

Q

What do we mean by safe and idempotent methods?

Or: are POST and GET equivalent ?

idempotent methods

<https://developer.mozilla.org/en-US/docs/Glossary/idempotent>

An HTTP method is **idempotent** if an **identical request** can be made once or several times in a row with the **same effect** while **leaving the server in the same state**.

An idempotent method should **not have any side-effects** (except for keeping statistics).

Implemented correctly, the GET, HEAD, PUT, and DELETE method are **idempotent**, but the POST method is not.

safe methods

<https://developer.mozilla.org/en-US/docs/Glossary/safe>

An HTTP method is safe if it **doesn't alter the state of the server**.

A method is safe if it leads to a read-only operation.

Several common HTTP methods are safe: GET, HEAD, or OPTIONS.

All safe methods are also idempotent, but not all idempotent methods are safe.

For example, PUT and DELETE are both idempotent but **unsafe**.

Safe, unsafe and idempotent methods

- **GET /pageX HTTP/1.1 is idempotent**. Called several times in a row, the client gets the same results:
 - GET /pageX HTTP/1.1
 - GET /pageX HTTP/1.1
 - GET /pageX HTTP/1.1
- **POST /add_row HTTP/1.1 is not idempotent**; if it is called several times, it adds several rows:
 - POST /add_row HTTP/1.1
 - POST /add_row HTTP/1.1 -> Adds a 2nd row
 - POST /add_row HTTP/1.1 -> Adds a 3rd row
- **DELETE /idX/delete HTTP/1.1 is idempotent**, even if the returned status code may change between requests:
 - DELETE /idX/delete HTTP/1.1 -> Returns 200 if idX exists
 - DELETE /idX/delete HTTP/1.1 -> Returns 404 as it just got deleted
 - DELETE /idX/delete HTTP/1.1 -> Returns 404

Q

What is Aspect Oriented Programming?

AOP

Aspect-oriented programming (AOP) attempts to aid programmers in the **separation of concerns**, specifically **cross-cutting concerns**, as an advance in modularization.

Logging and authorization offer two examples of crosscutting concerns:

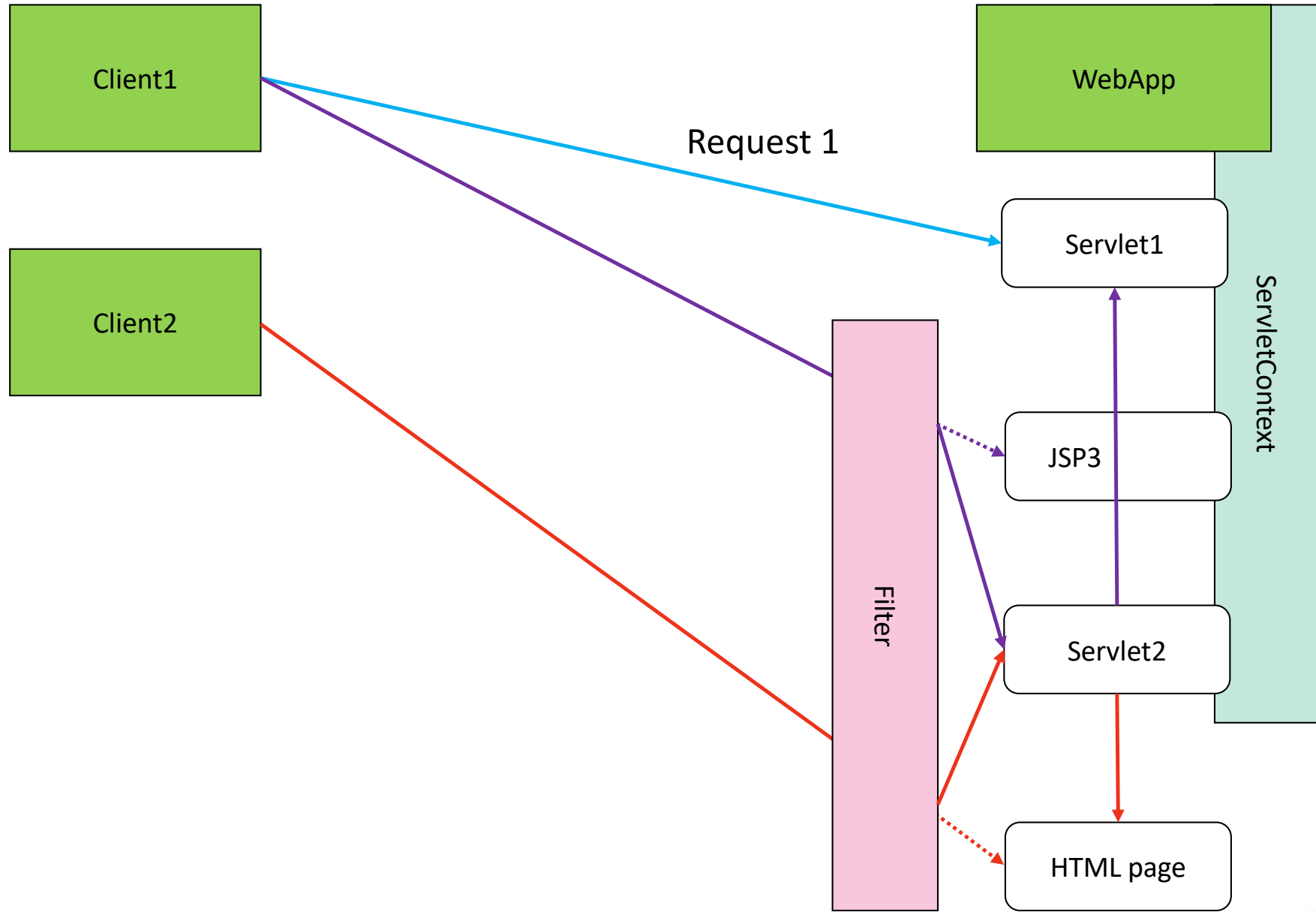
a logging strategy necessarily affects every single logged part of the system. Logging thereby crosscuts all logged classes and methods.

Same is true for authorization.



Q

What are Filters?



Filters (javax.servlet.filter)

Classes that preprocess/postprocess request/response

A filter is an object that performs **filtering tasks** on either the

- request to a resource (a servlet or static content),
- the response from a resource.

Filters perform filtering in the **doFilter** method. Every Filter has access to a **FilterConfig** object from which it can obtain its initialization parameters and a reference to the **ServletContext**.

They provide the ability to **encapsulate recurring tasks in reusable units**.



Filters (javax.servlet.filter)

Filters are configured:

- in the deployment descriptor of a web application
- via annotation (See <https://docs.oracle.com/javaee/7/api/javax/servlet/annotation/WebFilter.html>)



Filters

Filters can perform many different types of functions:

- * **Authentication** -> Blocking requests based on user identity
- * **Logging and auditing** -> Tracking users of a web application
- * **Image conversion** -> Scaling maps, and so on.
- * **Data compression** -> Making downloads smaller.
- * **Localization** -> Targeting the request and response to a particular locale.
- * **XSL/T** -> transformations of XML content-Targeting web application responses to more than one type of client.

There are many more applications of filters, such as **encryption**, **tokenizing**, **triggering resource access events**, **mime-type chaining**, and **caching**.



Filters

The filtering API is defined by the **Filter**, **FilterChain**, and **FilterConfig** interfaces in the javax.servlet package.

You define a filter by implementing the Filter interface: its most important method in this interface is **doFilter**, which has parameters **request**, **response**, and **filter chain** objects. It perform the e.g. following actions:

1. Examine the request headers.
2. Customize the request object and response objects if needed
3. Invoke the next entity in the filter chain (as configured in the web.xml) by calling the doFilter method on the chain object (passing in the request and response it was called with, or the wrapped versions it may have created).



Filters methods (javax.servlet.filter)

- **public void doFilter (ServletRequest, ServletResponse, FilterChain)**
 - This method is called by the container **each time a request/response pair is passed through the chain** due to a client request for a resource at the end of the chain.
- **public void init(FilterConfig filterConfig)**
 - This method is called by the web container to indicate to a filter that it is **being placed into service**.
- **public void destroy()**
 - This method is called by the web container to indicate to a filter that it is **being taken out of service**.



Filter example

```
import javax.servlet.*; import javax.servlet.http.*;
import java.io.*;

public class LoginFilter implements Filter {
    protected FilterConfig filterConfig;
    public void init(FilterConfig filterConfig) throws
        ServletException{this.filterConfig =filterConfig; }
    public void destroy() {this.filterConfig = null; }
    public void doFilter(ServletRequest req,ServletResponse res,
        FilterChain chain) throws java.io.IOException, ServletException {
        HttpServletRequest hreq=(ServletRequest)req;
        String username = hreq.getParameter("j_username");
        if (isUserOk(username)) chain.doFilter(request,response);
        res.sendError(
            javax.servlet.http.HttpServletResponse.SC_UNAUTHORIZED);
    }
}
```



Configuration

```
<filter id="Filter_1">  
    <filter-name>LoginFilter</filter-name>  
    <filter-class>LoginFilter</filter-class>  
    <description>Performs pre-login and post-login  
operation</description>  
</filter-id>  
  
<filter-mapping>  
    <filter-name>LoginFilter</filter-name>  
    <url-pattern>/*</url-pattern>  
</filter-mapping>
```



Filters Application Order

The order of filter-mapping elements in **web.xml** determines the order in which the web container applies the filter to the servlet.



Filter sequencing example

```
<filter>
  <filter-name>Uncompress</filter-name>
  <filter-class>compressFilters.createUncompress</filter-
  class>
</filter>
<filter>
  <filter-name>Authenticate</filter-name>
  <filter-class>authentication.createAuthenticate</filter-
  class>
</filter>
<filter-mapping>
  <filter-name>Uncompress</filter-name>
  <url-pattern>/status/compressed/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>Authenticate</filter-name>
  <url-pattern>/status/compressed/*</url-pattern>
</filter-mapping>
```

Both Uncompress and Authenticate appear on the filter chain for servlets located at /status/compressed/.*.

The Uncompress filter precedes the Authenticate filter in the chain because the Uncompress filter appears before the Authenticate filter in the web.xml file.



Example: Filters and sessions

```
public void doFilter(ServletRequest req,
    ServletResponse res, FilterChain chain) throws java.io.IOException,
    ServletException {
    HttpServletRequest hreq=(HttpServletRequest) req;
    HttpSession session = hreq.getSession(false);
    if (null == session |
        !(Boolean)session.getAttribute("auth")) {
        if (isUserOk(hreq.getParameter("user"))){
            session=hreq.getSession(true);
            session.setAttribute("auth",new Boolean(true));
        } else res.sendError(
            javax.servlet.http.HttpServletResponse.SC_UNAUTHORIZED);
    }
    chain.doFilter(request, response);
}
private boolean isUserOk(String name) {...}
```



Example: Filters and parameters

```
java.util.ArrayList userList=null;
public void init(FilterConfig fc) throws ServletException {
    this.filterConfig = fc;
    BufferedReader in;
    userList = new java.util.ArrayList();
    if ( fc != null ) {
        try {
            String filename = fc.getInitParameter("Users");
            in = new BufferedReader( new FileReader(filename));
        } catch ( FileNotFoundException fnfe) {
            writeErrorMessage(fnfe); return;
        }
        String userName;
        try {
            while ( (userName = in.readLine()) != null )
                userList.add(userName);
        } catch ( IOException ioe) {writeErrorMessage(ioe);return;}
    }
}
public void destroy() { this.filterConfig = null; userList = null;}
```



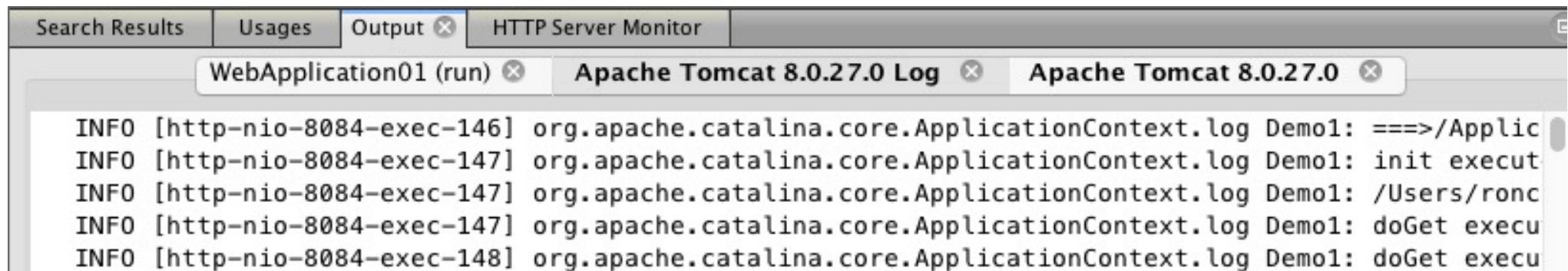
Filters and parameters

```
<filter id="Filter_1">  
  <filter-name>LoginFilter</filter-name>  
  <filter-class>LoginFilter</filter-class>  
  <description>Performs pre-login and post-login  
operation</description>  
  <init-param>  
    <param-name>Users</param-name>  
    <param-value>c:\mydir\Users.lst</param-value>  
  </init-param>  
</filter>
```



```
public void log(String msg) {  
    filterConfig.getServletContext().log(msg);  
}
```

Logging in filters



Further examples

<http://www.oracle.com/technetwork/java/filters-137243.html>

<https://www.tutorialspoint.com/servlets/servlets-writing-filters.htm>



A

2nd Assignment

See web site of the course

Q

Where do we go from here ?

Let's follow the historical evolution of the web paradigm

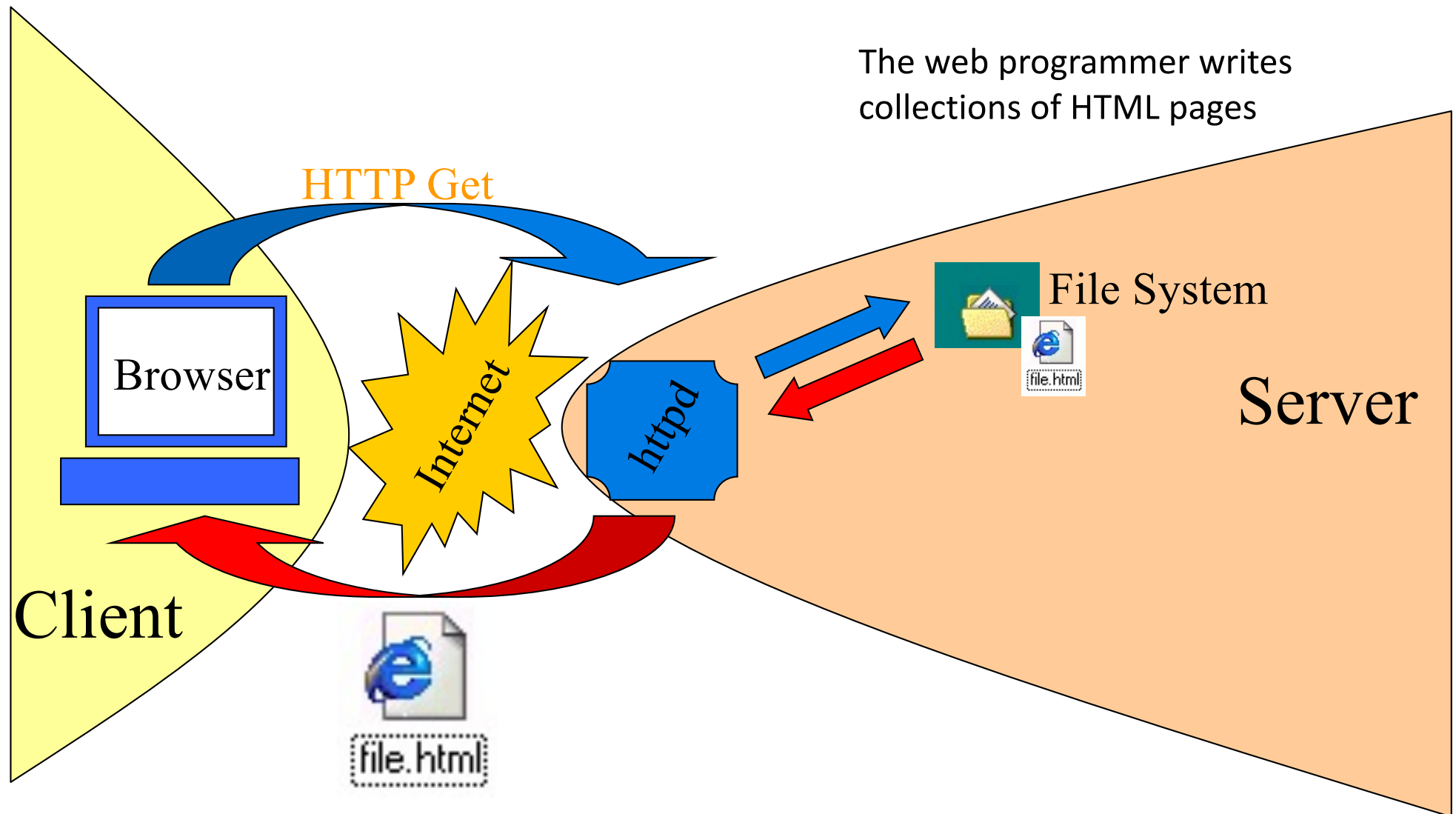
History of the web

- 1989-1990 – **Tim Berners-Lee** invents World Wide Web at CERN.



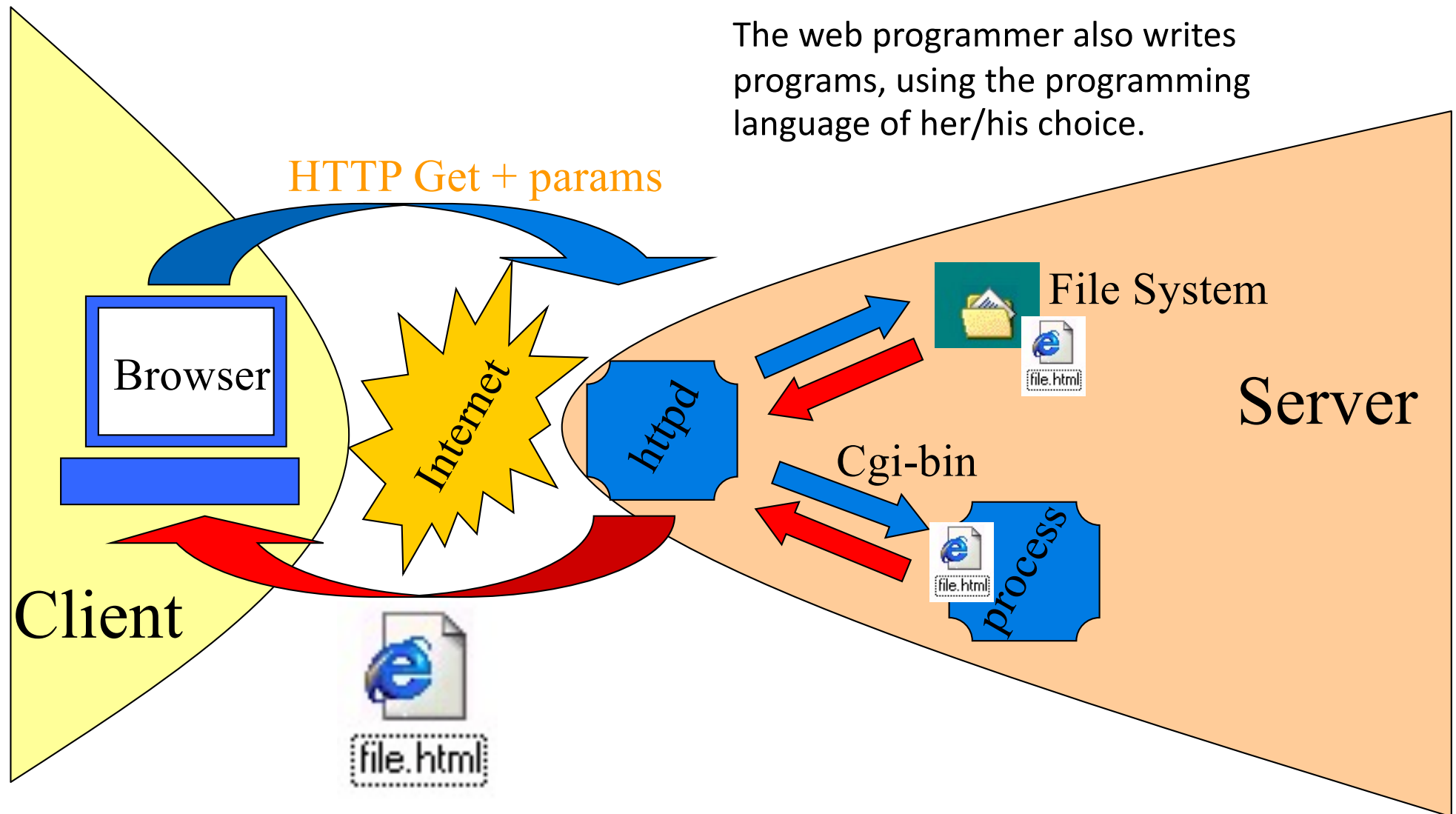
- On 30 April 1993, CERN put the World Wide Web software in the public domain. Later, CERN made a release available with an open license, a more sure way to maximize its dissemination.
- TB-L moved from CERN to the Massachusetts Institute of Technology in 1994 to found the **World Wide Web Consortium (W3C)**, an international community devoted to developing open web standards.

The original web architecture: static pages



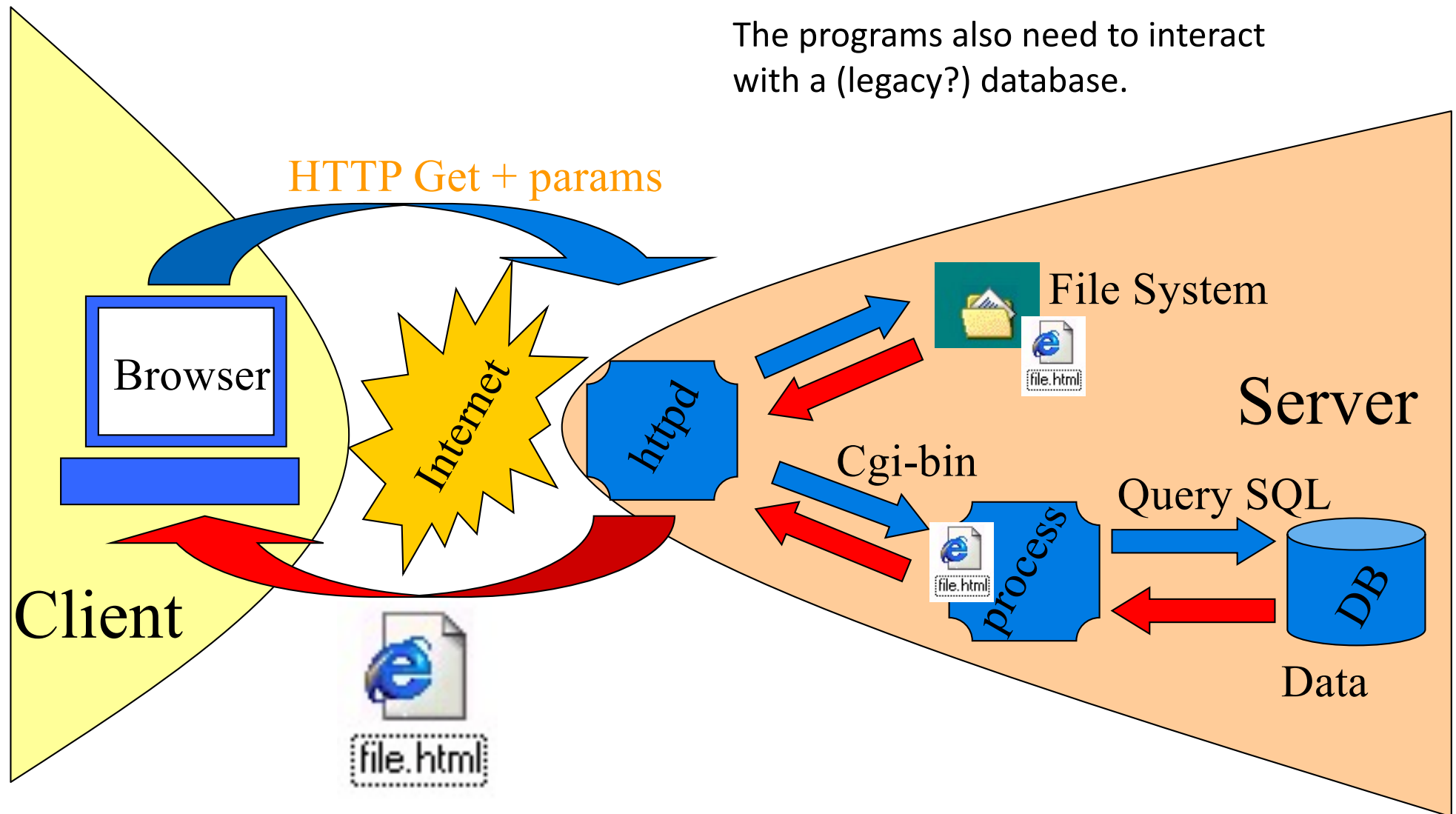
Initial idea: get (static) interlinked documents

The original web architecture: dynamic pages



Evolution 1: dynamically create (interlinked) documents

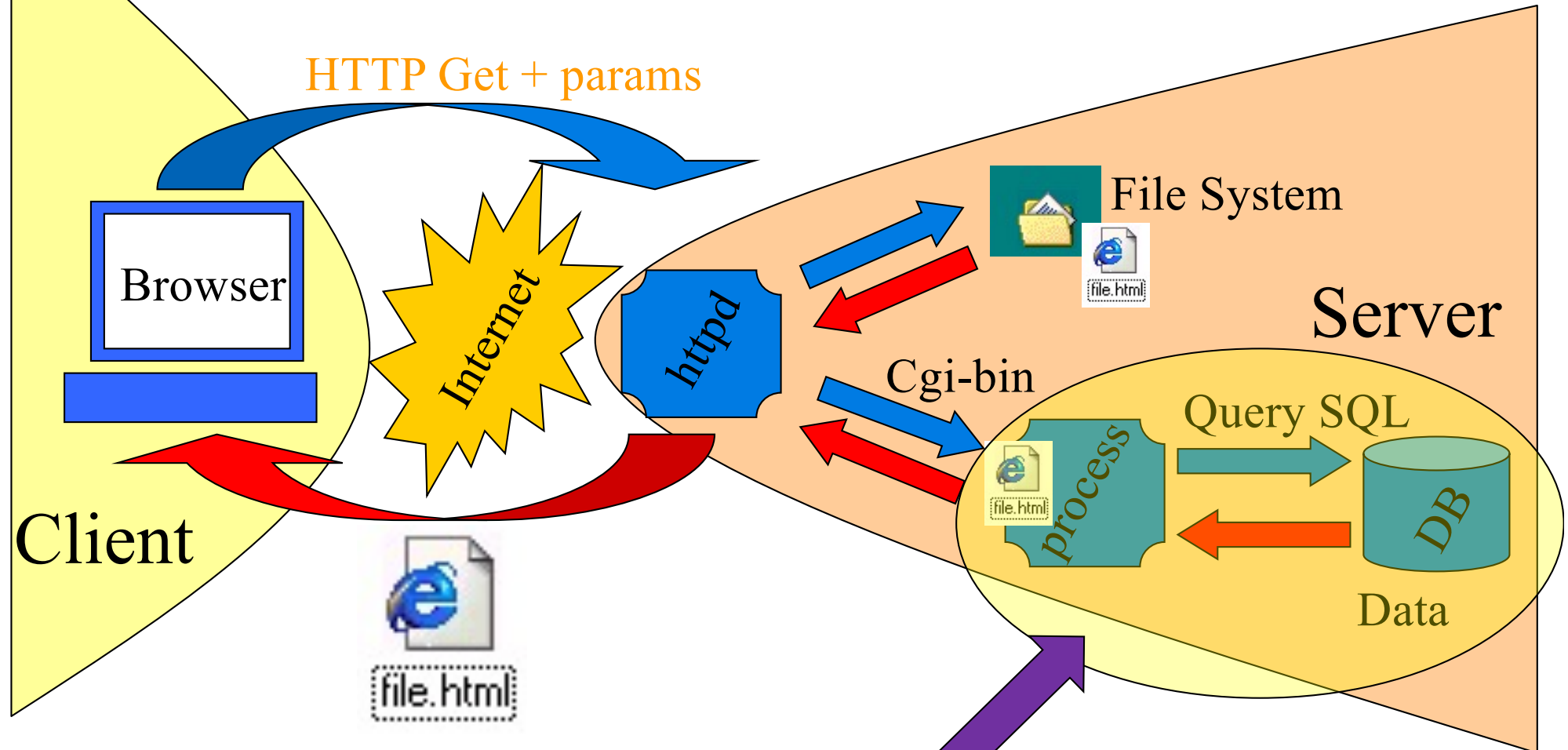
The original web architecture: dynamic pages with DB



Evolution 2: dynamically create (interlinked) documents interacting with a persistent data storage

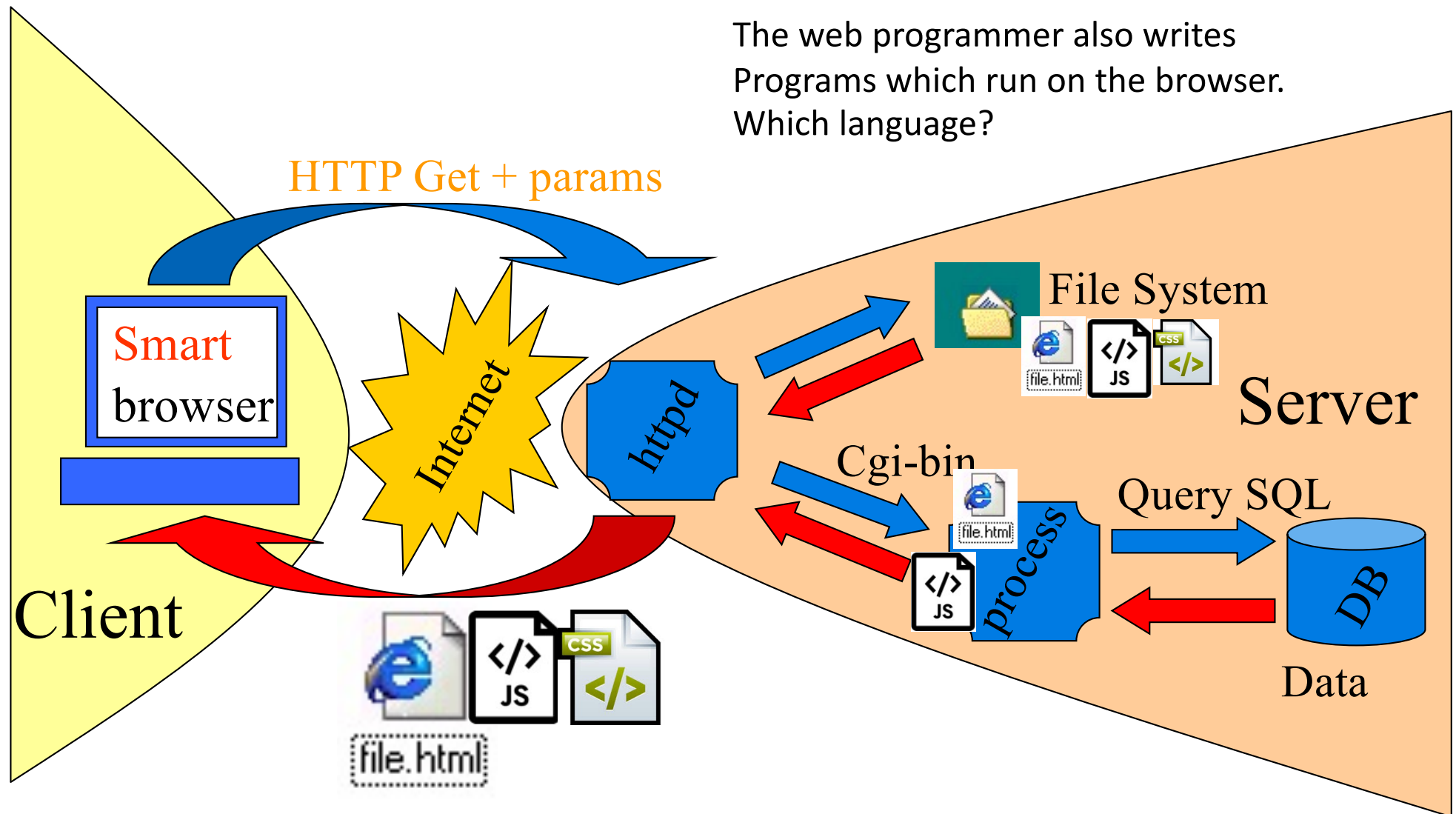
The original web architecture: dynamic pages with DB

The programs also need to interact with a (legacy?) database.



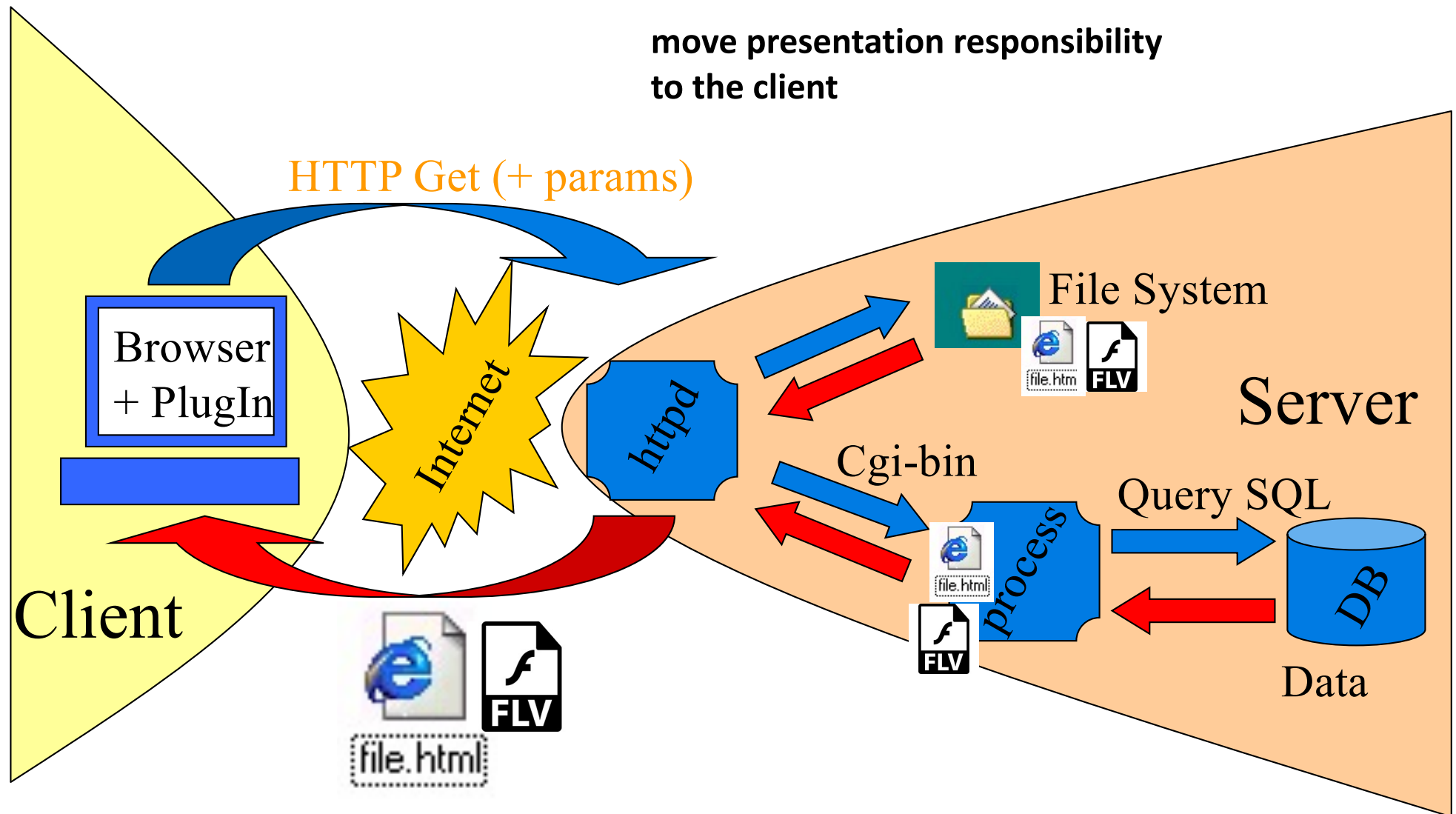
We need to focus on this area

The web architecture with smart browser



Evolution 3: execute code also on client! (How ?)

The web architecture: plug in

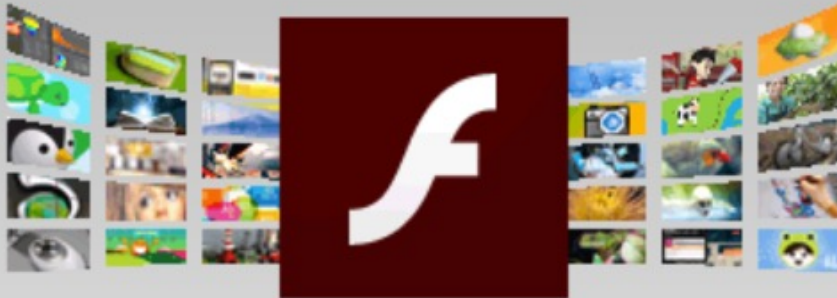


Evolution 4: augment browser with an ad-hoc engine to be able to execute a (proprietary) language

Plug in: Adobe Flash (former Macromedia Flash)

FLV-Video-Media-Content played by the Adobe Flash OCX Plugin.

Adobe Flash Player



About:

Adobe® Flash® Player is a lightweight browser plug-in and rich Internet application runtime that delivers consistent and engaging user experiences, stunning audio/video playback, and exciting gameplay.

Installed on more than 1.3 billion systems, Flash Player is the standard for delivering high-impact, rich Web content.

Plug in: Silverlight



SEARCH WITH BING



[Home](#) [About](#) [Features](#) [Partners](#) [Resources](#) [Forums](#) [Download](#)

Prepare for Silverlight 5 end of support after October 2021. [Learn more >](#)

Get Silverlight 5

Silverlight is a powerful development tool for creating engaging, interactive user experiences for Web and mobile applications. Silverlight is a free plug-in, powered by the .NET framework and compatible with multiple browsers, devices and operating systems, bringing a new level of interactivity wherever the Web works.

DOWNLOAD NOW



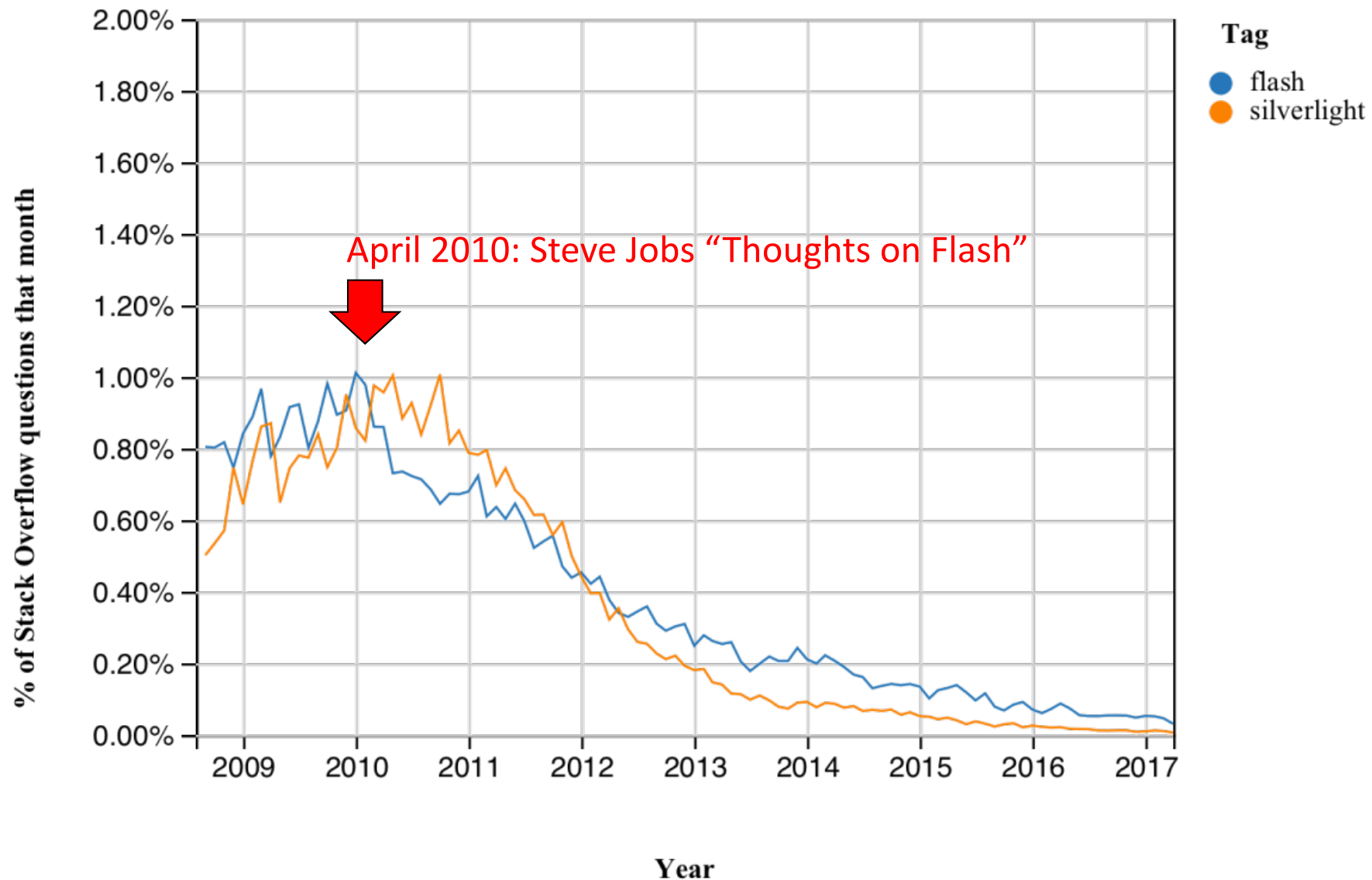
Silverlight 5
now available

[Download now >](#)

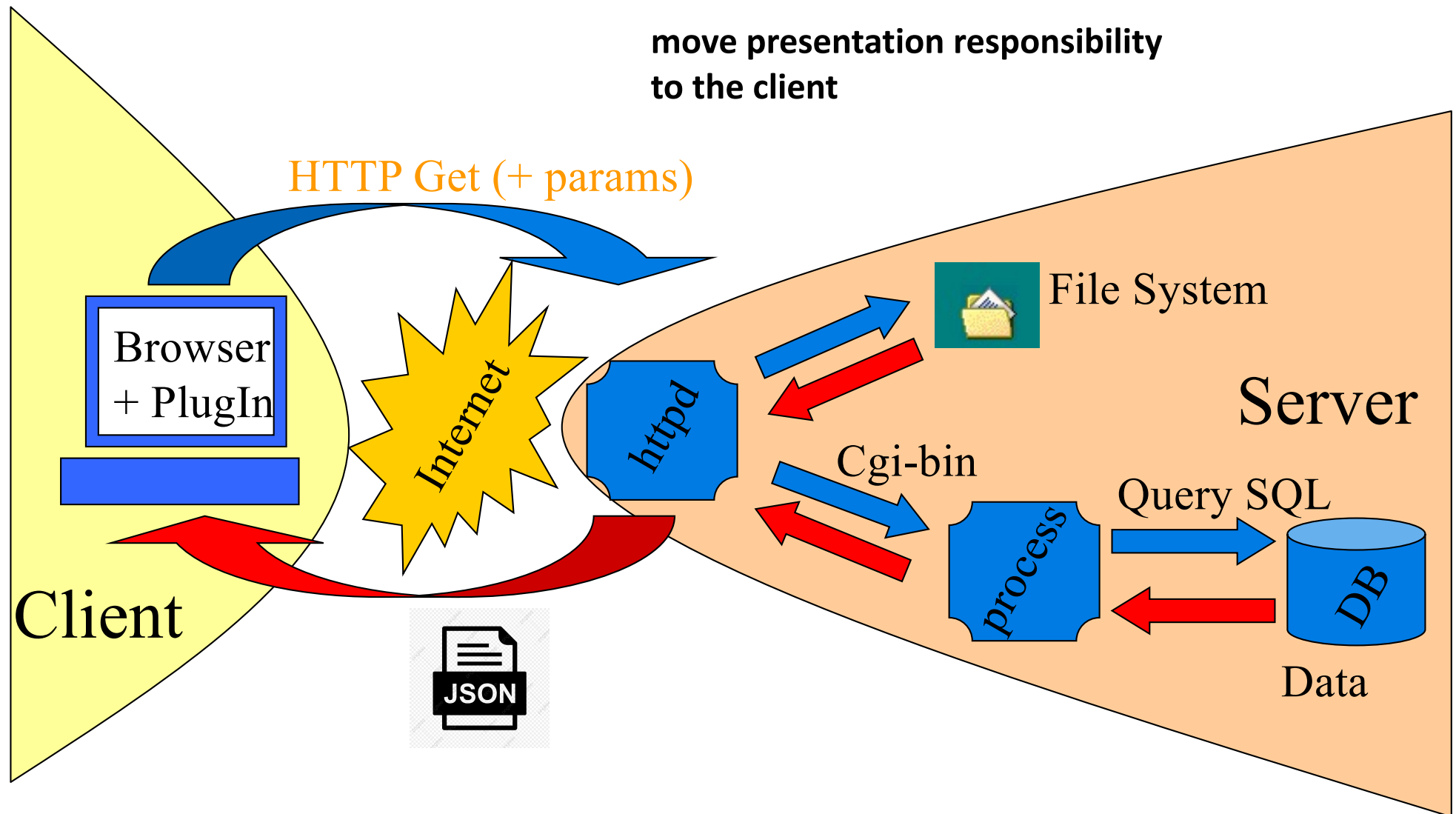


Get the Silverlight 5: A powerful tool for creating interactive web and mobile app

Flash and Silverlight decline



The web architecture: Ajax and SPA



Evolution 5: deliver data to single page applications

Q

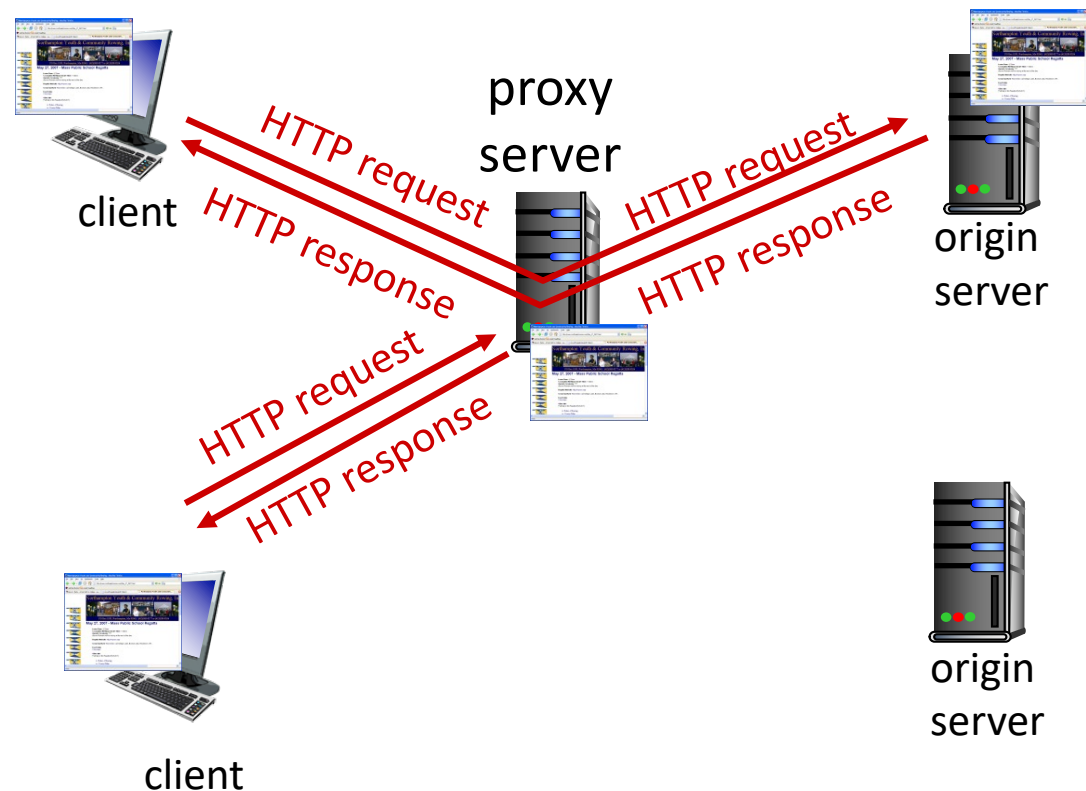
How can we optimize traffic ?

1: Proxies

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
 - object in cache: cache returns object
 - else cache requests object from origin server, then returns object to client



More about Web caching

- cache acts as both client and server
 - server for original requesting client
 - client to origin server
 - typically cache is installed by ISP (university, company, residential ISP)
- why Web caching?
- reduce response time for client request
 - reduce traffic on an institution's access link
 - Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)

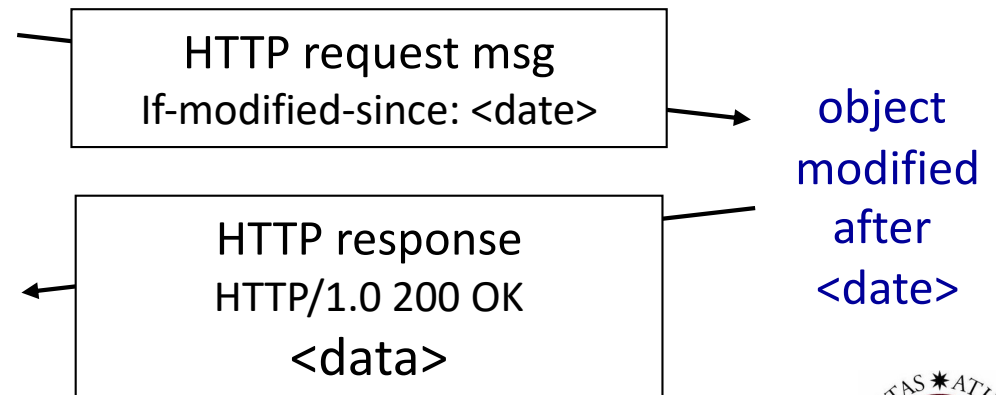
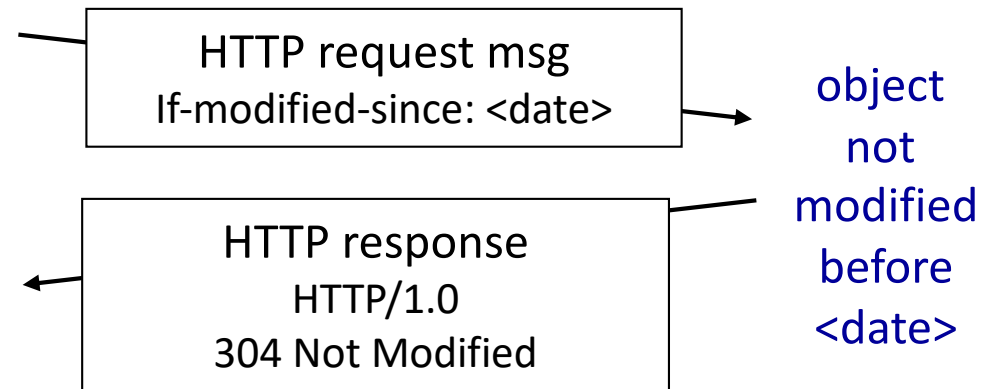
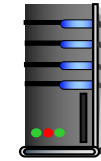
Conditional GET

- Goal: don't send object if cache has up-to-date cached version
 - no object transmission delay
 - lower link utilization
- cache: specify date of cached copy in HTTP request
 - If-modified-since: <date>
- server: response contains no object if cached copy is up-to-date:
 - HTTP/1.0 304 Not Modified

client



server



Q

How can we optimize traffic ?

2: Video streaming and CDNs

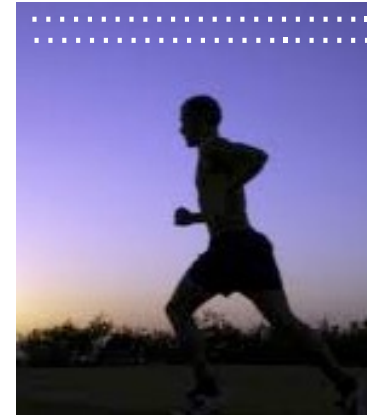
Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
 - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
 - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
 - single mega-video server won't work (why?)
- challenge: heterogeneity
 - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- solution: distributed, application-level infrastructure

Multimedia: video

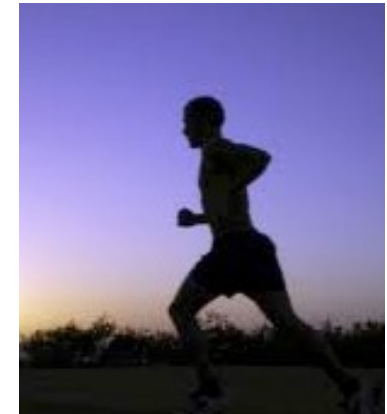
- video: sequence of images displayed at constant rate
 - e.g., 24 images/sec
- digital image: array of pixels
 - each pixel represented by bits
- coding: use redundancy within and between images to decrease # bits used to encode image
 - spatial (within image)
 - temporal (from one image to next)

spatial coding example: instead of sending N values of same color (all purple), send only two values: color value (purple) and number of repeated values (N)



frame i

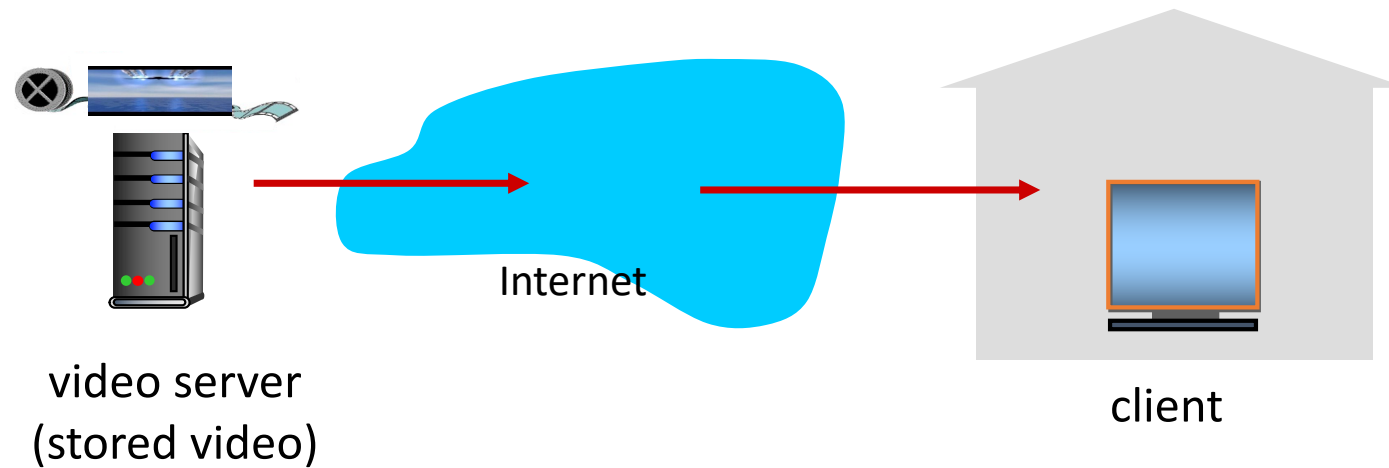
temporal coding example: instead of sending complete frame at i+1, send only differences from frame i



frame i+1

Streaming stored video

- Simple scenario:



Streaming multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

- server:
 - divides video file into multiple chunks
 - each chunk stored, encoded at different rates
 - manifest file: provides URLs for different chunks
- client:
 - periodically measures server-to-client bandwidth
 - consulting manifest, requests one chunk at a time
 - chooses maximum coding rate sustainable given current bandwidth
 - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

- “intelligence” at client: client determines
 - when to request chunk (so that buffer starvation, or overflow does not occur)
 - what encoding rate to request (higher quality when more bandwidth available)
 - where to request chunk (can request from URL server that is “close” to client or has high available bandwidth)

Content distribution networks

- challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- option 1: single, large “mega-server”
 - single point of failure
 - point of network congestion
 - long path to distant clients
 - multiple copies of video sent over outgoing link
- quite simply: this solution doesn't scale

Content distribution networks

- challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- option 2: store/serve multiple copies of videos at multiple geographically distributed sites (CDN)
 - enter deep: push CDN servers deep into many access networks
 - close to users
 - used by Akamai, 1700 locations
 - bring home: smaller number (10's) of larger clusters in POPs near (but not within) access networks
 - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
 - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
 - directed to nearby copy, retrieves content
 - may choose different copy if network path congested

