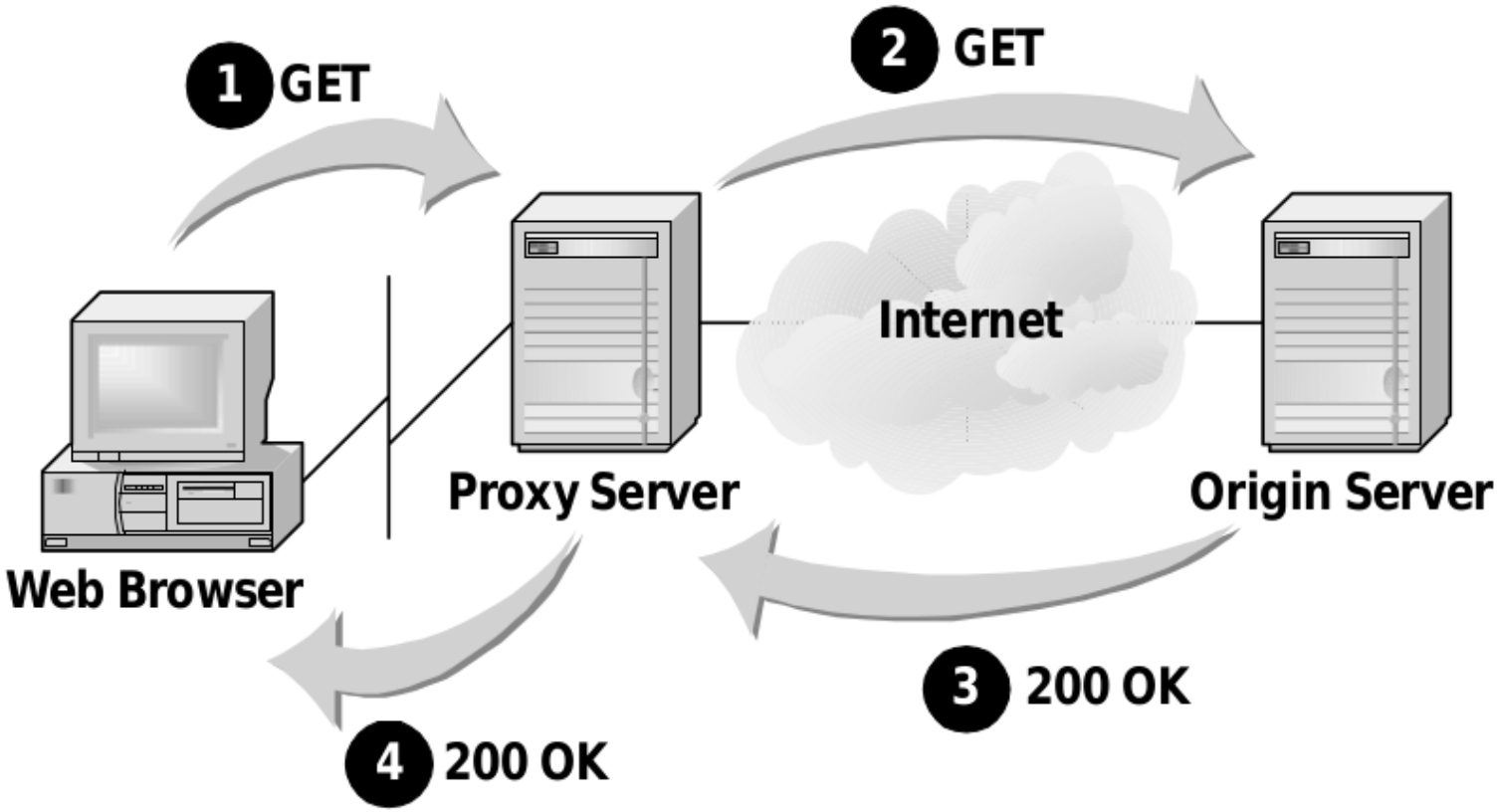


Figure 18: Proxy server

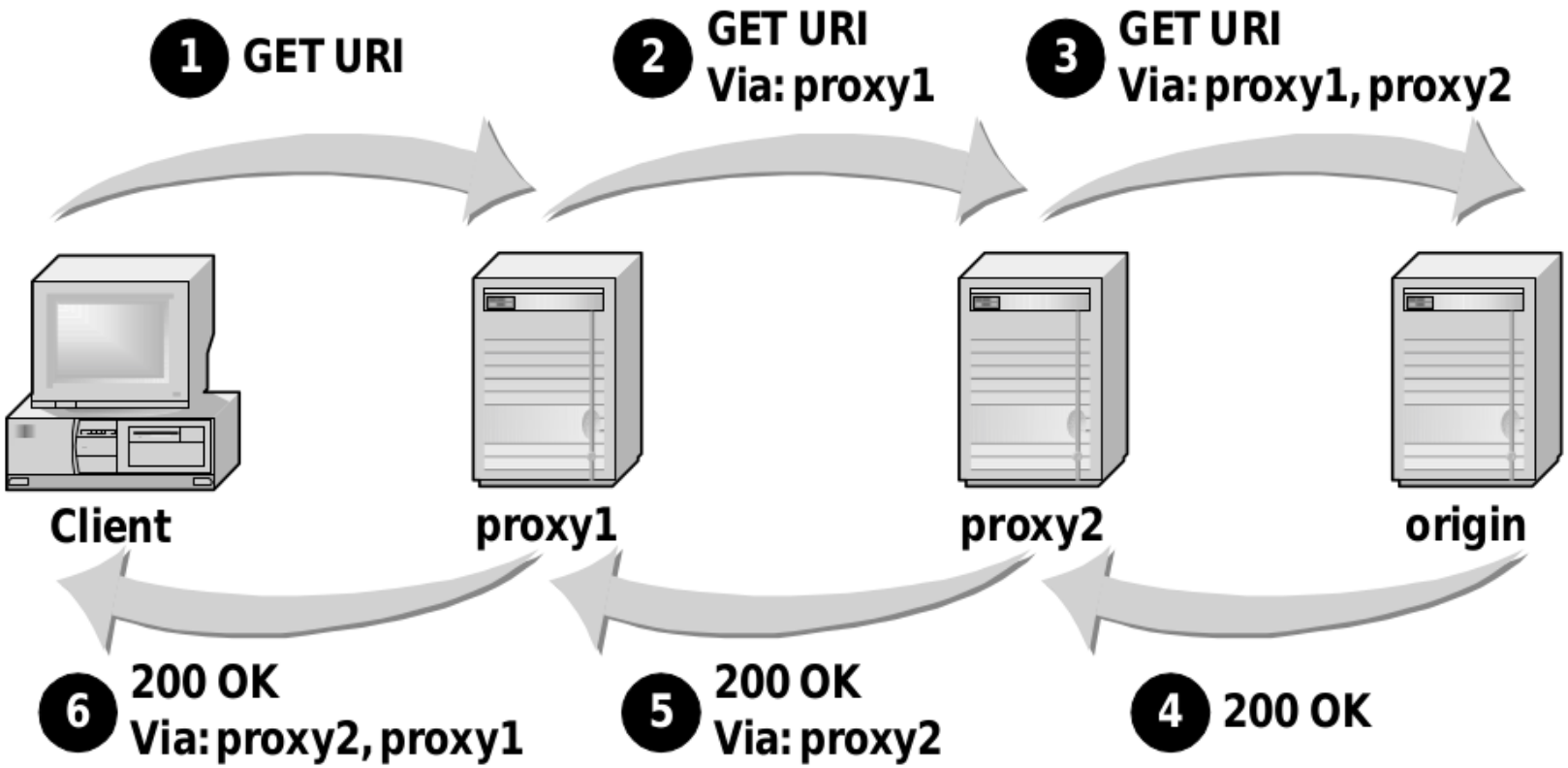


Figure 19: Proxy server

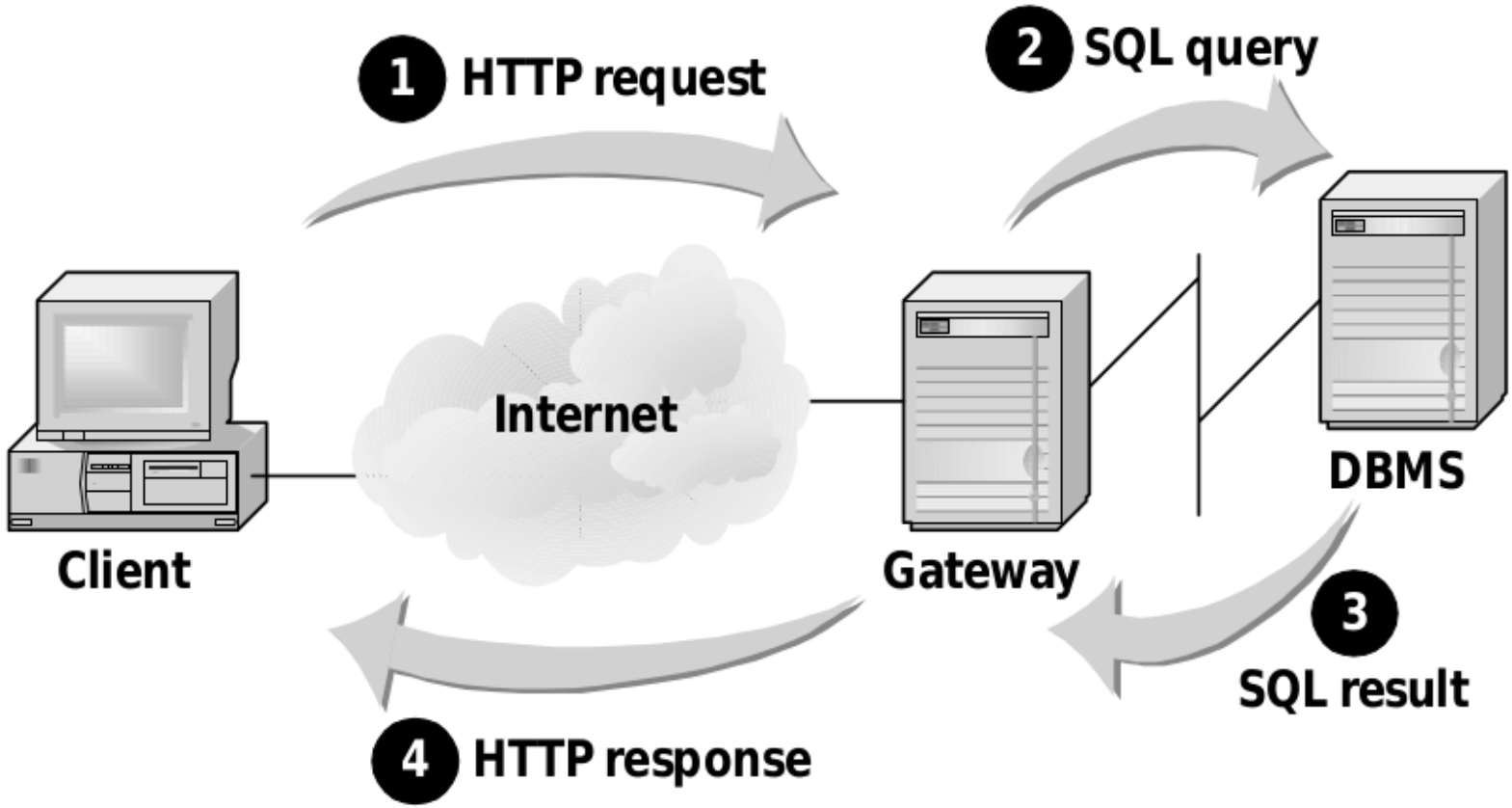
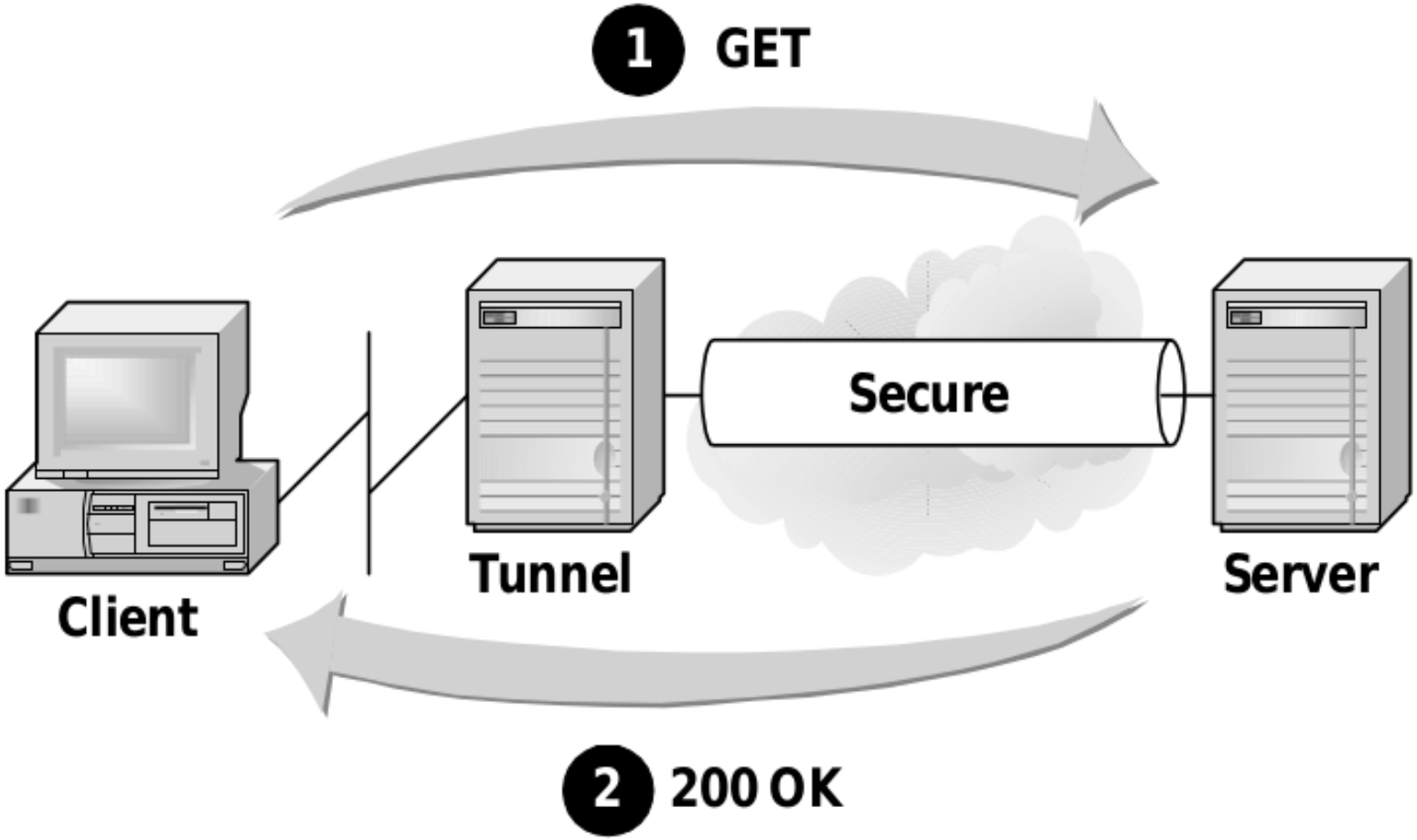
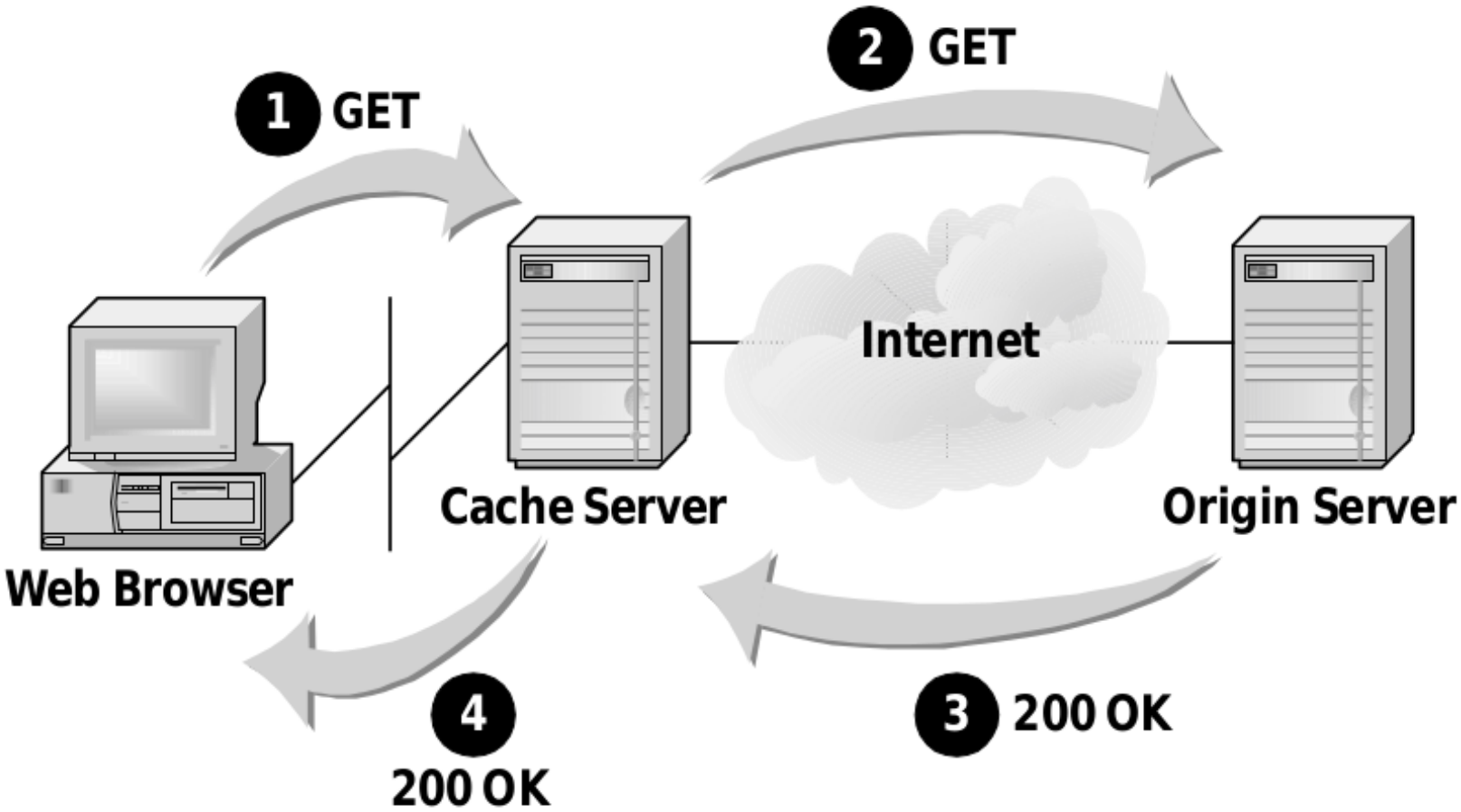


Figure 20: gateway

Proxies and tunnels



Impact of cache servers on web performance



Impact of cache servers on web performance

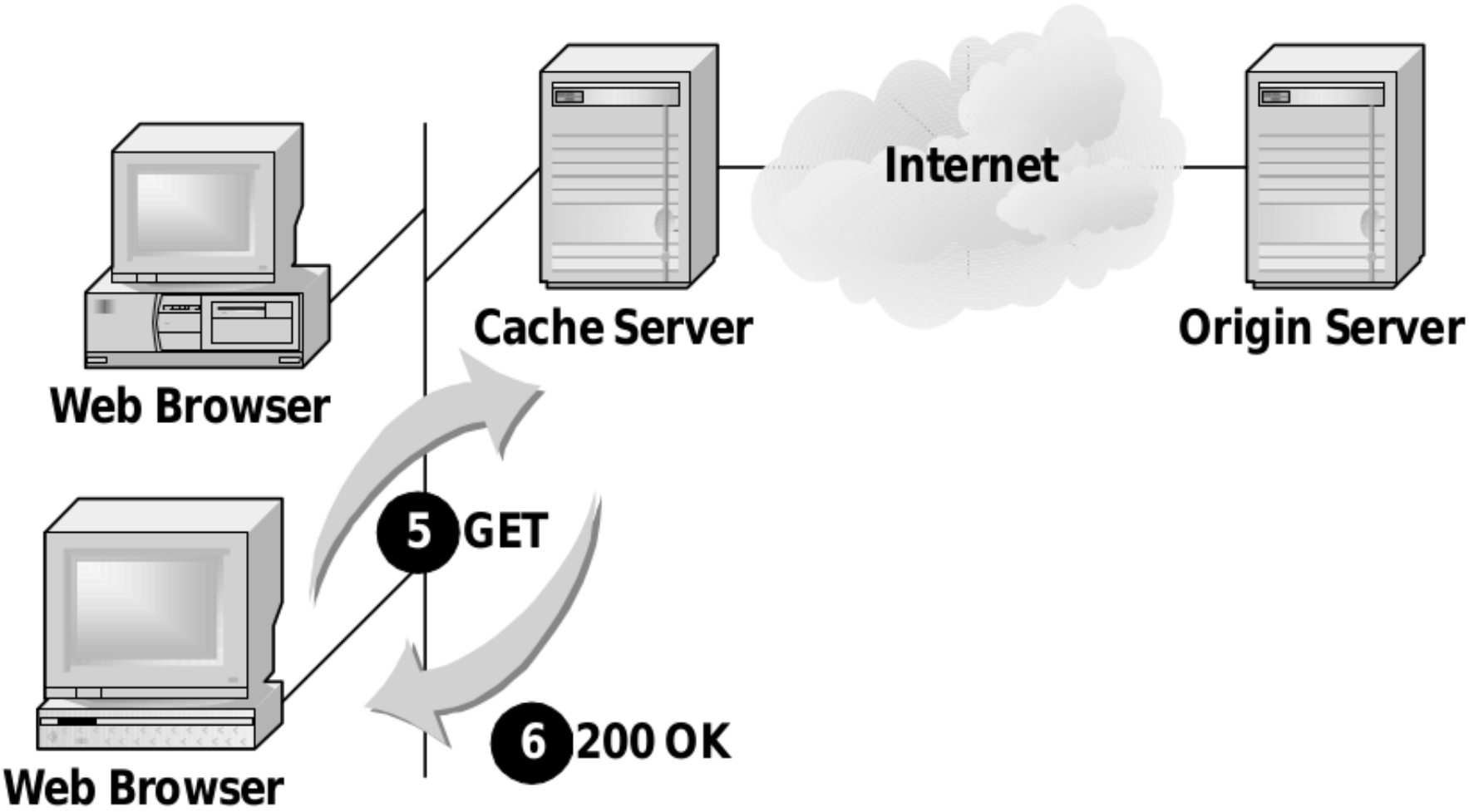
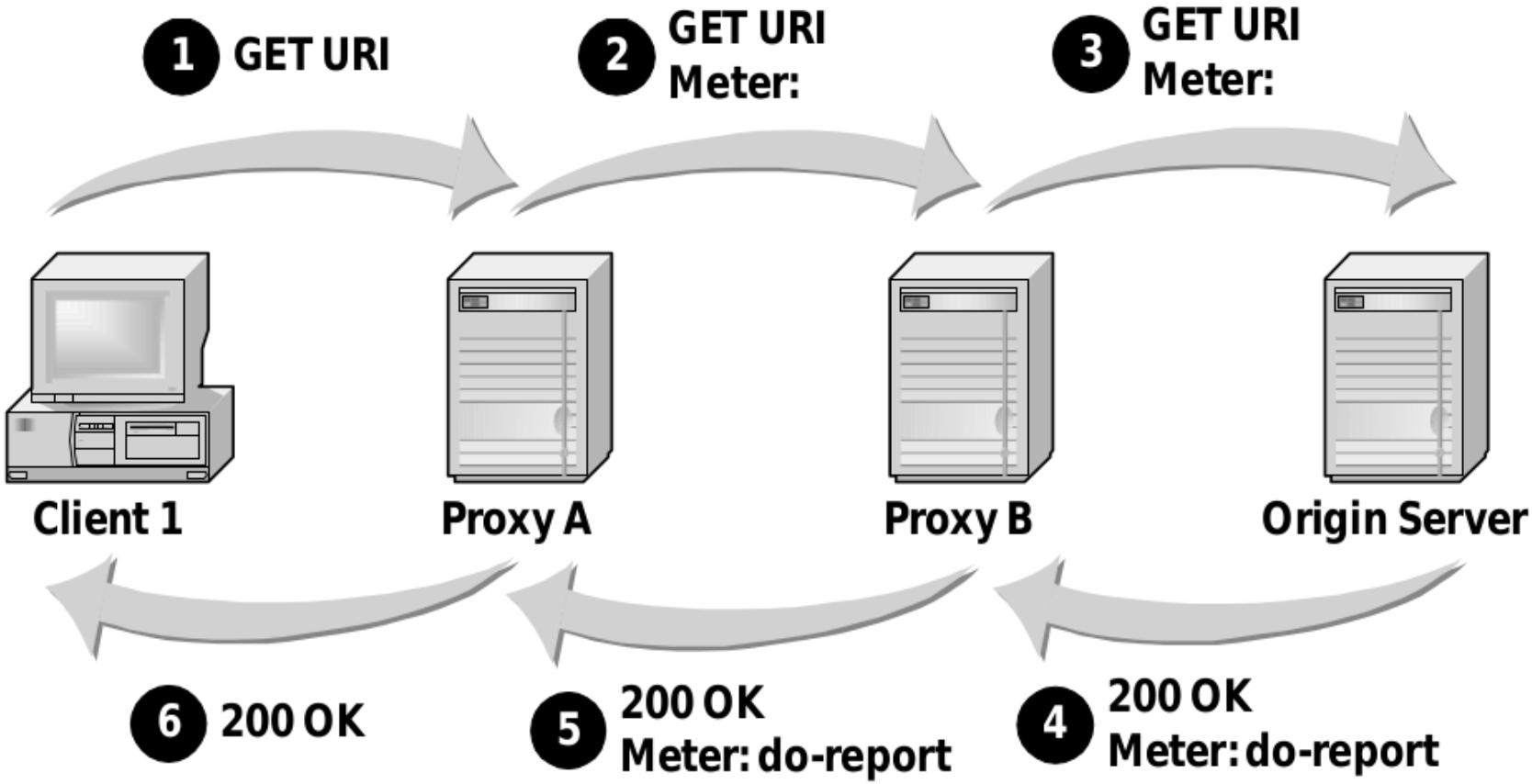
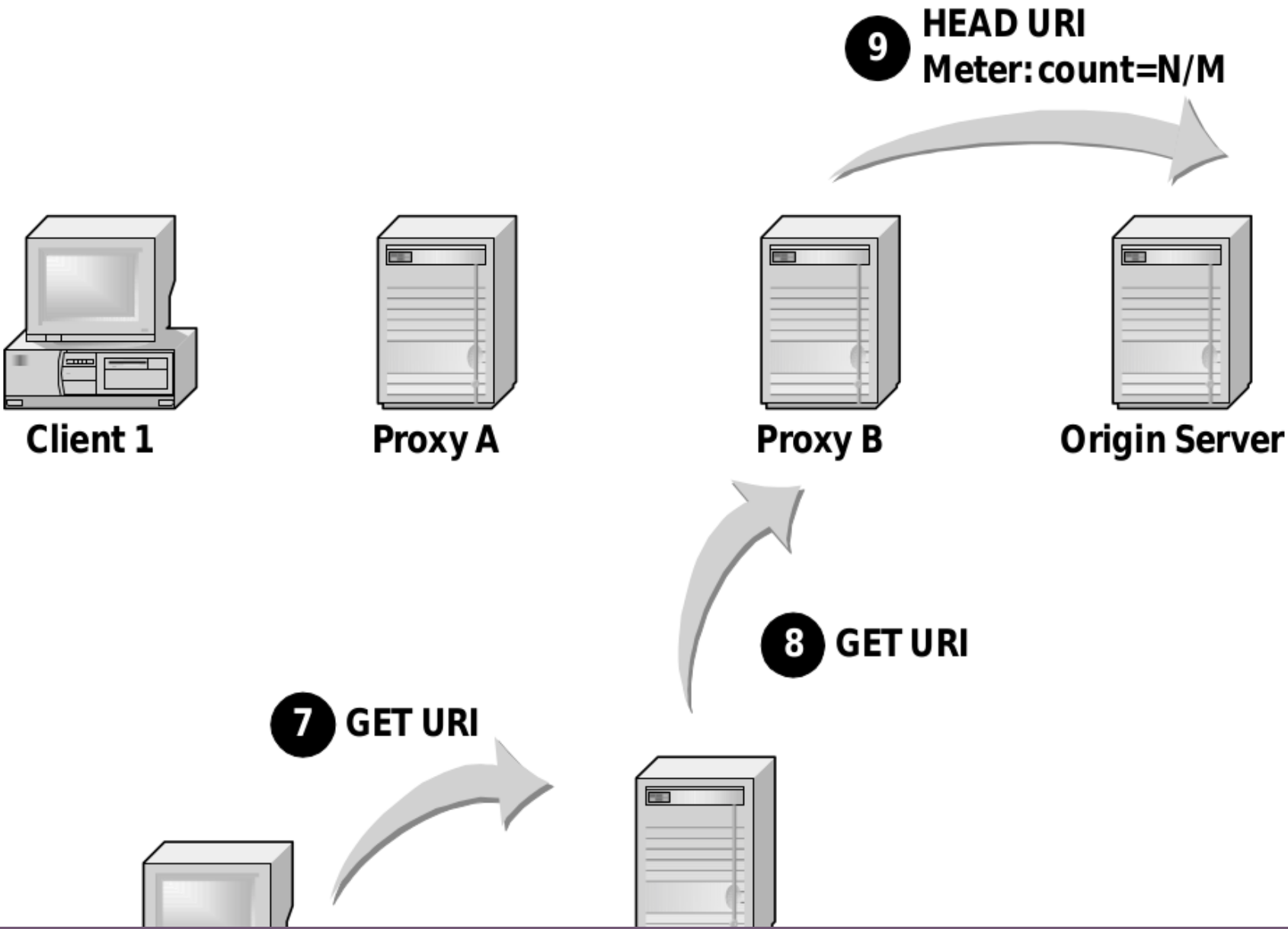


Figure 23: cache servers

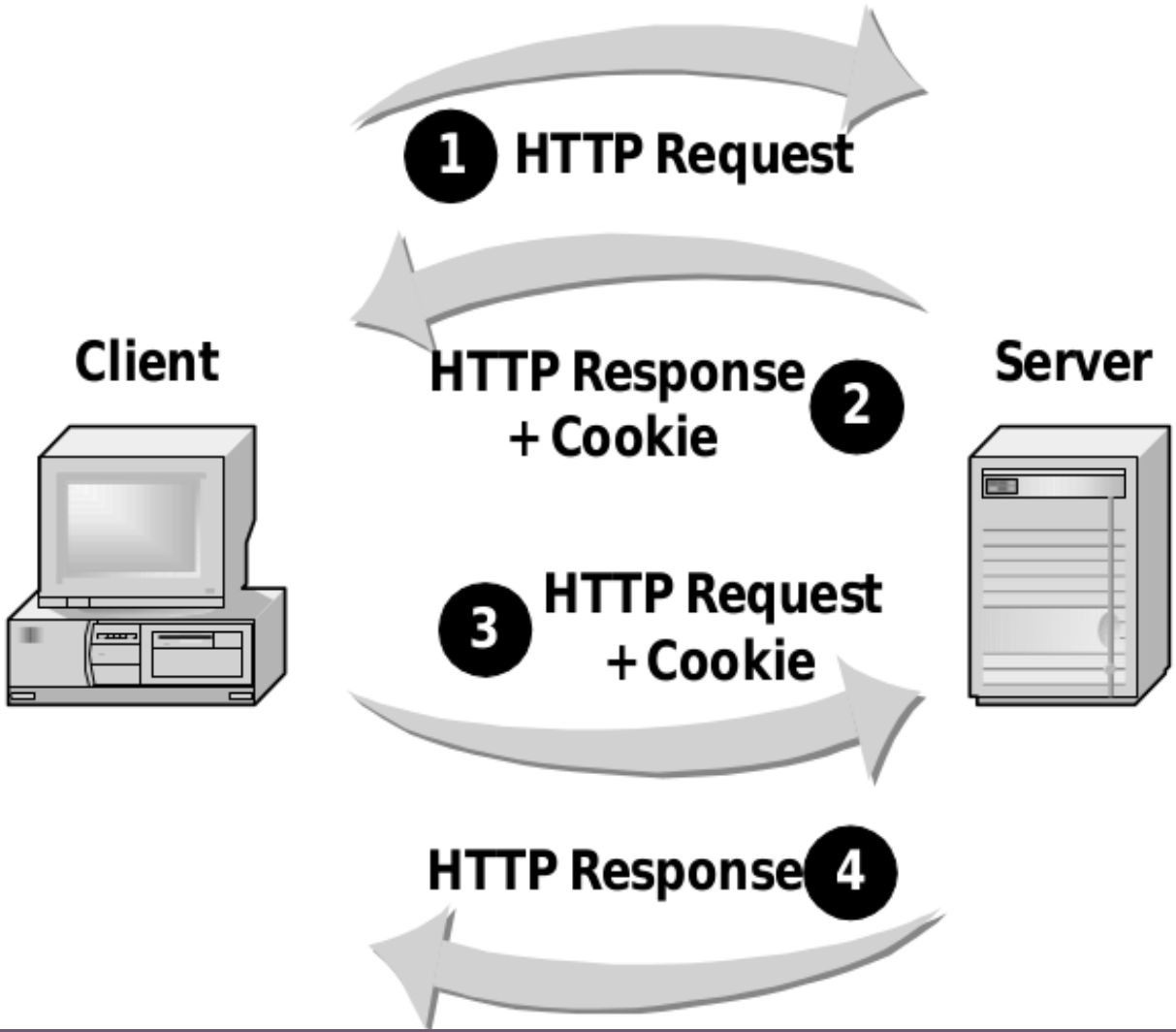
Counting and Limiting Page Views



Meter count


9 HEAD URI
Meter: count=N/M
Client 1
Proxy A
Proxy B
Origin Server
8 GET URI
7 GET URI

Cookies


1 HTTP Request
Client
HTTP Response
+ Cookie
2
Server
3 HTTP Request
+ Cookie
HTTP Response
4

Cookies

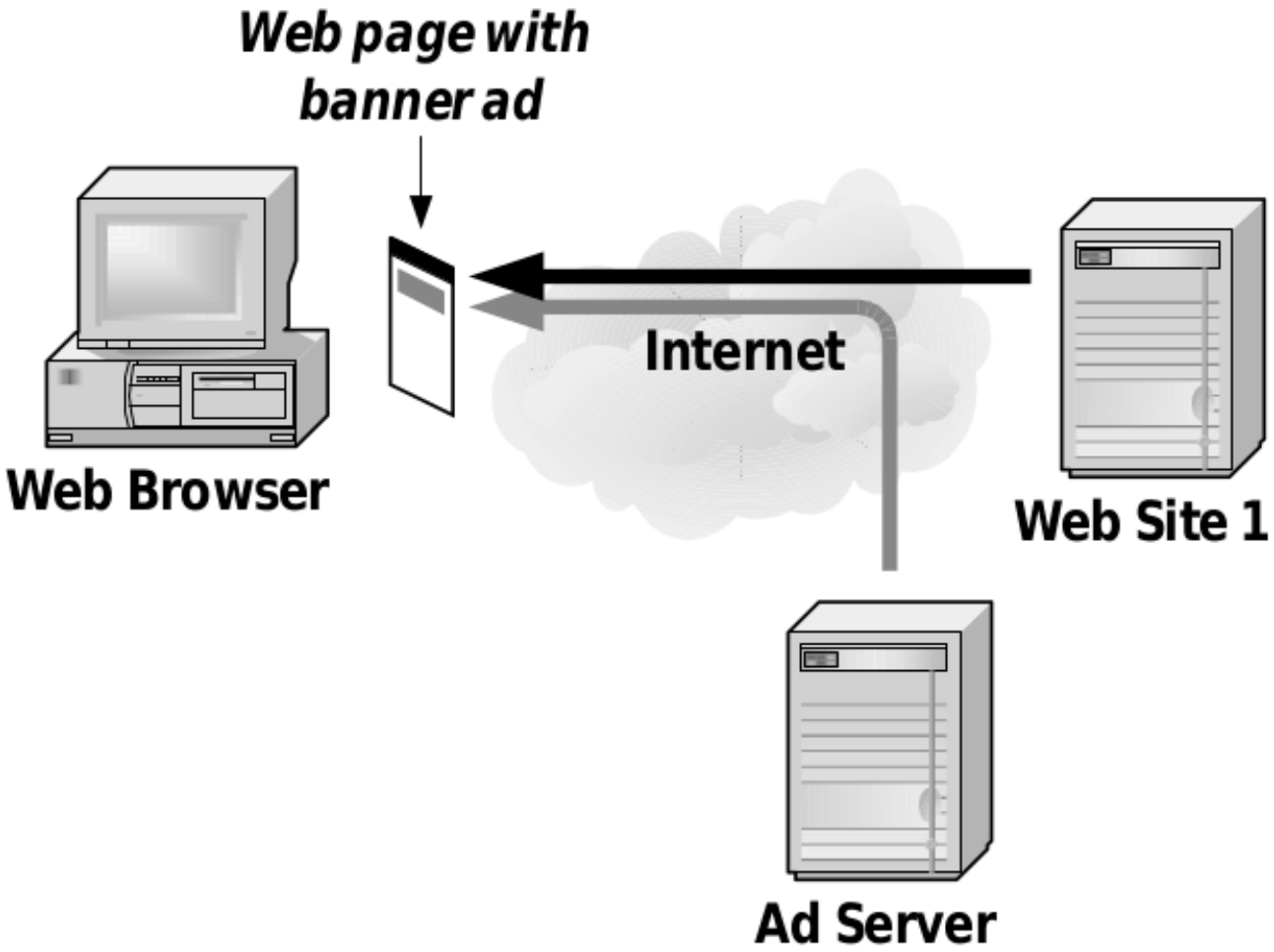

Web page with
banner ad
Web Browser
Internet
Web Site 1
Ad Server

Table 2.1 Cookie Attributes (continued)

Attribute	Status	Notes
Domain	Optional	The domain (from the Domain Name System) for which the cookie is valid; a server may not specify a domain other than one to which itself belongs, but it may specify a domain more general than a single server.
Max-Age	Optional	The lifetime of the cookie, in seconds.
Path	Optional	The URLs on the server to which the cookie applies.
Port	Optional	A list of TCP ports for which the cookie applies.
Secure	Optional	Instructs the client to only return the cookie in subsequent requests if those requests are secure; it may be used for cookies that should not be exposed to eavesdroppers. Note, however, that HTTP does not specify what "secure" means in this context.

Table 2.2 continued

Attribute	Default Value if Missing
Max-Age	Keep the cookie only as long as the current user session is active (e.g., do not store the cookie on the user's hard disk).
Path	The URL for which the cookie was originally returned, up to, but not including, the file specified by that URL.
Port	The cookie applies to any ports. (Note that if the `Port` attribute is present in the cookie but has no value, then the client sets the value of the attribute to the port of its original request.)
Secure	The cookie may be returned with insecure requests.

Figure 29: cookies table

Table 2.3 Rules for Rejecting Cookies

Conditions Under Which a Client Rejects a Cookie
• The value of the `Path` attribute is not a prefix of the URL in the client's request.
• The value for the `Domain` attribute does not have any dots within it (not just at the beginning), unless the value is ".local".
• The server that returned the cookie does not belong to the domain specified by the `Domain` attribute.
• The host part of the `Domain` attribute, if present, contains a dot within it.
• The port of the client's request is not included in the `Port` attribute (unless the `Port` attribute is absent).

Figure 30: cookies table

Conditions Under Which a Client Returns a Cookie

- The domain name for the new request must belong to the domain specified by the cookie's `Domain` attribute.
- The port for the new request must be included in the list of ports of the cookie's `Port` attribute, unless the `Port` attribute was absent from the cookie (indicating all ports).
- The path for the new request must match the cookie's `Path` attribute, or represent a child of the `Path` attribute.
- The cookie must not have expired, as per its `Max-Age` attribute.

Figure 31: cookies table

MIME

Multipurpose Internet Mail Extensions. Internet hosts many thousands of different data types, HTTP carefully tags each object being transported through the Web with a data format label called a MIME type. MIME (Multipurpose Internet Mail Extensions) was originally designed to solve problems encountered in moving messages between different electronic mail systems. MIME worked so well for email that HTTP adopted it to describe and label its own multimedia content.

Figure 1-3. MIME types are sent back with the data content

A MIME type is a textual label, represented as a primary object type and a specific subtype, separated by a slash. For example:

- An HTML-formatted text document would be labeled with type `text/html`.
- A plain ASCII text document would be labeled with type `text/plain`.
- A JPEG version of an image would be `image/jpeg`.
- A GIF-format image would be `image/gif`.
- An Apple QuickTime movie would be `video/quicktime`.
- A Microsoft PowerPoint presentation would be `application/vnd.ms-powerpoint`.

Figure 32: Mime types

Figure 33: URI

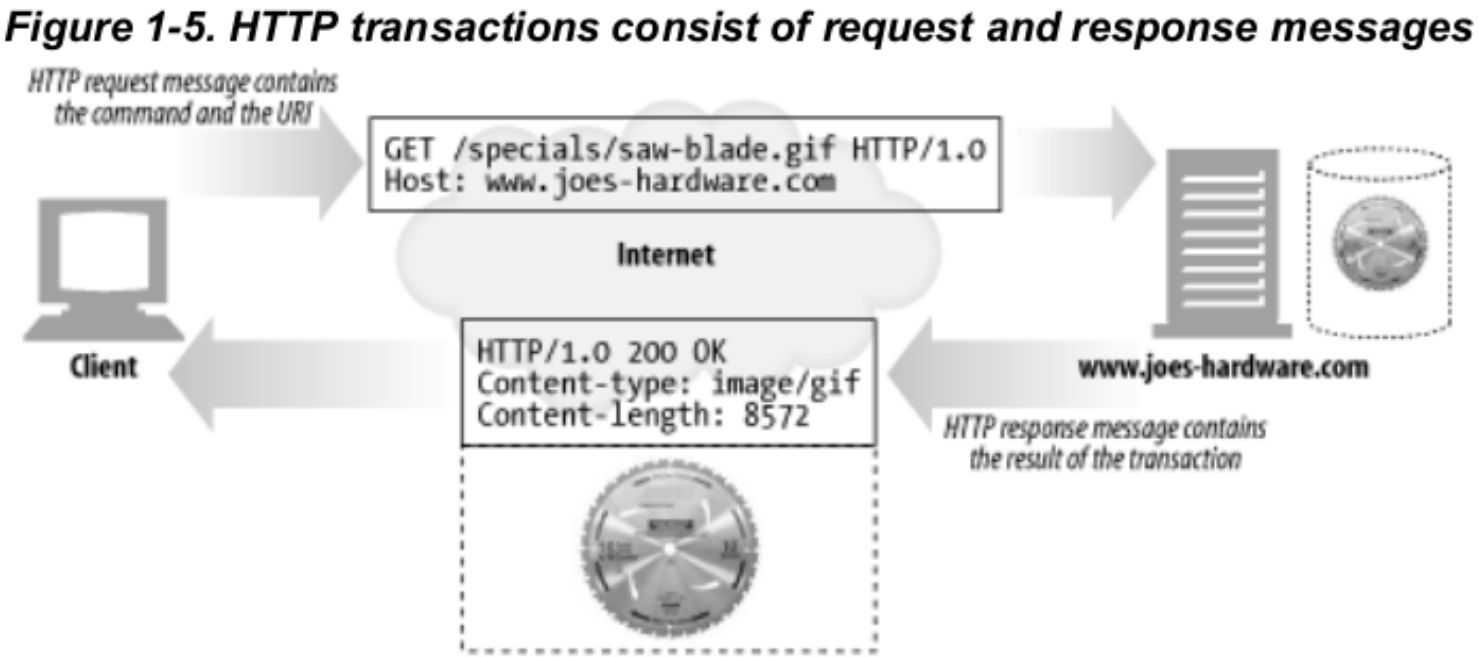


Figure 34: HTTP Transactions

HTTP Message Types

HTTP messages consist of three parts

1. **Start line**: The first line of the message is the start line, indicating what to do for a request or what happened for a response.
2. **Header fields**: Zero or more header fields follow the start line. Each header field consists of a name and a value, separated by a colon (:) for easy parsing. The headers end with a blank line. Adding a header field is as easy as adding another line.
3. **Body**: After the blank line is an optional message body containing any kind of data. Request bodies carry data to the web server; response bodies carry data back to the client. Unlike the start lines and headers, which are textual and structured, the body can contain arbitrary binary data (e.g., images, videos, audio tracks, software applications). Of course, the body can also contain text.

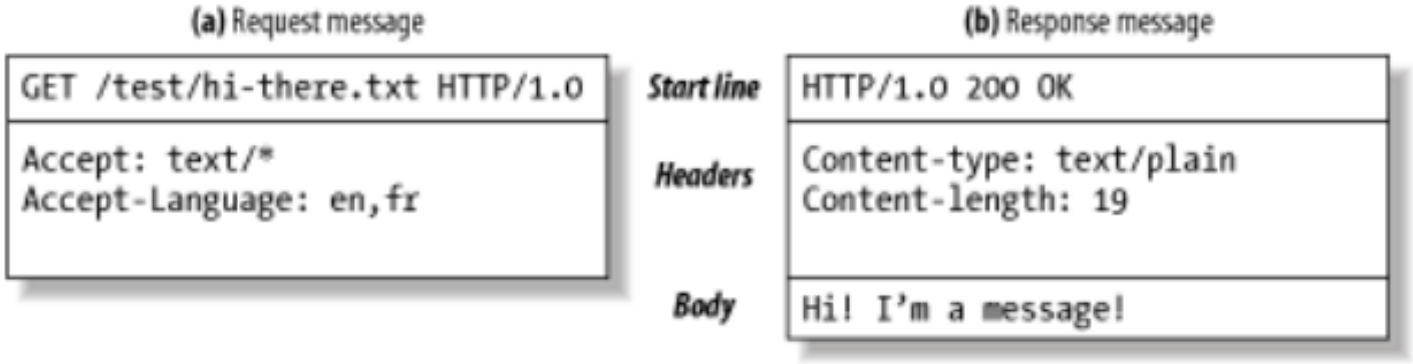


Figure 35: HTTP Message structure

Get Transaction

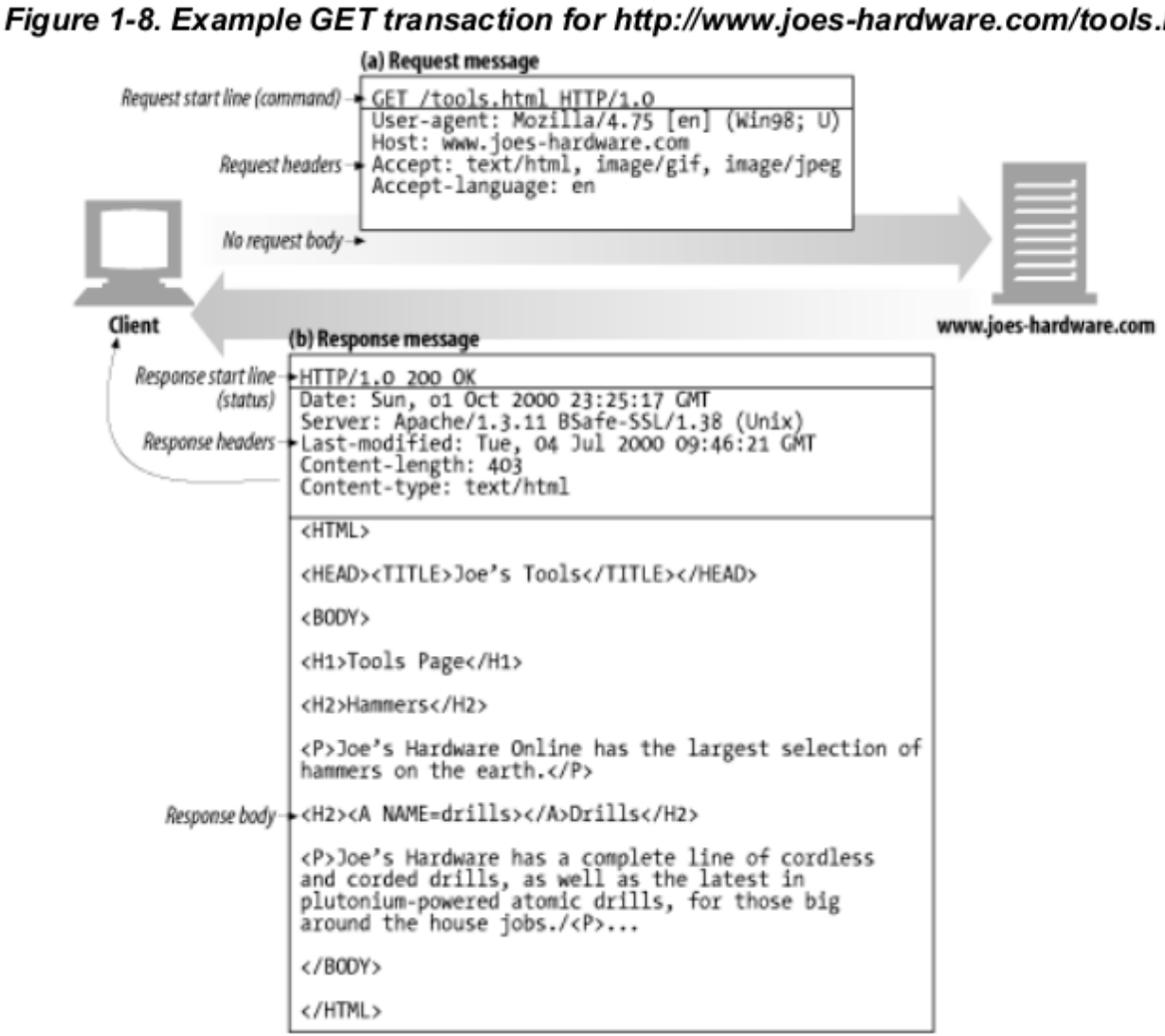


Figure 36: Get Transaction

Browser connection

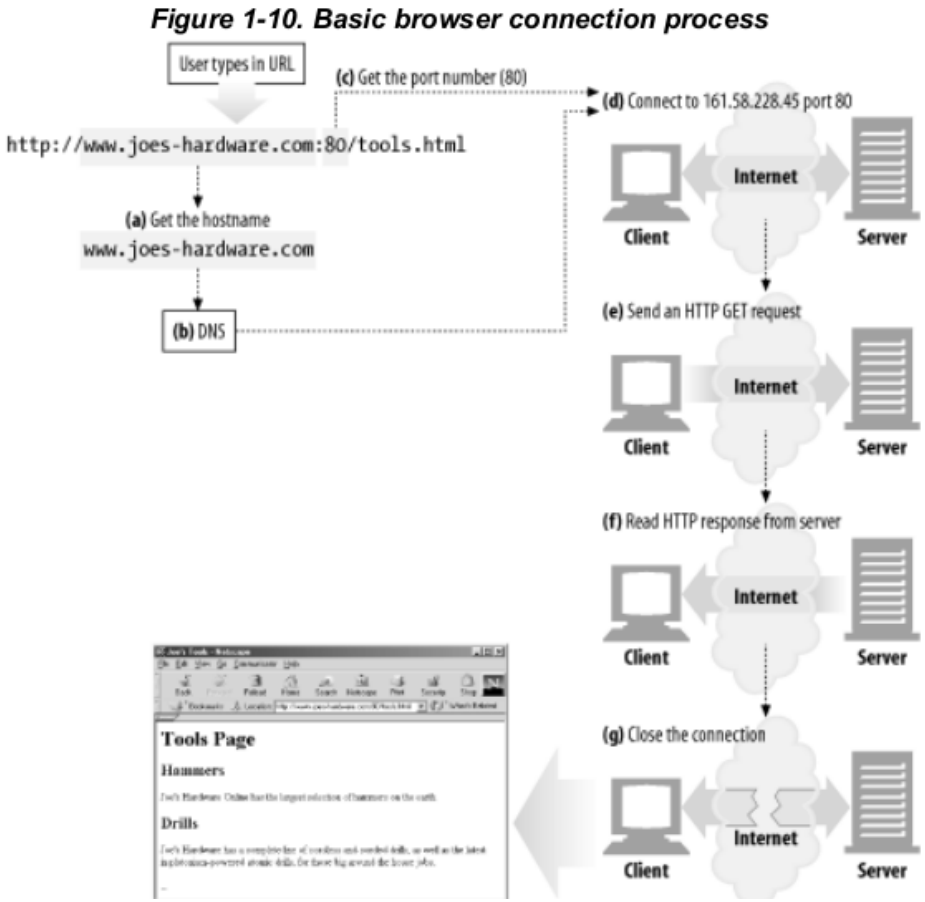


Figure 37: Browser connection

Telnet Transaction

Example 1-1. An HTTP transaction using telnet

```
% telnet www.joes-hardware.com 80
Trying 161.58.228.45...
Connected to joes-hardware.com.
Escape character is '^]'.
GET /tools.html HTTP/1.1
Host: www.joes-hardware.com

HTTP/1.1 200 OK
Date: Sun, 01 Oct 2000 23:25:17 GMT
Server: Apache/1.3.11 BSafe-SSL/1.38 (Unix) FrontPage/4.0.4.3
Last-Modified: Tue, 04 Jul 2000 09:46:21 GMT
ETag: "373979-193-3961b26d"
Accept-Ranges: bytes
Content-Length: 403
Connection: close
Content-Type: text/html

<HTML>
<HEAD><TITLE>Joe's Tools</TITLE></HEAD>
<BODY>
<H1>Tools Page</H1>
<H2>Hammers</H2>
<P>Joe's Hardware Online has the largest selection of hammers
on the earth.</P>
<H2><A NAME=drills></A>Drills</H2>
```

Protocol versions

1. **HTTP/0.9**: The 1991 prototype version of HTTP is known as HTTP/0.9. This protocol contains many serious design flaws and should be used only to interoperate with legacy clients. HTTP/0.9 supports only the GET method, and it does not support MIME typing of multimedia content, HTTP headers, or version numbers. HTTP/0.9 was originally defined to fetch simple HTML objects. It was soon replaced with HTTP/1.0.
2. **HTTP/1.0**: 1.0 was the first version of HTTP that was widely deployed. HTTP/1.0 added version numbers, HTTP headers, additional methods, and multimedia object handling. HTTP/1.0 made it practical to support graphically appealing web pages and interactive forms, which helped promote the wide-scale adoption of the World Wide Web. This specification was never well specified. It represented a collection of best practices in a time of rapid commercial and academic evolution of the protocol.

1. **HTTP/1.0+**: virtual hosting support, andproxy connection support, were added to HTTP and became unofficial, de facto standards.
2. **HTTP/1.1**: HTTP/1.1 focused on correcting architectural flaws in the design of HTTP, specifying semantics, introducing significant performance optimizations, and removing mis-features. HTTP/1.1 also included support for the more sophisticated web applications and deployments that were under way in the late 1990s
3. **HTTP-NG (a.k.a. HTTP/2.0)**: HTTP-NG is a prototype proposal for an architectural successor to HTTP/1.1 that focuses on significant performance optimizations and a more powerful framework for remote execution of server logic.

Architectural Components of the Web

1. Proxies
2. Caches
3. Gateways
4. Tunnels
5. Agents

Types of headers

General headers

These are generic headers used by both clients and servers. They serve general purposes that are useful for clients, servers, and other applications to supply to one another. For example, the Date header is a general-purpose header that allows both sides to indicate the time and date at which the message was constructed:

```
Date: Tue, 3 Oct 1974 02:16:00 GMT
```

Request headers

As the name implies, request headers are specific to request messages. They provide extra information to servers, such as what type of data the client is willing to receive. For example, the following Accept header tells the server that the client will accept any media type that matches its request:

```
Accept: */*
```

Response headers

Response messages have their own set of headers that provide information to the client (e.g., what type of server the client is talking to). For example, the following Server header tells the client that it is talking to a Version 1.0 Tiki-Hut server:

```
Server: Tiki-Hut/1.0
```

Entity headers

Entity headers refer to headers that deal with the entity body. For instance, entity headers can tell the type of the data in the entity body. For example, the following Content-Type header lets the application know that the data is an HTML document in the iso-latin-1 character set:

```
Content-Type: text/html; charset=iso-latin-1
```

Extension headers

Extension headers are nonstandard headers that have been created by application developers but not yet added to the sanctioned HTTP specification. HTTP programs need to tolerate and forward extension headers, even if they don't know what the headers mean.

HyperText Markup Language

Dedicated HTML books invariably begin with a brief history of HTML. Such a history might begin with the ARPANET of the late 1960s, jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991, and then to HTML's codification by the World Wide Web Consortium (better known as the W3C) in 1997.

To publish information for global distribution, one needs a universally understoodvlanguage, a kind of publishing mother tongue that all computers may potentiallyvunderstand. The publishing language used by the World Wide Web is HTML (from HyperText Markup Language)

HyperText Markup Language

A markup language is simply a way of annotating a document in such a way as to make the annotations distinct from the text being annotated. Markup languages such as HTML, Tex, XML, and XHTML allow users to control how text and visual elements will be laid out and displayed.

In addition to specifying ?information about content? many markup languages are able to encode information how to display the content for the end user. These presentation semantics can be as simple as specifying a bold weight font for certain words, and were a part of the earliest HTML specification. Although combining semantic markup with presentation markup is no longer permitted in HTML5, "formatting the content" for display remains a key reason why HTML was widely adopted.

HyperText Markup Language

the World Wide Web Consortium (W3C) is the main standards organization for the World Wide Web (WWW). It promotes compat- ibility, thereby ensuring web technologies work together in a predictable way. To help in this goal, the W3C produces Recommendations (also called specifications). These Recommendations are very lengthy documents that are meant to guide manufacturers in their implementations of HTML, XML, and other web protocols.

XHTML

As the web evolved in the 1990s, web browsers evolved into quite permissive and lenient programs. They could handle sloppy HTML, missing or malformed tags, and other syntax errors. However, it was somewhat unpredictable how each browser would handle such errors. The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors. To help web authors, two versions of XHTML were created: XHTML 1.0 Strict and XHTML 1.0 Transitional. The strict version was meant to be rendered. Like HTML, XML is a textual markup language. Also like HTML, the formal rules for XML were set by the W3C.

XML Based syntax rules

XML-based syntax rules The main rules are:

1. There must be a single root element
2. Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML
3. Element names can?t start with a number.
4. Element and attribute names are case sensitive.
5. Attributes must always be within quotes
6. All elements must have a closing element (or be self-closing)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

Figure 40: XML code

HTML5

At around the same time the XHTML 2.0 specification was being developed, a group of developers at Opera and Mozilla formed the WHATWG (Web Hypertext Application Technology Working Group) group within the W3C. This group was not convinced that the W3C?s embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web. Thus the WHATWG charter announced:

"The Web Hypertext Applications Technology working group therefore intends to address the need for one coherent development environment for Web applications, through the creation of technical specifications that are intended to be implemented in mass-market Web browsers."

W3C stopped work on XHTML 2.0 and instead adopted the work done by WHATWG and named it HTML5. There are three main aims to HTML5:

1. Specify unambiguously how browsers should deal with invalid markup.
2. Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.
3. Be backwards compatible with the existing web.

Elements and Attributes

HTML documents are composed of textual content and HTML elements. The term HTML element is often used interchangeably with the term tag. However, an HTML element is a more expansive term that encompasses the element name within angle brackets (i.e., the tag) and the content within the tag (though some elements contain no extra content)

1. An HTML element is identified in the HTML document by tags. A tag consists of the element name within angle brackets. The element name appears in both the beginning tag and the closing tag, which contains a forward slash followed by the element's name, again all enclosed within angle brackets. The closing tag acts like an off-switch for the on-switch that is the start tag.
2. An HTML attribute is a name=value pair that provides more information about the HTML element. In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional, though many web authors still maintain the practice of enclosing attribute values in quotes. Some HTML attributes expect a number for the value. These will just be the numeric value; they will never include the unit.

Nesting HTML Elements

Often an HTML element will contain other HTML elements. In such a case, the container element is said to be a parent of the contained, or child, element. Any elements contained within the child are said to be descendants of the parent element; likewise, any given child element, may have a variety of ancestors.

Nesting HTML Elements

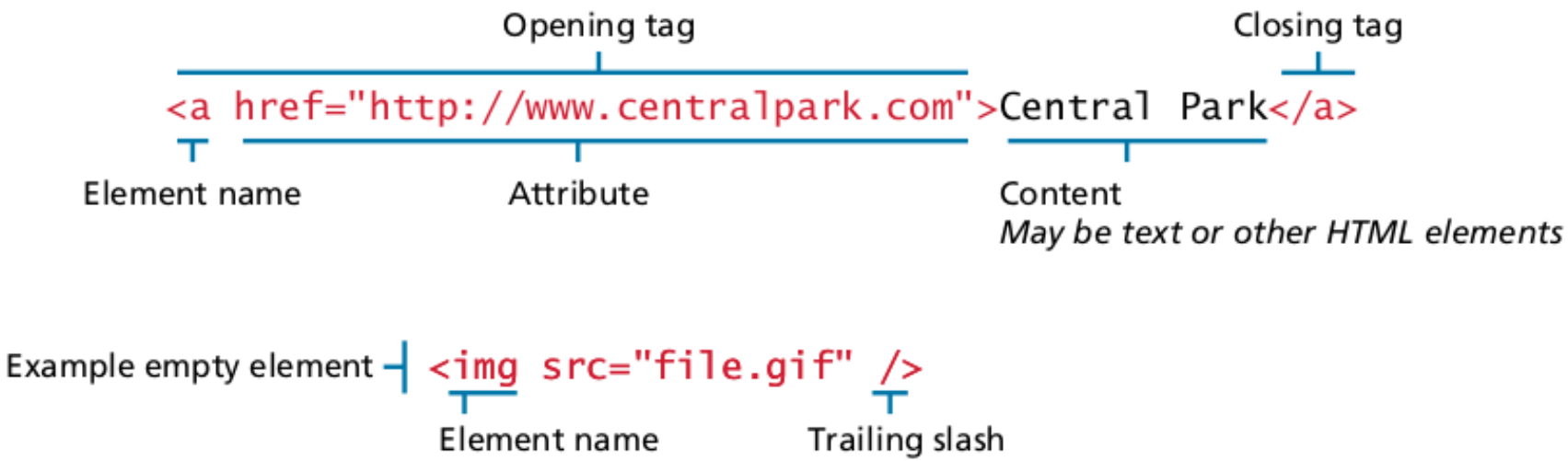


Figure 41: XML code

Nesting HTML Elements

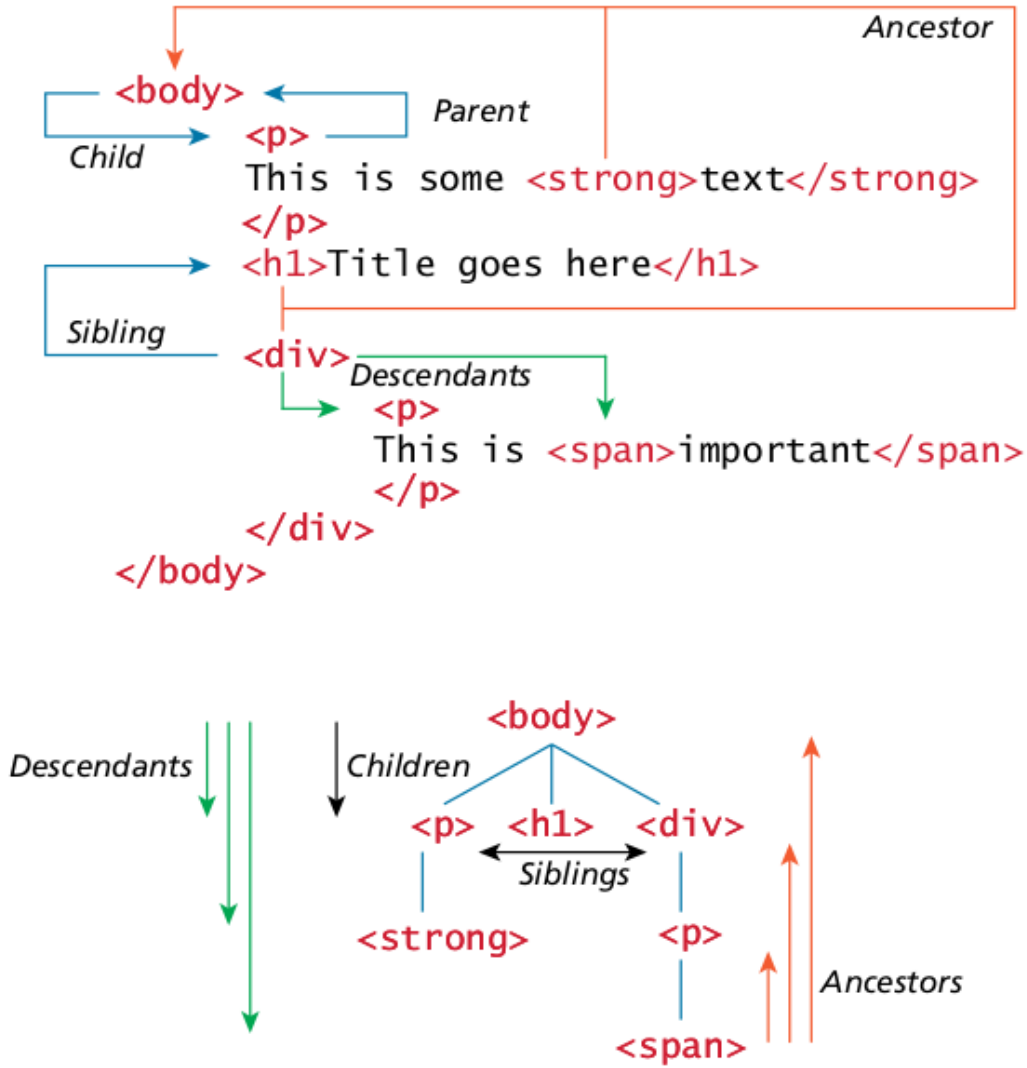


Figure 42: XML code

Nesting HTML Elements

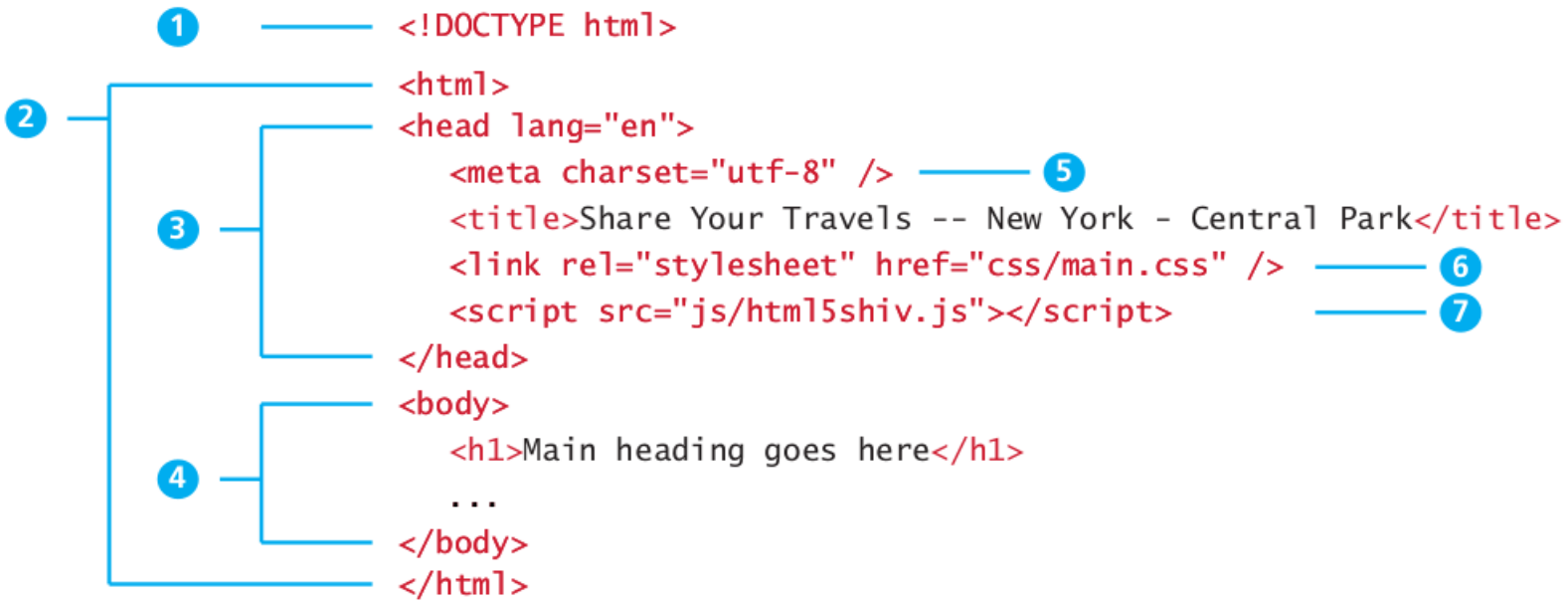
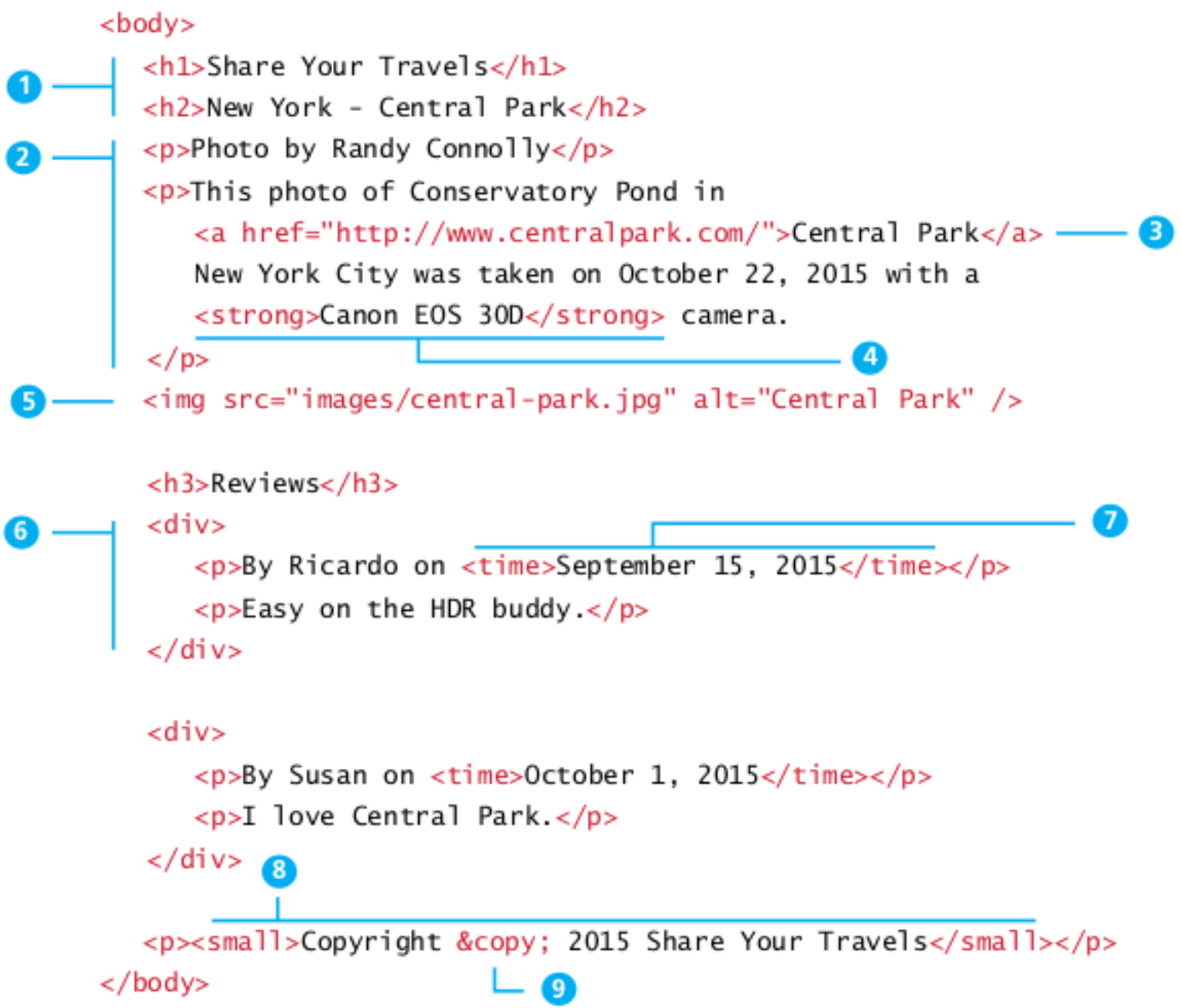


Figure 43: structure of HTML code

Nesting HTML Elements



HTML5 semantic elements Articles and Sections

Element	Description
<a>	Anchor used for hyperlinks.
<abbr>	An abbreviation
 	Line break
<cite>	Citation (i.e., a reference to another work).
<code>	Used for displaying code, such as markup or programming code.
	Emphasis
<mark>	For displaying highlighted text
<small>	For displaying the fine-print, i.e., "non-vital" text, such as copyright or legal notices.
	The inline equivalent of the element. It is generally used to mark text that will receive special formatting using CSS.
	For content that is strongly important.
<time>	For displaying time and date data

TABLE 2.2 Common Text-Level Semantic Elements

Figure 45: sample html

HTML5 semantic elements Articles and Sections

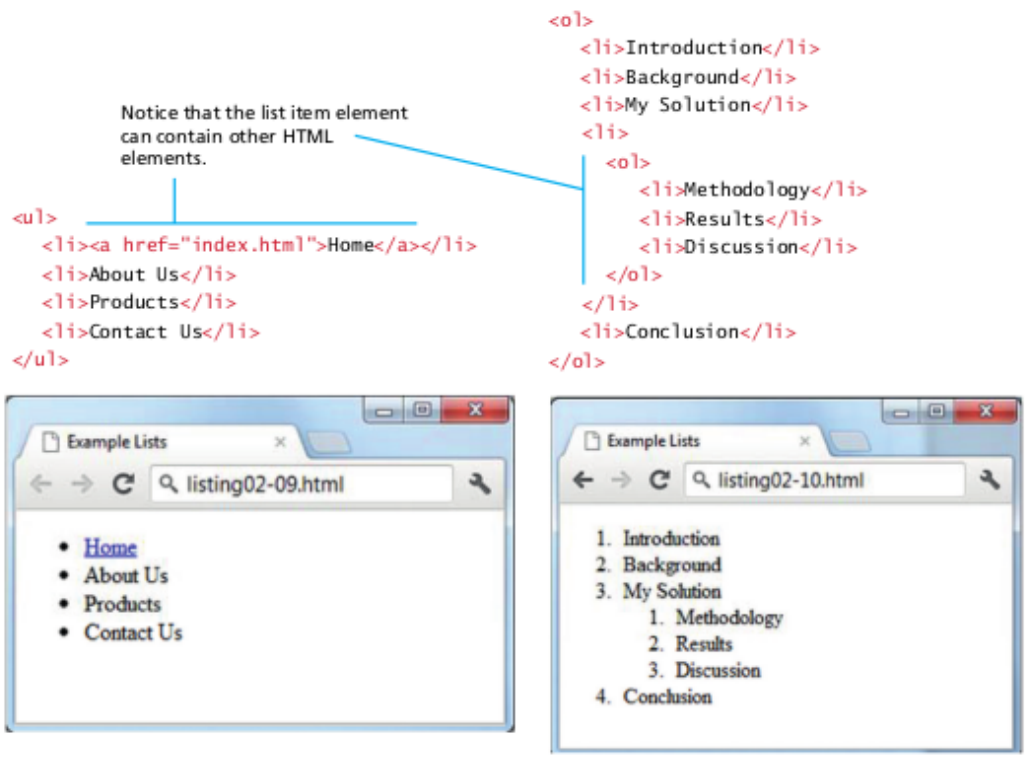


Figure 46: sample html

HTML5 semantic elements Articles and Sections

```
<header>
  <img src="logo.gif" alt="logo" />
  <h1>Fundamentals of Web Development</h1>
  <nav role="navigation">
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="about.html">About Us</a></li>
      <li><a href="browse.html">Browse</a></li>
    </ul>
  </nav>
</header>
```

LISTING 2.3 nav example

Figure 47: sample html

HTML5 semantic elements Articles and Sections

```
<header>
<img src="logo.gif" alt="logo" />
<h1>Fundamentals of Web Development</h1>
...
</header>
<article>
   <header>
      <h2>HTML5 Semantic Structure Elements</h2>
      <p>By <em>Randy Connolly</em></p>
      <p><time>September 30, 2015</time></p>
   </header>
   ...
</article>
```

LISTING 2.1 Heading example

Figure 48: sample html

HTML5 semantic elements Articles and Sections

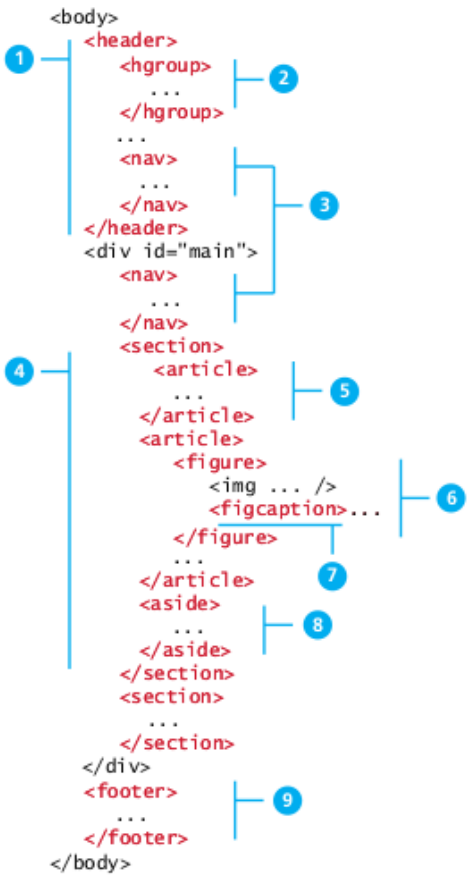


FIGURE 2.21 Sample layout using new HTML5 semantic structure elements

Figure 49: sample html

Steps to run html pages on localhost

1. Create your own directory in the path /var/www. gci is the directory created in the path.
2. change the ownership by executing the command chown

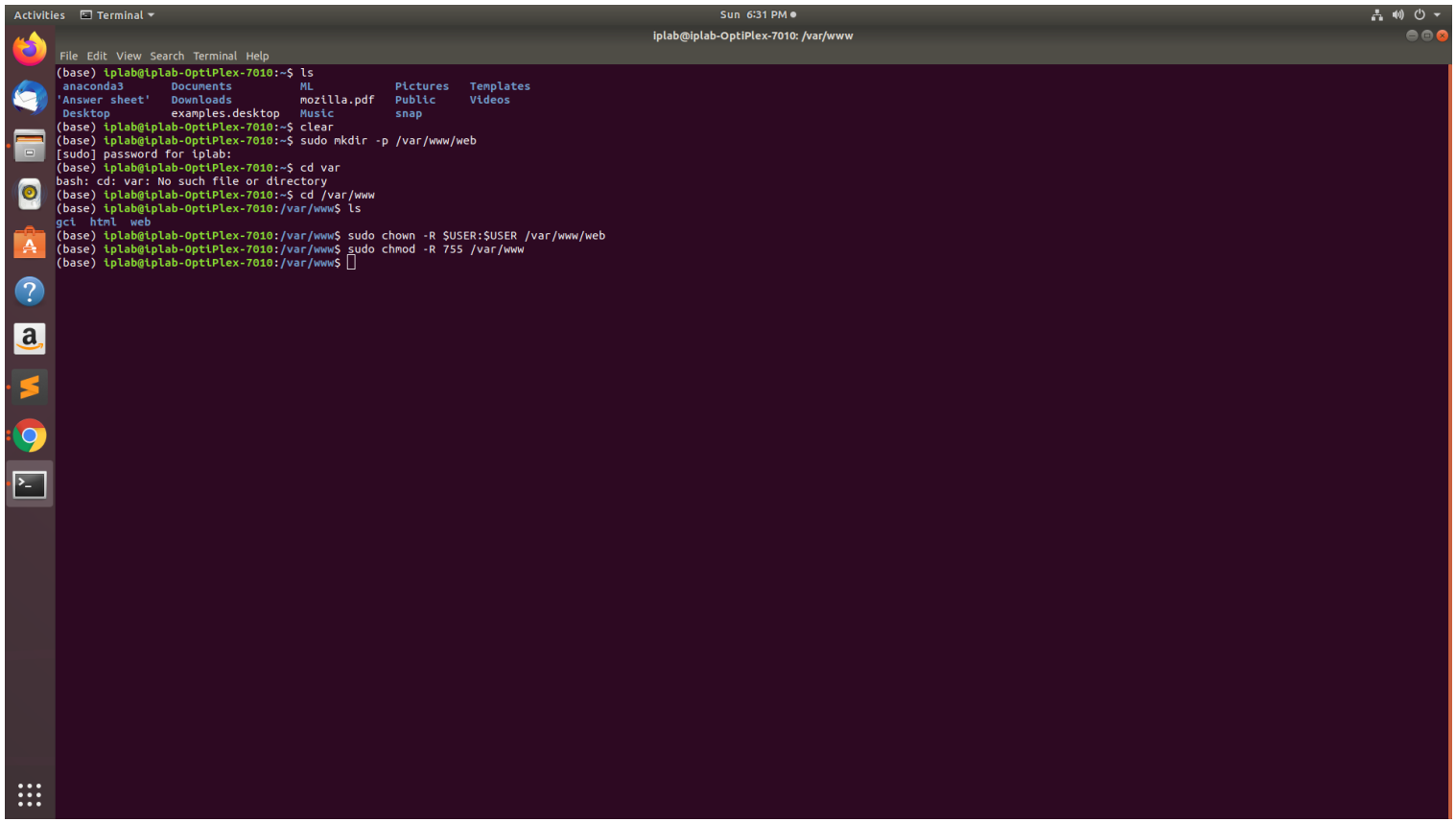


Figure 50: create own directory

Steps to run html pages on localhost

1. Create your own html programs in gci directory.
2. Visit the directory where apache2 server is installed. Go to sites enabled to look for the conf files.
3. create gci conf file (look for the command in the terminal)

Steps to run html pages on localhost

```
File  Edit  View  Search  Terminal  Help
(base) iplab@iplab-OptiPlex-7010:/var/www$ ls
gci  html
(base) iplab@iplab-OptiPlex-7010:/var/www$ cd html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ ls
index.html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ cd gci
bash: cd: gci: No such file or directory
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ ls
index.html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ cd .
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ cd ..
(base) iplab@iplab-OptiPlex-7010:/var/www$ cd gci
(base) iplab@iplab-OptiPlex-7010:/var/www/gci$ sudo nano index1.htm
(base) iplab@iplab-OptiPlex-7010:/var/www/gci$ cd ..
(base) iplab@iplab-OptiPlex-7010:/var/www$ cd html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ ls
index.html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ sudo cp index.html /var/www/gci/index1.html
(base) iplab@iplab-OptiPlex-7010:/var/www/html$ cd ..
(base) iplab@iplab-OptiPlex-7010:/var/www$ cd gci
(base) iplab@iplab-OptiPlex-7010:/var/www/gci$ ls
index1.htm  index1.html
(base) iplab@iplab-OptiPlex-7010:/var/www/gci$ sudo nano index1.html
(base) iplab@iplab-OptiPlex-7010:/var/www/gci$ cd ..
(base) iplab@iplab-OptiPlex-7010:/var/www$ cd ..
(base) iplab@iplab-OptiPlex-7010:/var$ cd ..
(base) iplab@iplab-OptiPlex-7010:/$ cd /etc/apache2
(base) iplab@iplab-OptiPlex-7010:/etc/apache2$ ls
apache2.conf    conf-enabled  magic          mods-enabled  sites-available
conf-available  envvars       mods-available  ports.conf    sites-enabled
(base) iplab@iplab-OptiPlex-7010:/etc/apache2$ cd sites-enabled/
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ ls
000-default.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ sudo nano gci.conf
[sudo] password for iplab:
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ ls
000-default.conf  gci.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ sudo nano gci.conf
Use "fg" to return to nano.

[1]+  Stopped                 sudo nano gci.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ rm -r gci.conf
rm: remove write-protected regular file 'gci.conf'? y
rm: cannot remove 'gci.conf': Permission denied
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ sudo rm -r gci.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ ls
000-default.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ sudo nano gci.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ ls
000-default.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$ sudo nano 000-default.conf
Use "fg" to return to nano.

[2]+  Stopped                 sudo nano 000-default.conf
(base) iplab@iplab-OptiPlex-7010:/etc/apache2/sites-enabled$
```

Figure 51: create own directory

Steps to run html pages on localhost

1. copy default conf file to gci conf file
2. enable gci conf file
3. disable 000default conf file

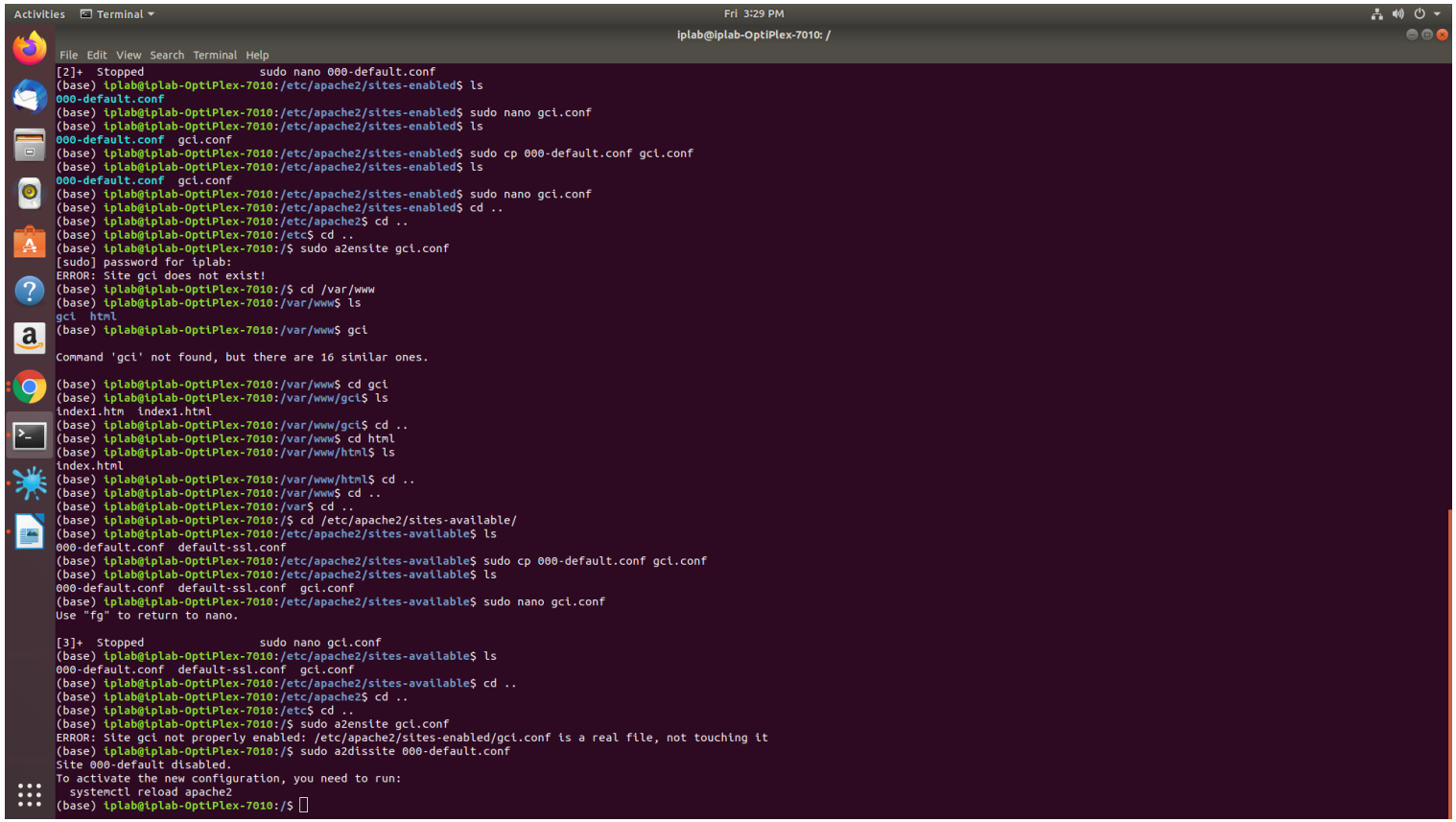


Figure 52: create own directory

Steps to run html pages on localhost

1. open the gci file using vi editor
2. copy the contents shown in the figure to gci conf file

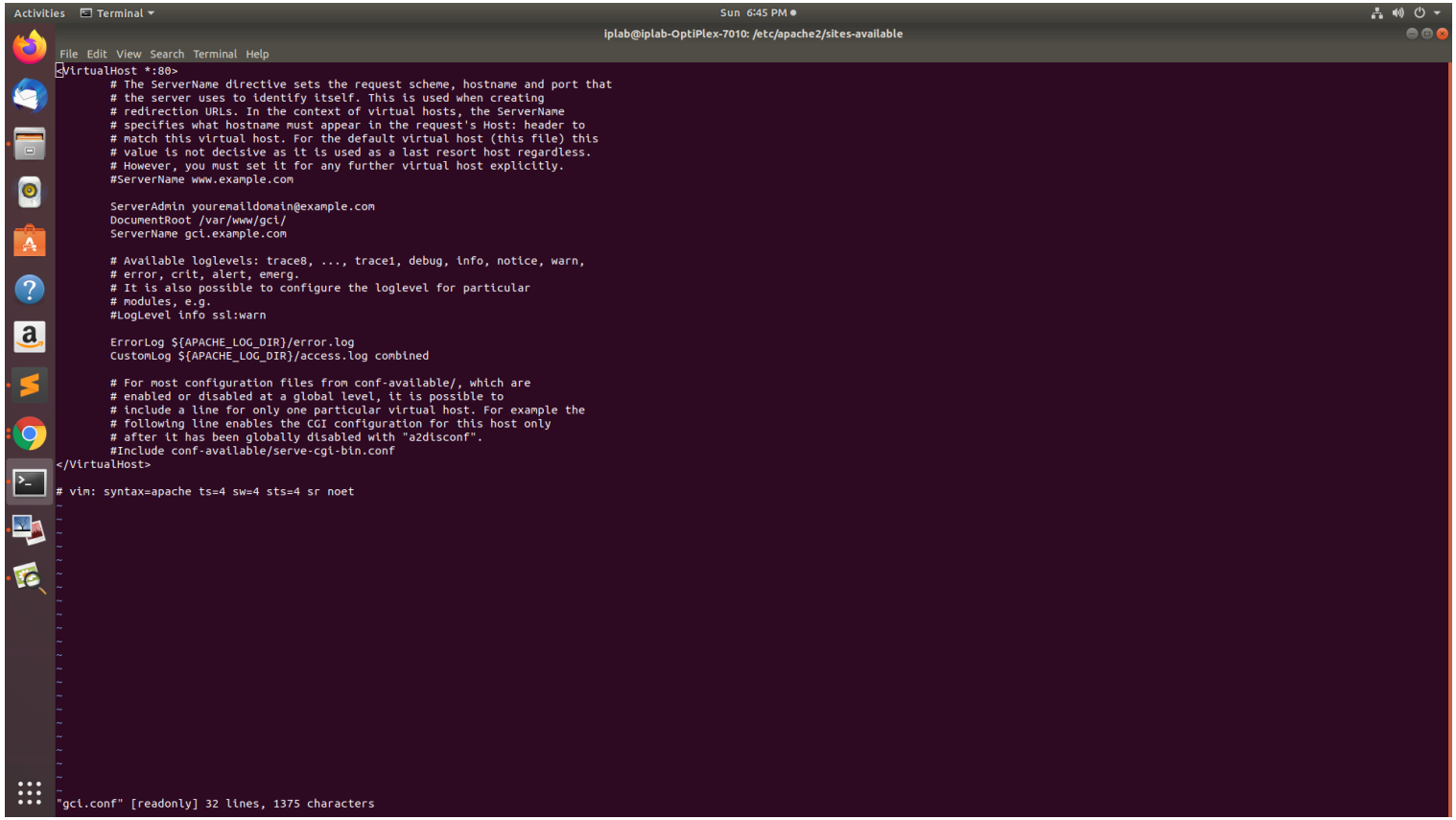


Figure 53: write to gci conf file

Steps to run html pages on localhost

1. copy the html programs in the sublime path into to /var/www/gci directory

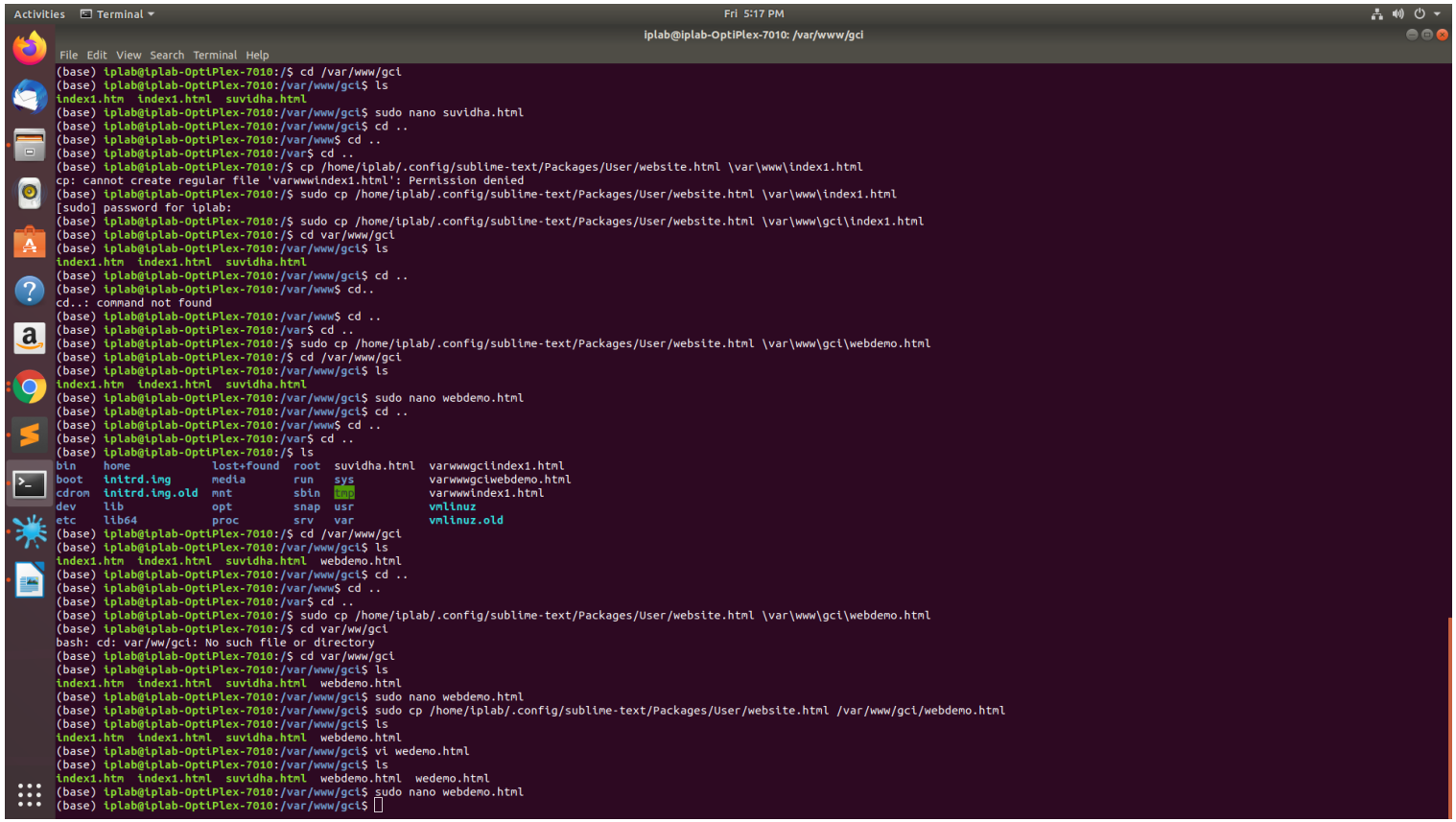


Figure 54: copy program files to gci directory

Steps to run html pages on localhost

1. Run programs on webserver

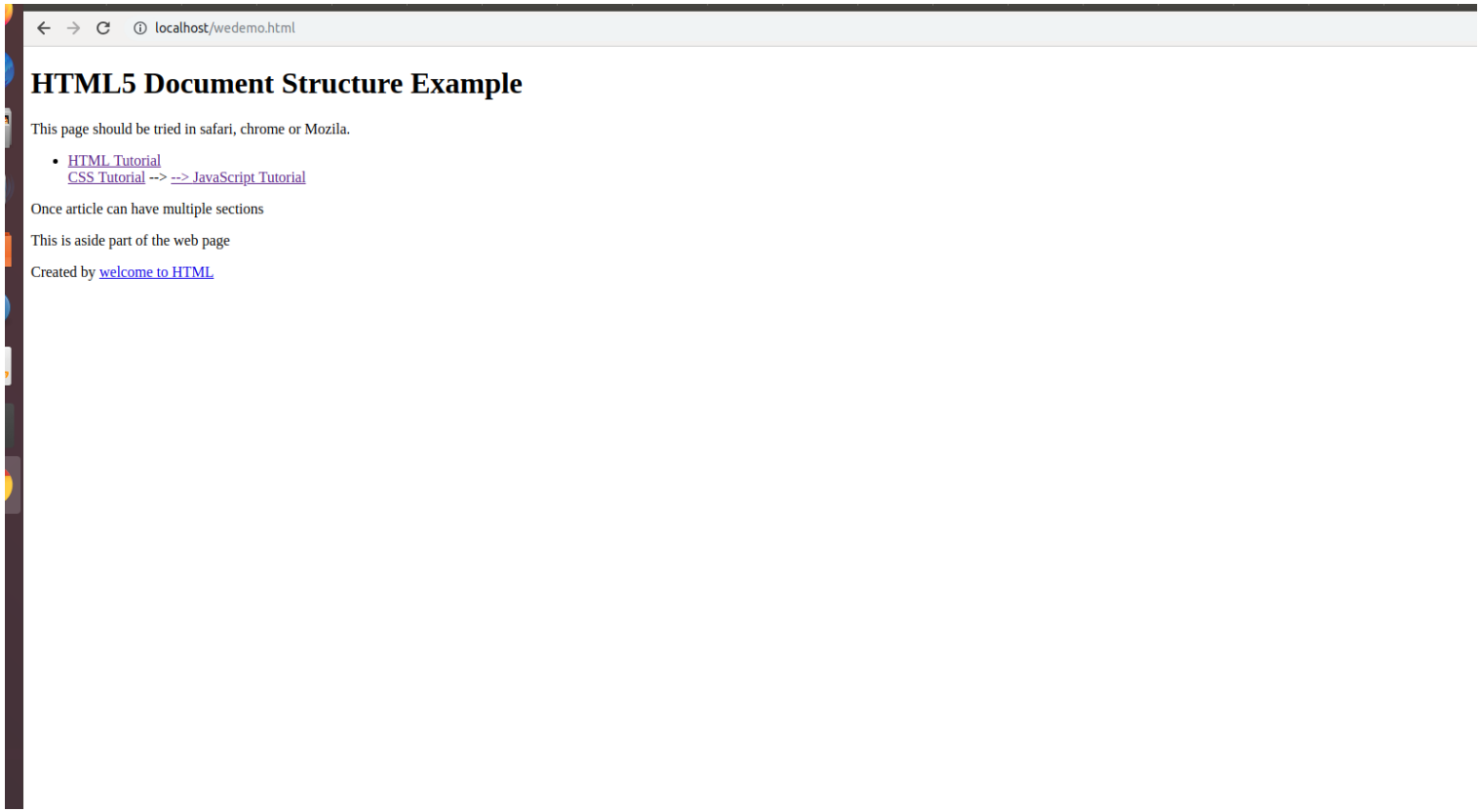


Figure 55: Deploy website on webserver

CSS

1. HTML
2. CSS
3. DOM

CSS Selectors

1. .class .intro Selects all elements with class="intro"
2. element p Selects all <p> elements
3. element.class p.intro Selects all <p> elements with class="intro"
4. element,element div, p Selects all elements and all <p> elements

CSS Font family

1. Arial (sans-serif)
2. Verdana (sans-serif)
3. Helvetica (sans-serif)
4. Tahoma (sans-serif)
5. Trebuchet MS (sans-serif)
6. Times New Roman (serif)
7. Georgia (serif)
8. Garamond (serif)
9. Courier New (monospace)
10. Brush Script MT (cursive)

1. Run programs on webserver

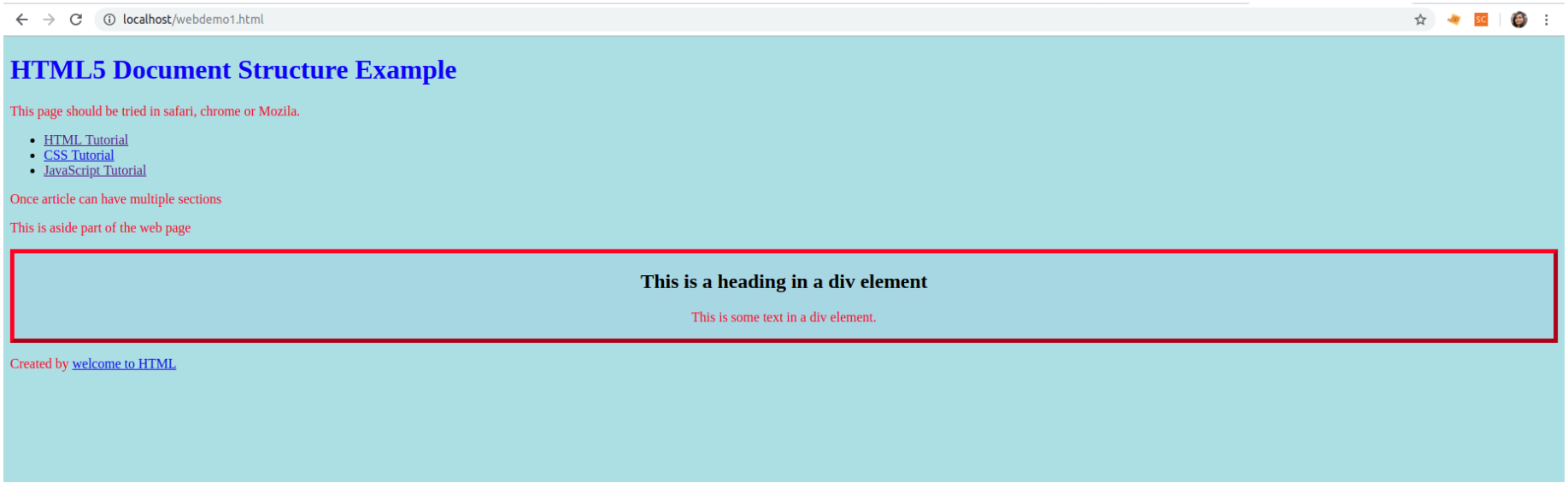


Figure 56

Steps to run html pages on localhost

1. Run programs on webserver

Figure 57

Steps to run html pages on localhost

1. Run programs on webserver

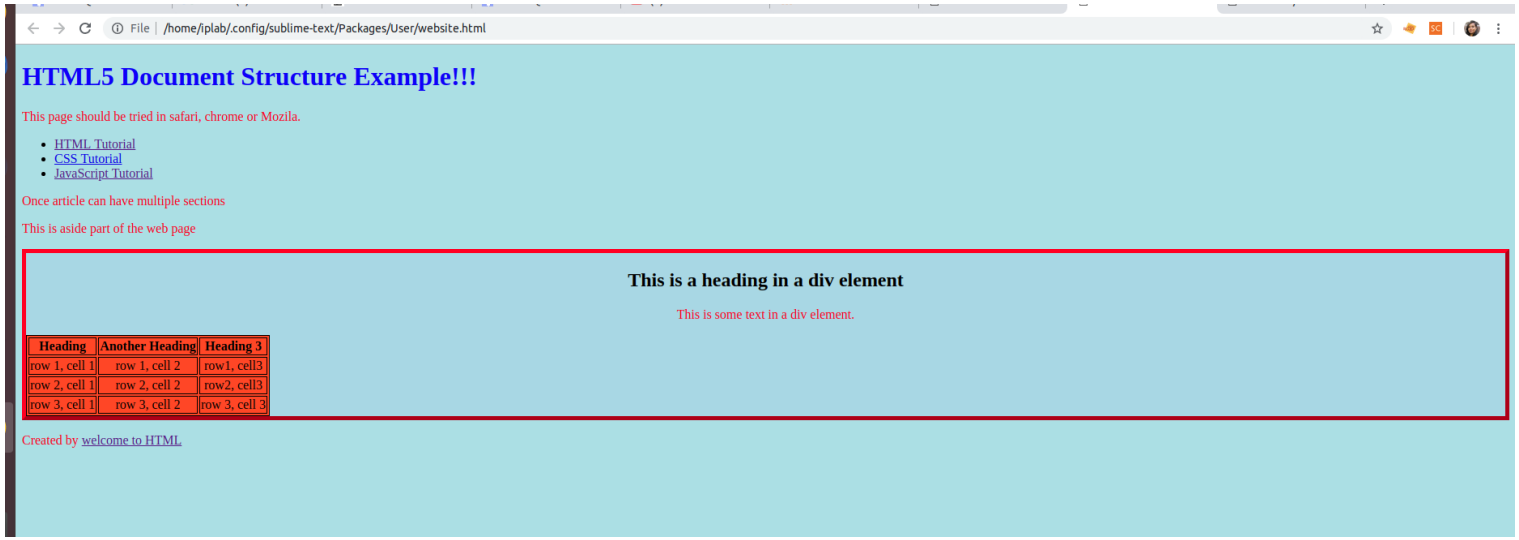


Figure 58: html with css style

Javascript

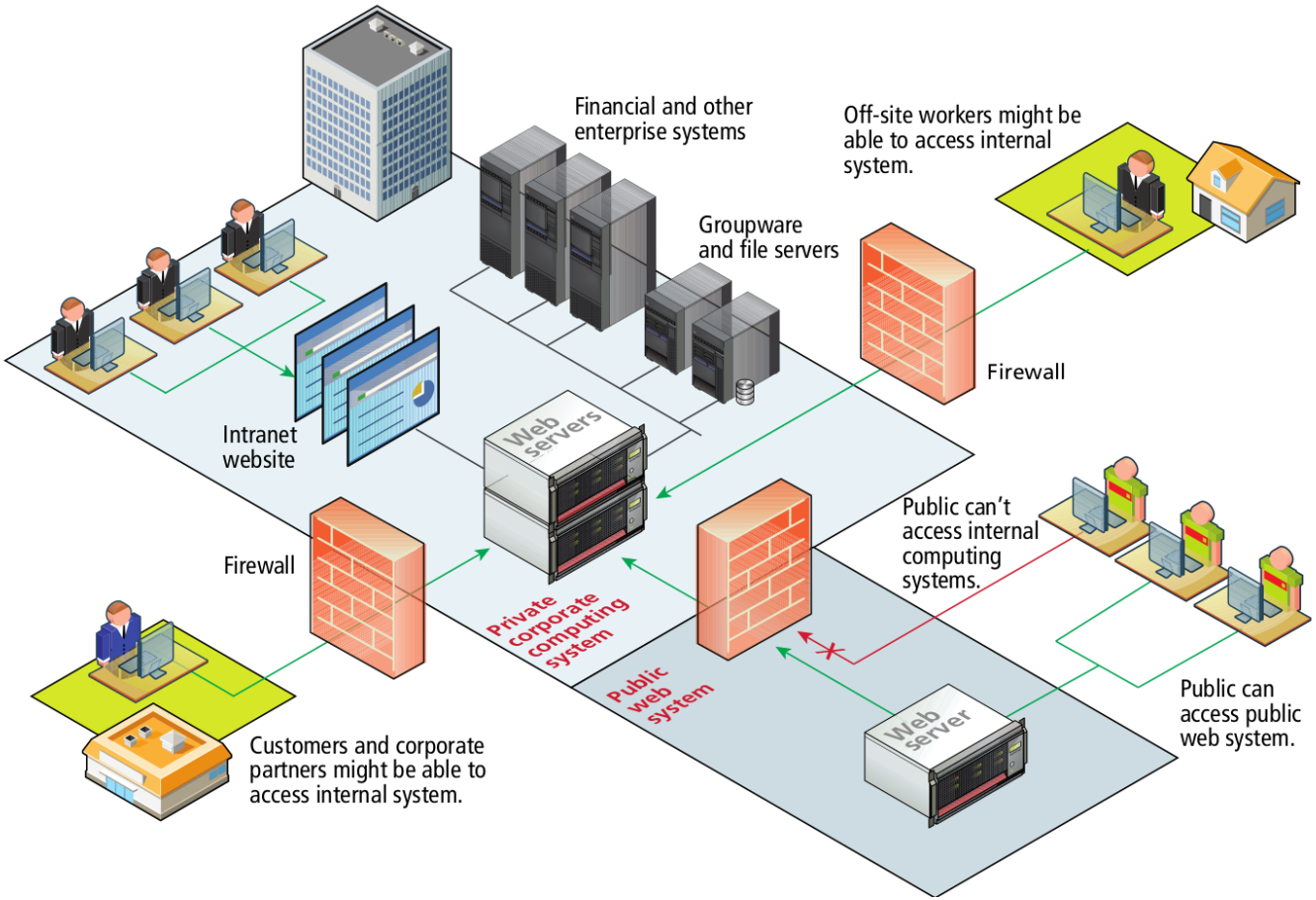


Figure 59

Javascript

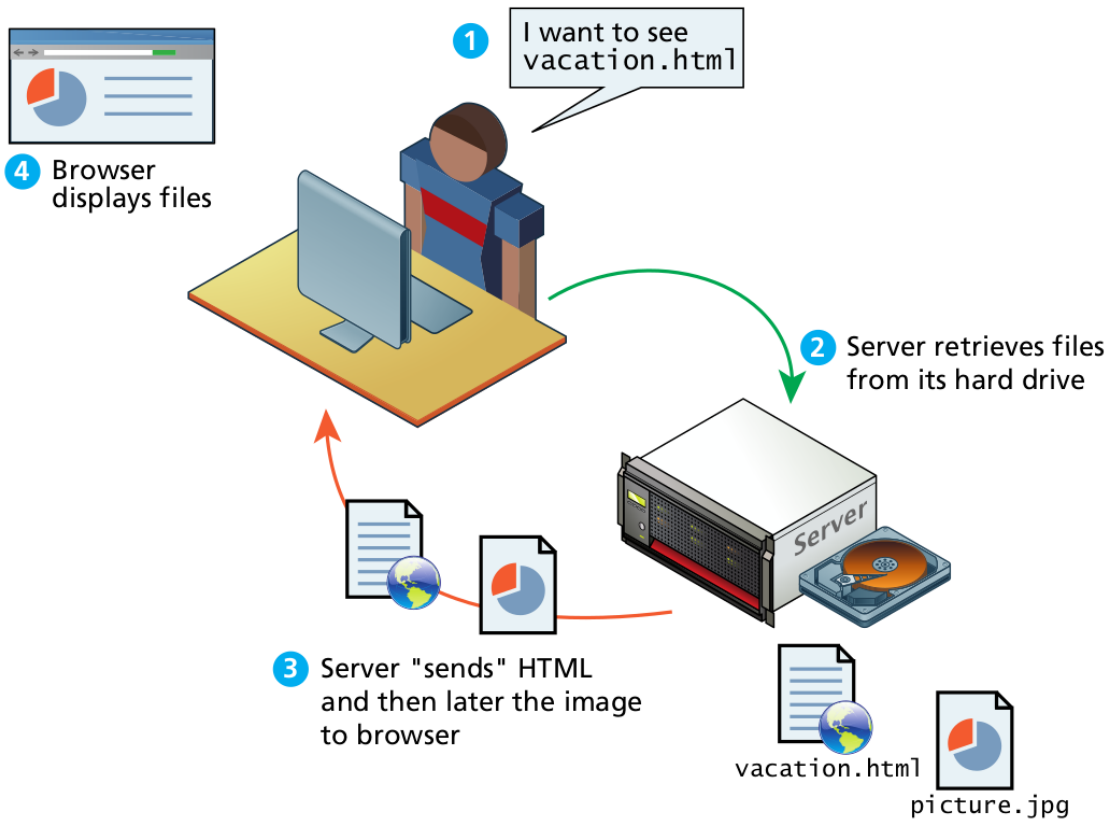


Figure 60: staticwebsite

Javascript

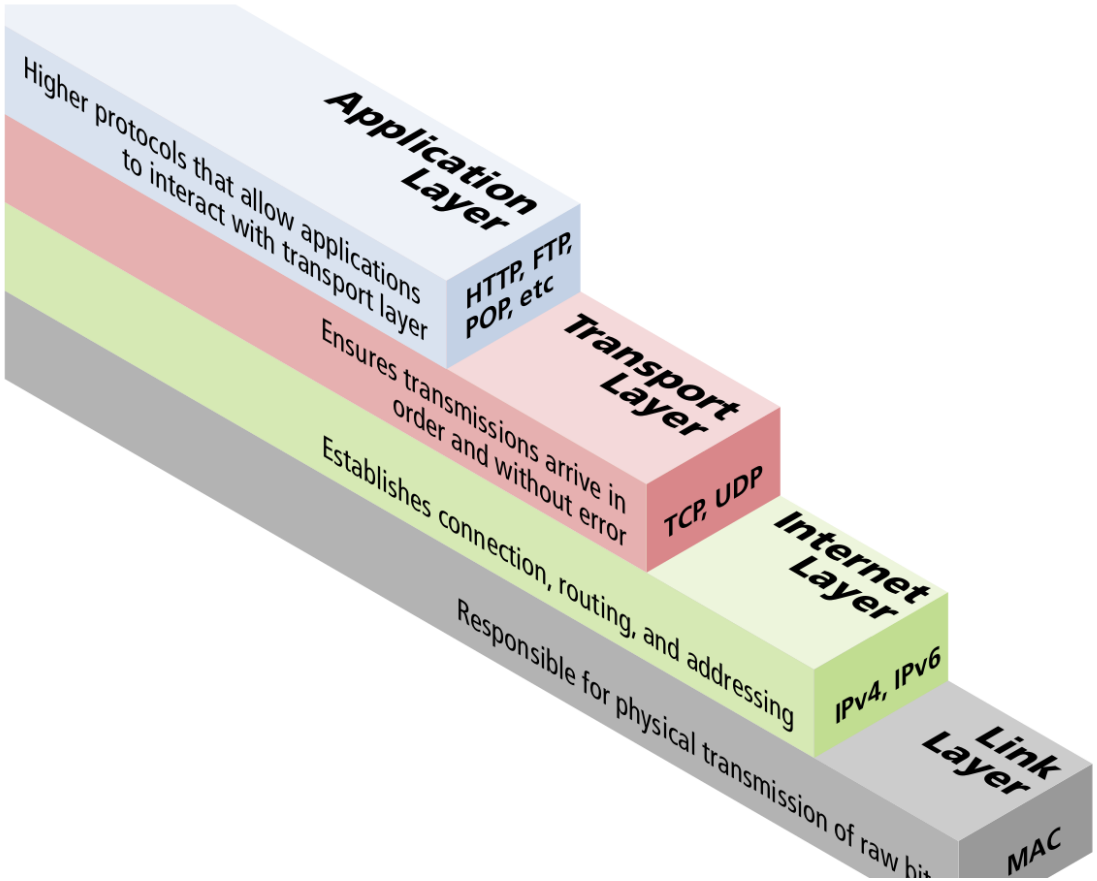


Figure 61: staticwebsite

Javascript

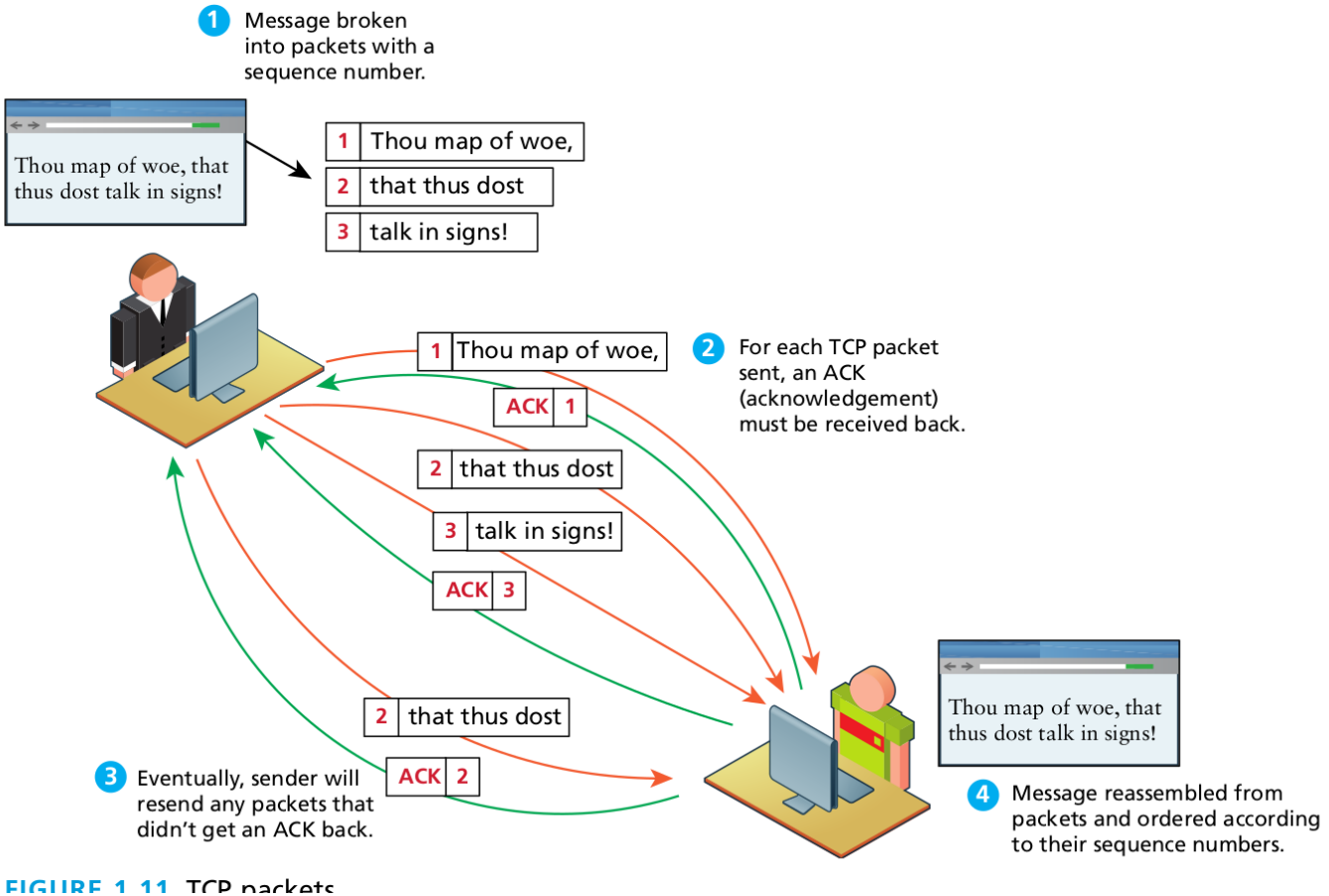


FIGURE 1.11 TCP packets

Figure 62: staticwebsite

Javascript

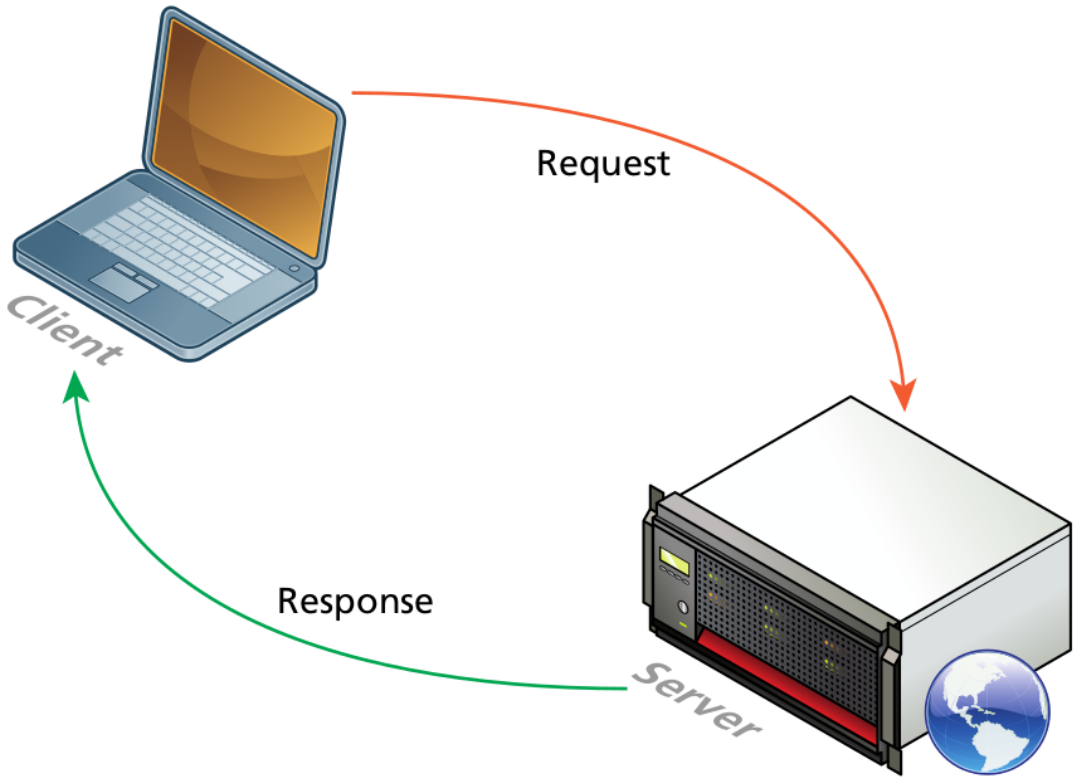


Figure 63: staticwebsite

Javascript

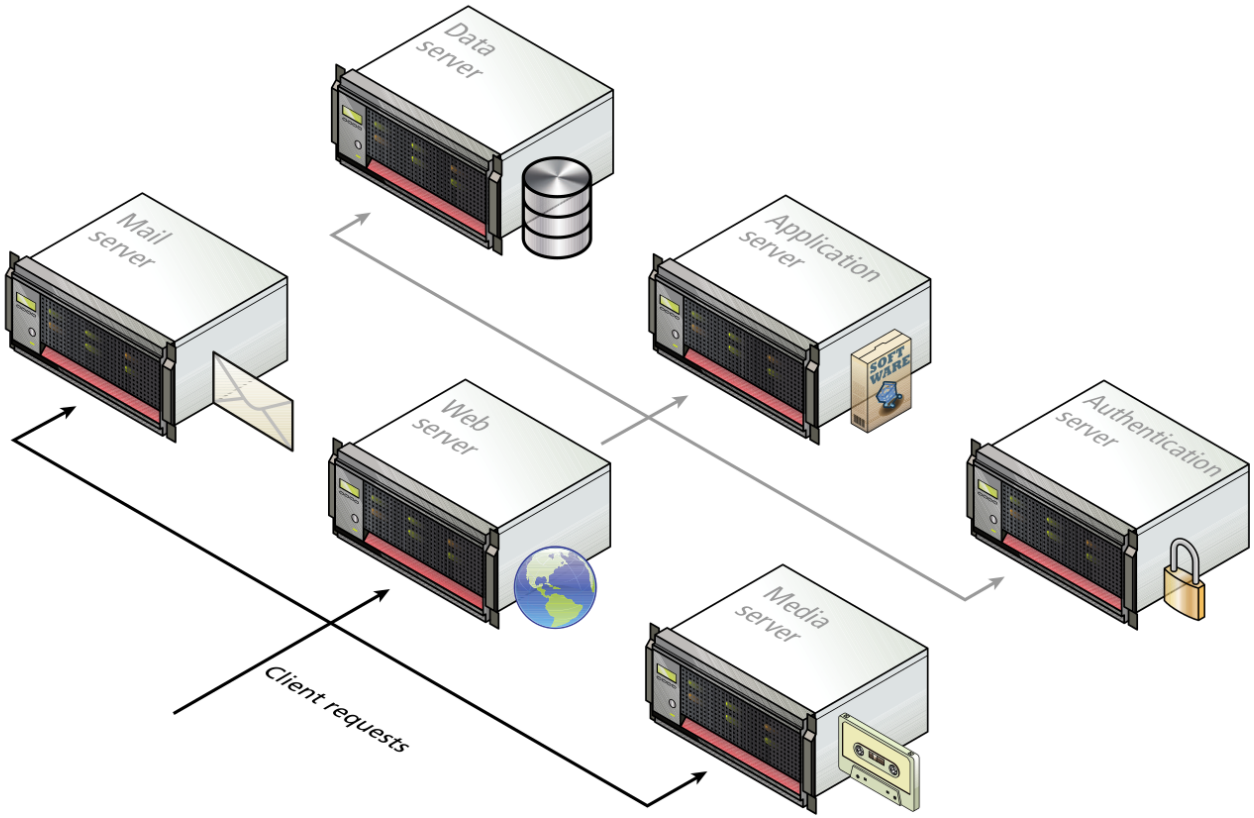


Figure 64: staticwebsite

PhP

PHP is an acronym for "PHP: Hypertext Preprocessor". PHP is a widely-used, open source scripting language. PHP scripts are executed on the server.
What is a PHP File

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP runs efficiently on the server side

To start using PHP, you can:
Find a web host with PHP and MySQL support Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

- If your server has activated support for PHP you do not need to do anything.
- Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.
- Because PHP is free, most web hosts offer PHP support.

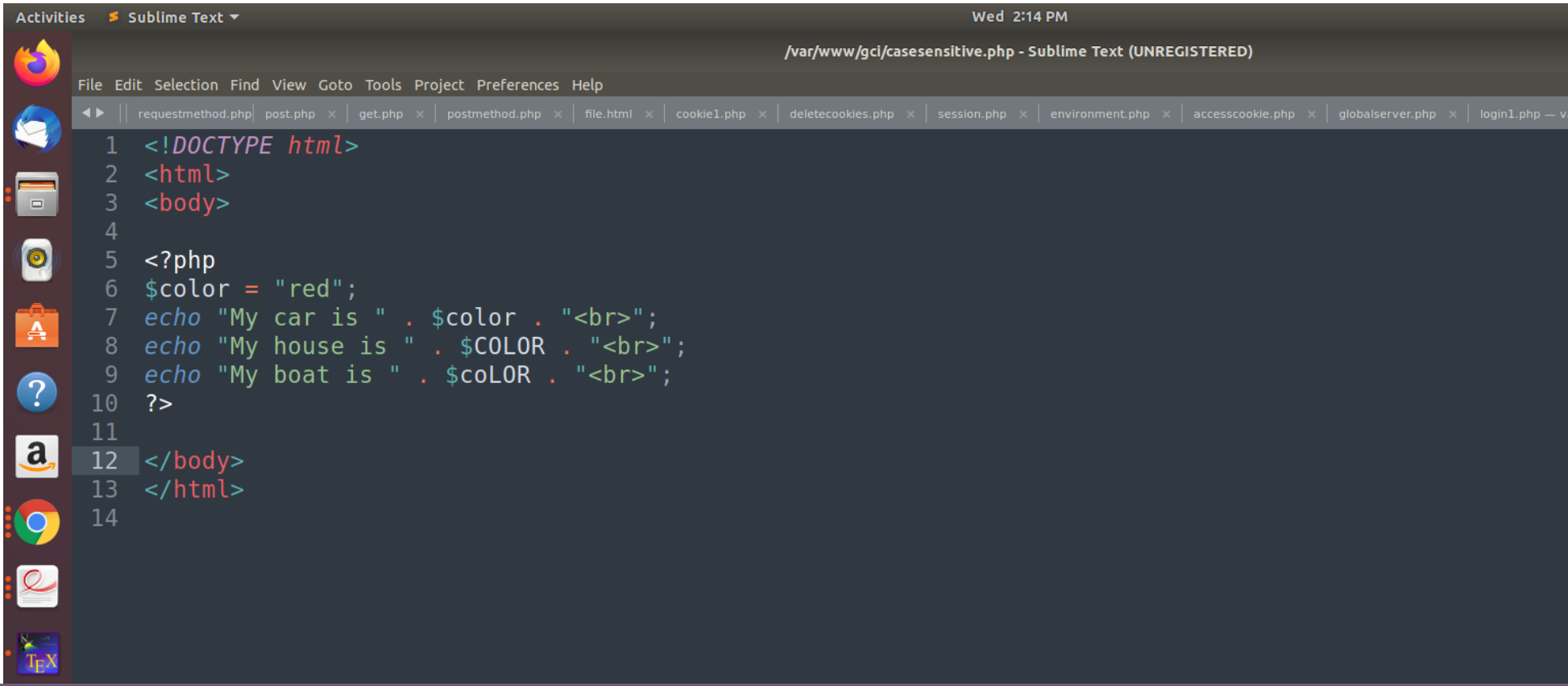
Set Up PHP on Your Own PC

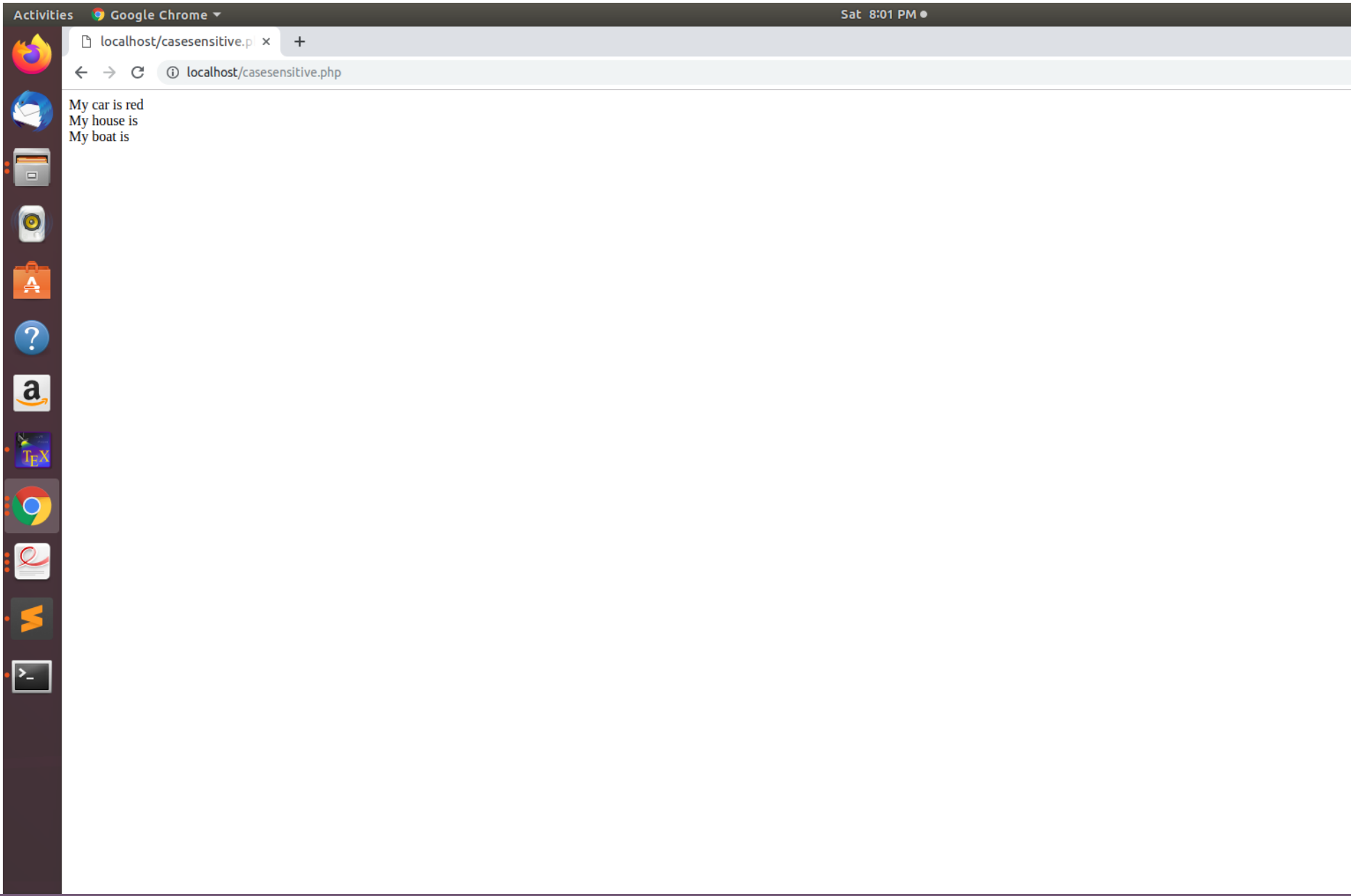
However, if your server does not support PHP, you must:
install a web server install PHP install a database, such as MySQL
Basic PHP Syntax

PHP Case Sensitivity

- In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

Look at the example below; only the first statement will display the value of the $color variable! This is because $color, $COLOR, and $coLOR are treated as three different variables:




Activities
Google Chrome
Sat 8:01 PM
localhost/casesensitive.p
localhost/casesensitive.php
My car is red
My house is
My boat is

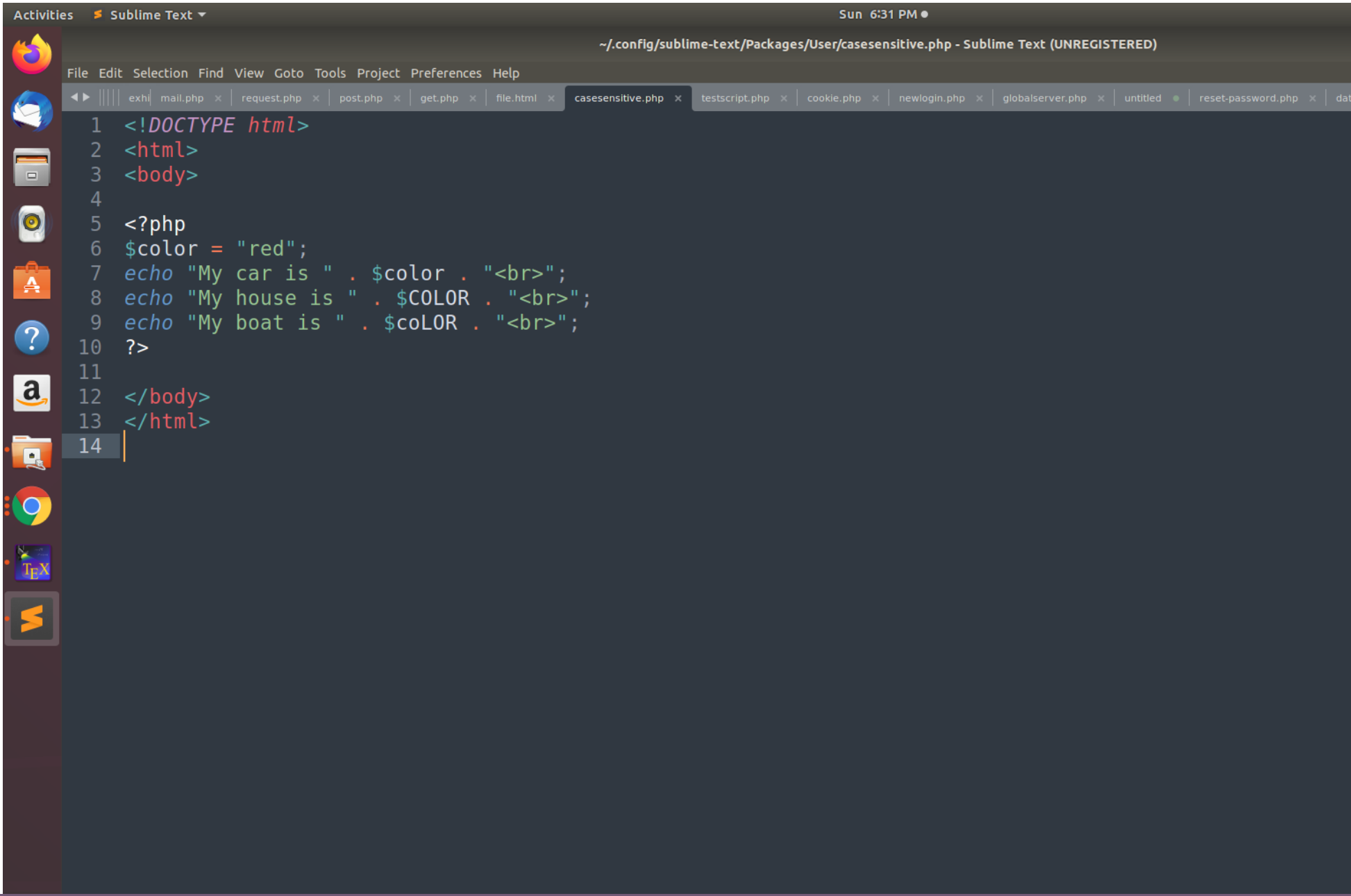
PHP Variables

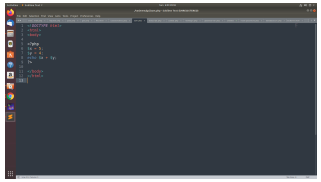
- In PHP, a variable starts with the $ sign, followed by the name of the variable:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

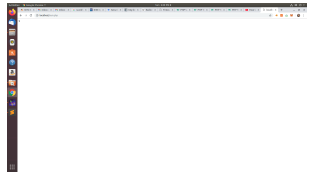
Rules for PHP variables:

1. A variable starts with the $ sign, followed by the name of the variable
2. A variable name must start with a letter or the underscore character
3. A variable name cannot start with a number
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
5. Variable names are case-sensitive ($age and $AGE are two different variables)


Activities
Sublime Text
Sun 6:31 PM
~/.config/sublime-text/Packages/User/casesensitive.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
exhi
mail.php
request.php
post.php
get.php
file.html
casesensitive.php
testscript.php
cookie.php
newlogin.php
globalserver.php
untitled
reset-password.php
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <?php
6 $color = "red";
7 echo "My car is " . $color . "
";
8 echo "My house is " . $COLOR . "
";
9 echo "My boat is " . $coLOR . "
";
10 ?>
11
12 </body>
13 </html>
14



(a) Addition of two numbers



(b) output

Figure 68: A figure with two subfigures

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

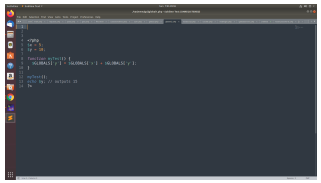
PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

PHP Variables Scope In PHP, variables can be declared anywhere in the script. The scope of a variable is the part of the script where the variable can be referenced.

PHP has three different variable scopes:

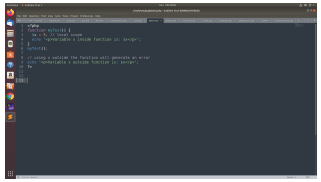
1. local
2. global
3. static



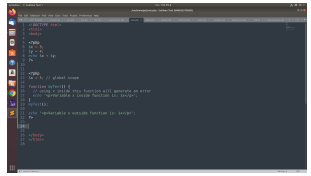
(a) global scope variable

(b) global output

Figure 69: A figure with two subfigures

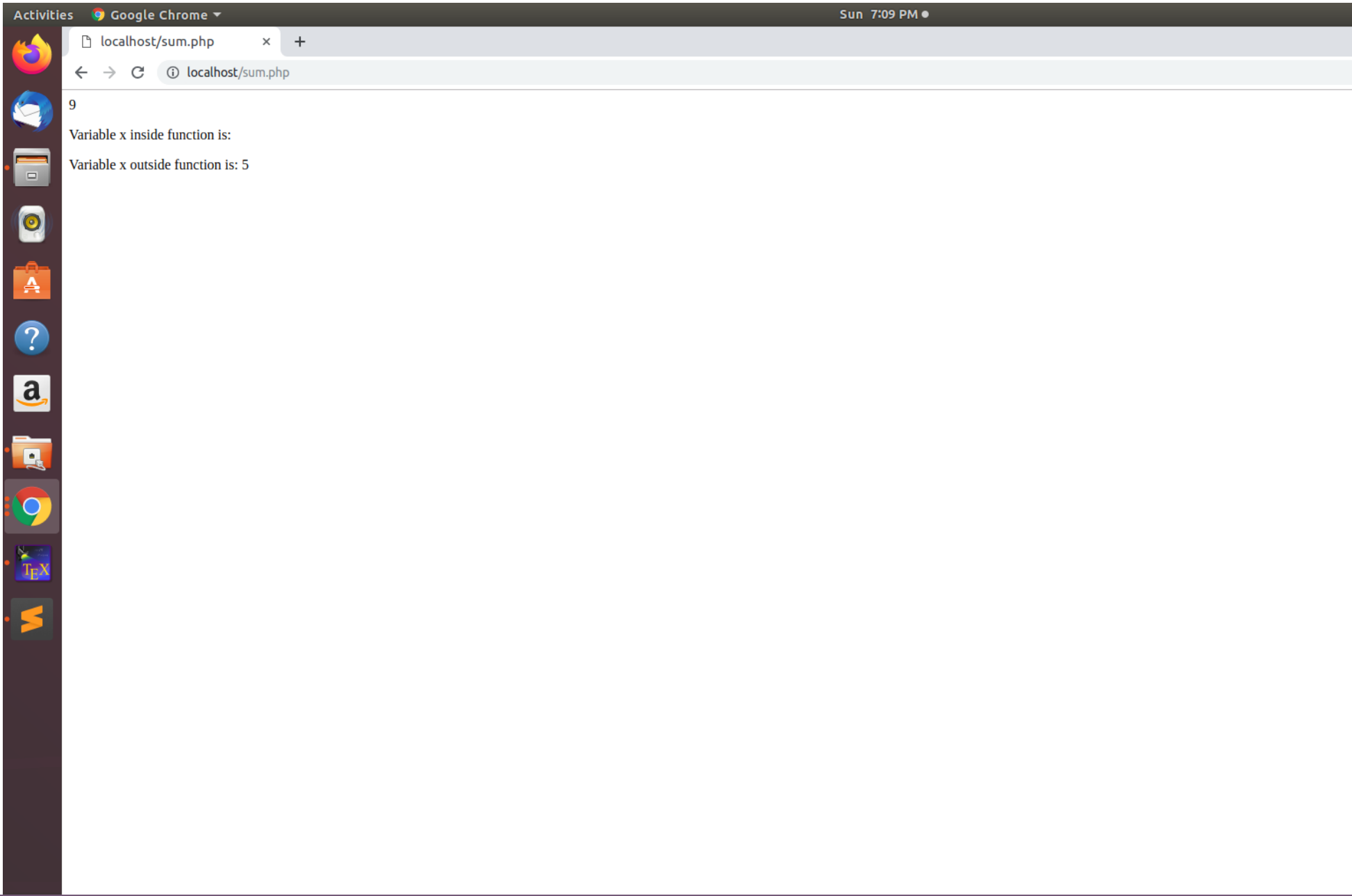


(a) local scope



(b) local scope variable

Figure 70: A figure with two subfigures


Activities
Google Chrome
Sun 7:09 PM
localhost/sum.php
localhost/sum.php
9
Variable x inside function is:
Variable x outside function is: 5

PHP Data Types

1. String
2. Integer
3. Float (floating point numbers - also called double)
4. Boolean
5. Array
6. Object
7. NULL
8. Resource

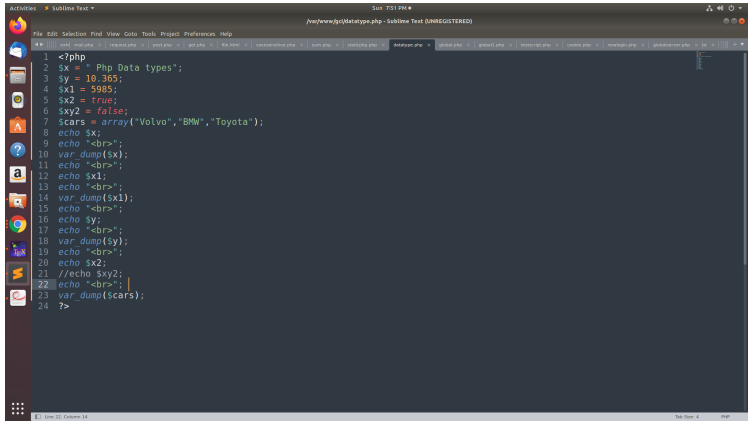


Figure 72: Datatypes

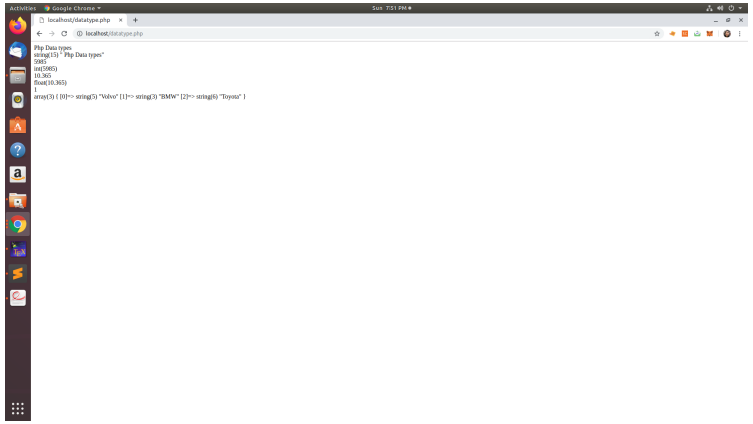


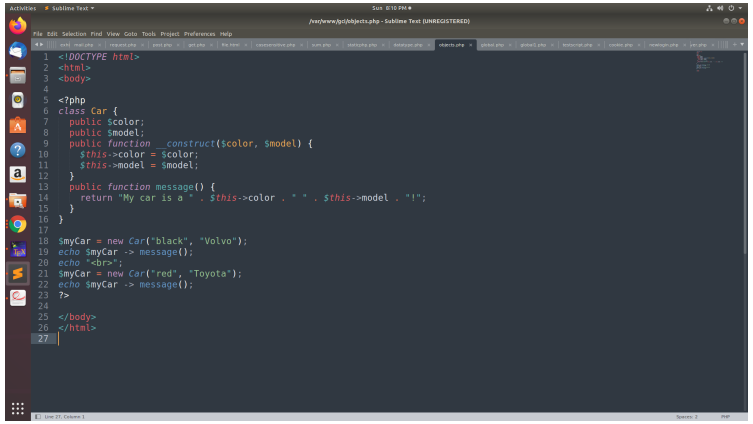
Figure 73: Datatypes

PHP Object

Classes and objects are the two main aspects of object-oriented programming. A class is a template for objects, and an object is an instance of a class. When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties. When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a _construct() function, PHP will automatically call this function when you create an object from a class.



```php
<!DOCTYPE html>
<html>
<body>

<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>

</body>
</html>
```

Figure 74: Classes and objects

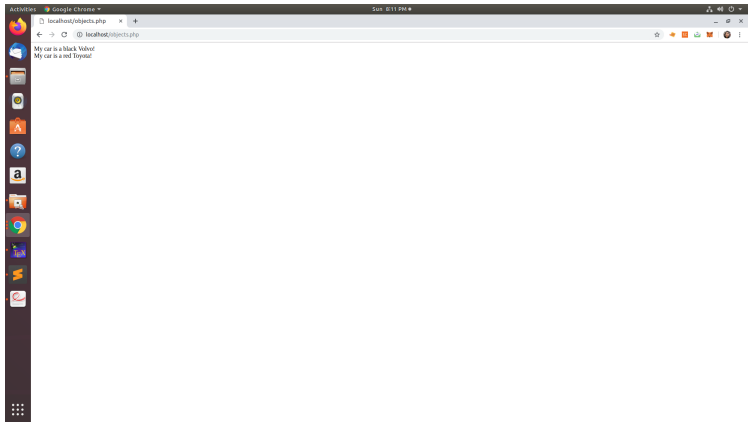


Figure 75: output

PHP Strings

1. The PHP strlen() function returns the length of a string.
2. The PHP str_word_count() function counts the number of words in a string.
3. The PHP strrev() function reverses a string.
4. strpos() - Search For a Text Within a String
5. str_replace() - Replace Text Within a String

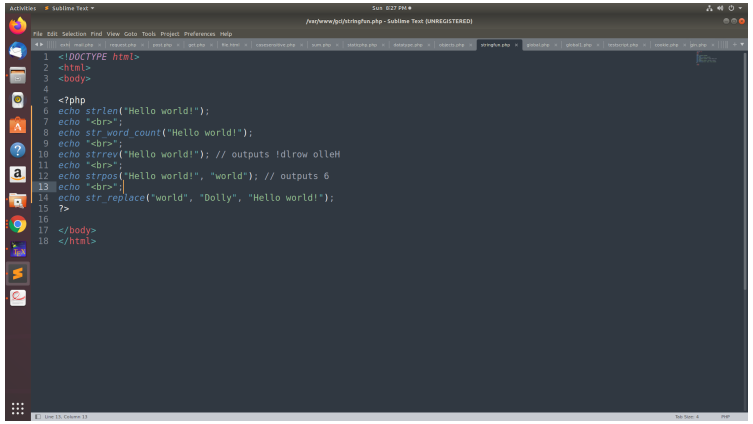


Figure 76: String Functions

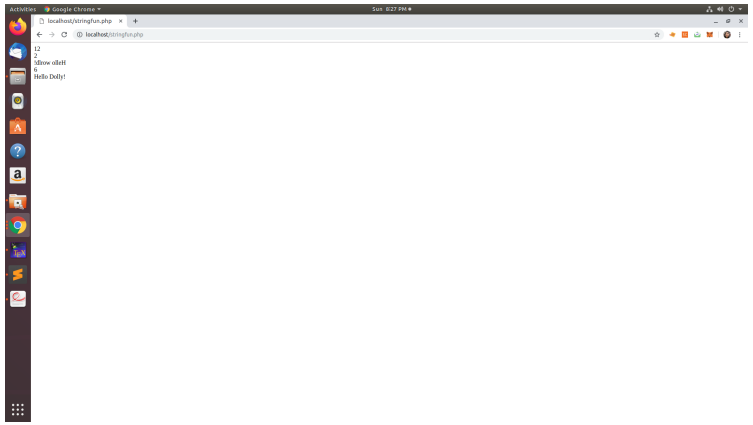


Figure 77: output

PHP Numbers

One thing to notice about PHP is that it provides automatic data type conversion. So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string. This automatic conversion can sometimes break your code.
PHP Integers 2, 256, -256, 10358, -179567 are all integers.
An integer is a number without any decimal part.
An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems, and between -9223372036854775808 and 9223372036854775807 in 64 bit systems.
A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
Note: Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

PhP: Server side scripting

A web server has many responsibilities beyond responding to requests for HTML files. These include

1. Handling HTTP connections, responding to requests for static and dynamic resources
2. Managing permissions and access for certain resources, encrypting and compressing data, managing multiple domains and URLs
3. managing database connections, cookies, and state, and uploading and managing files.

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when you first declare the variable:

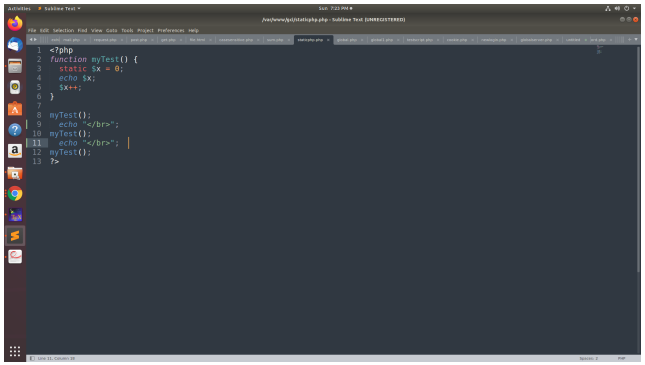


Figure 78: local variable output

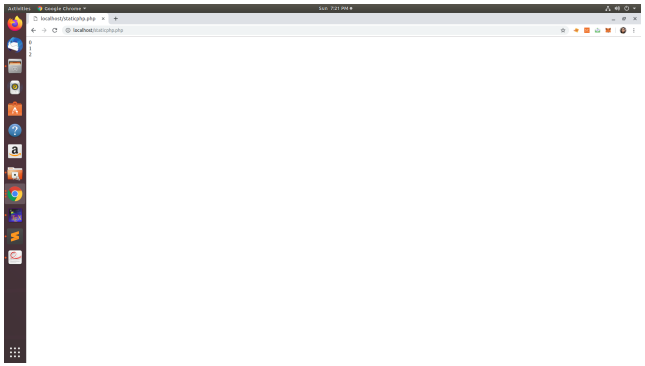


Figure 79: local variable output

PhP: Global variables

The PHP superglobal variables are:

1. GLOBALS
2. $_SERVER
3. $_REQUEST
4. $_POST
5. $_GET
6. $_FILES
7. $_ENV
8. $_COOKIE
9. $_SESSION

Get method

There are two ways the browser client can send information to the web server.

1. The GET Method
2. The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.
name1=value1&name2=value2&name3=value3

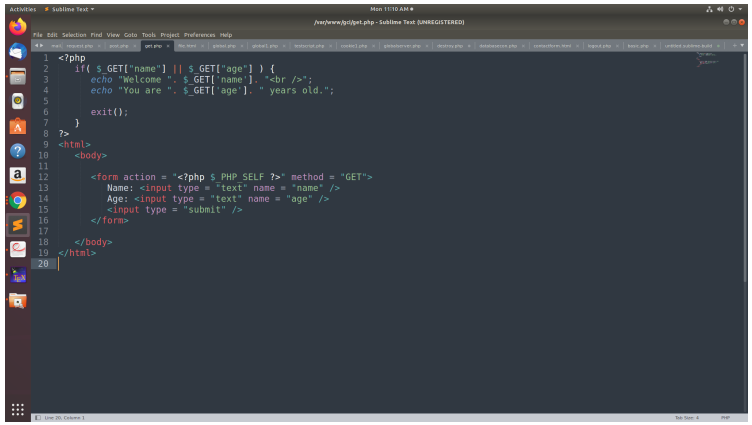


Figure 80: get method

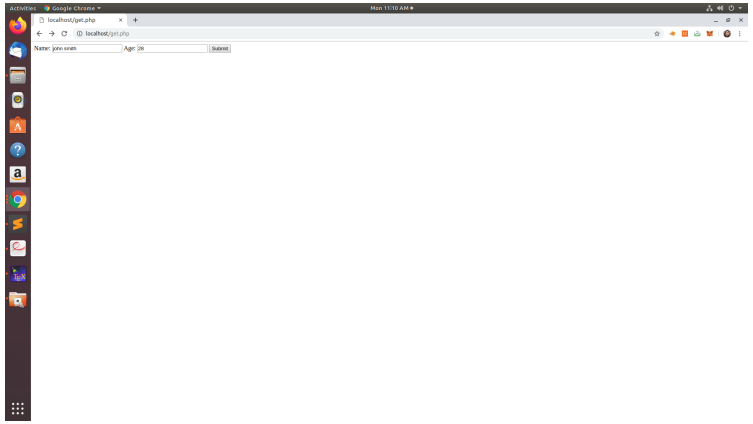


Figure 81: get method

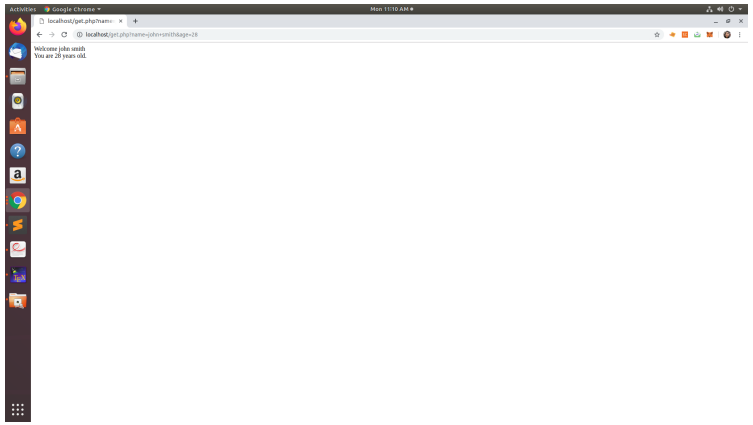


Figure 82: get method

POST Method

The POST Method The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides $_POST associative array to access all the sent information using POST method.

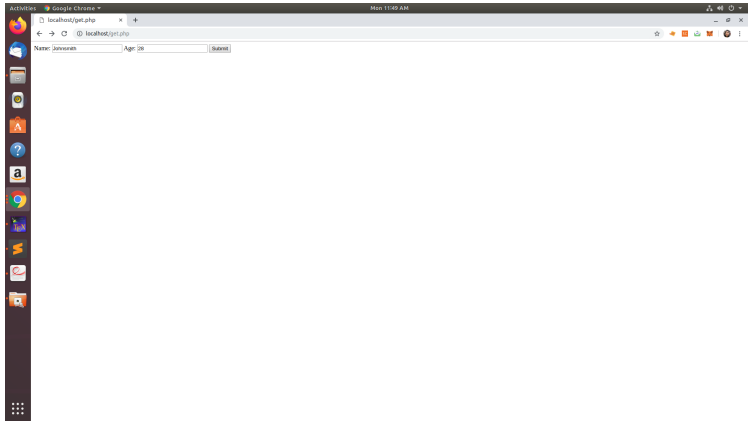


Figure 83: post method

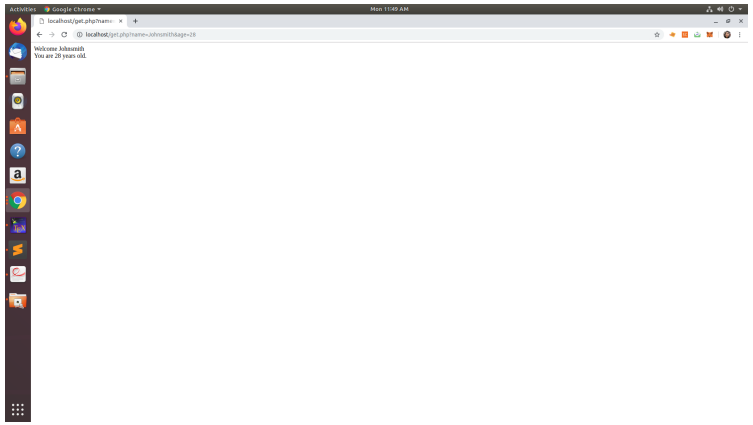
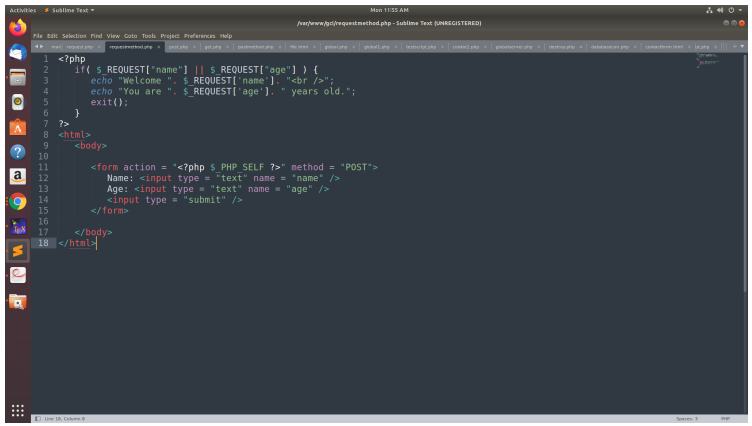


Figure 84: post method

The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.
The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.



```php
<?php
   if( $_REQUEST["name"] || $_REQUEST["age"] ) {
      echo "Welcome ". $_REQUEST['name']. "<br />";
      echo "You are ". $_REQUEST['age']. " years old.";
      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "POST">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

Figure 85: Request method

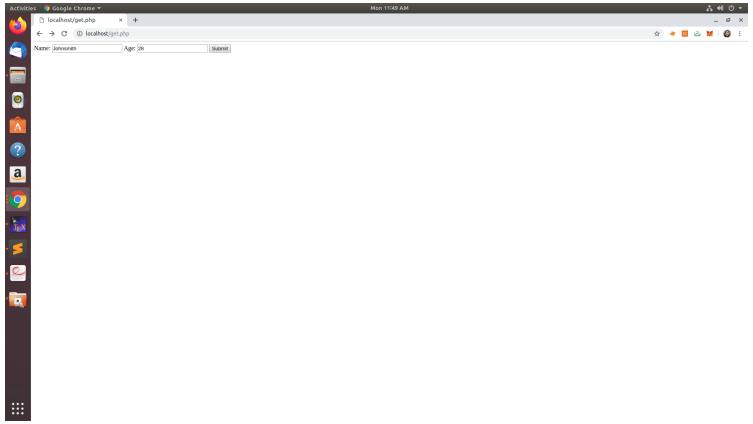


Figure 86: post method

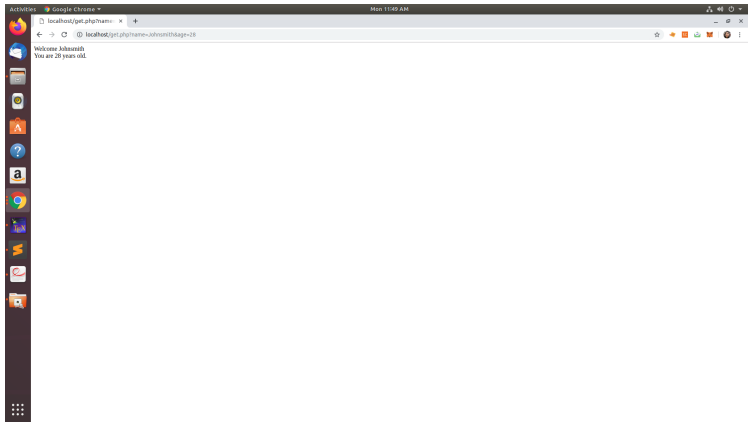


Figure 87: post method

HTTP_POST_FILES also contains the same information, but is not a superglobal, and now been deprecated
The _FILES array contains following properties -

1. _FILES['file']['name'] - The original name of the file to be uploaded.
2. _FILES['file']['type'] - The mime type of the file.
3. _FILES['file']['size'] - The size, in bytes, of the uploaded file.
4. _FILES['file']['tmp_name'] - The temporary filename of the file in which the uploaded file was stored on the server.
5. _FILES['file']['error'] - The error code associated with this file upload.

Cookie

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.
There are three steps involved in identifying returning users

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

- Name - This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- Value - This sets the value of the named variable and is the content that you actually want to store.
- Expiry - This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- Path - This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- Domain - This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- Security - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

cookie program

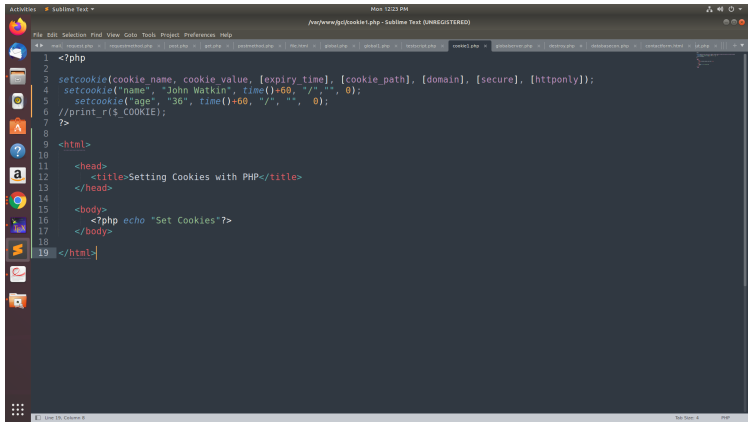


Figure 88: cookies

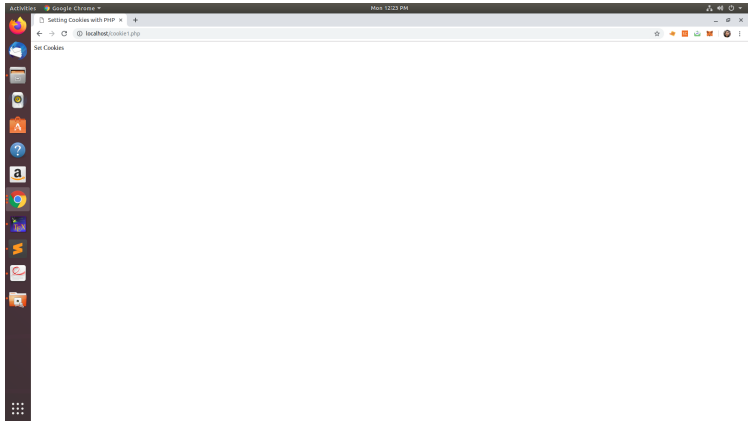
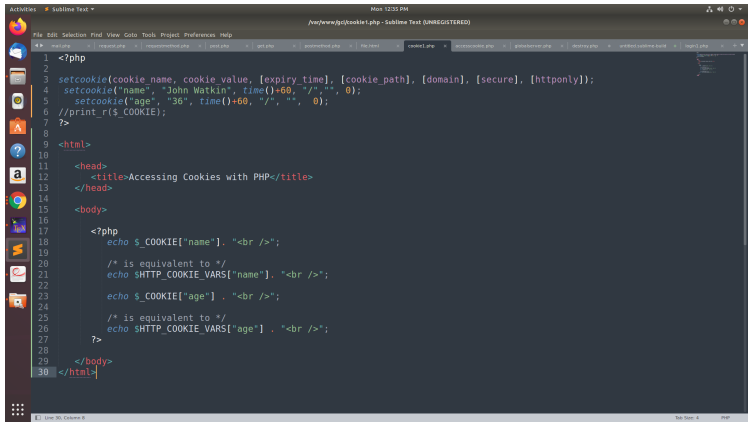


Figure 89: cookies output

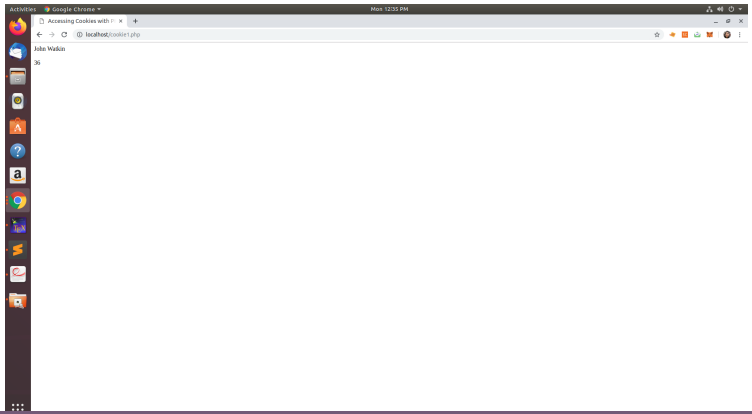
Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.



```php
<?php

setcookie(cookie_name, cookie_value, [expiry_time], [cookie_path], [domain], [secure], [httponly]);
 setcookie("name", "John Watkin", time()+60, "/","", 0);
   setcookie("age", "36", time()+60, "/", "",  0);
//print_r($_COOKIE);
?>

<html>

   <head>
      <title>Accessing Cookies with PHP</title>
   </head>

   <body>

      <?php
         echo $_COOKIE["name"]. "<br />";

         /* is equivalent to */
         echo $HTTP_COOKIE_VARS["name"]. "<br />";

         echo $_COOKIE["age"] . "<br />";

         /* is equivalent to */
         echo $HTTP_COOKIE_VARS["age"] . "<br />";
      ?>

   </body>
</html>
```

Figure 90: cookies



Delete Cookies with PHP

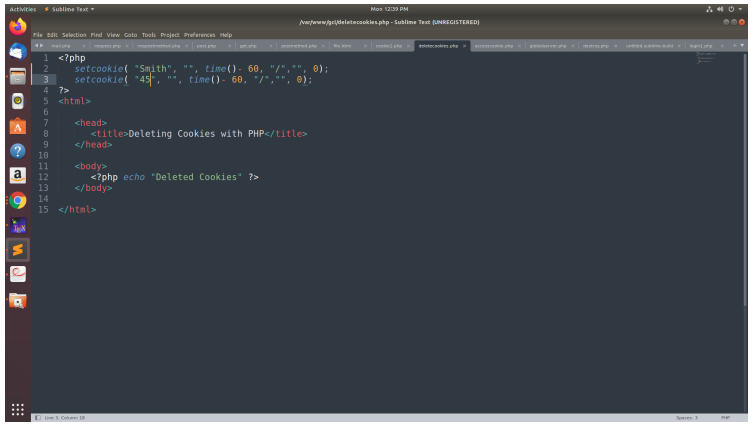


Figure 92: cookies

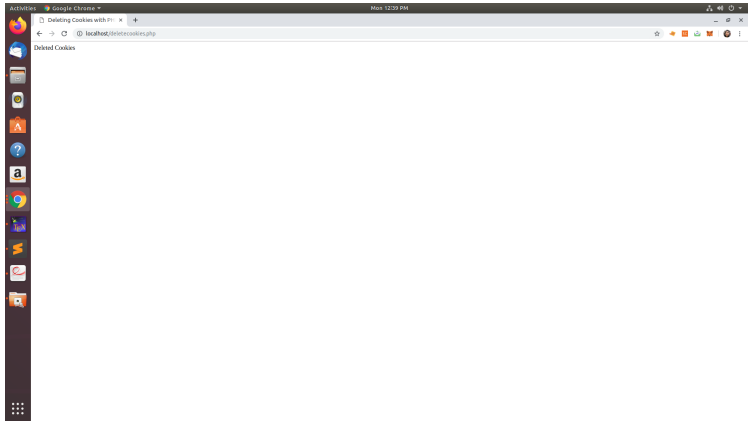


Figure 93: cookies output

session

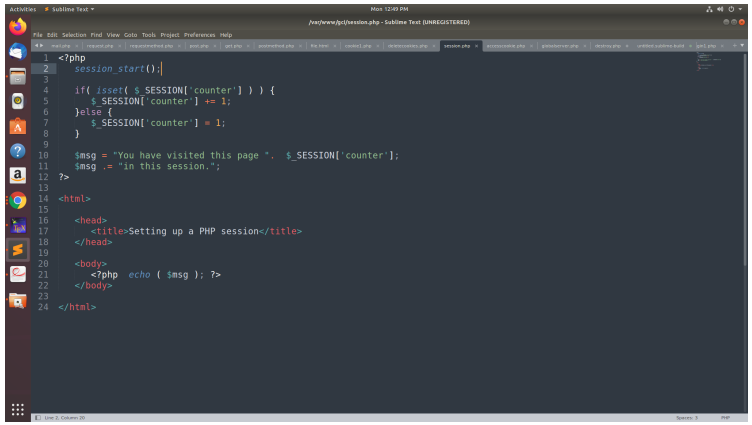


Figure 94: cookies

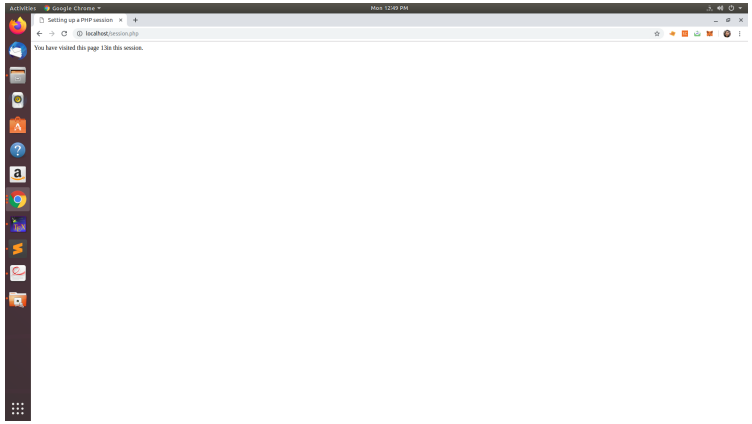


Figure 95: cookies output

AJAX

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.

- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

1. AJAX is based on JavaScript and HTTP requests.
2. AJAX is not a new programming language, but a new way to use existing standards with DOM manipulating.
3. JavaScript is help to a AJAX to make background AJAX calls for fetching certain amount of data.
4. Without AJAX, traditional web page takes a longer time to finishing round trip process for getting a data form the server. So it's time-consuming process even if small changes are in web page, entire web page reload.
5. In a traditional web page you can't update small portion without reloading page. So here Ajax is help us to background server calls for fetching data and update new contain without reloading page.
6. AJAX is a type of programming made popular in 2005 by Google (with Google Suggest).

Ajax behind Process (How to Work Ajax)

- HTML/CSS is website markup language for defining web page layout, such as fonts style and colors.
- JavaScript is a web scripting language. JavaScript special object XMLHttpRequest that was designed by Microsoft. XMLHttpRequest provides an easy way to retrieve data from web server without having to do full page refresh. Web page can update just part of the page without interrupting what the users are doing.
- Document Object Model (DOM) method provides a tree structure as a logical view of web page.
- XML is a format for retrieve any type of data, not just XML data from the web server. However you can use other formats such as Plain text, HTML or JSON (JavaScript Object Notation). and it supports protocols HTTP and FTP.

Benefits of Ajax

1. Using AJAX you can create better, faster, and more user-friendly web applications.
2. Ajax is based on JavaScript, CSS, HTML and XML etc. So you can easily learn.
3. Ajax behavior and works is like a desktop application. So Ajax use for creating a rich web application.

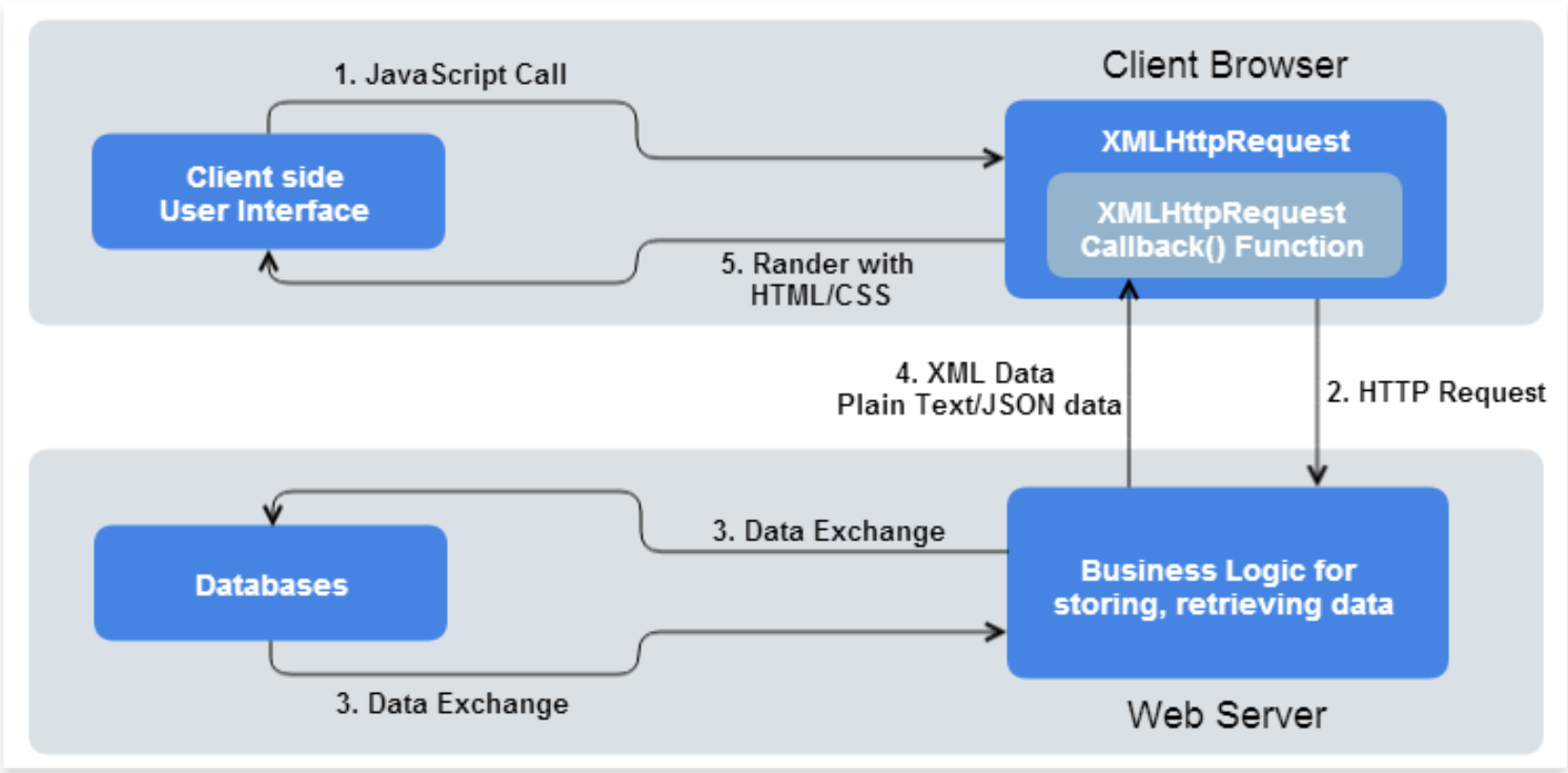


Figure 96: Ajax

1. Above figure (visual diagram) illustrates how AJAX technologies work together to handle user action. User action are triggers an AJAX response.
2. Client side user perform action to generate event that event call to a JavaScript function.
3. JavaScript function create XMLHttpRequest object, XMLHttpRequest object specify the JavaScript callback function.
4. JavaScript XMLHttpRequest object call as a asynchronous HTTP request to the Server.
5. Web Server process the request and return XML contain data.
6. XMLHttpRequest object calls to a callback function along with response from the web server.
7. Client browser updates the HTML DOM representing the web page along with new data.

Advantages of AJAX

- AJAX Concept Before you starting AJAX you'll need to have a strong knowledge of JavaScript. AJAX is not a difficult, you can easy implement AJAX in a meaningful manner. Some IDE are help us to implement AJAX.
- Speed Reduce the server traffic in both side request. Also reducing the time consuming on both side response.
- Interaction AJAX is much responsive, whole page(small amount of) data transfer at a time.
- XMLHttpRequest has an important role in the Ajax web development technique. XMLHttpRequest is special JavaScript object that was designed by Microsoft. XMLHttpRequest object call as a asynchronous HTTP request to the Server for transferring data both side. It's used for making requests to the non-Ajax pages.
- Asynchronous calls AJAX make asynchronous calls to a web server. This means client browsers are avoid waiting for all data arrive before start the rendering.
- Form Validation This is the biggest advantage. Form are common element in web page. Validation should be instant and properly, AJAX gives you all of that, and more.

Disadvantages of AJAX

- AJAX application would be a mistake because search engines would not be able to index an AJAX application.
- Open Source: View source is allowed and anyone can view the code source written for AJAX.
- ActiveX requests are enabled only in Internet Explorer and newer latest browser.
- The last disadvantage, XMLHttpRequest object itself. For a security reason you can only use to access information from the web host that serves initial pages. If you need to fetching information from another server, it's is not possible with in the AJAX.

AJAX XMLHttpRequest Object

XMLHttpRequest object is an API for fetching any text base format data, including XML without user/visual interruptions. All most all browser platform support XML-HttpRequest object to make HTTP requests. Using Ajax XMLHttpRequest object you can make many things easier. So many new things can't possible using HEAD request. This object allows you to making HTTP requests and receive responses from the server in the background, without requiring the user to submit the page to the server (without round trip process).

Using DOM to manipulate received data from the server and make responsive contents are added into live page without user/visual interruptions.

Using this object you can make very user interactive web application.

Following are sequence of step for working with XMLHttpRequest object:

1. Define instance of this XMLHttpRequest.
2. Create a asynchronous call to a server page, also defining a callback function that will automatically execute when the server response is received.
3. Callback function getting server response.
4. DOM manipulate received data and added into live page.

XMLHttpRequest

An object of XMLHttpRequest is used for asynchronous communication between client and server. It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it

Properties of XMLHttpRequest object

The common properties of XMLHttpRequest object are as follows:

Property	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	represents the state of the request. It ranges from 0 to 4. **0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	returns response as text.
responseXML	returns response as XML

Figure 97: Ajax properties

Properties of XMLHttpRequest object

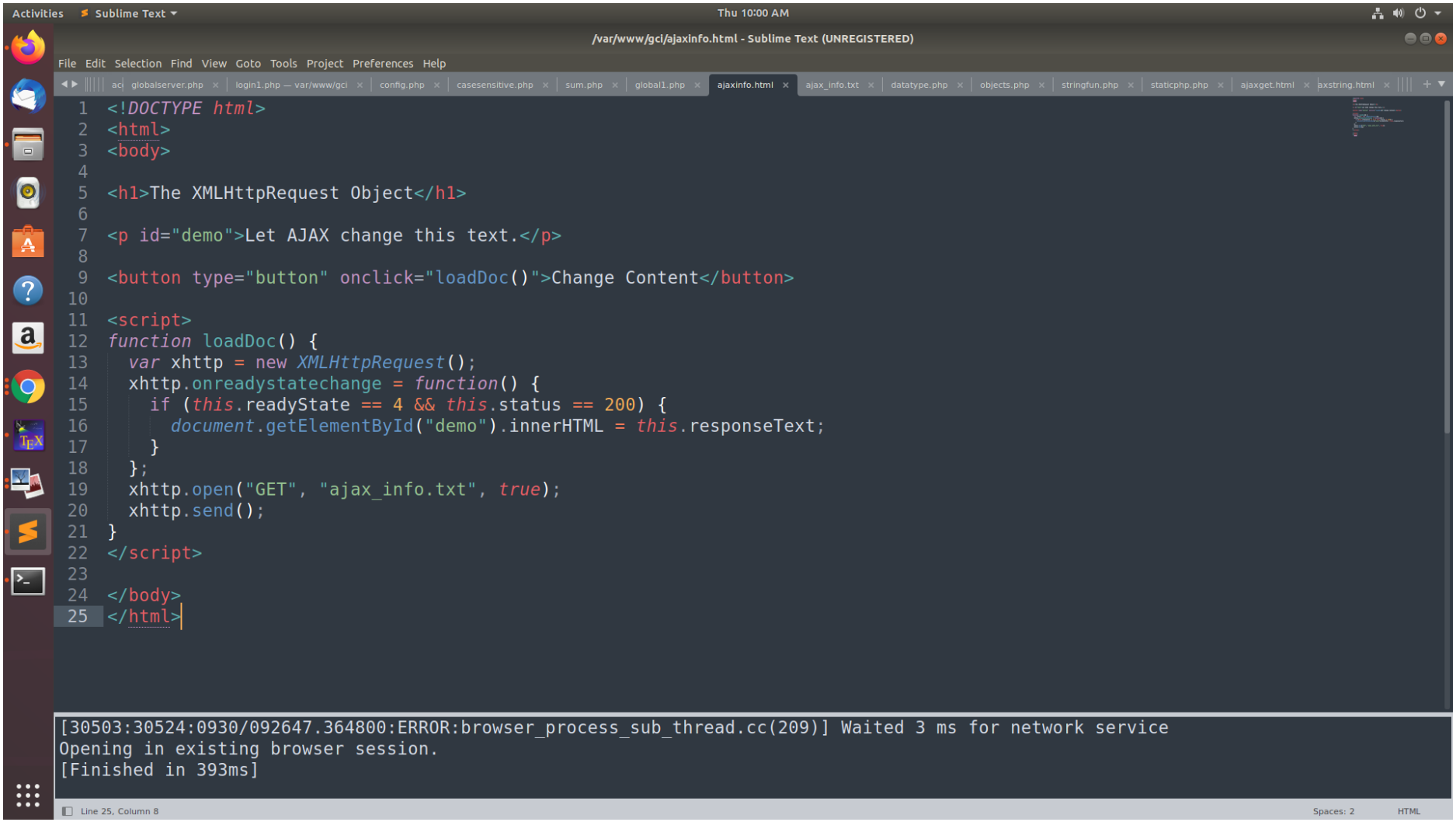
The common properties of XMLHttpRequest object are as follows:

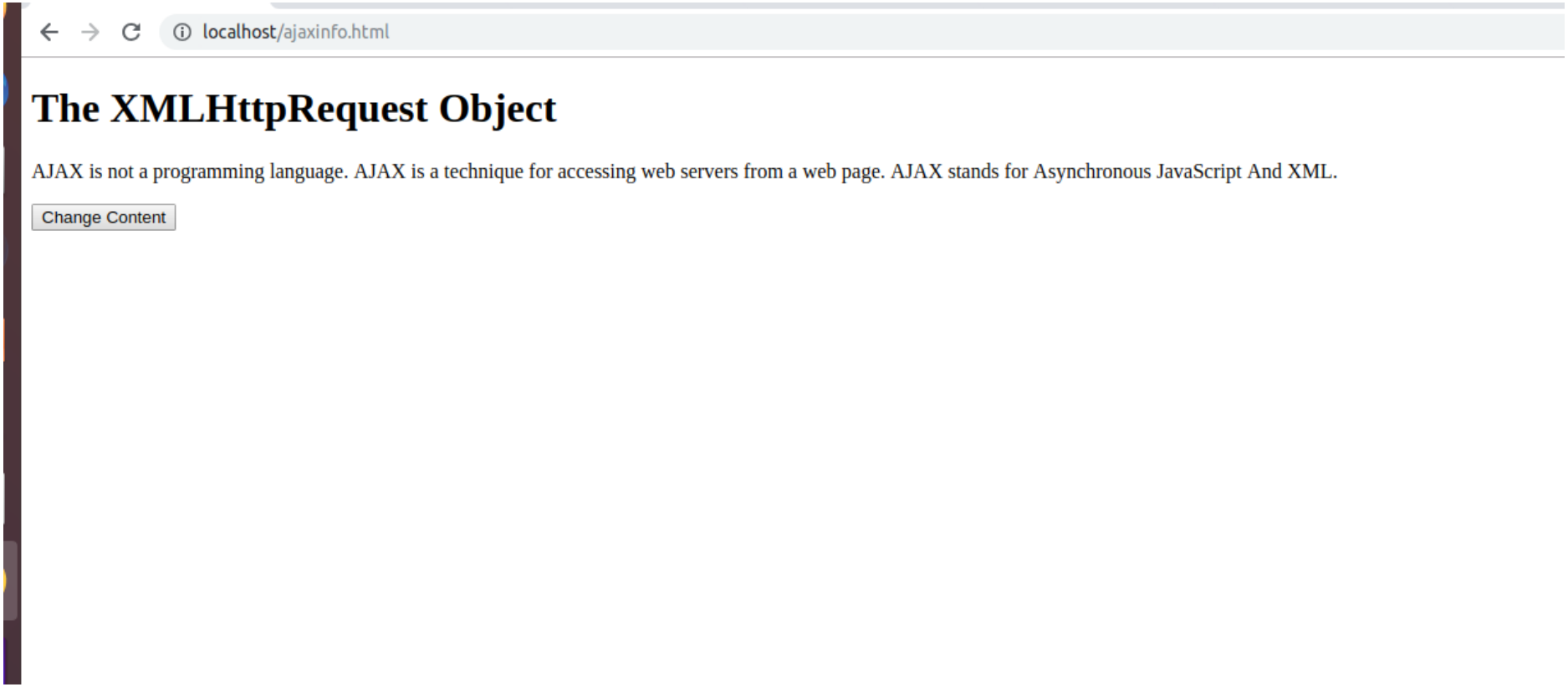
Method	Description
open(*method*, *url*, *async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

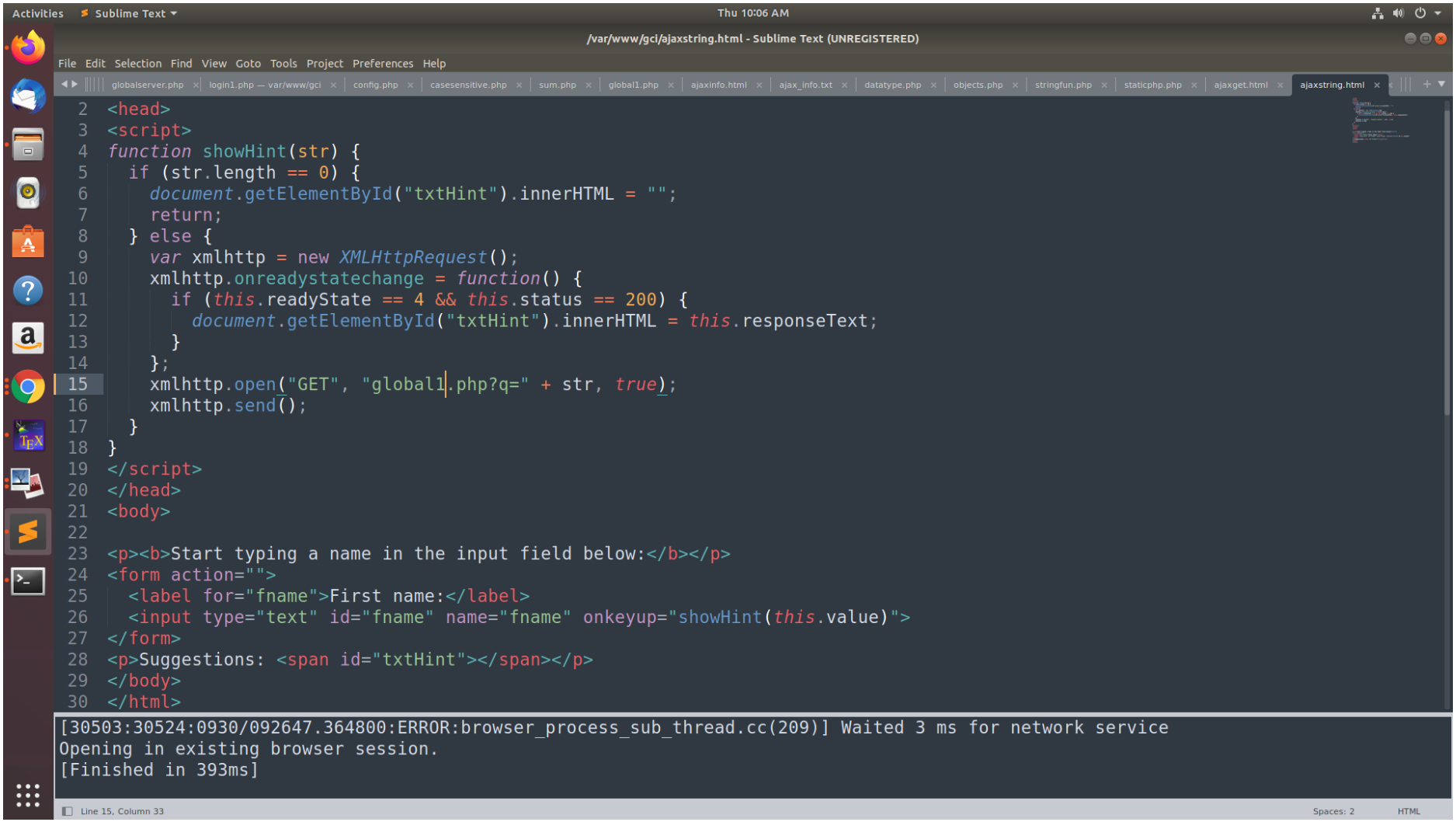
Figure 98: Ajax properties

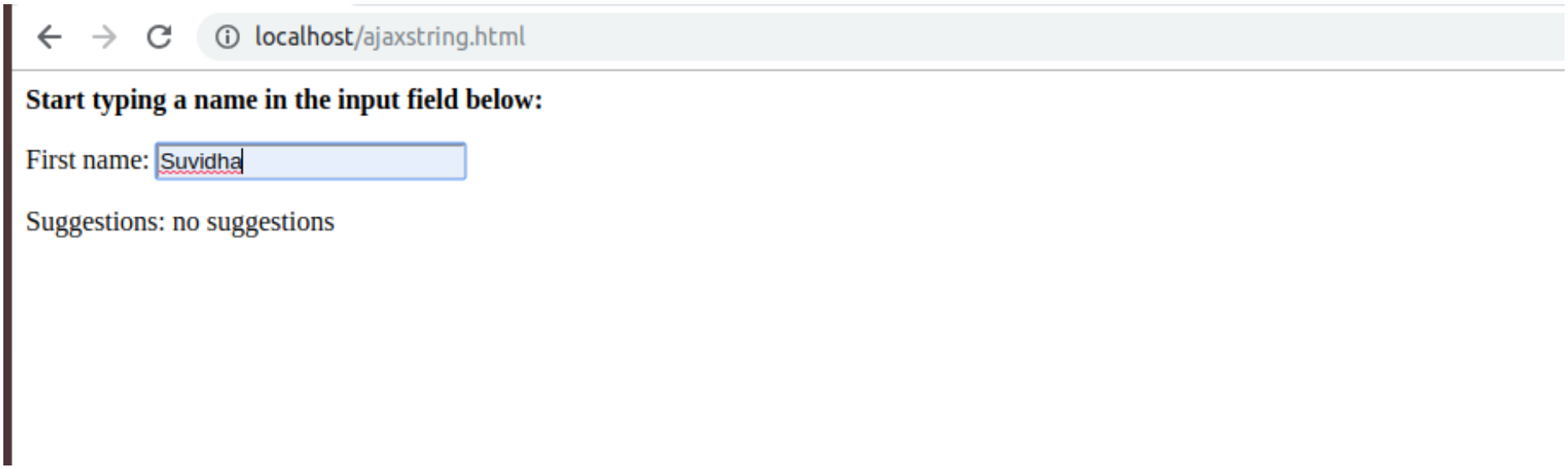
Send a Request To a Server

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```


Activities
Sublime Text
Thu 10:00 AM
/var/www/gci/ajaxinfo.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
globalserver.php
login1.php — var/www/gci
config.php
casesensitive.php
sum.php
global1.php
ajaxinfo.html
ajax_info.txt
datatype.php
objects.php
stringfun.php
staticphp.php
ajaxget.html
axstring.html
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>The XMLHttpRequest Object</h1>
6
7 Let AJAX change this text.
8
9 <button type="button" onclick="loadDoc()">Change Content</button>
10
11 <script>
12 function loadDoc() {
13 var xhttp = new XMLHttpRequest();
14 xhttp.onreadystatechange = function() {
15 if (this.readyState == 4 && this.status == 200) {
16 document.getElementById("demo").innerHTML = this.responseText;
17 }
18 };
19 xhttp.open("GET", "ajax_info.txt", true);
20 xhttp.send();
21 }
22 </script>
23
24 </body>
25 </html>
[30503:30524:0930/092647.364800:ERROR:browser_process_sub_thread.cc(209)] Waited 3 ms for network service
Opening in existing browser session.
[Finished in 393ms]
Line 25, Column 8
Spaces: 2
HTML


localhost/ajaxinfo.html
The XMLHttpRequest Object
AJAX is not a programming language. AJAX is a technique for accessing web servers from a web page. AJAX stands for Asynchronous JavaScript And XML.
Change Content


Activities
Sublime Text
Thu 10:06 AM
/var/www/gci/ajaxstring.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
globalserver.php
login1.php — var/www/gci
config.php
casesensitive.php
sum.php
global1.php
ajaxinfo.html
ajax_info.txt
datatype.php
objects.php
stringfun.php
staticphp.php
ajaxget.html
ajaxstring.html
2 <head>
3 <script>
4 function showHint(str) {
5 if (str.length == 0) {
6 document.getElementById("txtHint").innerHTML = "";
7 return;
8 } else {
9 var xmlhttp = new XMLHttpRequest();
10 xmlhttp.onreadystatechange = function() {
11 if (this.readyState == 4 && this.status == 200) {
12 document.getElementById("txtHint").innerHTML = this.responseText;
13 }
14 };
15 xmlhttp.open("GET", "global1.php?q=" + str, true);
16 xmlhttp.send();
17 }
18 }
19 </script>
20 </head>
21 <body>
22
23 Start typing a name in the input field below:
24 <form action="">
25 <label for="fname">First name:</label>
26 <input type="text" id="fname" name="fname" onkeyup="showHint(this.value)">
27 </form>
28 Suggestions:
29 </body>
30 </html>
[30503:30524:0930/092647.364800:ERROR:browser_process_sub_thread.cc(209)] Waited 3 ms for network service
Opening in existing browser session.
[Finished in 393ms]
Line 15, Column 33
Spaces: 2
HTML


localhost/ajaxstring.html
Start typing a name in the input field below:
First name: Suvidha
Suggestions: no suggestions


THANK YOU