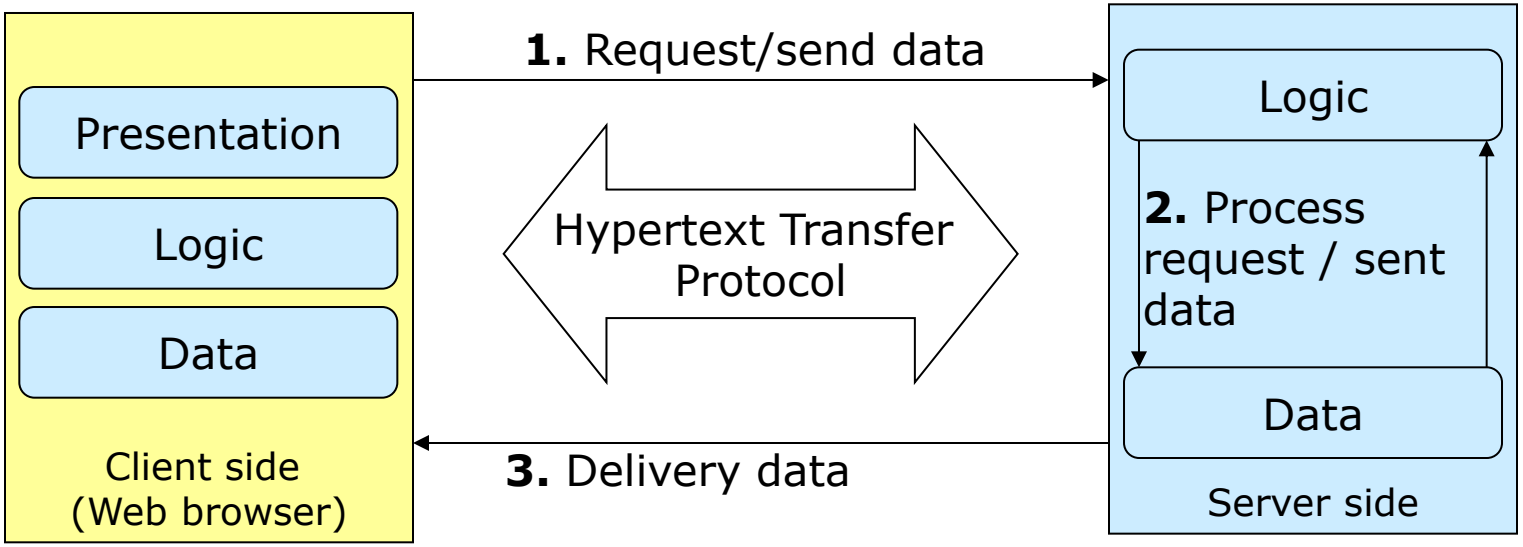
Basic aspects of Web Applications

Introduction



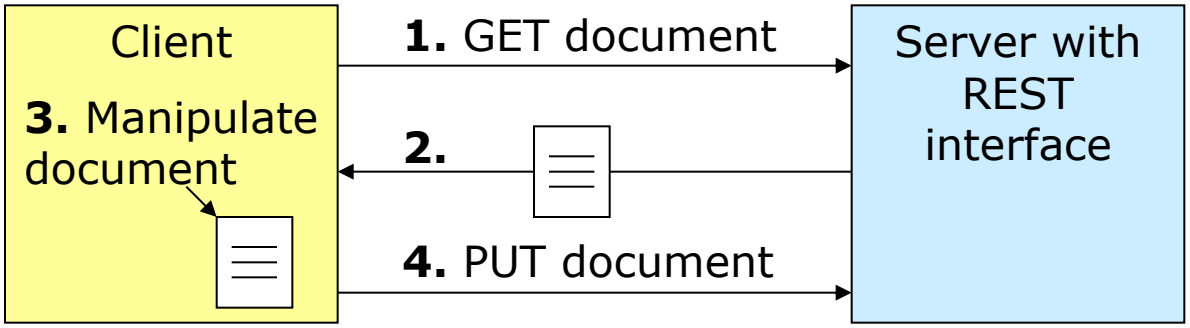
- Means for providing modern User Interfaces apart from form-based UIs with rich media and comfortable user interaction?

- Transfer efficiency?
- Protocol features?
- Functional range?

- Efficiency?
- Integration of existing logic?
- Simplicity of application development?

REST model

- The Representational State Transfer (REST) is a model for distributed hypermedia systems that predominantly fits to applications in the World Wide Web
- It is based on a client/server architecture with a stateless communication protocol
- Every message contains all necessary context information thus neither the server nor the client has to store context data
- In contrast to Remote Procedure Call (RPC) **requests in a REST system are not directed to procedures but to resources** (documents) using a generic interface with standard semantics
- Every resource has to be available through a unique identifier

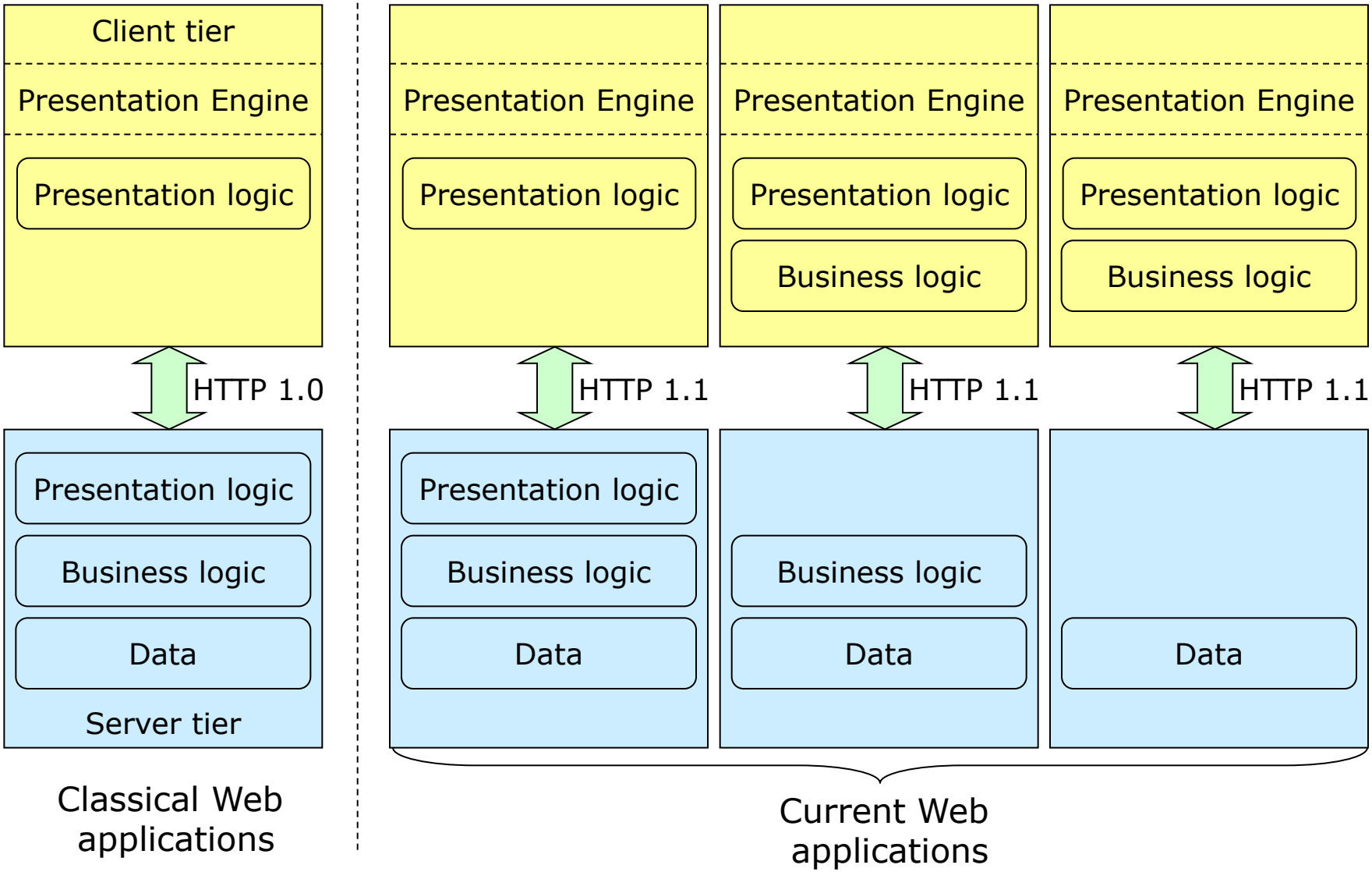


Outline

The lecture presents selected principles and technologies of Web applications:

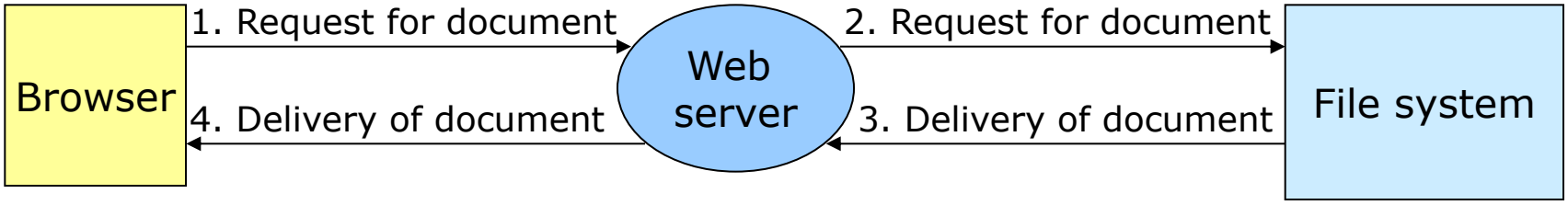
- **Evolution of web technology**
  - Short survey of technology evolution
- **Server-side aspects**
  - Architecture of a web server
- **Communication aspects**
  - HTTP Communication details
  - Aspects of HTTP 1.1
  - HTTP/2
- **Client-side aspects**
  - Web browser architecture
  - Overview and selected features of HTML 5

Development



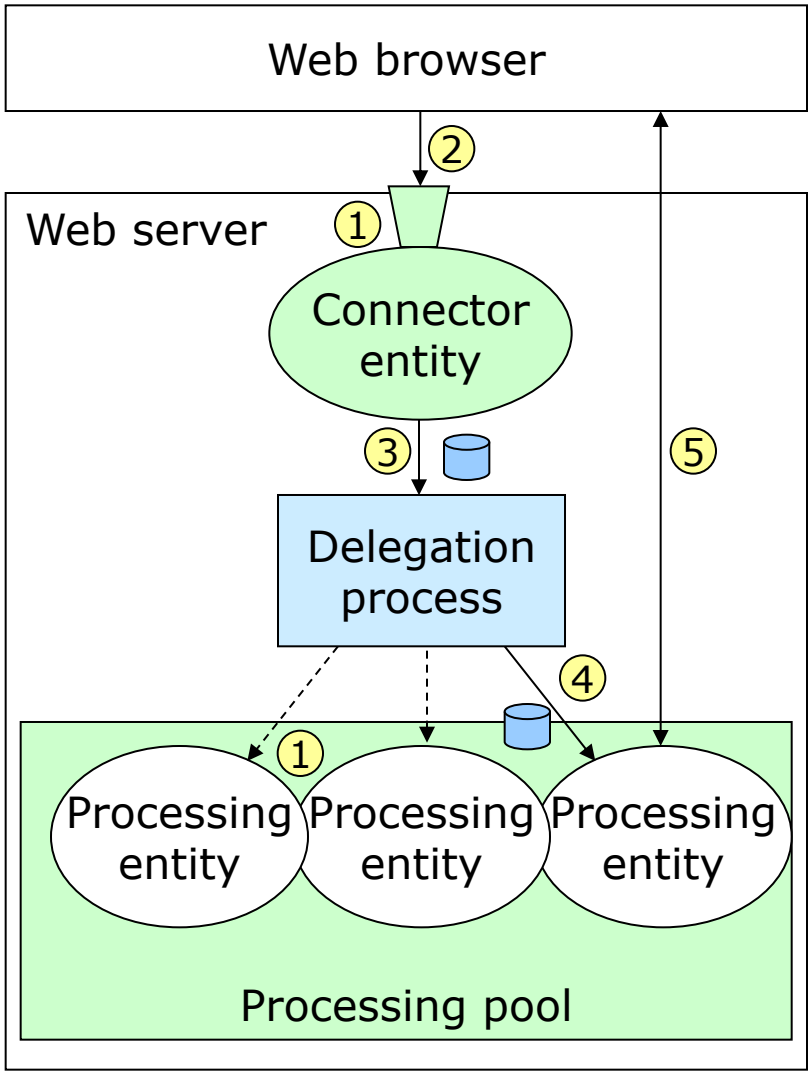
Web server architecture

- A web server is a software that makes resources available through an interface that is accessible by HTTP
- Web servers that deliver static content only have to extract a requested document from the file system and pass it to the web browser



- Due to the need for low latency time and to handle high load current web servers are structured in a hierarchical way that allows parallel request processing

Web server architecture



1. On start-up the web server creates a pool of entities for client handling and opens an interface for incoming connection requests

---

2. Web browser initiates communication on the basis of TCP through this interface which is handled by a connector entity

---

3. The connector entity creates a client descriptor which is the endpoint of a communication path to the client and passes it to the delegation process

---

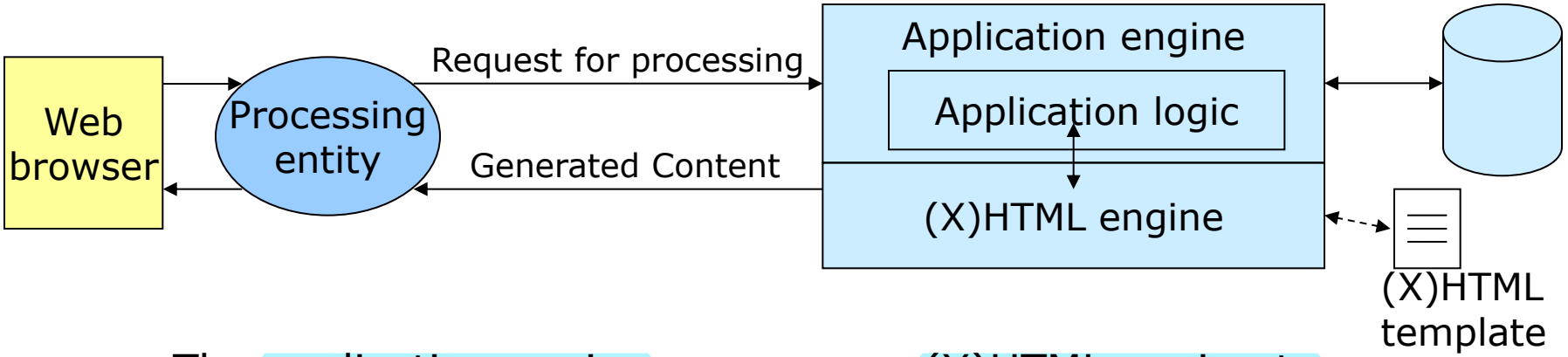
4. The delegation process chooses an available pregenerated processing entity and forwards the client descriptor to it

---

5. Communication with the web browser is finally realized through HTTP

Server-side logic

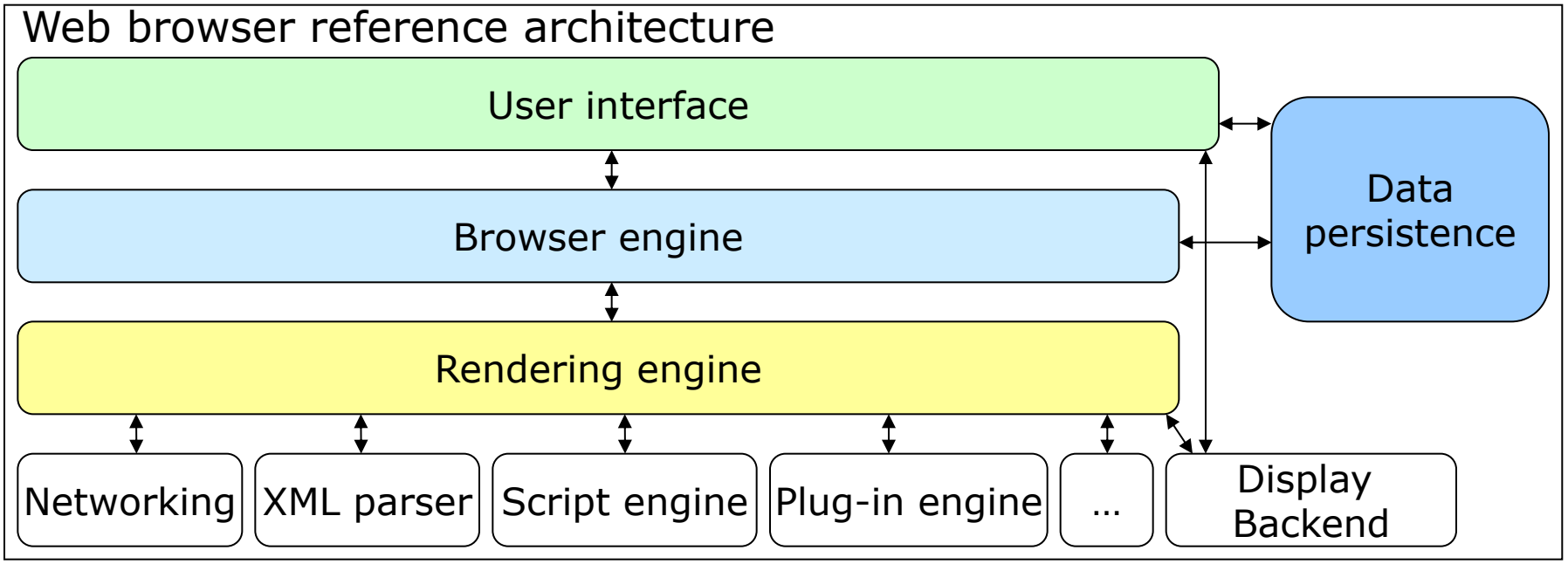
- Web applications are based on dynamic content which is generated by application logic based on e.g.:
  - Common Gateway Interface (CGI) Scripts
  - Java Servlet Container
  - Application Server that provides further services such as transaction, security or directory services and thus enables development of complex logic



- The application engine may use an (X)HTML engine to combine dynamic content with (X)HTML templates to generate resulting documents that finally are delivered to the web browser

Web browser architecture

- Early web browsers only requested documents and presented them after a rendering process to the user
- Today web browsers are extended by further technologies as especially XML support, script engines and plug-in engines

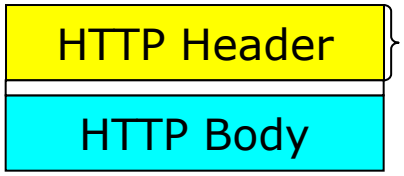


Web browser architecture

- **User interface:** Front-end for displaying a page to the user
- **Browser engine:** Embedded component that provides a high level interface for querying and using the rendering engine
- **Rendering engine:** Performs parsing and layout for (X)HTML documents enriched with other languages such as CSS
- **Networking:** Realises HTTP communication with the server
- **XML parser:** Parses XML content
- **Script engine:** Executes scripts embedded in (X)HTML pages
- **Display backend:** Provides drawing and windowing primitives, user interface widgets and fonts (e.g. GNU Image Manipulation Program Toolkit (GTK+))
- **Data persistence:** Stores associated data (cached pages, cookies etc.)
- **Plug-in engine:** Dynamic extension point for plug-ins e.g. an Web feed reader plug-in
- Further subsystems such as e.g. an integrated Extensible Stylesheet Transformation (XSLT) processor are not shown

HTTP

- The Hypertext Transfer Protocol (version 1.1 RFCs 7230-7235) realises communication on top of TCP by the exchange of messages in a request-response manner
- Every message is divided into a header and a body



Divided into several sub-headers

- The header specifies the operation that should be performed on the addressed resource and includes parameters (passed as key-value pairs)
- Resources are addressed by the request through Uniform Resource Identifiers (URI) which in this context are simple strings that identify a resource via name, location or any other characteristic:
  - **Request-URI = "*" | absolute URI | absolute Path**

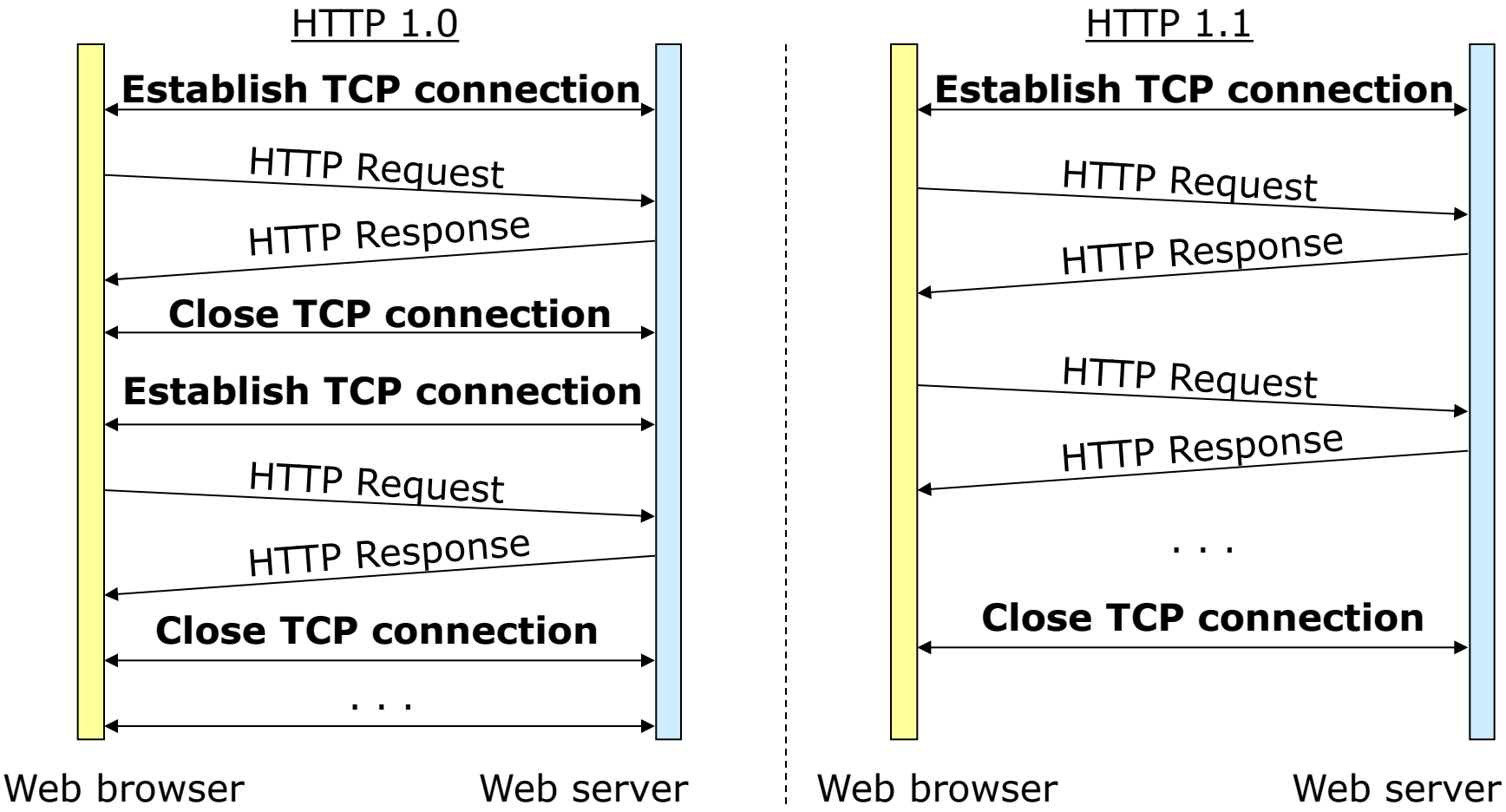
Used if no resource is associated with the request

Path to the resource on the server combined with the server address e.g. http://www.serv.com/index.html

Absolute path to the resource on the server e.g. /index.html

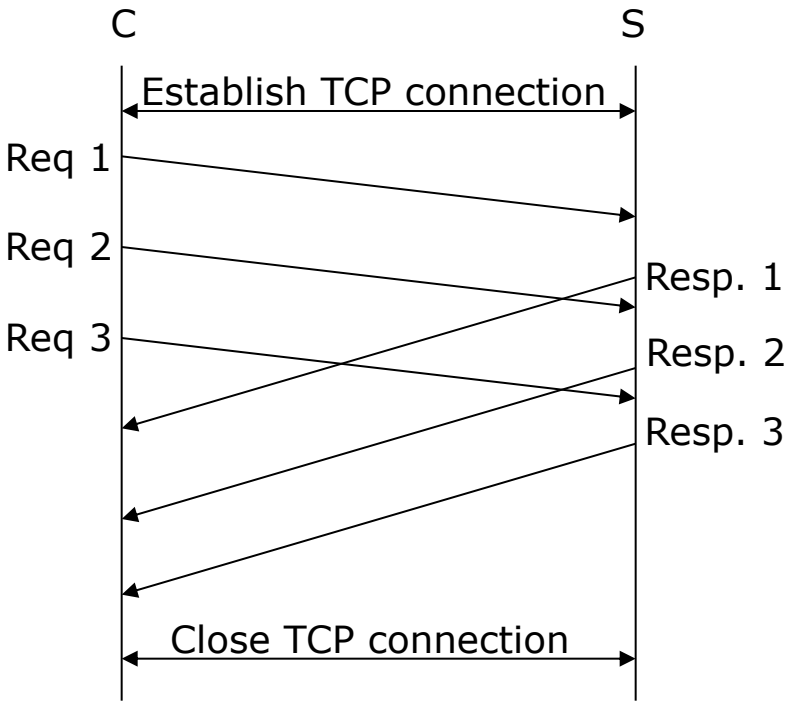
HTTP 1.1

- In contrast to version 1.0 default behaviour of HTTP 1.1 defines establishment of a persistent connection ("connection: keep-alive") that makes communication more efficient



HTTP 1.1 pipelining

- Pipelining is a technique that makes it possible to send further requests to the server without the need to wait for the response of previous requests
- Responses are transmitted in the order in which the associated requests were sent
- Not supported by all common browsers

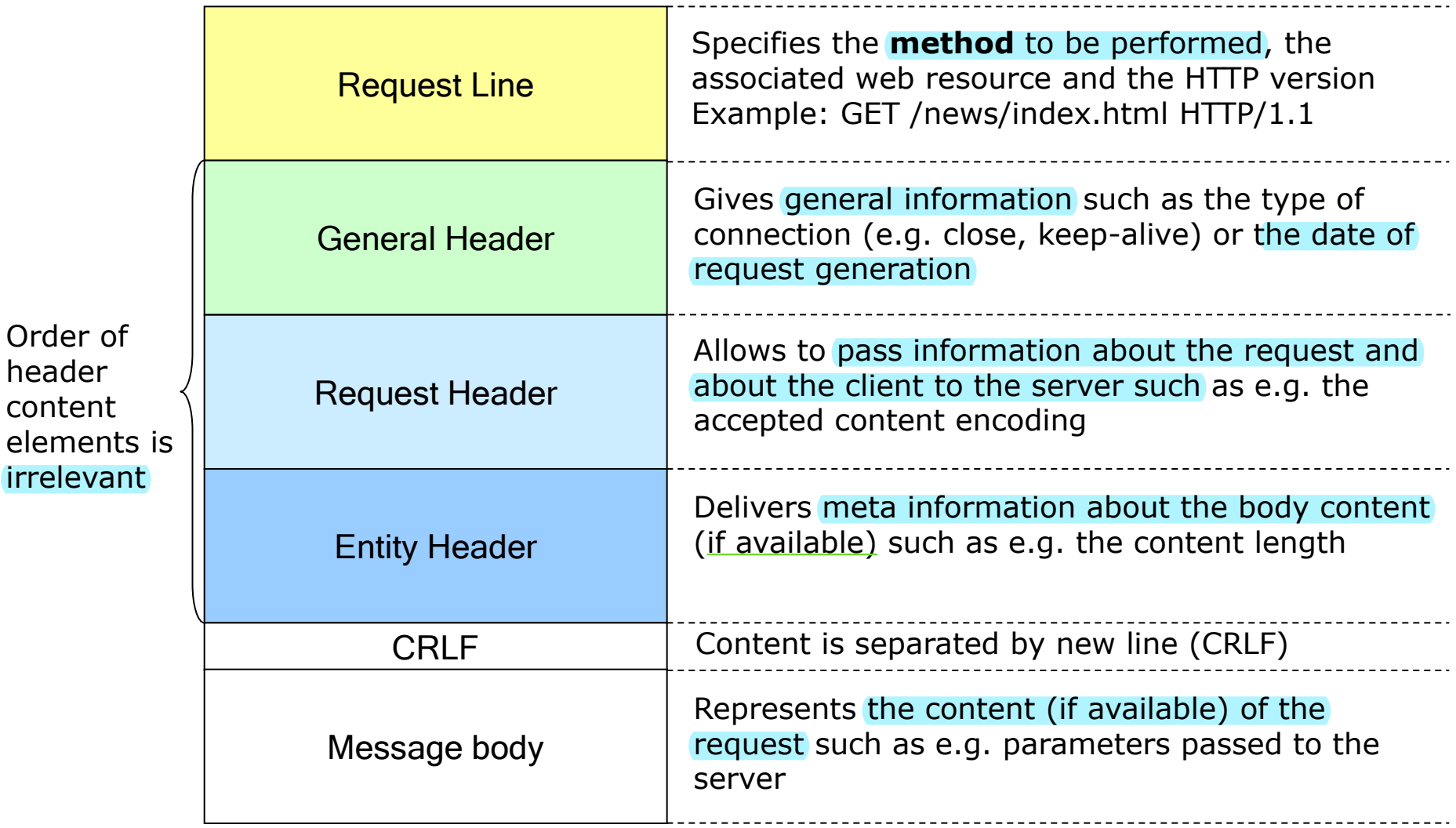


	HTTP/1.0	HTTP/1.1 Persistent	HTTP/1.1 Pipeline
Packets from client to server	226	70	25
Packets from server to client	271	153	58
Total number of packets	497	223	83

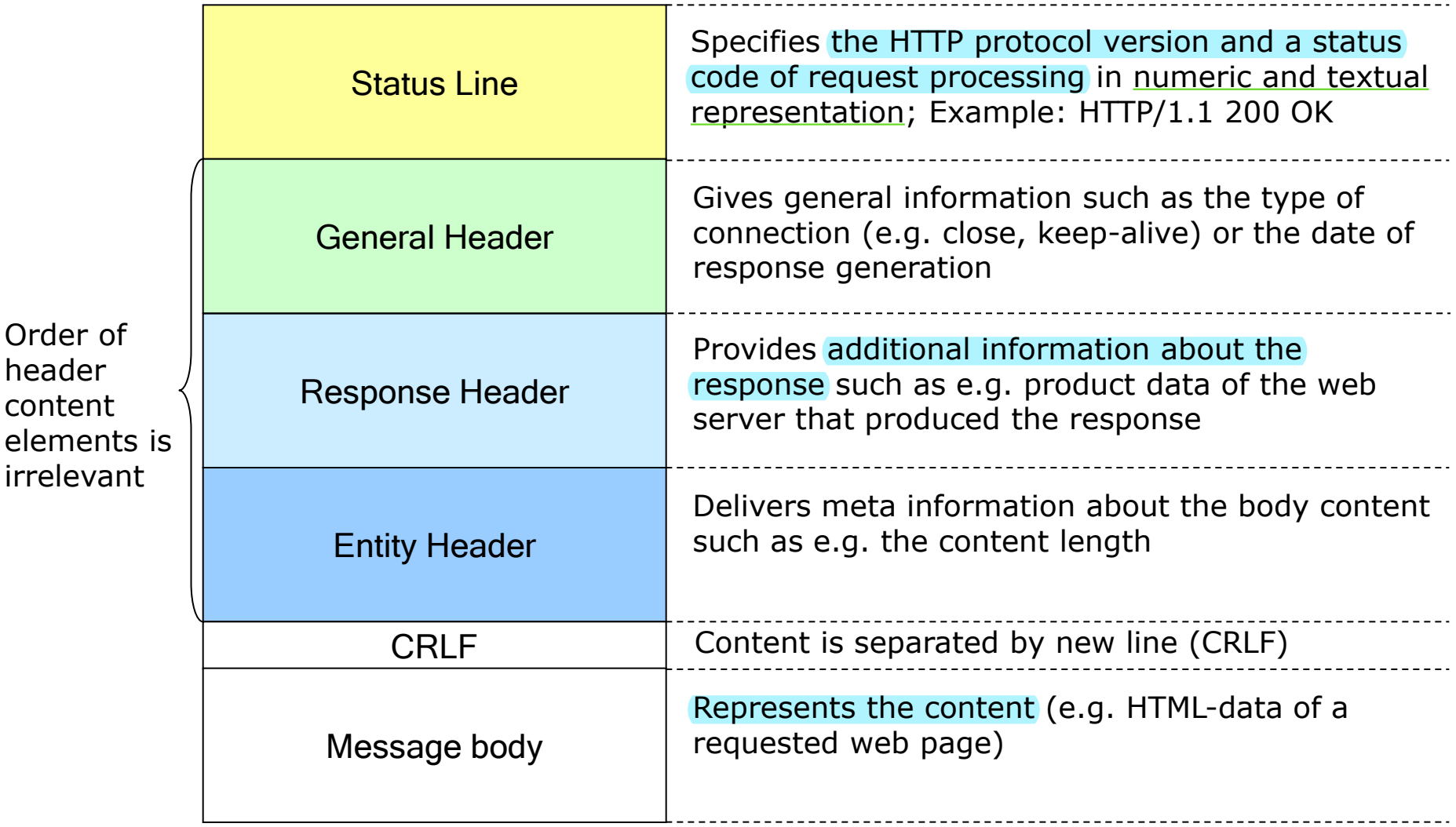
Multiple HTTP messages can be sent with the same TCP segment

Taken from http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html

HTTP 1.1 request message



HTTP 1.1 response message



HTTP 1.1 methods

- Methods specify the resource by passing its absolute or relative Uniform Resource Identifier (URI)
- If no resource is needed an asterisk is passed
- HTTP 1.1 defines an expandable set of methods:

GET	Requests for delivery of the specified resource
HEAD	Identical to GET except the message body of the response is empty thus the client only receives meta information of the header; can e.g. be used to determine the type of a file or its length without the need to receive the file
POST	Submits the message body content to the specified resource
PUT*	Instructs the server to make the passed content available under the specified URL
DELETE*	Deletes the specified resource
OPTIONS*	Requests communication options from server such as e.g. the methods that are supported by the server
TRACE*	Echoes back the send request; can be used for diagnostic purposes
CONNECT*	Reserved name for use by proxy servers

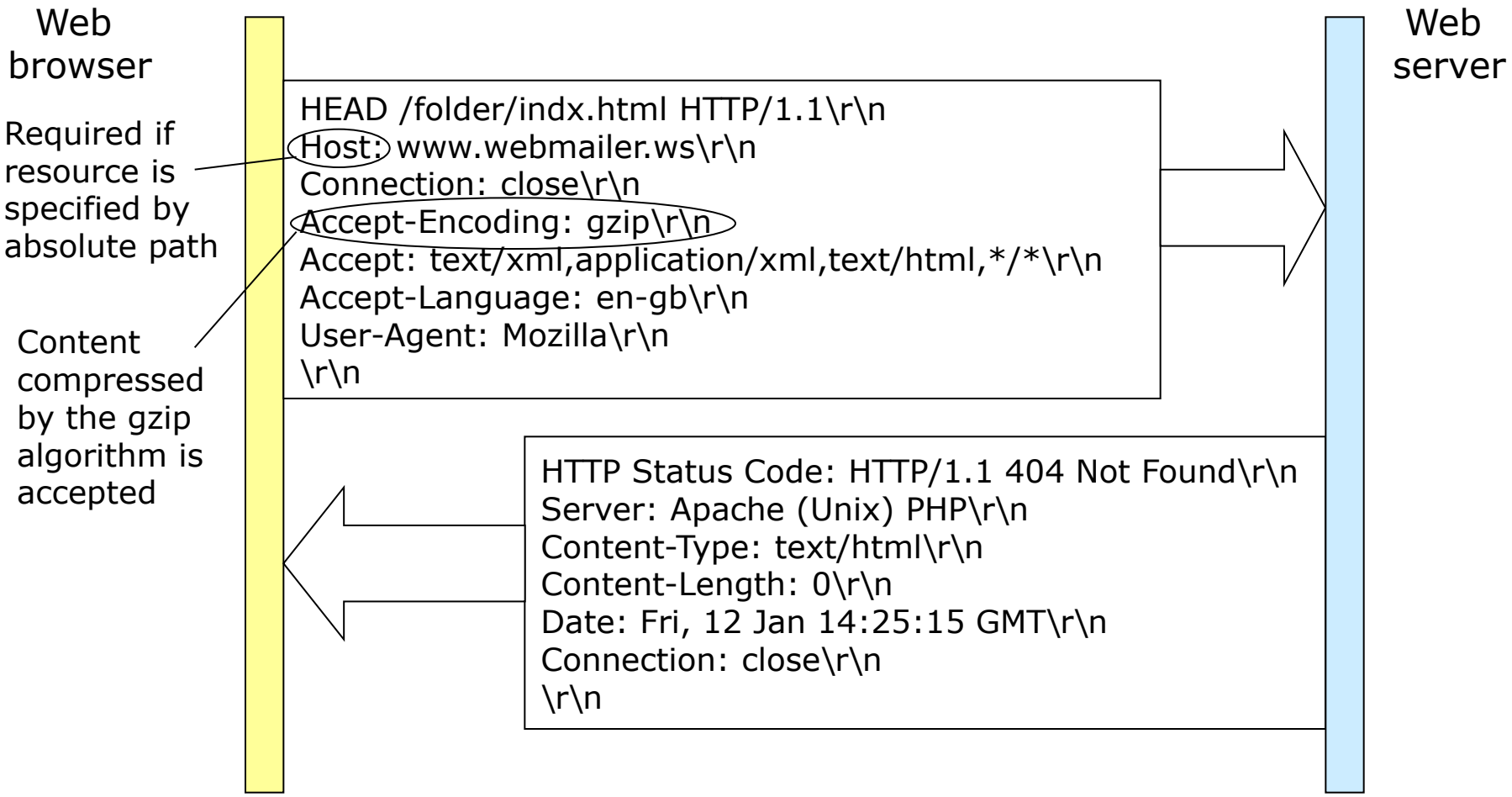
* Not defined by HTTP 1.0

HTTP 1.1 status codes

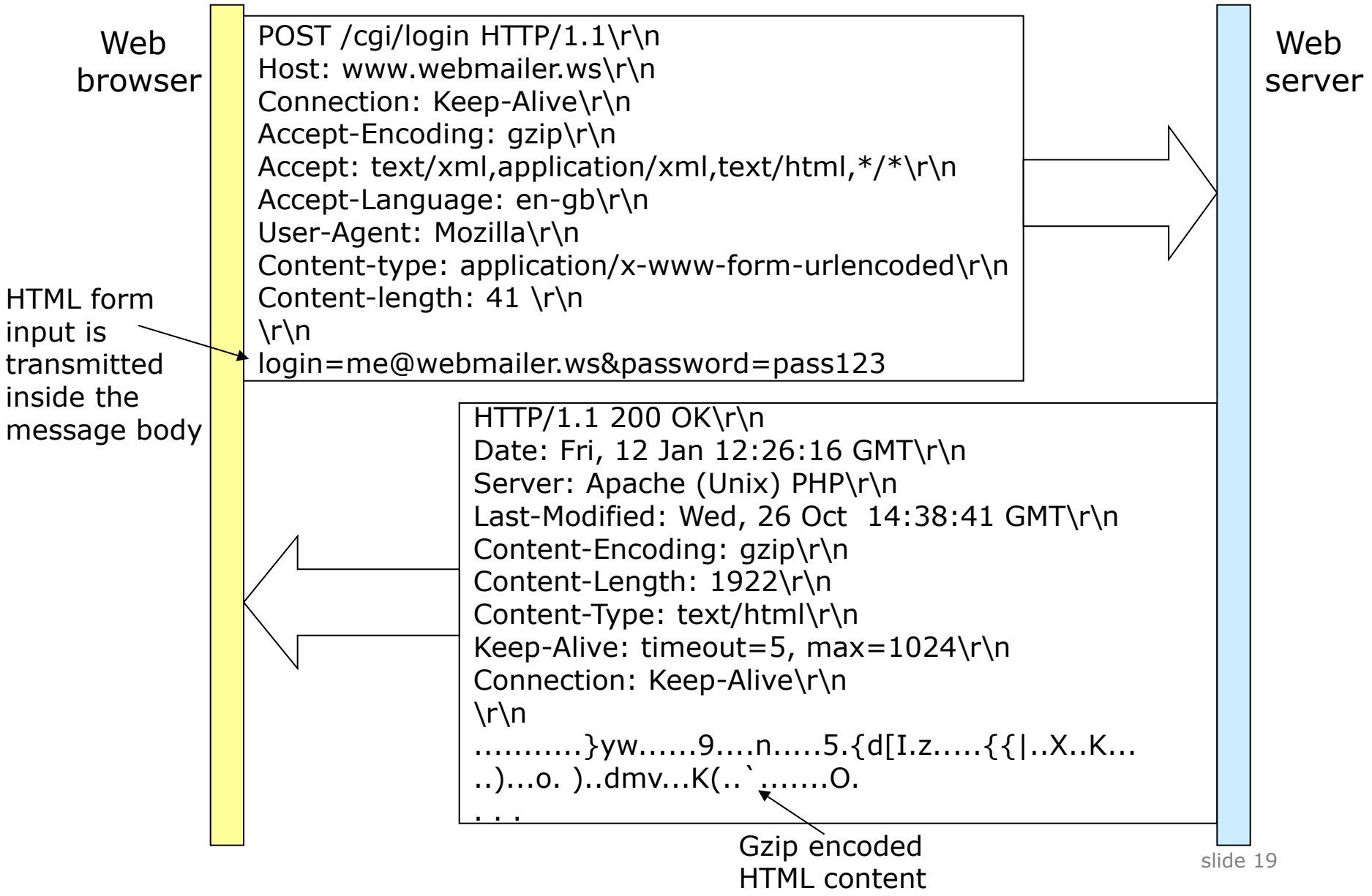
- Numeric representation of status codes consists out of a three digit value
- The first digit describes the category of the status

Code	Status category	Description	Example : Explanation
1xx	**Informational**	Request received, continuing processing	**100** : Client should continue sending the request (e.g. extensive request)
2xx	**Success**	The action was successfully received, understood and accepted	**200** : Request has succeeded
3xx	**Redirection**	Further action must be taken in order to complete the request	**301** : Requested resource has been moved permanently to new URI
4xx	**Client error** (Request error)	The request contains bad syntax or cannot be fulfilled	**404** : Server has not found anything matching the Request-URI
5xx	**Server error**	The server failed to fulfil an apparently valid request	**501** : Server does not support the functionality required to fulfil the request

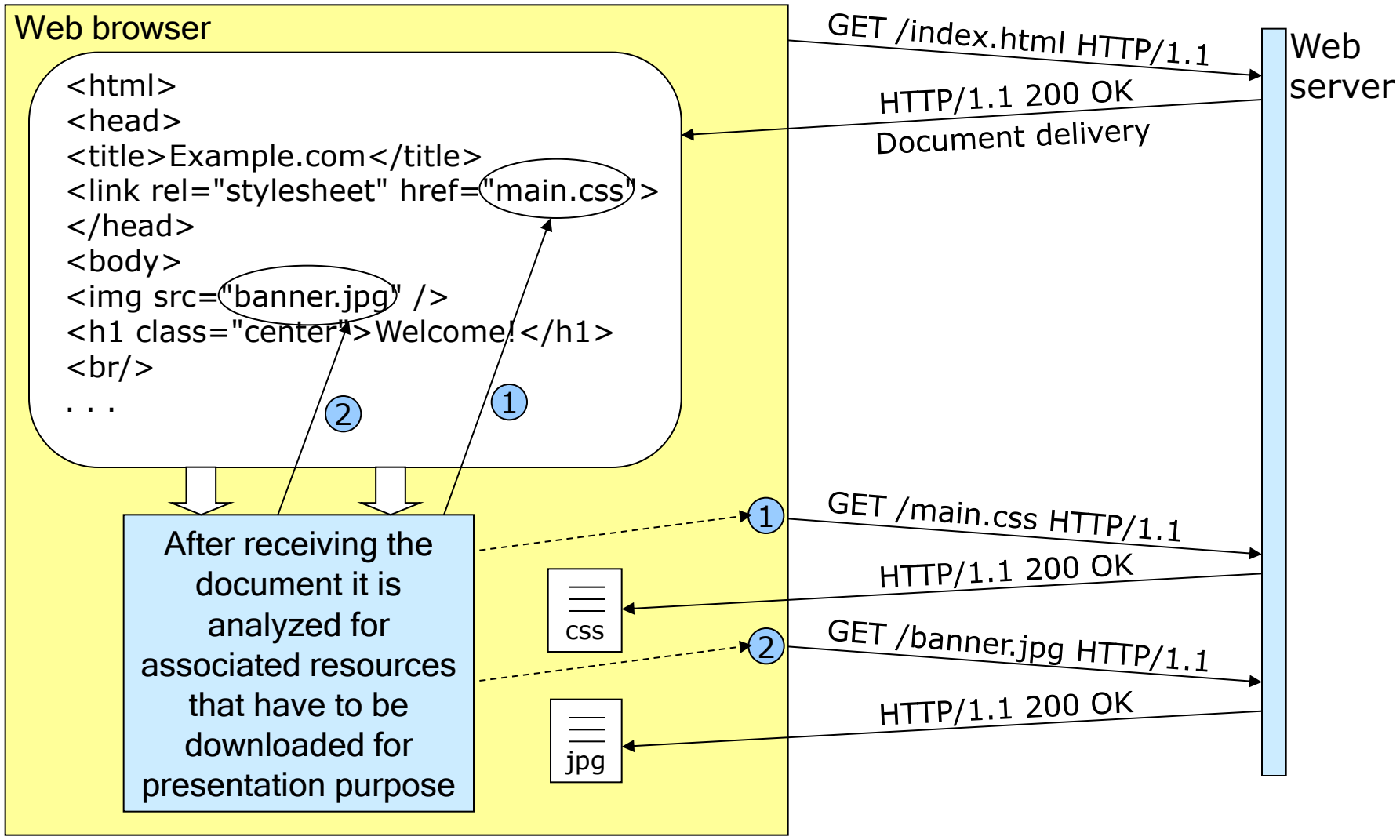
HTTP 1.1 communication details



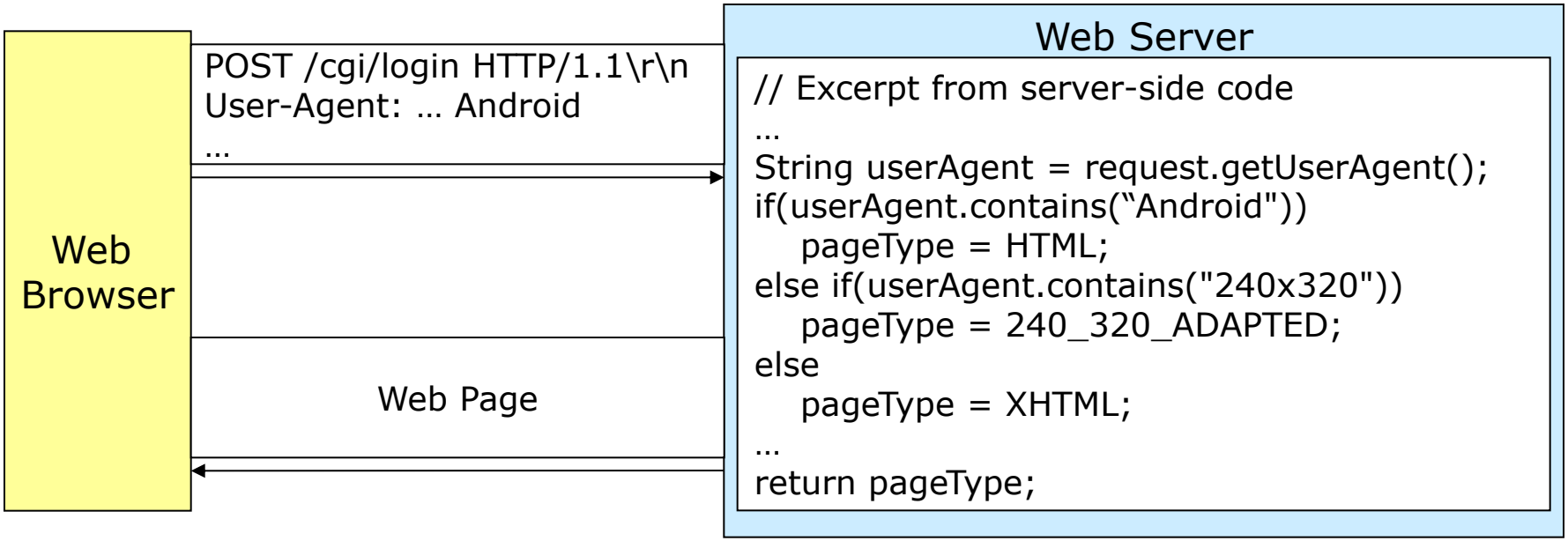
HTTP 1.1 communication details



HTTP download of associated resources



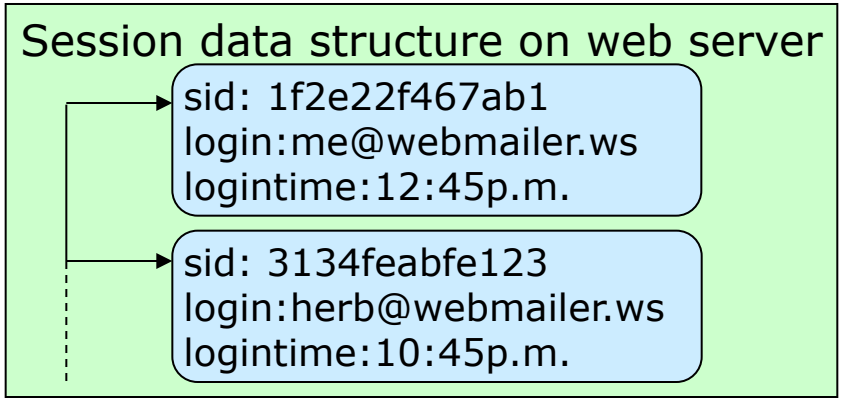
HTTP user agent detection

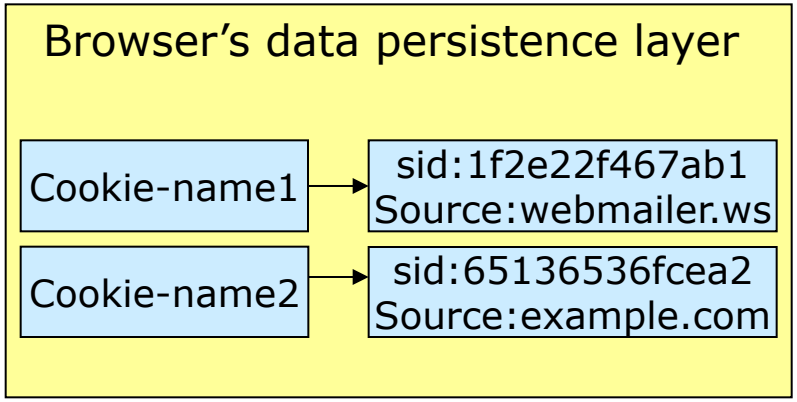


- HTTP header information makes adaptation of delivered content to the needs of user agents possible
- Example for user-agent strings:
  - "Mozilla (compatible; MSIE; Windows; .NET CLR; .NET CLR)"
  - "Mozilla [en] (Android)"
  - "Mozilla (compatible; MSIE; Windows Mobile; PPC; 240x320)"
- Analogous with e.g. content language adaptation ("Accept-Language")

Session handling

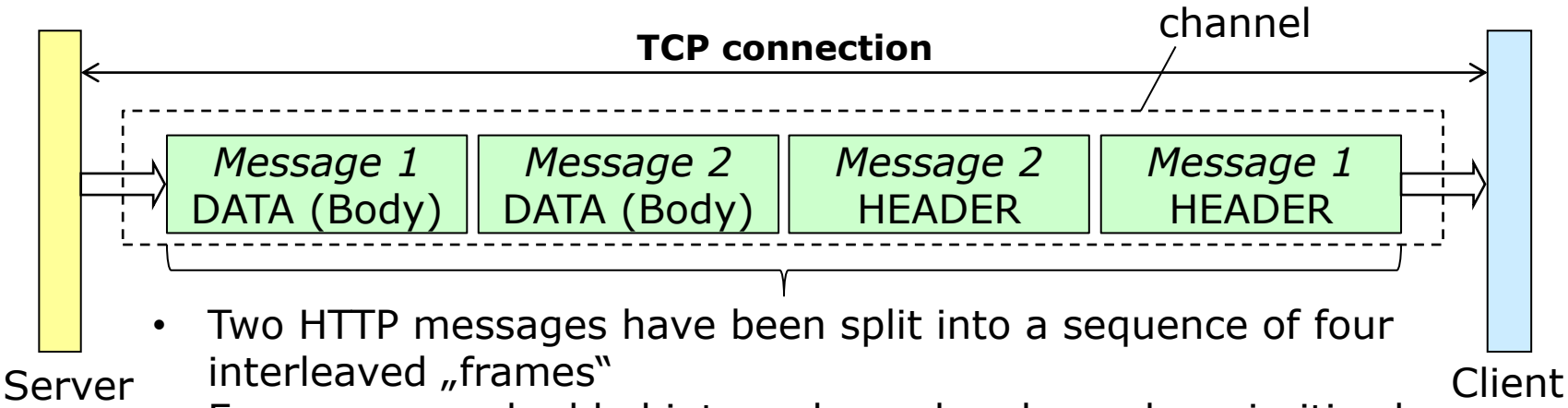
- A session context establishes a state for Web applications
- During a session the browser can store information about a user in a special data structure
- For identification purpose the web server sends a session id to the web browser which is included into later requests thus allowing the server to associate the right session data structure with the originator of a request
- Famous technique for session id exchange is the use of cookies which are small pieces of data stored on client-side





HTTP/2

- HTTP 1.1 has been optimized for simplicity and for applicability to a wide range of use cases
- Though the performance has been improved by HTTP 1.1, the protocol *has not been optimized for performance*
- Goal of HTTP/2: Provide an optimized transport for HTTP semantics using **multiplexing** and **header compression**
- HTTP methods, status codes and semantics are not modified by HTTP/2
- Has been approved by the steering group of the IETF for publication as standards-track RFCs in February 2015



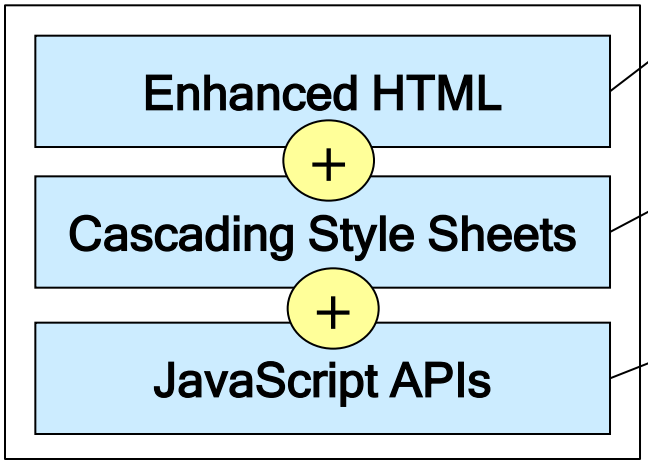
- Two HTTP messages have been split into a sequence of four interleaved „frames"
- Frames are embedded into a channel and may be prioritized enabling faster delivery of selected frames
- Per TCP connection several parallel channels can be established

→ HTTP/2 will be further discussed in the exercise

HTML5 - Overview

- HTML5, being still in the standardization process, is intended to be the next generation of HTML addressing current Web applications' requirements
- Enables mobile and desktop Web site designers to deliver the advantages of client-side and server-side development to their users simultaneously
- Proposed standard of HTML5 aggregates several previously separated developments

~=



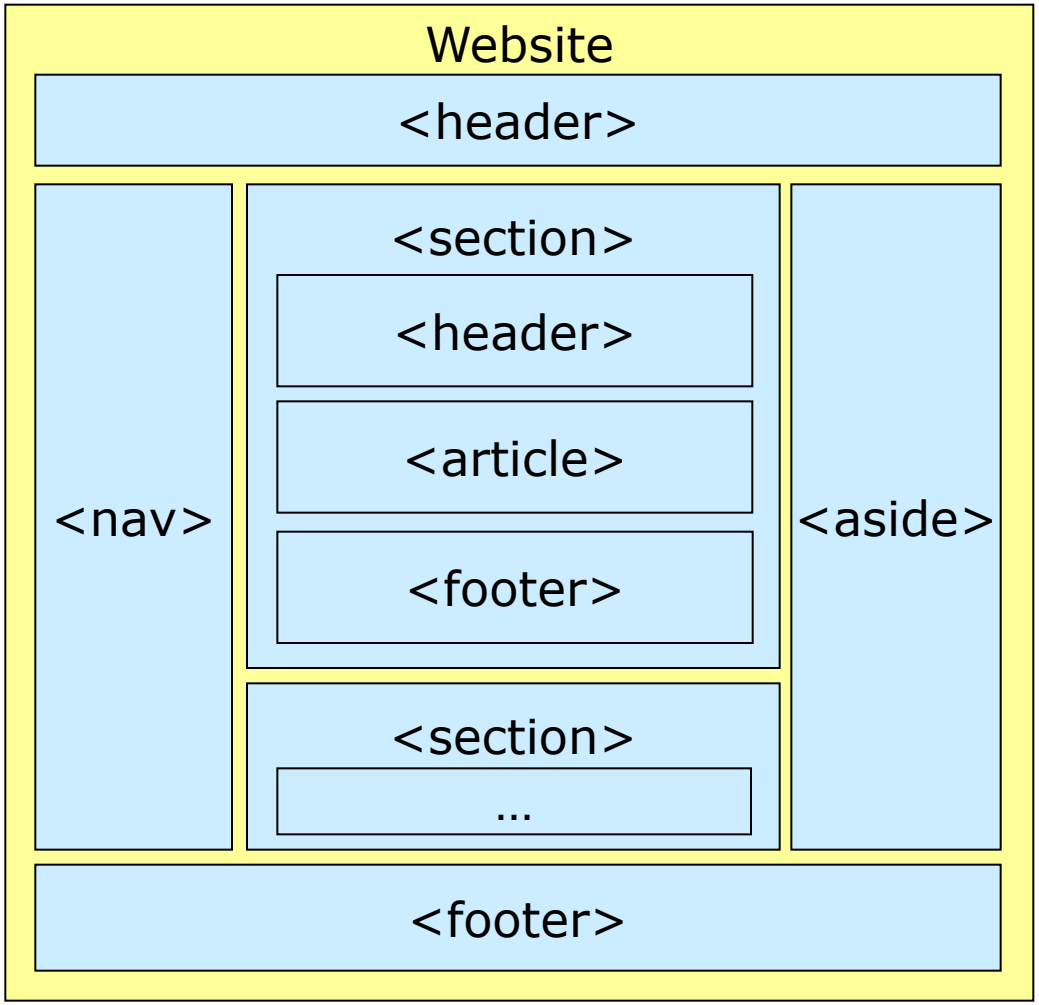
Enhanced Web document structure with new HTML elements and attributes

Improved text and object styling and improved animation effects via CSS 3

Standardization of partially existing APIs and introduction of new APIs: e.g. Web storage for storing data on client-side, Web Workers for parallelizing tasks, JavaScript Geolocation API for locating a user

HTML5 - Document structure

- HTML5 introduces several important structural elements used for grouping (not for positioning – for this purpose CSS is applied):



<header>: group of introductory or navigational aids

<section>: generic document or application section - used for grouping purposes

<article>: *independent* piece of content of a document, such as blog entry or news article

<footer>: represents a footer for the section or the page containing information e.g. about the author or copyright information

<nav>: section of navigation

<aside>: represents content that is only slightly related to the rest of the page (e.g. a list of links to external Web resources)

HTML5 - Audio and Video support

- HTML5 introduces built-in media support via the <audio> and <video> elements in addition to well-defined JavaScript APIs for media control

Document type of HTML5 pages is specified as "html"

Alternative video formats can be provided

specifies text tracks e.g. in the form of subtitles

Implementation of start/stop video control functionality

```
<!doctype html>
<html>
…
<button onclick="playPause()">Play/Pause</button>
<br />
<video id="video1" width="320" height="240">
  <source src="lecture.mp4" type="video/mp4" />
  <source src="lecture.ogg" type="video/ogg" />
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en"/>
  <track src="subtitles_de.vtt" kind="subtitles" srclang="de"/>
</video>
<script type="text/javascript">
    var myVideo=document.getElementById("video1");
    function playPause(){
        if (myVideo.paused)
          myVideo.play();
        else
          myVideo.pause();
    }
</script>
</html>
```

Example subtitles

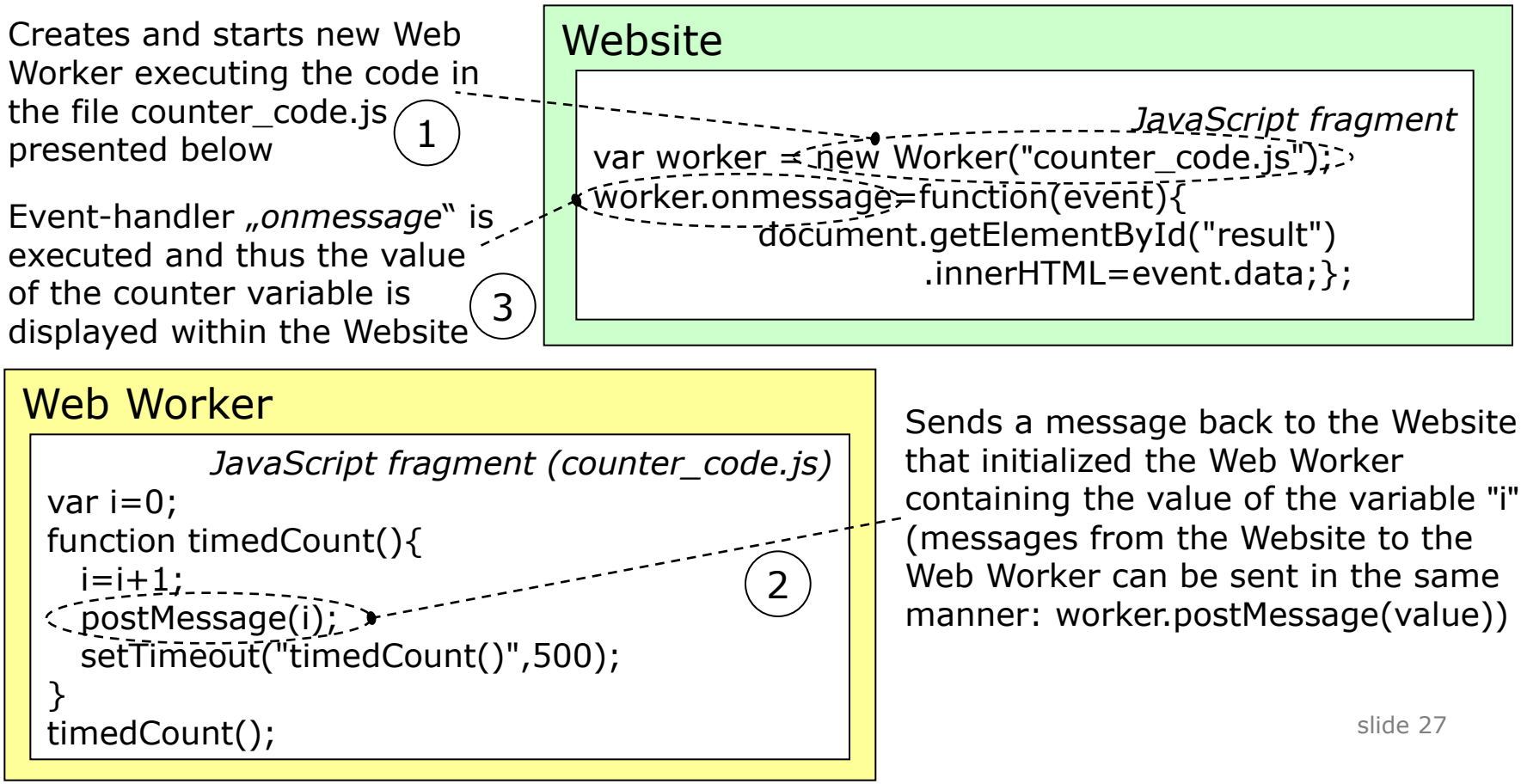
```
WEBVTT

00:11.000 --> 00:13.000
Overview of HTML5
00:13.000 --> 00:16.000
HTML5 document structure
```

Web VTT (Web Video Text Tracks) format:
http://dev.w3.org/html5/webvtt

HTML5 - Web Workers

- Web Worker = JavaScript that runs in the background, independently of further user interface (UI) scripts thus allowing parallel processing
- Important for performing computationally expensive tasks without interrupting the UI
- Interaction between Web Workers and UI thread is event-based



HTML5 - Storage

- HTML5 offers three different means for storing data on client side: Web Storage, Indexed Database API, Web SQL Database
- All of them apply the "same-origin-policy": script can only access data that a script from the same source (identified by protocol+hostname+port) has stored

Web Storage

→ simple key/value persistence system
Advantage: Very simple API, supported by most browsers
Disadvantage: only for simple use cases:
- does not offer any query languages, no schemas, no transactional safety

---

Indexed Database API

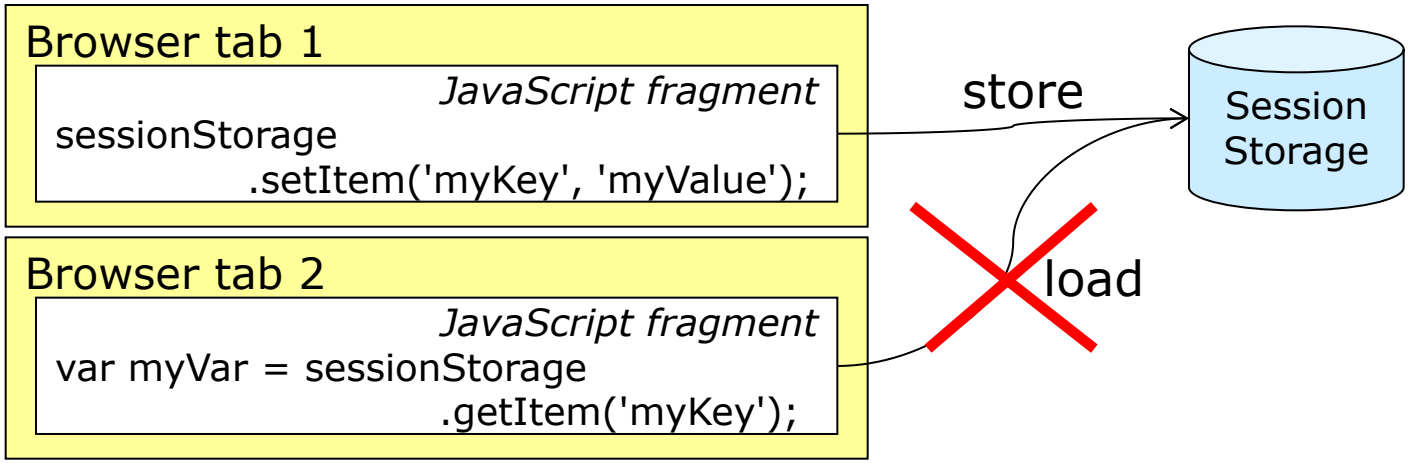
→ API that allows to operate on a set of indexable object stores persisted in a client's web browser environment
Advantage:simplifies the programming model for interacting with databases though offering a powerful persistence layer
Disadvantage: Not implemented in all browsers yet

Web SQL Database

→ simplified SQL implementation embedded into the browser
Advantage: Efficient and feature-rich SQL implementation (including joins, inner selects etc.)
Disadvantage: Specification effort has been stopped by World Wide Web Consortium due to lack of implementation

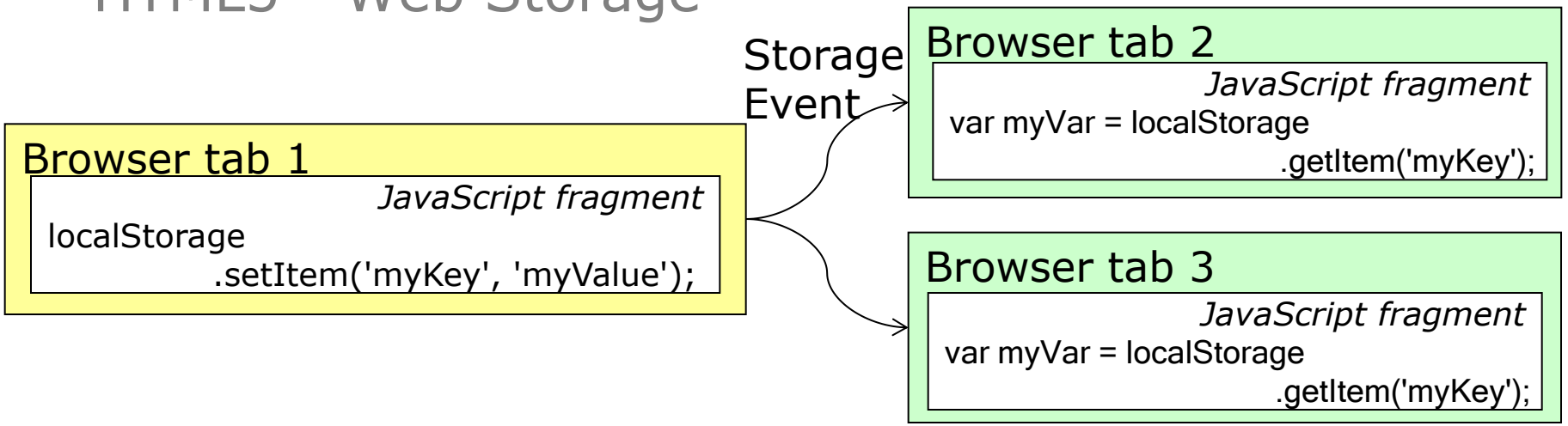
HTML5 - Web Storage

- A key-value pair storage framework that provides behavior similar to cookies but with larger storage capacity and improved API
- Access is realized via getter and setter methods
- Two scopes of storage are available:
  - Session: valid as long as top-level browsing context (=usually window or tab of browser) is active



  - Local: no temporal restrictions, data can be accessed from different windows or tabs of browser
- Via removeItem('itemKey') a single item with key 'itemKey' can be removed and via clear() all stored items can be removed

HTML5 - Web Storage



- Data in localStorage can be accessed from different browser windows/tabs
- As soon as data in localStorage is modified, a storage event is propagated and can be handeled via registered event listeners

Event name for which listener is registered

```
window.addEventListener('storage',
function(event) {
        console.log('Value for ' + event.key + ' was changed from'
        + event.oldValue + ' to ' + event.newValue);
        });
```

function is called if storage event appears

HTML5 - Indexed Database API

- Indexed Database API (IndexedDB) is an API for client-side storage of significant amounts of **structured data** and for high performance searches on this data using keys, indexes or cursors
- Database operation result in DOM event that has to be handeled

1

```
var request = window.indexedDB.open("BookDB");
request.onsuccess = function(event){
            var database = event.result;
            write("Database Opened", database);
};
```

Open a database with name "BookDB" – if the method call is successful, the onsuccess event handler is executed

2

```
objectStore = database.createObjectStore
("BookList", "id");
```

Object store named "BookList" with index named "id" is created in the database

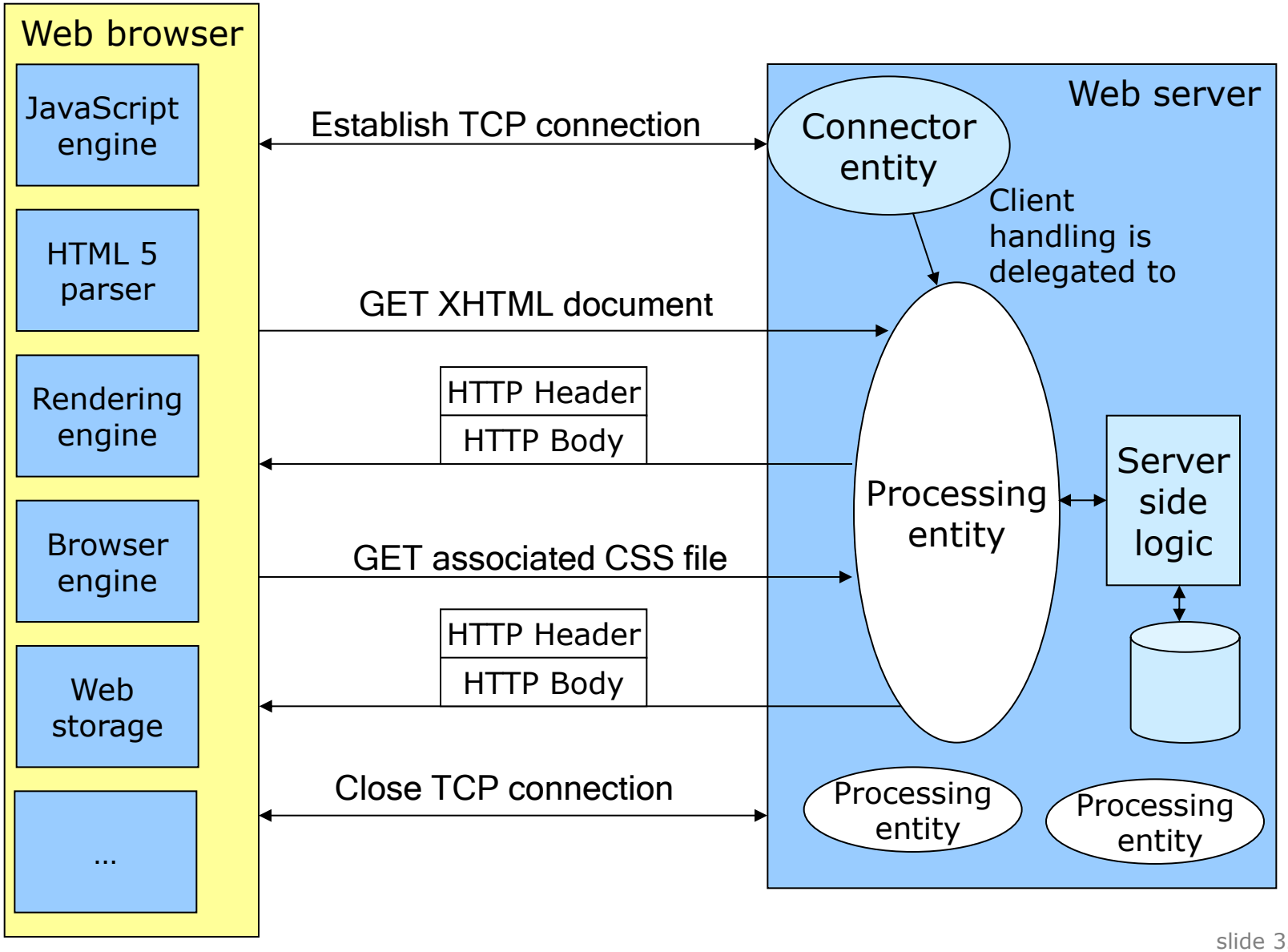
3

```
var book = {"bookName" : " Web Applications",
            "price" : 20, "rating":"excellent"};
var request = objectStore.add(book);
request.onsuccess = function(event){
document.write("Saved using id ", event.result)
var key = event.result; };
```

Stores the value of the variable "book" in the created object store – if this step is successful, the onsuccess event handler is executed and the key of the object can be read

Realization of key-, index- and cursor-based access to the database will be discussed in the exercise

Conclusion


Web browser
JavaScript engine
HTML 5 parser
Rendering engine
Browser engine
Web storage
...
Establish TCP connection
GET XHTML document
HTTP Header
HTTP Body
GET associated CSS file
HTTP Header
HTTP Body
Close TCP connection
Web server
Connector entity
Client handling is delegated to
Processing entity
Server side logic
Processing entity
Processing entity

References

HTML5 Working Draft	http://www.w3.org/TR/html5/
HTML5 demos and examples	http://html5demos.com/

RFCs:

HTTP 1.0 spec.	http://tools.ietf.org/html/rfc1945
HTTP 1.1 spec. (besides further RFCs)	http://tools.ietf.org/html/rfc7230
	http://tools.ietf.org/html/rfc7231

Details about the REST architecture:

REST	http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

HTTP/2 spec.	http://tools.ietf.org/html/draft-ietf-httpbis-http2-17