

ADC 502

WEB DEVELOPMENT

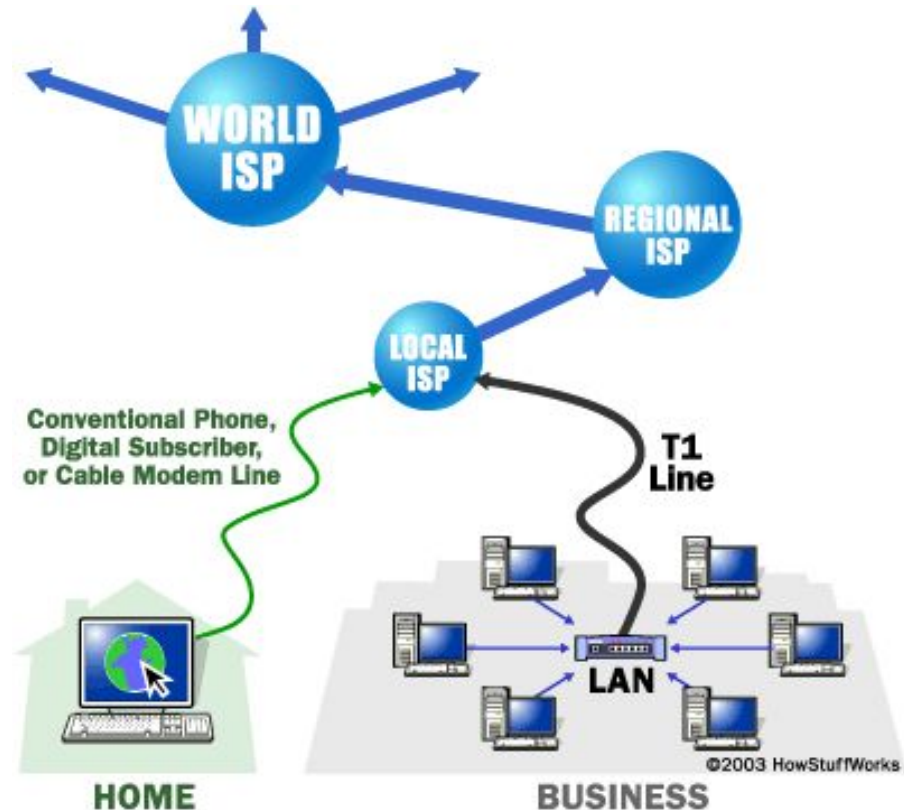
Instructor: Amit Singh

□ What is the internet?

What is the internet?

3

- Global network of billions of computers and other electronic devices that use a protocol to exchange data.
- Possible to access almost any information, communicate with anyone else in the world, and do much more.



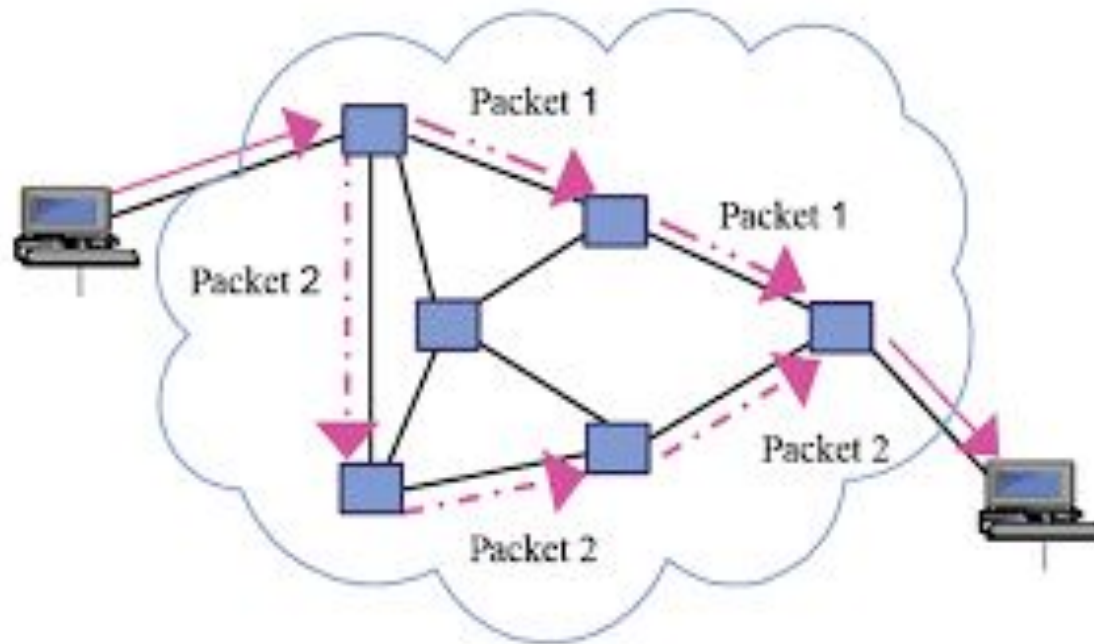
Brief history

4

- ❑ ARPANET was introduced in the year 1969 by Advanced Research Projects Agency (ARPA) of US Department of Defense.
- ❑ ARPANET's main use was for academic and research purposes.
- ❑ It was established using a bunch of PCs at various colleges and sharing of information and messages was done.
- ❑ Packet switching Network
- ❑ By 1987: Internet includes nearly 30,000 hosts

Brief history

5



Brief history (cont.)

6

- WWW created in 1989-91 by Tim Berners-Lee
- Popular web browsers released:
 - Netscape 1994
 - IE 1995
- Amazon.com opens in 1995
- Google January 1996
- Wikipedia launched in 2001
- MySpace opens in 2003
- Facebook February 2004



tweakers.net

SECOND LIFE

friendster

SMULWEB

Slashdot

မိမိ

Hyves.nl

PARTY FLOCK

myspace.com

orkut

del.icio.us



KOESKEURIG
interactieve koopgids

generationnext

SENIORWEB

Ouders Online

XING

listible

YouTube

twitter

Faceparty

upcoming

mypunchbowl

xanga.com

flickr

THE WELL

LinkedIn

vrouwonline

校内网
xiaonei.com

五座

ChinaRen

ryze
Business Networking

last.fm

wink

飽靈樓

beto2

China.y.com

NICKTROPOLES

why rnb rocks.com



5Q校园网
www.5q.com

谊多
www.eDorm.cn

底片网

facebook

VOX

EBO

iJOURNAL

DORM99.com

YEEJEE

neopets

tribe

XUQACM

FaceRen

YuMe netvibes

Blogger



jalku

Techmeme

RATE IT ALL

dodgeball

HABBO

digg

YAHOO! GEOCITIES

SchoolBANK

Key aspects of the internet

8

- Sub-networks are independent
- Computers can dynamically join and leave the network
- Built on open standards
- Everyone can use it with simple, commonly available software

People and organizations

9

- Internet Engineering Task Force (IETF): internet protocol standards
- Internet Corporation for Assigned Names and Numbers (ICANN): decides top-level domain names
- World Wide Web Consortium (W3C): web standards



Web Servers

10

- Web server: software that listens for web page requests
 - Apache
 - Microsoft Internet Information Server (IIS)



Application Server

11

- Software framework that provides an environment where applications can run
 - Apache
 - Glassfish
 - WebSphere
 - WebLogic



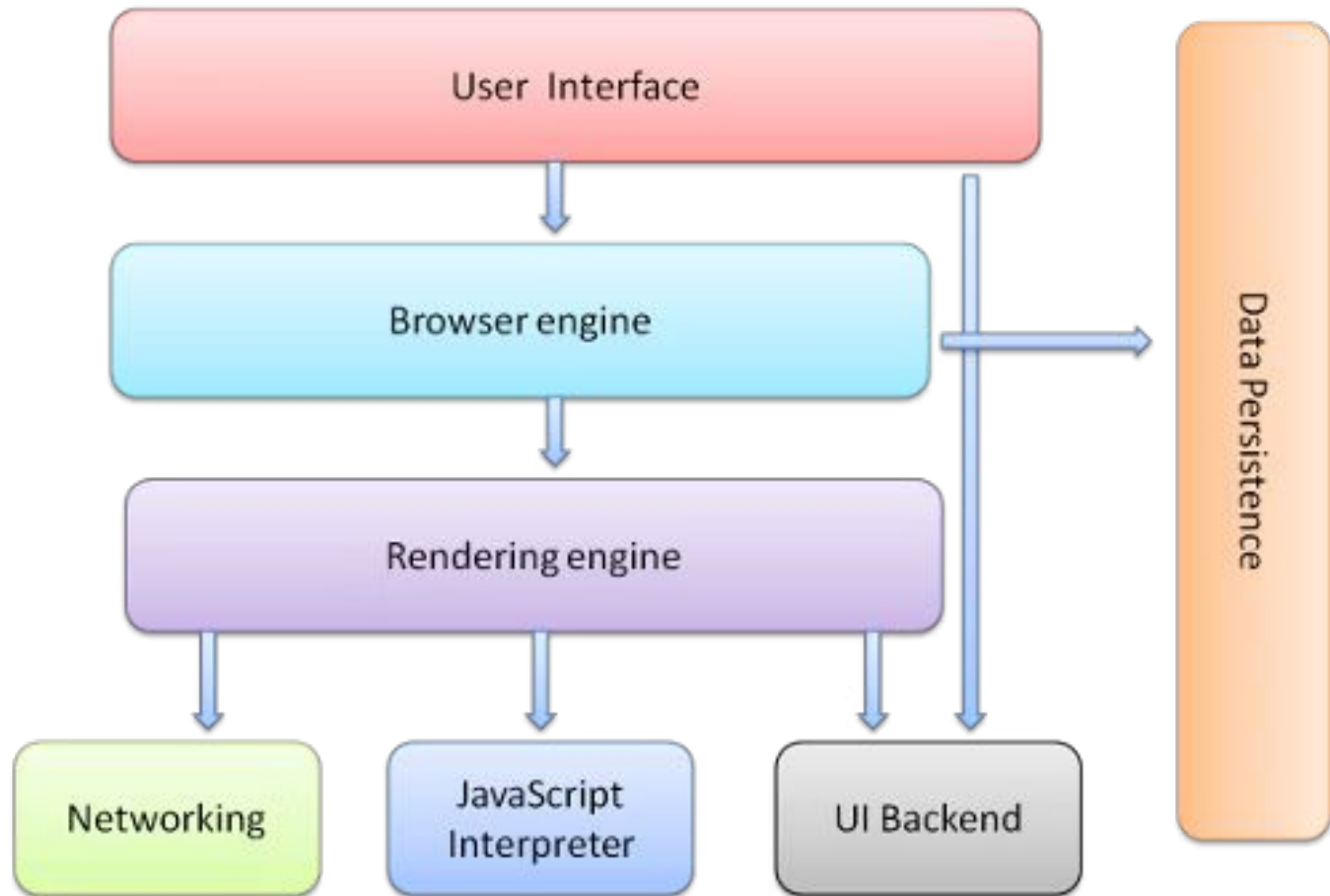
Web Browser

12

- Type of software that allows you to find and view websites from web servers on the Internet.
 - Mozilla Firefox
 - Microsoft Internet Explorer (IE)
 - Apple Safari
 - Google Chrome
 - Opera

Web Browser

13



Web Browser

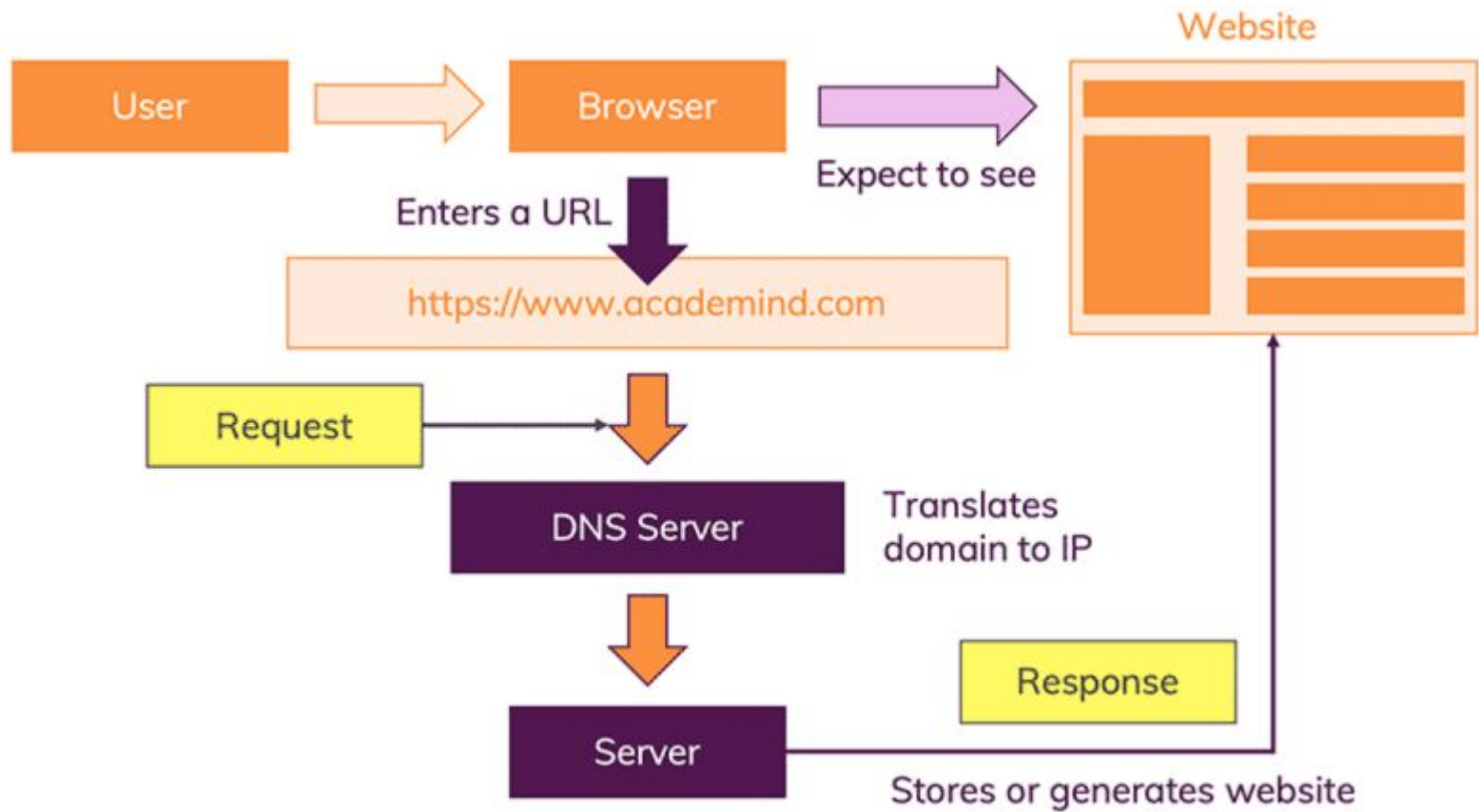
14

- The user interface: this includes the address bar, back/forward button, bookmarking menu, etc.
- The browser engine: marshals actions between the UI and the rendering engine.
- The rendering engine: responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.

Web Browser

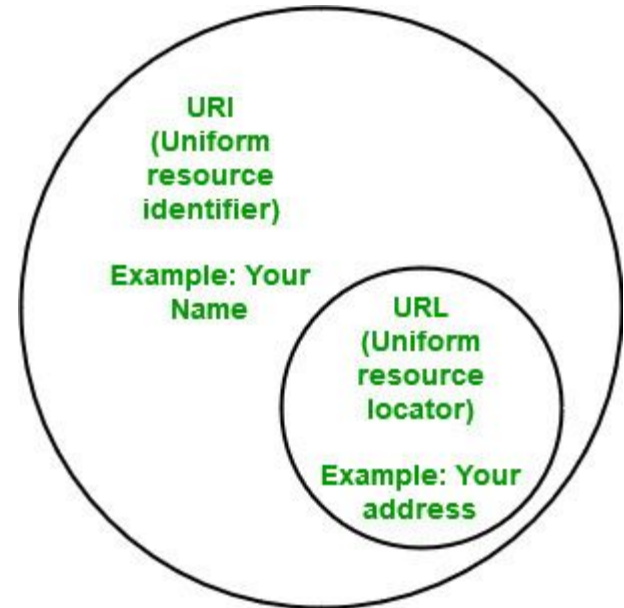
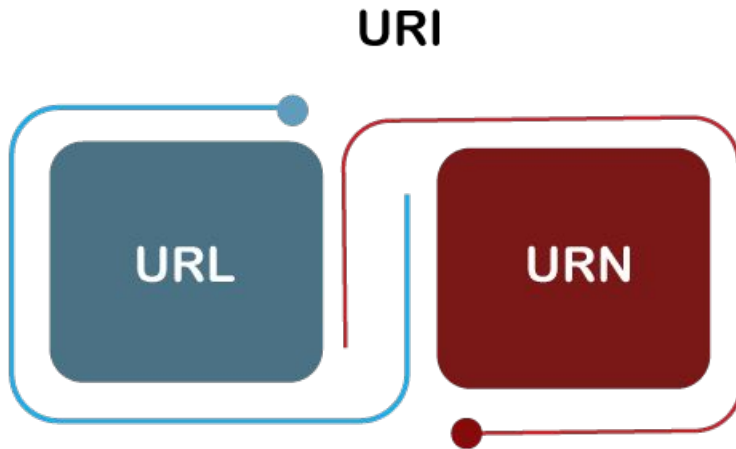
15

- Networking: for network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.
- UI backend: used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.
- JavaScript interpreter. Used to parse and execute JavaScript code.
- Data storage. This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem.



Uniform Resource Identifier (URI)

17



<https://auth0.com/blog/url-uri-urn-differences/>

URI, URL, AND URN

URIs

<https://myapi.com>

URLs

<https://auth0.com/blog>

URNs

<urn:isbn:1234567890>

URI

Uniform Resource Identifier

Identify a resource

All URLs and URNs are URIs

Can use any scheme
(http, https, file, etc.)

You can create your own URI,
although using a registered
domain name is recommended

URL

Uniform Resource Locator

Identify and locate a resource

All URLs are URIs.
Not all URIs are URLs.
URLs are not URNs.

The scheme specifies the
protocol to access the resource

You can create your own URL,
provided you control its
domain name

URN

Uniform Resource Name

Identify a resource

All URNs are URIs.
Not all URIs are URNs.
URNs are not URLs.

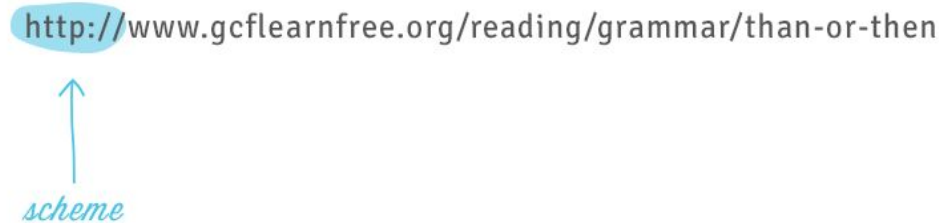
Use the *urn* scheme

URNs are usually assigned
by a specific standard
organizations

Uniform Resource Locator (URL)

19

- **Scheme:**
- Every URL begins with the Scheme.
- tells your browser what type of address it is so the browser connects to it correctly.
- http and https

The diagram shows the URL `http://www.gcflernfree.org/reading/grammar/than-or-then`. The text `http://` is enclosed in a light blue oval. A light blue arrow points from the word *scheme* (written in a light blue script font) below to the `http://` part of the URL.

Uniform Resource Locator (URL)

20

- ❑ **Domain name:**
- ❑ The most prominent part of a web address.
- ❑ Typically, different pages on the same site will continue to use the same domain name.

http://www.gcflearnfree.org/reading/grammar/than-or-then

↑
domain name

www.GCFLearnfree.org

↑ ↑ ↑
sub-domain *second level* *top level*

Uniform Resource Locator (URL)

21

- **File path:**
- often just called the path—tells your browser to load a specific page.
- **Parameters:**
- The parameter string can be clear or confusing to a human user, but it is critical information for the server.

<http://www.gcflearnfree.org/reading/grammar/than-or-then>

↑
file path

www.youtube.com/watch?v=dQw4w9WgXcQ

↑
parameters

Uniform Resource Locator (URL)

22

- **Anchor**
- Tells your browser to scroll to or load a specific part of the page.
- Begins with a hashtag and is used to direct your browser to a specific part of a very long page.
- Like a bookmark.

tolkiengateway.net/wiki/J.R.R._Tolkien#Writing

↑
anchor

http://en.wikipedia.org/w/index.php?title=Burrito#Breakfast_burrito

↑
scheme

↑
domain name

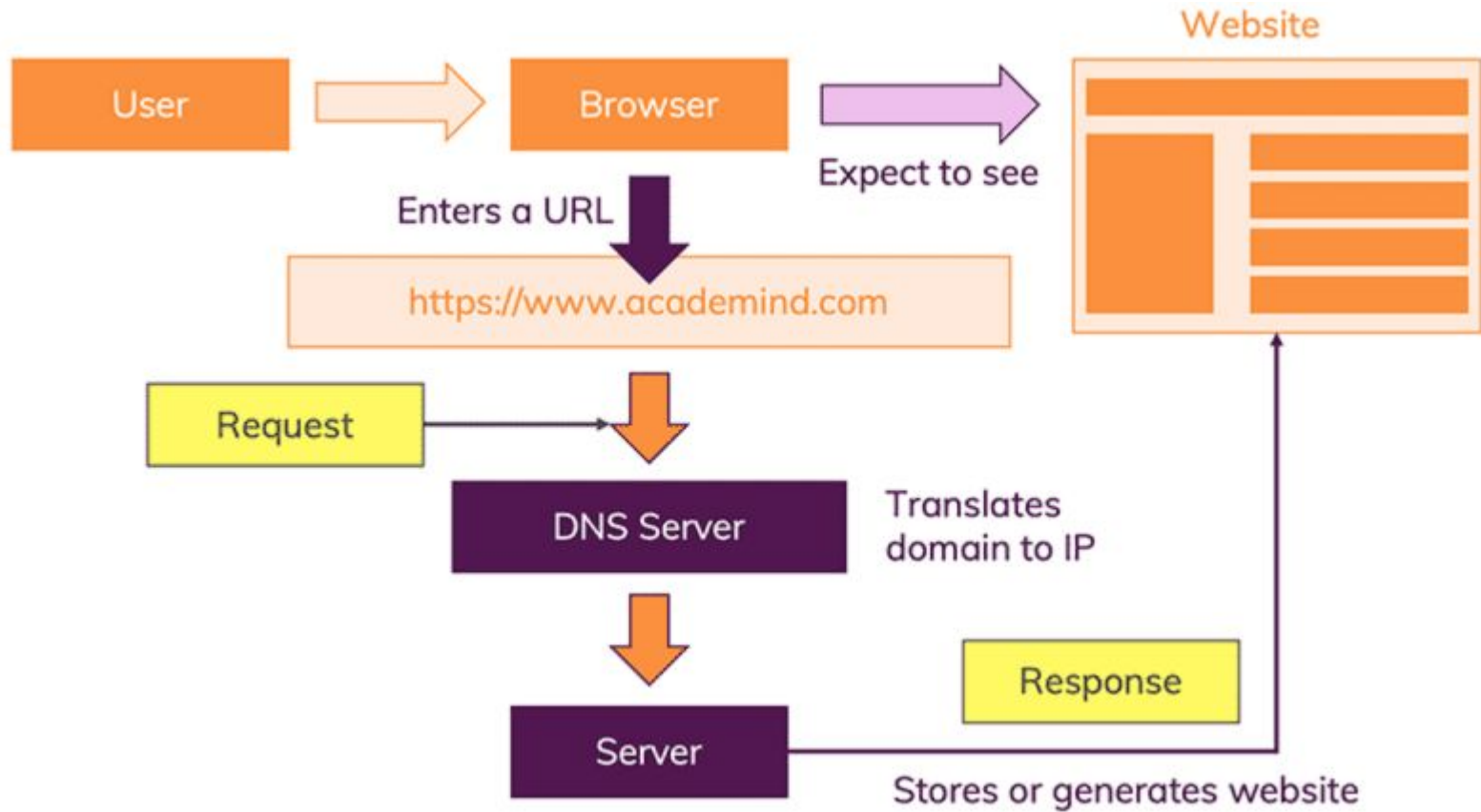
↑
file path

↑
parameters

↑
anchor

Quick Glance

23



Domain Name Server (DNS)

24

- Step 1: URL Gets Resolved
- The website code not stored on your machine and hence needs to be fetched from another computer where it is stored. This "other computer" is called a "server". Because it serves some purpose, in our case, it serves the website.
- The server which hosts the source code of a website, is identified via IP addresses. The browser sends a "request" to the server with the IP address you entered.

Domain Name Server (DNS)

25

- Translates domains to IP addresses.
- Huge dictionaries that store translation tables:
Domain => IP address.
 - Example: ju.edu → **204.29.160.73**
- Many systems maintain a local cache called a hosts file
 - Windows:
C:\Windows\system32\drivers\etc\hosts
 - Mac: /private/etc/hosts
 - Linux: /etc/hosts

Hypertext Transport Protocol (HTTP)

26

- Step 2 - Request Is Sent
- With the IP address resolved, the browser goes ahead and makes a request to the server with that IP address.
- The browser bundles up a bunch of information (What's the exact URL? Which kind of request should be made? Should metadata be attached) and sends that data package to the IP address.

Hypertext Transport Protocol (HTTP)

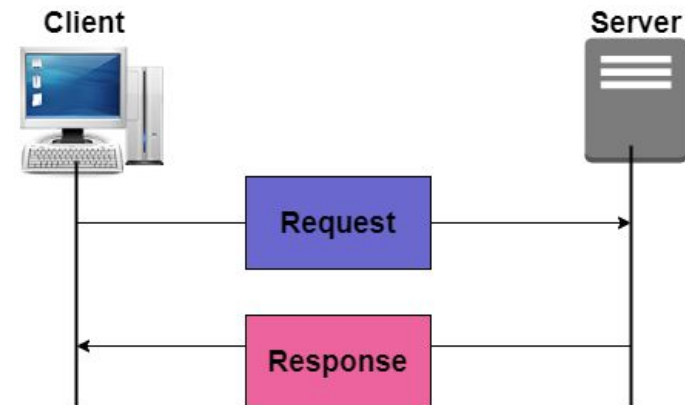
27

- HTTP stands for HyperText Transfer Protocol.
- Networking protocols are a set of rules for transferring data from one computer to another.
- Invented by Tim Berner.
- HTTP/2 is was published in 2015.
- HTTP/3 is was published in 2022.
- These protocols are used for communication between web server and (web) client.

Hypertext Transport Protocol (HTTP)

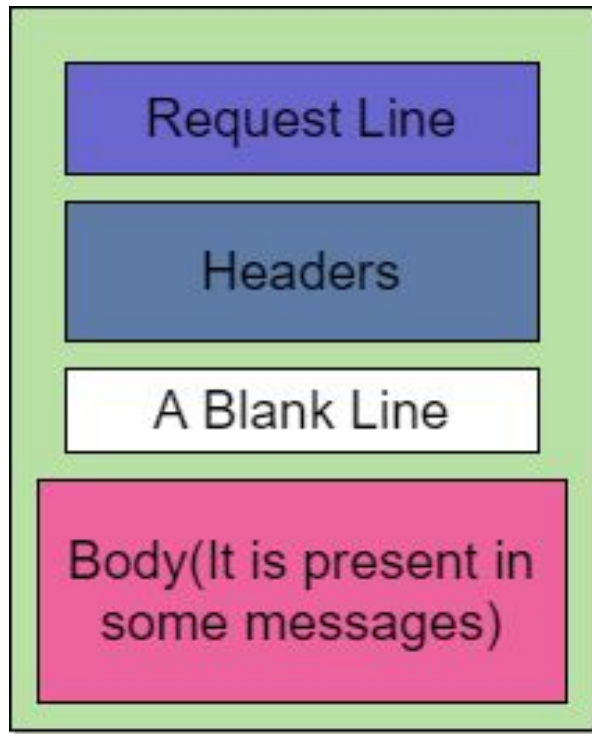
28

- HTTP requests:
- An HTTP request is made by a client, to a named host, which is located on a server. The aim of the request is to access a resource on the server.
- HTTP responses:
- An HTTP response is made by a server to a client. The aim of the response is to provide the client with the resource it requested, or inform the client that the action it requested has been carried out; or else to inform the client that an error occurred in processing its request.

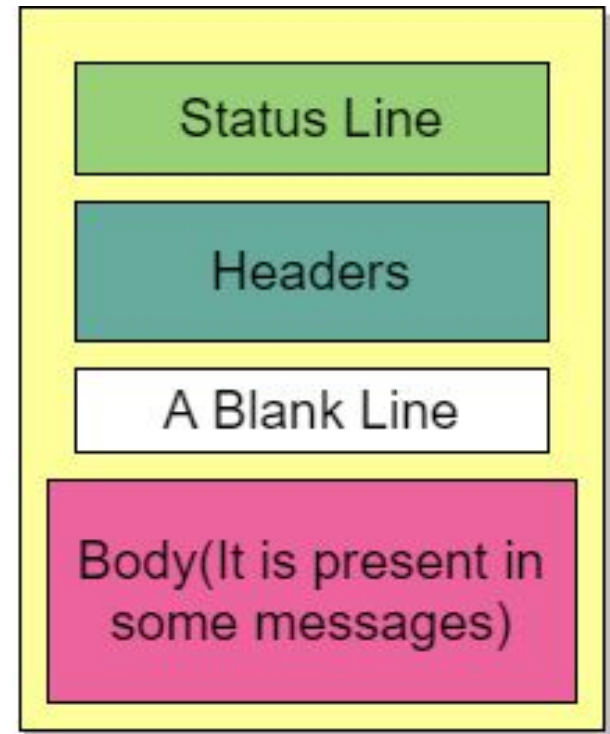


Hypertext Transport Protocol (HTTP)

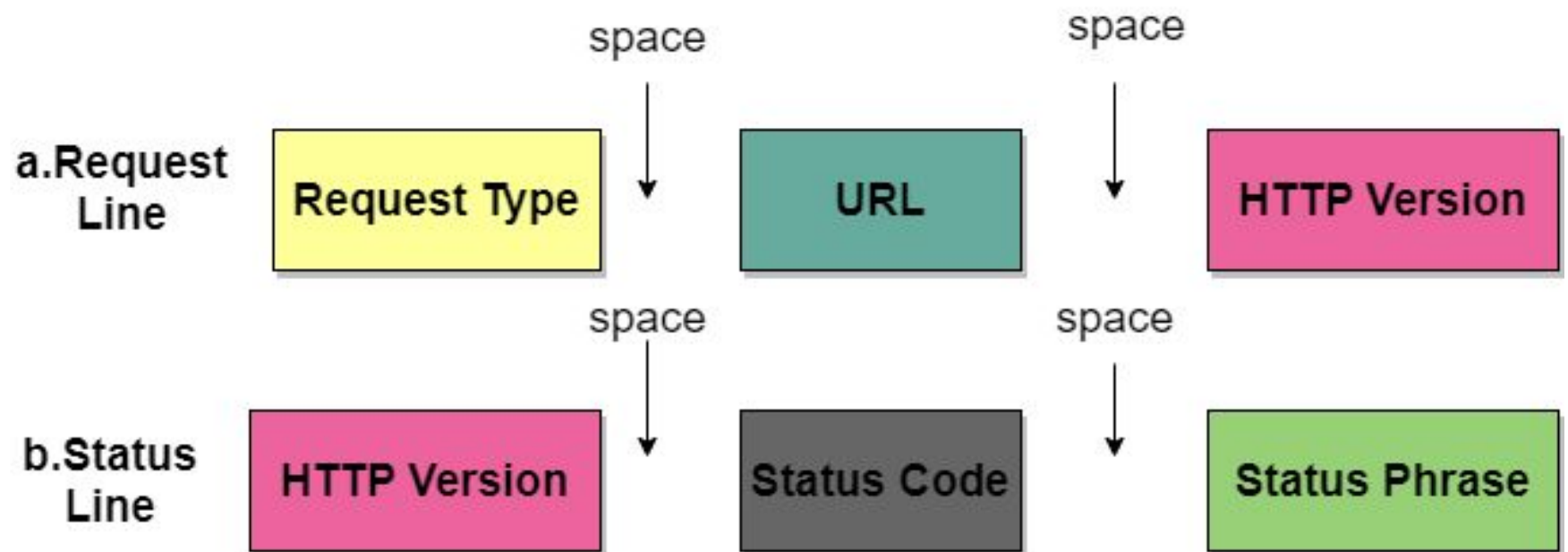
29



Request Message



Response Message



Hypertext Transport Protocol (HTTP)

31

- GET
- POST
- PUT
- DELETE
- ...

- Documentation
 - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

HTTP Header Fields

32

method	path	protocol
GET	/tutorials/other/top-20-mysql-best-practices/	HTTP/1.1

```
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q71b3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

HTTP headers as Name: Value

HTTP Header Fields

33

General:

Request URL

Request Method

Status Code

Remote Address

Referrer Policy

Response:

Server

Set-Cookie

Content-Type

Content-Length

Date

Request:

Cookies

Accept-xxx

Content-Type

Content-Length

Authorization

User-Agent

Referrer

Hypertext Transport Protocol (HTTP)

34

- A status code, indicating if the request was successful or not, and why.
- A status message, a non-authoritative short description of the status code.
- Optionally, a body containing the fetched resource.

Hypertext Transport Protocol (HTTP)

35

- Header
- The header is used to exchange the additional information between the client and the server. The header mainly consists of one or more header lines. Each header line has a header name, a colon, space, and a header value.

Hypertext Transport Protocol (HTTP)

36

- The header line is further categorized into four:
- General Header: It provides general information about the message and it can be present in both request and response.
- Request Header: It is only present in the request message and is used to specify the configuration of the client and the format of the document preferred by the client
- Response Header: This header is only present in the response header and mainly specifies the configuration of the server and also the special information about the request.
- Entity Header: It is used to provide information about the body of the document.

HTTP Error Codes

37

1xx : Informational

Request recieved / processing

2xx: Success

Successfully Recieved, understood and accepted

3xx: Redirect

Further action must be taken / redirect

4xx: Client Error

Request does not have what it needs

5xx: Server Error

Server failed to fulfil an apparent valid request

200 - OK

201 - OK created

301 - Moved to new URL

304 - Not modified (Cached version)

400 - Bad request

401 - Unauthorized

404 - Not found

500 - Internal server error

HTTP Error Codes

38

- When something goes wrong, the web server returns a special "error code" number
- Common error codes:

Number

Meaning

200

OK

301-303

page has moved (permanently or temporarily)

403

you are forbidden to access this page

404

page not found

500

internal server error

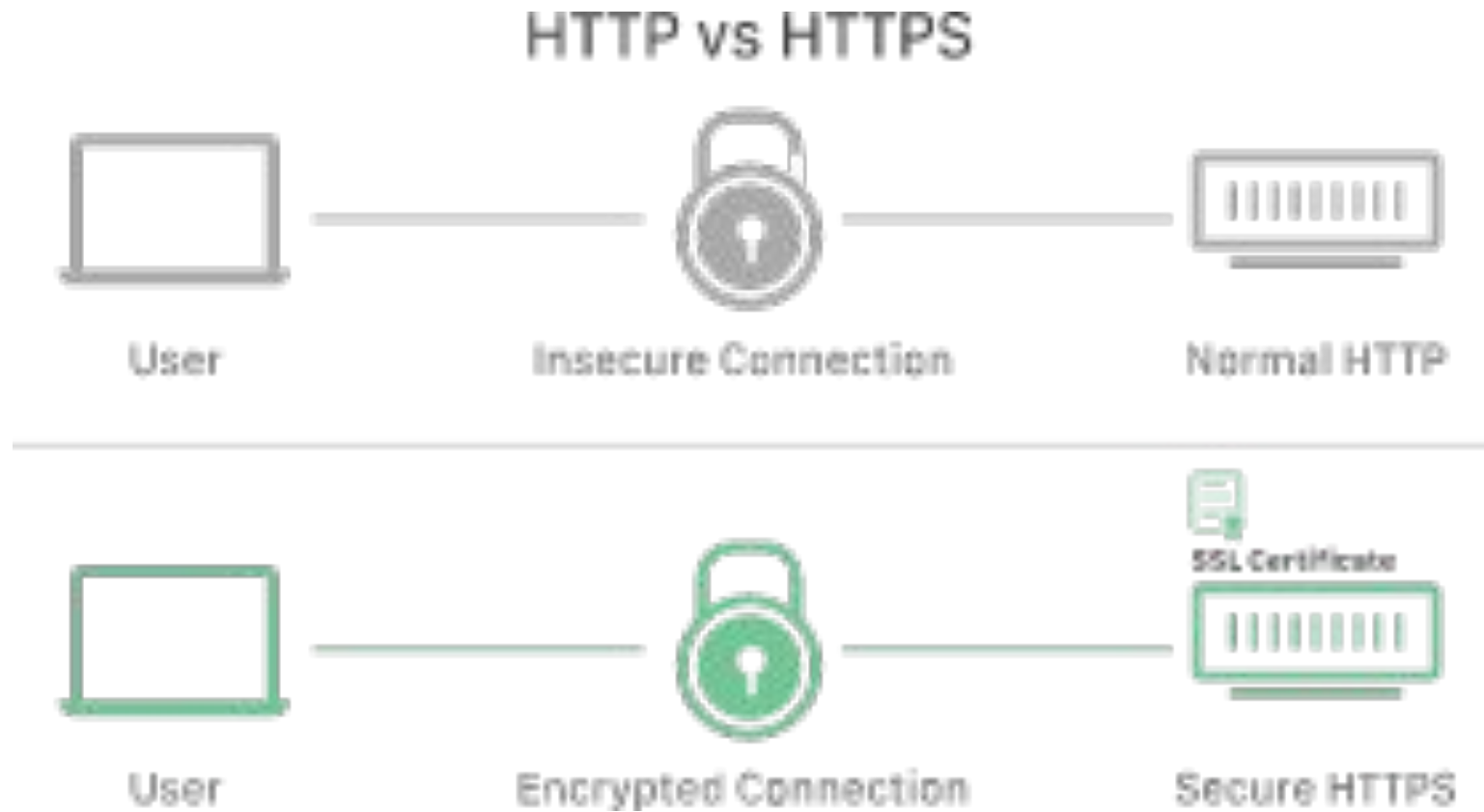
Internet Media (“MIME”) types

39

MIME type	file extension
text/html	.html
text/plain	.txt
image/gif	.gif
image/jpeg	.jpg
video/quicktime	.mov
application/octet-stream	.exe

HTTP vs HTTPS

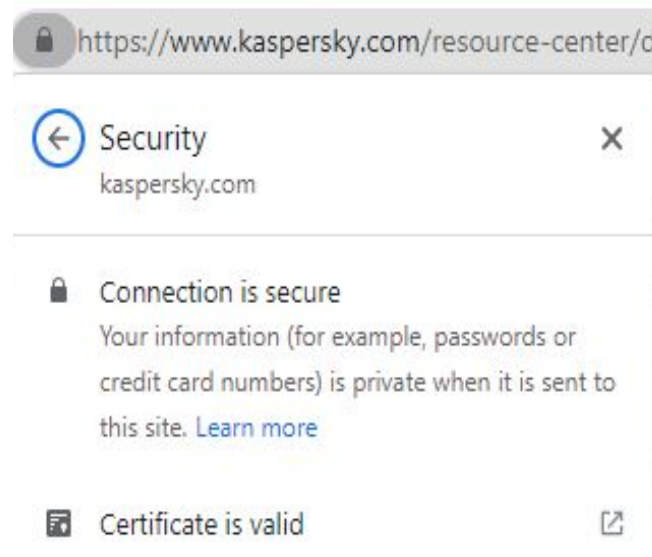
40



SSL- SECURE SOCKET LAYER

41

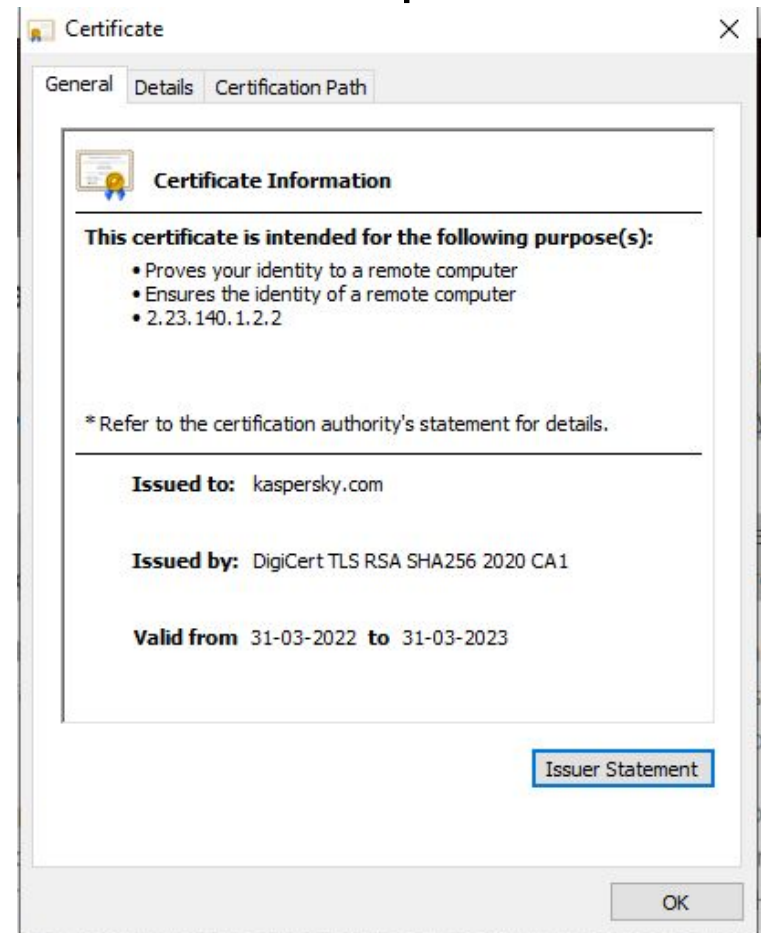
- An SSL certificate is a digital certificate that authenticates a website's identity and enables an encrypted connection. SSL stands for Secure Sockets Layer, a security protocol that creates an encrypted link between a web server and a web browser.



SSL- SECURE SOCKET LAYER

42

- Companies and organizations need to add SSL certificates to their websites to secure online transactions and keep customer information private and secure.



SSL- SECURE SOCKET LAYER

43

- The following steps are involved in the standard SSL handshake:
- Client Hello: Server communicates with the client using SSL. This includes the SSL version number, cipher settings, and session-specific data.
- Server Hello: The server responds with a “server hello” message. This includes the server’s SSL version number, cipher settings, session-specific data, an SSL certificate with a public key, and other information that the client needs to communicate with the server over SSL.
- Authentication: The client verifies the server’s SSL certificate from the CA (Certificate Authority) and authenticates the server. If the authentication fails, then the client refuses the SSL connection and throws an exception. If the authentication succeeds, then they proceed to the next step.

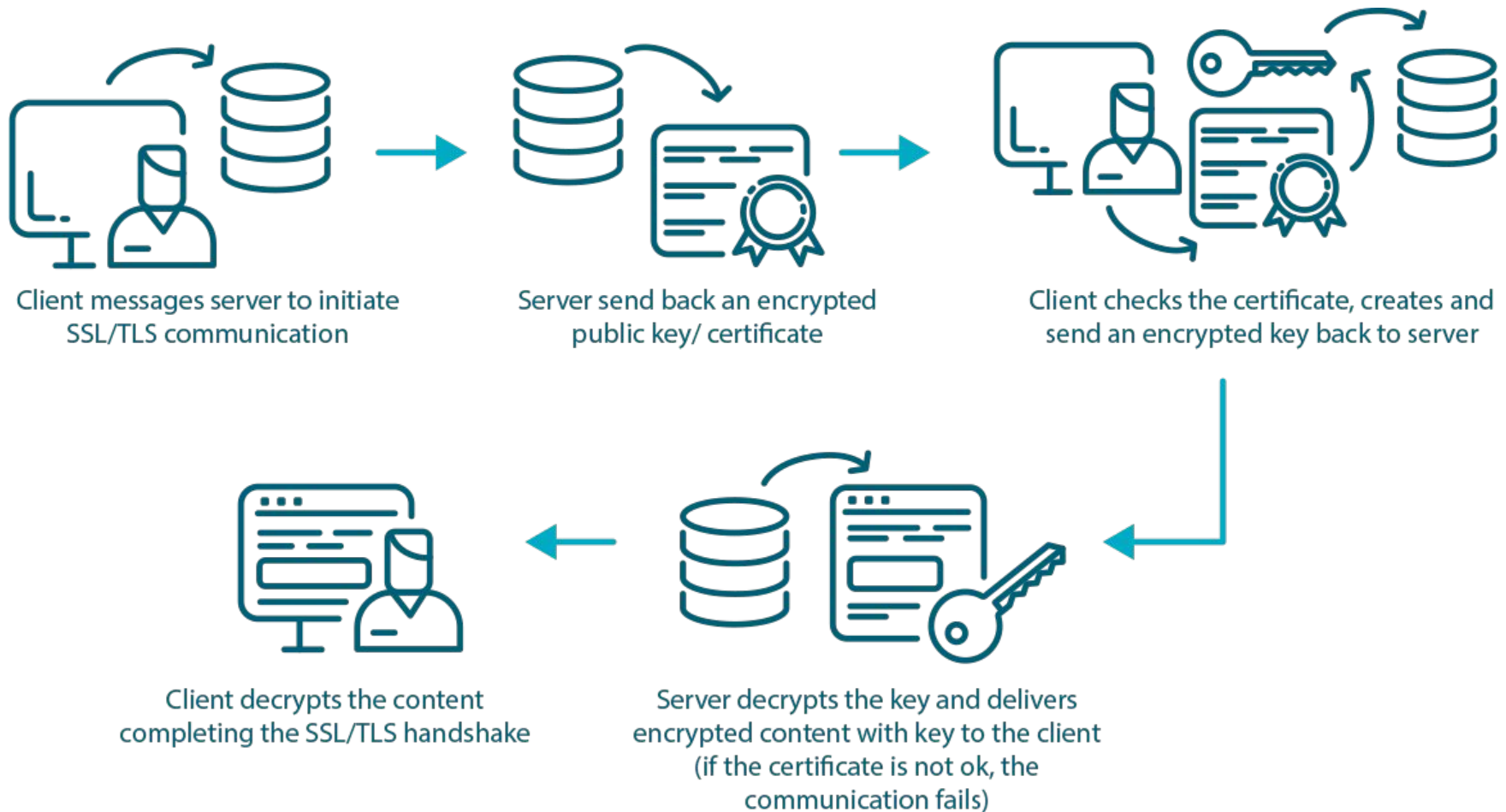
SSL- SECURE SOCKET LAYER

44

- ❑ Decryption: The client creates a session key, encrypts it with the server's public key and sends it to the server. If the server has requested client authentication (mostly in server to server communication), then the client sends their own certificate to the server.
- ❑ Encryption with Session Key: The server decrypts the session key with its private key and sends the acknowledgement to the client encrypted with the session key.
- ❑ Thus, at the end of the SSL handshake, both the client and the server have a valid session key which they will use to encrypt or decrypt the original data.

TLS- Transport Layer Security

45



TLS- Transport Layer Security

46

For Comparison	SSL	TLS
Abbreviation	SSL stands for "Secure Socket Layer."	TLS stands for "Transport Layer Security."
Cipher suites	SSL protocol offers support for Fortezza cipher suite	Cipher suites SSL protocol offers support for Fortezza cipher suite The TLS standardization process makes it much easier to define new cipher suites, such as RC4, Triple DES, AES, IDEA, etc.
Version	Three versions of SSL have been released: SSL 1.0, 2.0, and 3.0	Four versions of TLS have been released: TLS 1.0, 1.1, 1.2, and 1.3
Version Status	All versions of SSL have been found vulnerable, and they all have been deprecated	TLS 1.0 and 1.1 have been "broken" and are deprecated as of March 2020. TLS 1.2 is the most widely deployed protocol version
Secure Communication	SSL is a cryptographic protocol that uses explicit connections to establish secure communication between the web server and client	TLS is also a cryptographic protocol that provides secure communication between the web server and client via implicit connections. It is the successor to the SSL protocol.
Master Secret	SSL creates a master secret, the message digest of the pre-master secret is used	TLS uses a pseudorandom function to generate the master secret

Parameter	HTTP	HTTPS
Protocol	It is hypertext transfer protocol.	It is hypertext transfer protocol with secure.
Security	It is less secure as the data can be vulnerable to hackers.	It is designed to prevent hackers from accessing critical information. It is secure against such attacks.
Port	It uses port 80 by default	It was use port 443 by default.
Starts with	HTTP URLs begin with http://	HTTPSs URLs begin with https://
Used for	It's a good fit for websites designed for information consumption like blogs.	If the website needs to collect the private information such as credit card number, then it is a more secure protocol.
Scrambling	HTTP does not scramble the data to be transmitted. That's why there is a higher chance that transmitted information is available to hackers.	HTTPS scrambles the data before transmission. At the receiver end, it descrambles to recover the original data. Therefore, the transmitted information is secure which can't be hacked.
Protocol	It operates at application level.	HTTPS does not have any separate protocol. It operates using HTTP but uses encrypted TLS/SSL connection.
Domain Name Validation	HTTP website do not need SSL.	HTTPS requires SSL certificate.
Data encryption	HTTP website doesn't use encryption.	HTTPS websites use data encryption.
Search Ranking	HTTP does not improve search rankings.	HTTPS helps to improve search ranking.
Speed	Fast	Slower than HTTP
Vulnerability	Vulnerable to hackers	It Is highly secure as the data is encrypted before it is seen across a network.



FRONT-END

Development



BACK-END

Development



FULL-STACK

Development

**HTML**

Hypertext Markup Language

**CSS**

Cascading Stylesheet

**JavaScript**

HTML





CSS



JavaScript



Sarcastic Mommy

@sarcasticmommy4

 **Follow**

If you wait long enough to make dinner, everyone will just eat cereal.

It's science.

4:30 AM - 6 Aug 2014



1,663



2,507

**HTML**

Hypertext Markup Language

Markup Language

**CSS**

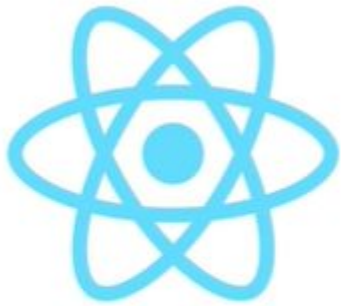
Cascading Stylesheet

Styling Language

**JavaScript**

Programming Language

FRAMEWORKS



REACT

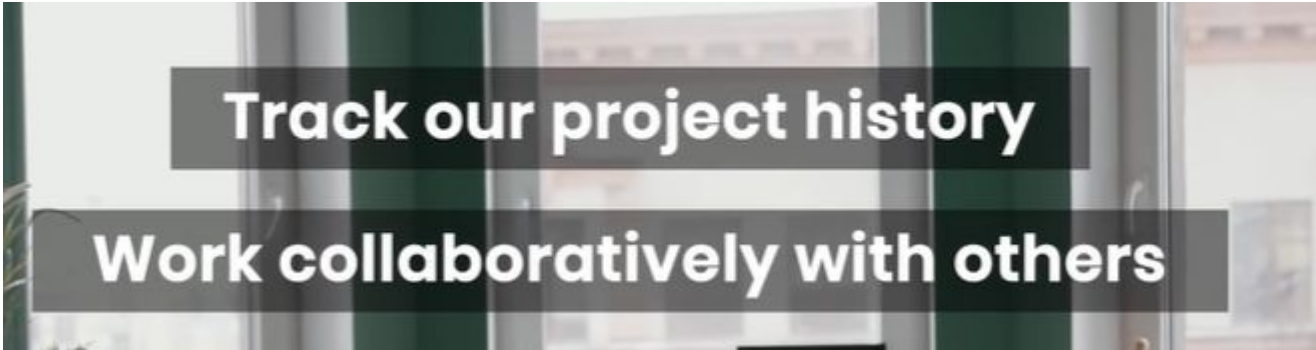


ANGULAR



VUE

VERSION CONTROL SYSTEMS



Track our project history

Work collaboratively with others

VERSION CONTROL SYSTEMS

- Git
- Subversion (SVN)
- Mercurial



HTML

61

- HTML: HTML (Hyper Text Markup Language) is used to create web pages and web applications. It is a markup language. By HTML we can create our own static page.
- It is used for displaying the data not to transport the data.
- HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages.
- A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly.

HTML

62

- `<!DOCTYPE html>`
- `<html>`
- `<head>`
- `<title>This is document title</title>`
- `</head>`
- `<body>`
- `<h1>This is a heading</h1>`
- `<p>Document content goes here.....</p>`
- `</body>`
- `</html>`

HTML

63

This is a heading

Document content goes here.....

HTML

64

- Web pages development - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- Internet Navigation - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- Responsive UI - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- Offline support HTML pages once loaded can be made available offline on the machine without any need of internet.
- Game development- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

Web Languages: XML

65

1.HTML is defined to describe the general form and layout of information in web documents without considering its meaning of the information.

To describe a particular kind of information, it would be necessary to have tags indicating the meaning of the element's content. That would allow the processing of specific categories of information in a document.

Web Languages: XML

66

2. HTML defines fixed set of tags and attributes. Given tags must fit every kind of document.
3. There are no restrictions on arrangement or order of tag appearance in HTML document. For example, an opening tag can appear in the content of an element, but its corresponding closing tag can appear after the end of the element in which it is nested.
Eg : ` Now is the time `

Web Languages: XML

67

- XML stands for Extensible Markup Language and is a text-based markup language
- A markup language is a set of codes, or tags, that describes the text in a digital document.
- The most famous markup language is hypertext markup language (HTML), which is used to format Web pages. XML, a more flexible cousin of HTML, makes it possible to conduct complex business over the Internet.

Web Languages: XML

68

The Difference between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags
- XML documents form a tree structure that starts at "the root" and branches to "the leaves".

Web Languages: XML

69

HTML includes a collection of predefined tags that you can use right away in editing your HTML files, e.g. `` and `<h1>`, but XML tags are not predefined.

To use XML, users have to define their own tags for their specific application before using them. For example, to describe a note, `<note>`, `<to>`, `<from>`, `<heading>`, and `<body>` are defined in advance and used in a nested fashion to present the following XML file as a note:

```
<note>
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
<heading>Reminder</heading>
```

```
<body>Don't forget the party!</body>
```

```
</note>
```

Web Languages: XML

70

With XML, it is illegal to omit the closing tag.

In HTML some elements do not have to have a closing tag to be able to present content

in the way the user wants them to. For example:

```
<p>This is a paragraph
```

```
<p>This is another paragraph
```

In XML, however, all elements must have a closing tag, like this:

```
<p>This is a paragraph</p>
```

```
<p>This is another paragraph</p>
```

Web Languages: XML

71

Unlike HTML, XML tags are case sensitive.

With XML, the tag `<Letter>` is different from the tag `<letter>`.

Opening and closing tags must therefore be written with the same case:

`<Message>This is incorrect</message>`

`<message>This is correct</message>`

Improper nesting of tags makes no sense to XML.

In HTML some elements can be improperly nested within each other and still display content in the desired way like this:

`<i>This text is bold and italic</i>`

In XML all elements must be properly nested within each other like this:

`<i>This text is bold and italic</i>`

Web Languages: XML

72

All XML documents must contain a single tag pair to define a root element.

All other elements must be within this root element. All elements can have sub elements(child elements). Sub elements must be correctly nested within their parent element:

```
<root>
```

```
<child>
```

```
<subchild>.....</subchild>
```

```
</child>
```

```
</root>
```

Web Languages: XML

73

With XML, it is illegal to omit quotation marks around attribute values. XML elements can have attributes in name/value pairs just like in HTML. In XML the attribute value must always be quoted. Study the two XML documents below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<note date=12/11/2002>
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
</note>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<note date="12/11/2002">
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
</note>
```

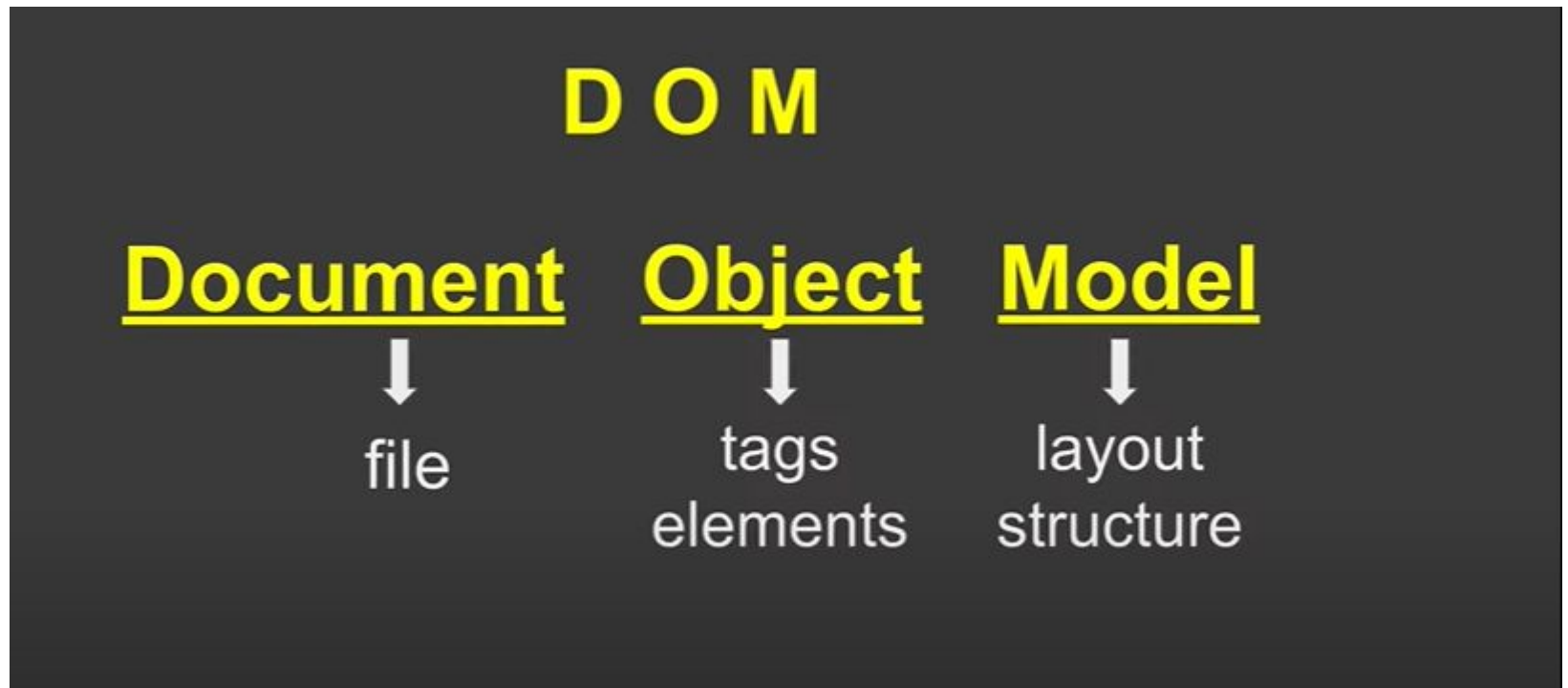
Web Languages: XML

74

- There is no length limitation for names.
- Space is Preserved in XML
- HTML truncates multiple white-space characters to one single white-space:
With XML, the whitespace in a document is not truncated
- Comments in XML are same as HTML `<!-- This is a comment -->`
- XML names must begin with a letter or underscore and can include digits, hyphens, and periods.

Document Object Model: DOM

75

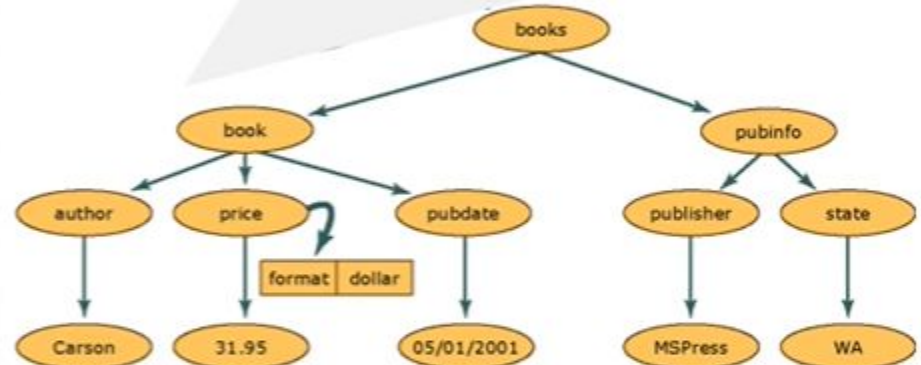


Document Object Model: DOM

76

```
<books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
  <pubinfo>
    <publisher>MSPress</publisher>
    <state>WA</state>
  </pubinfo>
</books>
```

DOM represents the content of xml or html document as tree structure



Can easily read, access, update the contents of the document

Document Object Model: DOM

77

- DOM is a way to represent the webpage in a structured hierarchical way so that it will become easier for programmers and users to glide through the document.
- With DOM, we can easily access and manipulate tags, IDs, classes, Attributes, or Elements of HTML using commands or methods provided by the Document object.
- Using DOM, the JavaScript gets access to HTML as well as CSS of the web page and can also add behavior to the HTML elements. so basically Document Object Model is an API that represents and interacts with HTML or XML documents.

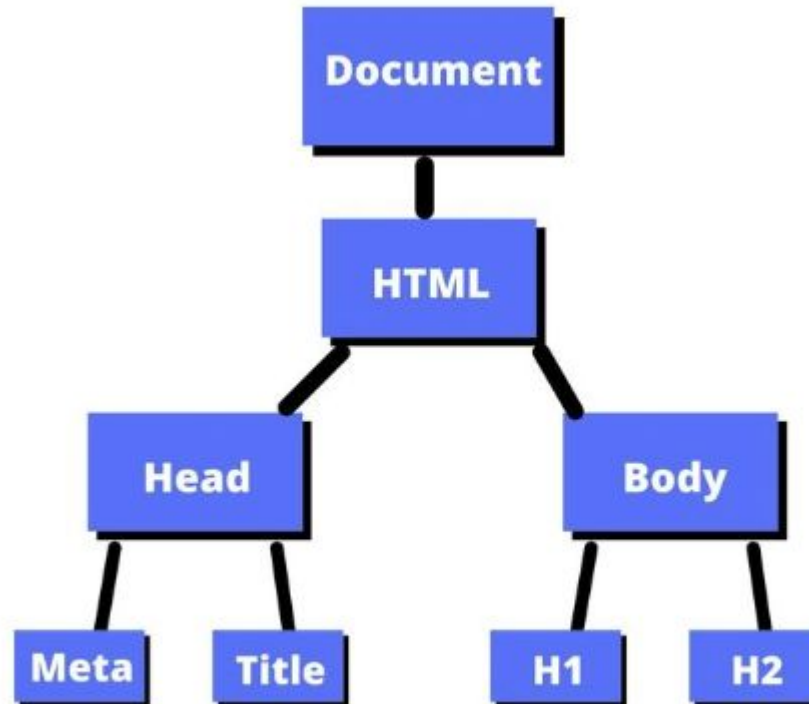
Document Object Model: DOM

78

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>DOM tree structure</title>
  </head>
  <body>
    <h1>DOM tree structure</h1>
    <h2>Learn about the DOM</h2>
  </body>
</html>
```

Document Object Model: DOM

79



How to Select Elements in the Document

80

- getElementById()
- In HTML, ids are used as unique identifiers for the HTML elements. This means you cannot have the same id name for two different elements.
- This would be incorrect:

```
<p id="para">This is my first paragraph.</p>  
<p id="para">This is my second paragraph.</p>
```

How to Select Elements in the Document

81

- You would have to make sure those ids are unique like this:

```
<p id="para1">This is my first paragraph.</p>  
<p id="para2">This is my second paragraph.</p>
```

- In JavaScript, we can grab an HTML tag by referencing the id name.

```
document.getElementById("id name goes here")
```

How to Select Elements in the Document

82

- This code tells the computer to get the `<p>` element with the id of `para1` and print the element to the console.

```
const paragraph1 = document.getElementById("para1");  
console.log(paragraph1);
```

Console

```
▼ <p id="para1">This is my first paragraph.</p>
```

JavaScript Object Notation JSON

83

□ JSON

- stands for JavaScript Object Notation.
- an open standard data-interchange format.
- lightweight and self-describing.
- originated from JavaScript.
- easy to read and write.
- language independent.
- supports data structures such as arrays and objects.

JavaScript Object Notation JSON

84

- In JSON, objects refer to dictionaries, which are enclosed in curly brackets, i.e., { }.
- These objects are written in key/value pairs, where the key has to be a string and values have to be a valid JSON data type such as string, number, object, Boolean or null.
- Here the key and values are separated by a colon, and a comma separates each key/value pair.
- For example:

```
{"name" : "Jack", "employeeid" : 001, "present" : false}
```


JSON Examples

85

```
'{"name": "John", "age": 30, "car": null}'
```

- It defines an object with 3 properties:
 - name
 - age
 - car
- Each property has a value.

JavaScript Object Notation JSON

86

- ❑ Less Verbose: In contrast to XML, JSON follows a compact style to improve its users' readability. While working with a complex system, JSON tends to make substantial enhancements.
- ❑ Faster: The JSON parsing process is faster than that of the XML because the DOM manipulation library in XML requires extra memory for handling large XML files. However, JSON requires less data that ultimately results in reducing the cost and increasing the parsing speed.

JavaScript Object Notation JSON

87

- Readable: The JSON structure is easily readable and straightforward. Regardless of the programming language that you are using, you can easily map the domain objects.
- Structured Data: In JSON, a map data structure is used, whereas XML follows a tree structure. The key-value pairs limit the task but facilitate the predictive and easily understandable model.

JavaScript Object Notation JSON

88

- Basic data types supported by json are:
 - Strings: Characters that are enclosed in single or double quotation marks.
 - Number: A number could be integer or decimal, positive or negative.
 - Booleans: The Boolean value could be either true or false without any quotation marks.
 - Null: Here, null means nothing without any quotation marks.

JavaScript Object Notation JSON

89

- Arrays in JSON
- Arrays are the lists that are represented by the square brackets, and the values have commas in between them. They can contain mix data types, i.e., a single array can have strings, Boolean, numbers.
- For example:
 - Example 1: [1, 2, 7.8, 5, 9, 10];
 - Example 2: ["red", "yellow", "green"];
 - Example 3: [8, "hello", null, true];
- In the above, example 1 is an array of numbers, example 2 is an array of strings, and example 3 is an array of mix data types.

JavaScript Object Notation JSON

90

- Objects
 - Objects are JSON dictionaries that are enclosed in curly brackets.
 - In objects, keys and values are separated by a colon ':', pairs are separated by comma.
 - Keys and values can be of any type, but the most common type for the keys is a string.
 - For example: `{"red" : 1, "yellow" : 2, "green" : 3};`

JavaScript Object Notation JSON

91

- Nesting
 - Nesting involves keeping the arrays and objects inside of each other.
 - We can put the arrays inside objects, objects inside arrays, arrays inside arrays, etc.
 - We can say that json file is a big object with lots of objects and arrays inside.

JavaScript Object Notation JSON

92

```
{  
  "song" :  
    {  
      "title" : "Hey Dude";  
      "artist": "The Beatles";  
      "musicians": ["John Lennon", "Paul McCratney",  
"Ringo Starr"];  
    }  
}
```

- In the above code, the song starts with a curly bracket. Therefore, a song is an object. It contains three key-value pairs wherein title, artist and musicians are the keys.'

JSON Nested Object Example

93

- A JSON object can have another object also.
- {
 - "firstName": "ABC",
 - "lastName": "PQR",
 - "age": 27,
 - "address" : {
 - "streetAddress": "Plot-6, Mohan Nagar",
 - "city": "Navi Mumbai",
 - "state": "Maharashtra",
 - "postalCode": "402107"
 - }
- }

JavaScript Object Notation JSON

94

- JSON Multidimensional Array
- We can store array inside JSON array, it is known as array of arrays or multidimensional array.
- [
 - ["a", "b", "c"],
 - ["m", "n", "o"],
 - ["x", "y", "z"]
-]

JSON Comments

95

- JSON doesn't support comments. It is not a standard.
- But you can do some tricks such as adding extra attribute for comment in JSON object, for example:
- {
- "employee": {
- "name": "Bob",
- "salary": 56000,
- "comments": "He is a nice man"
- }
- }
- Here, "comments" attribute can be treated as comment.

JSON vs XML

The object created in JSON has some type.	XML data does not have any type.
The data types supported by JSON are strings, numbers, Booleans, null, array.	XML data is in a string format.
It does not have any capacity to display the data.	XML is a markup language, so it has the capacity to display the content.
JSON has no tags.	XML data is represented in tags, i.e., start tag and end tag.
	XML file is larger. If we want to represent the data in XML then it would create a larger file as compared to JSON.
JSON is quicker to read and write.	XML file takes time to read and write because the learning curve is higher.
JSON can use arrays to represent the data.	XML does not contain the concept of arrays.
It can be parsed by a standard javascript function. It has to be parsed before use.	XML data which is used to interchange the data, must be parsed with respective to their programming language to use that.
It can be easily parsed and little bit code is required to parse the data.	It is difficult to parse.
File size is smaller as compared to XML.	File size is larger.
JSON is data-oriented.	XML is document-oriented.
It is less secure than XML.	It is more secure than JSON.

JSON vs XML

97

- {"employees":[
- {"name":"Shyam", "email":"shyamjaiswal@gmail.com"},
- {"name":"Bob", "email":"bob32@gmail.com"},
- {"name":"Jai", "email":"jai87@gmail.com"}
-]}

JSON vs XML

98

- <employees>
- <employee>
- <name>Shyam</name>
- <email>shyamjaiswal@gmail.com</email>
- </employee>
- <employee>
- <name>Bob</name>
- <email>bob32@gmail.com</email>
- </employee>
- <employee>
- <name>Jai</name>
- <email>jai87@gmail.com</email>
- </employee>
- </employees>

REST API

99

- RESTful API is an interface that two computer systems use to exchange information securely over the internet.
- An application programming interface (API) defines the rules that you must follow to communicate with other software systems. Developers expose or create APIs so that other applications can communicate with their applications programmatically.
- Representational State Transfer (REST) is a software architecture that imposes conditions on how an API should work. REST was initially created as a guideline to manage communication on a complex network like the internet. You can use REST-based architecture to support high-performing and reliable communication at scale. You can easily implement and modify it, bringing visibility and cross-platform portability to any API system.
- API developers can design APIs using several different architectures. APIs that follow the REST architectural style are called REST APIs. Web services that implement REST architecture are called RESTful web services.

Uniform interface

100

- The uniform interface is fundamental to the design of any RESTful webservice. It indicates that the server transfers information in a standard format. The formatted resource is called a representation in REST. This format can be different from the internal representation of the resource on the server application. For example, the server can store data as text but send it in an HTML representation format.
- Uniform interface imposes four architectural constraints: Requests should identify resources. They do so by using a uniform resource identifier.
- Clients have enough information in the resource representation to modify or delete the resource if they want to. The server meets this condition by sending metadata that describes the resource further.
- Clients receive information about how to process the representation further. The server achieves this by sending self-descriptive messages that contain metadata about how the client can best use them.
- Clients receive information about all other related resources they need to complete a task. The server achieves this by sending hyperlinks in the representation so that clients can dynamically discover more resources.

Statelessness

101

- In REST architecture, statelessness refers to a communication method in which the server completes every client request independently of all previous requests. Clients can request resources in any order, and every request is stateless or isolated from other requests. This REST API design constraint implies that the server can completely understand and fulfill the request every time.

Layered system

102

- In a layered system architecture, the client can connect to other authorized intermediaries between the client and server, and it will still receive responses from the server. Servers can also pass on requests to other servers. You can design your RESTful web service to run on several servers with multiple layers such as security, application, and business logic, working together to fulfill client requests. These layers remain invisible to the client.

Cacheability

103

- RESTful web services support caching, which is the process of storing some responses on the client or on an intermediary to improve server response time. For example, suppose that you visit a website that has common header and footer images on every page. Every time you visit a new website page, the server must resend the same images. To avoid this, the client caches or stores these images after the first response and then uses the images directly from the cache. RESTful web services control caching by using API responses that define themselves as cacheable or noncacheable.

Code on demand

104

- In REST architectural style, servers can temporarily extend or customize client functionality by transferring software programming code to the client. For example, when you fill a registration form on any website, your browser immediately highlights any mistakes you make, such as incorrect phone numbers. It can do this because of the code sent by the server.

Code on demand

105

- The general steps for any REST API call:
 - The client sends a request to the server. The client follows the API documentation to format the request in a way that the server understands.
 - The server authenticates the client and confirms that the client has the right to make that request.
 - The server receives the request and processes it internally.
 - The server returns a response to the client. The response contains information that tells the client whether the request was successful. The response also includes any information that the client requested.

Code on demand

106

HTTP METHODS

GET

POST

PUT

DELETE

Code on demand

107

GET CUSTOMERS

Request

```
GET /api/customers
```

Response

```
[  
  { id: 1, name: '' },  
  { id: 2, name: '' },  
  ...  
]
```

Code on demand

108

GET A CUSTOMER

Request

```
GET /api/customers/1
```

Response

```
{ id: 1, name: '' }
```


Code on demand

109

UPDATE A CUSTOMER

Request

```
PUT /api/customers/1
```

```
{ name: '' }
```

Response

```
{ id: 1, name: '' }
```

Code on demand

110

DELETE A CUSTOMER

Request

```
DELETE /api/customers/1
```

Response

Code on demand

111

CREATE A CUSTOMER

Request

```
POST /api/customers
```

```
{ name: '' }
```

Response

```
{ id: 1, name: '' }
```

END