SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

WEB TECHNOLOGY

III YEAR - V SEM

UNIT 1 – Web Site Basics And HTML

TOPIC 4 - HTTP request/response Message

Hypertext Transfer Protocol (HTTP)

- HTTP is based on the request-response communication model:
  - Client sends a request
  - Server sends a response
- HTTP is a stateless protocol:
  - The protocol does not require the server to remember anything about the client between requests.

HTTP

- Normally implemented over a TCP connection (80 is standard port number for HTTP)
- Typical browser-server interaction:
  - User enters Web address in browser
  - Browser uses DNS to locate IP address
  - Browser opens TCP connection to server
  - Browser sends HTTP request over connection
  - Server sends HTTP response to browser over connection
  - Browser displays body of response in the client area of the browser window

HTTP

- The information transmitted using HTTP is often entirely text
- Can use the Internet's Telnet protocol to simulate browser request and view server response

HTTP

Connect

Send Request

Receive Response

```
$ telnet www.example.org 80
Trying 192.0.34.166...
Connected to www.example.com (192.0.34.166).
Escape character is '^]'.
GET / HTTP/1.1
Host: www.example.org

HTTP/1.1 200 OK
Date: Thu, 09 Oct 2020 20:30:49 GMT
…
```

HTTP Request

- Structure of the request:
  - start line
  - header field(s)
  - blank line
  - optional body

HTTP Request

- Structure of the request:
  - **start line**
  - header field(s)
  - blank line
  - optional body

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - HTTP request method
  - Request-URI (Uniform Resource Identifier)
  - HTTP version

HTTP Request

- Start line
  - Example: `GET / HTTP/1.1`
- Three space-separated parts:
  - HTTP request method
  - **Request-URI**
  - HTTP version

HTTP Request

- Uniform Resource Identifier (URI)
  - Syntax: *`scheme : scheme-depend-part`*
    - Ex: In `http://www.example.com/` the scheme is `http`
  - Request-URI is the portion of the requested URI that follows the host name (which is supplied by the required Host header field)
    - Ex: / is Request-URI portion of `http://www.example.com/`

URI

- URI's are of two types:
  - Uniform Resource Name (URN)
    - Can be used to identify resources with unique names, such as books (which have unique ISBN's)
    - Scheme is `urn`
  - Uniform Resource Locator (URL)
    - Specifies location at which a resource can be found
    - In addition to `http`, some other URL schemes are `https`, `ftp`, `mailto`, and `file`

HTTP Request

- Start line
  - Example: GET / HTTP/1.1
- Three space-separated parts:
  - **HTTP request method**
  - Request-URI
  - HTTP version

HTTP Request

- Common request methods:
  - GET
    - Used if link is clicked or address typed in browser
    - No body in request with GET method
  - POST
    - Used when submit button is clicked on a form
    - Form information contained in body of request
  - HEAD
    - Requests that only header fields (no body) be returned in the response

HTTP Request

- Structure of the request:
  - start line
  - **header field(s)**
  - blank line
  - optional body

HTTP Request

- Header field structure:
  - *field name : field value*
- Syntax
  - Field name is not case sensitive
  - Field value may continue on multiple lines by starting continuation lines with white space
  - Field values may contain MIME types, quality values, and wildcard characters (*'s)

Multipurpose Internet Mail Extensions (MIME)

- Convention for specifying content type of a message
  - In HTTP, typically used to specify content type of the body of the response
- MIME content type syntax:
  - `top-level type / subtype`
- Examples: text/html, image/jpeg

HTTP Quality Values and Wildcards

- Example header field with quality values:
```
accept:
  text/xml,text/html;q=0.9,
  text/plain;q=0.8, image/jpeg,
  image/gif;q=0.2,*/*;q=0.1
```
- Quality value applies to all preceding items
- Higher the value, higher the preference
- Note use of wildcards to specify quality 0.1 for any MIME type not specified earlier

HTTP Request

- Common header fields:
  - Host: host name from URL (required)
  - User-Agent: type of browser sending request
  - Accept: MIME types of acceptable documents
  - Connection: value `close` tells server to close connection after single request/response
  - Content-Type: MIME type of (POST) body, normally application/x-www-form-urlencoded
  - Content-Length: bytes in body
  - Referer: URL of document containing link that supplied URI for this HTTP request

HTTP Response

- Structure of the response:
  - status line
  - header field(s)
  - blank line
  - optional body

HTTP Response

- Structure of the response:
  - **status line**
  - header field(s)
  - blank line
  - optional body

HTTP Response

- Status line
  - Example: HTTP/1.1 200 OK
- Three space-separated parts:
  - HTTP version
  - status code
  - reason phrase (intended for human use)

HTTP Response

- Status code
  - Three-digit number
  - First digit is class of the status code:
    - 1=Informational
    - 2=Success
    - 3=Redirection (alternate URL is supplied)
    - 4=Client Error
    - 5=Server Error
  - Other two digits provide additional information
  - See http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

HTTP Response

- Structure of the response:
  - status line
  - **header field(s)**
  - blank line
  - optional body

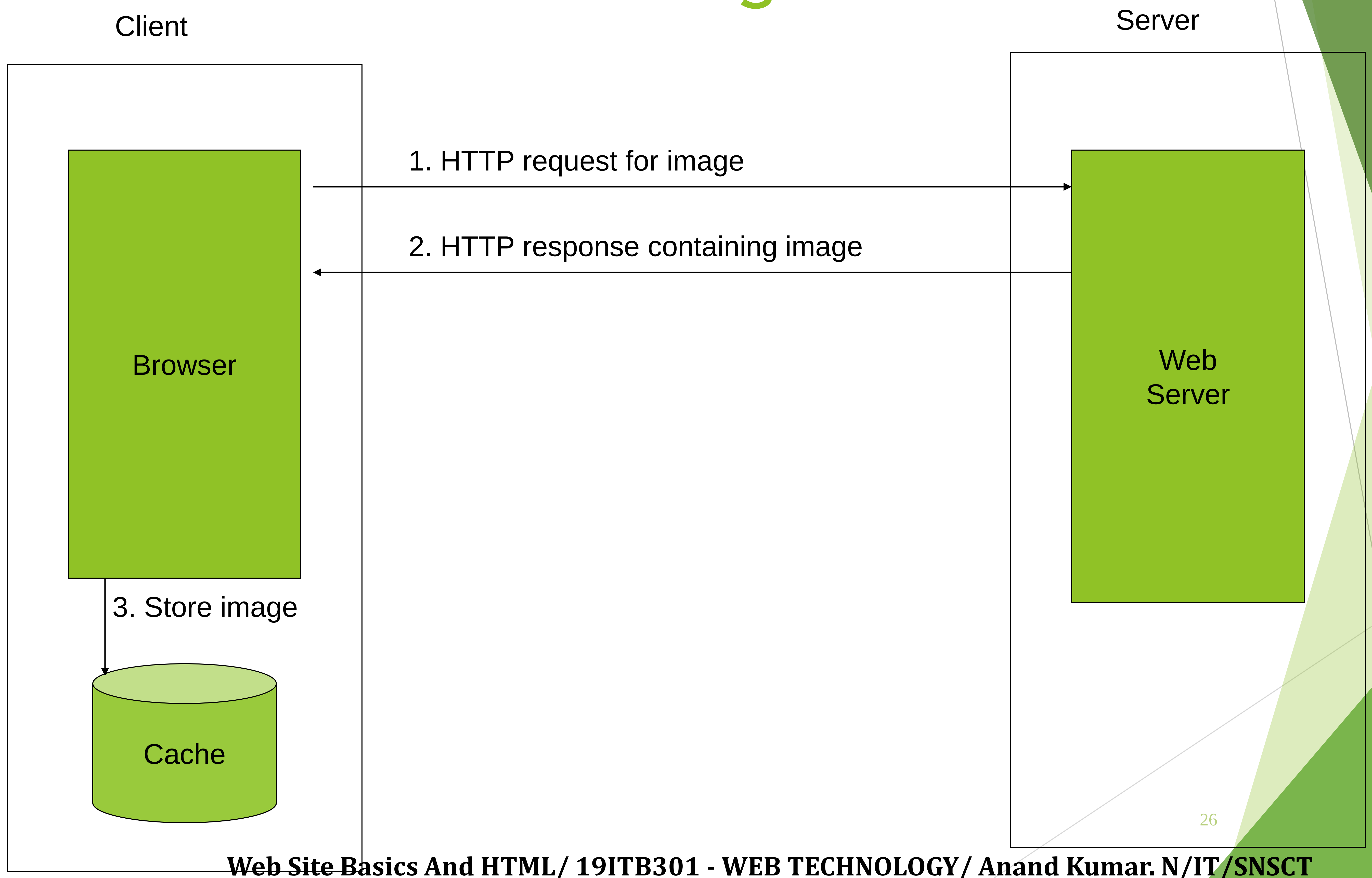
HTTP Response

- Common header fields:
  - Connection, Content-Type, Content-Length
  - Date: date and time at which response was generated (required)
  - Location: alternate URI if status is redirection
  - Last-Modified: date and time the requested resource was last modified on the server
  - Expires: date and time after which the client's copy of the resource will be out-of-date
  - ETag: a unique identifier for this version of the requested resource (changes if resource changes)

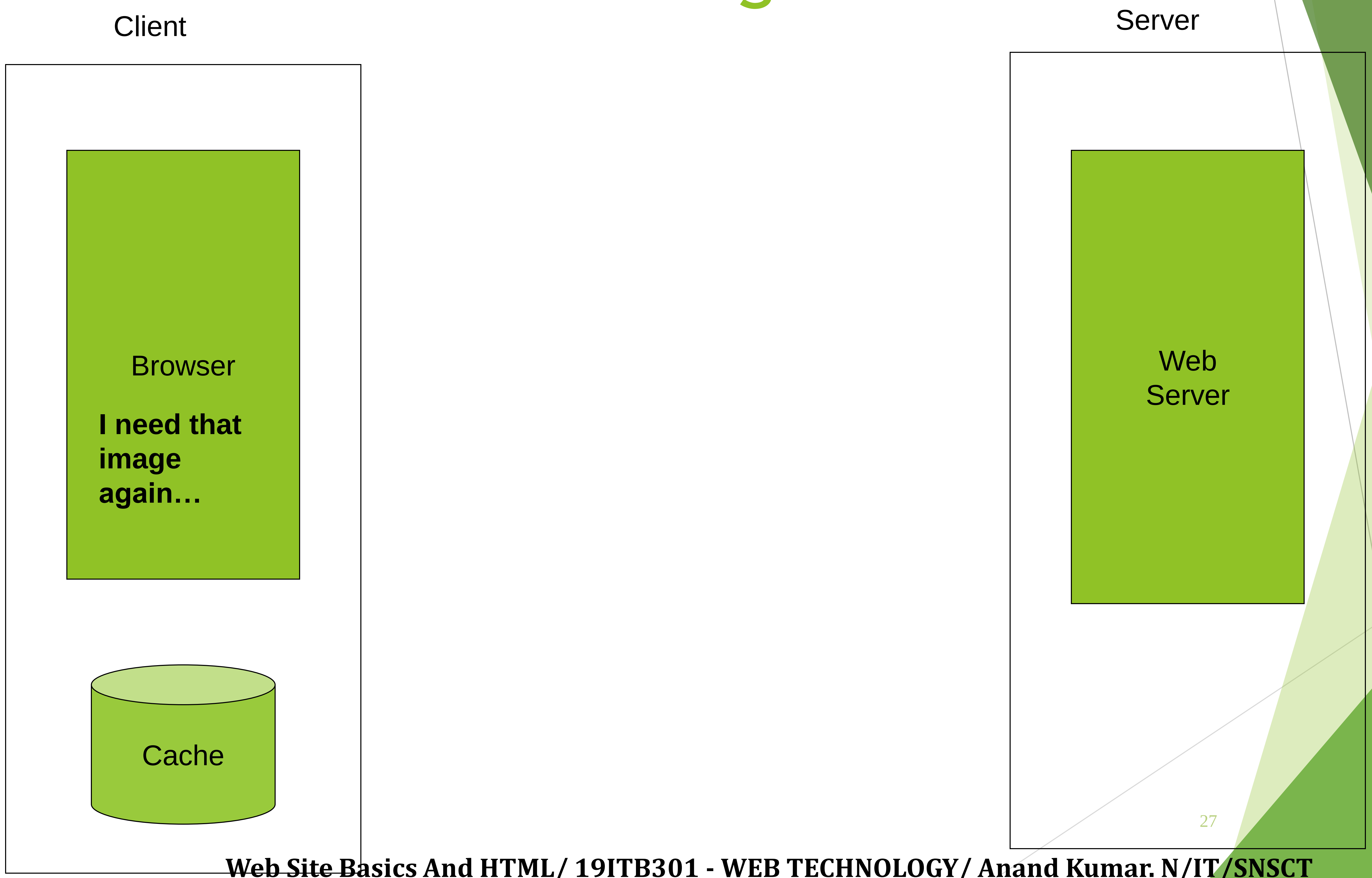
Client Caching

- A cache is a local copy of information obtained from some other source
- Most web browsers use cache to store requested resources so that subsequent requests to the same resource will not necessarily require an HTTP request/response
  - Ex: icon appearing multiple times in a Web page

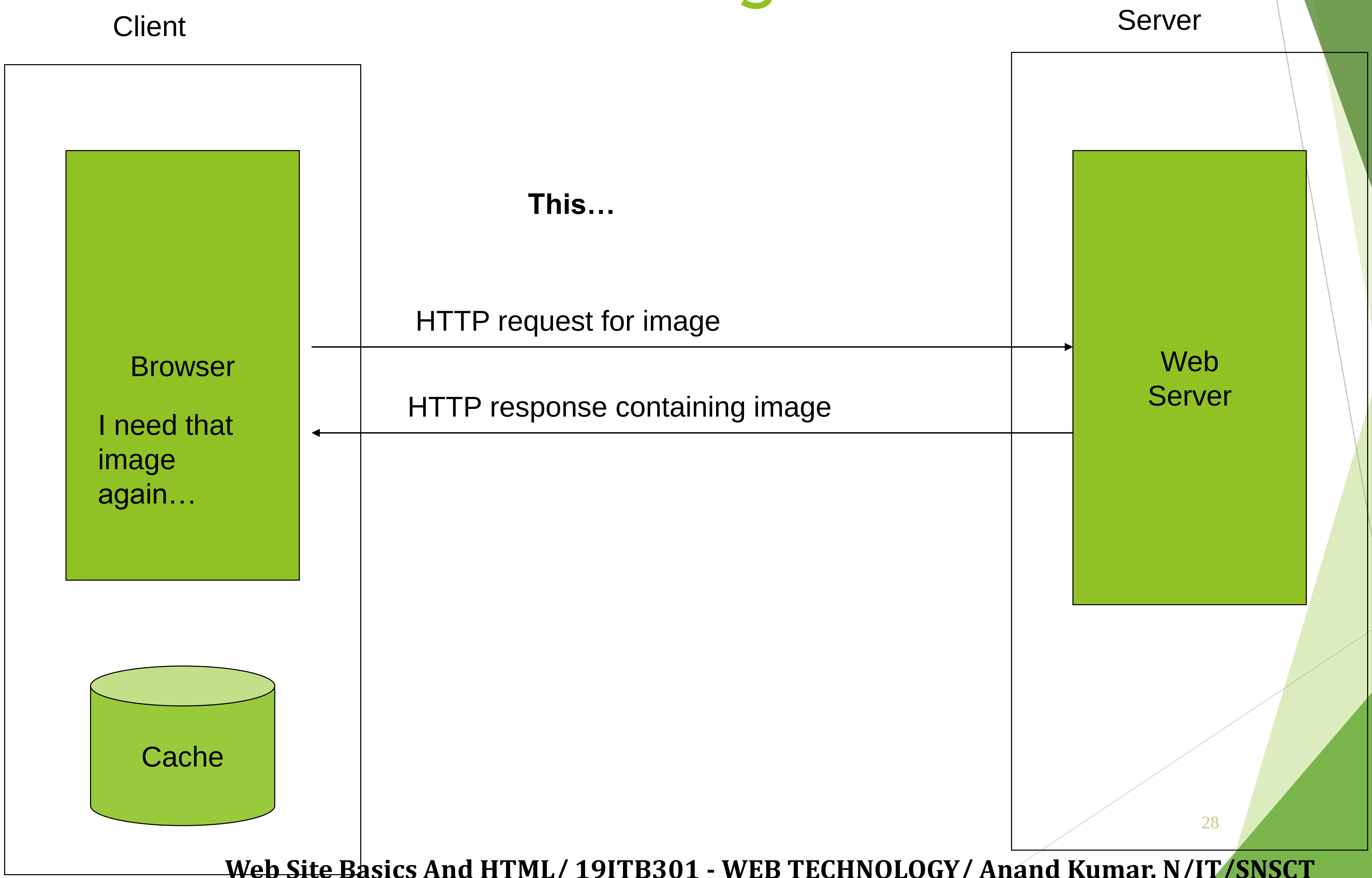
Client Caching



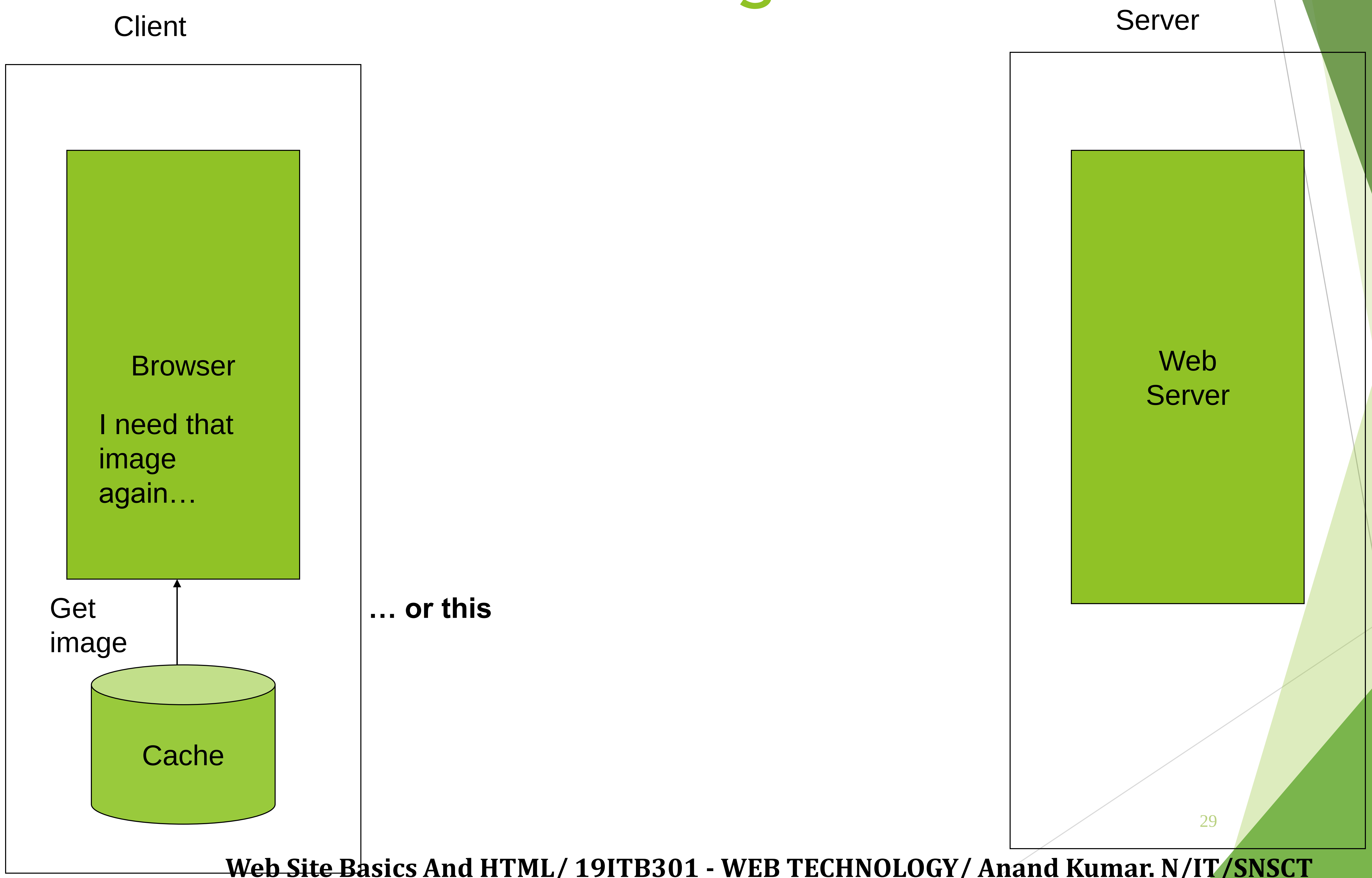
Client Caching



Client Caching



Client Caching



Client Caching

- Cache advantages
  - (Much) faster than HTTP request/response
  - Less network traffic
  - Less load on server
- Cache disadvantage
  - Cached copy of resource may be invalid (inconsistent with remote version)

Client Caching

- Validating cached resource:
  - Send HTTP HEAD request and check Last-Modified or ETag header in response
  - Compare current date/time with Expires header sent in response containing resource
  - If no Expires header was sent, use heuristic algorithm to estimate value for Expires
    - Ex: Expires = 0.01 * (Date – Last-Modified) + Date

Character Sets

- Every document is represented by a string of integer values (code points)
- The mapping from code points to characters is defined by a character set
- Some header fields have character set values:
  - Accept-Charset: request header listing character sets that the client can recognize
    - Ex: accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.5
  - Content-Type: can include character set used to represent the body of the HTTP message
    - Ex: Content-Type: text/html; charset=UTF-8

Character Sets

- Technically, many “character sets” are actually character encodings
  - An encoding represents code points using variable-length byte strings
  - Most common examples are Unicode-based encodings UTF-8 and UTF-16
- IANA maintains complete list of Internet-recognized character sets/encodings

Character Sets

- Typical US PC produces ASCII documents
- US-ASCII character set can be used for such documents, but is not recommended
- UTF-8 and ISO-8859-1 are supersets of US-ASCII and provide international compatibility
  - UTF-8 can represent all ASCII characters using a single byte each and arbitrary Unicode characters using up to 4 bytes each
  - ISO-8859-1 is 1-byte code that has many characters common in Western European languages, such as é