UNIVERSITÉ CÔTE D'AZUR

Inria INVENTEURS DU MONDE NUMÉRIQUE

02. Web browsers and Web applications

Nataliia Bielova

@nataliabielova

September 17th, 2018

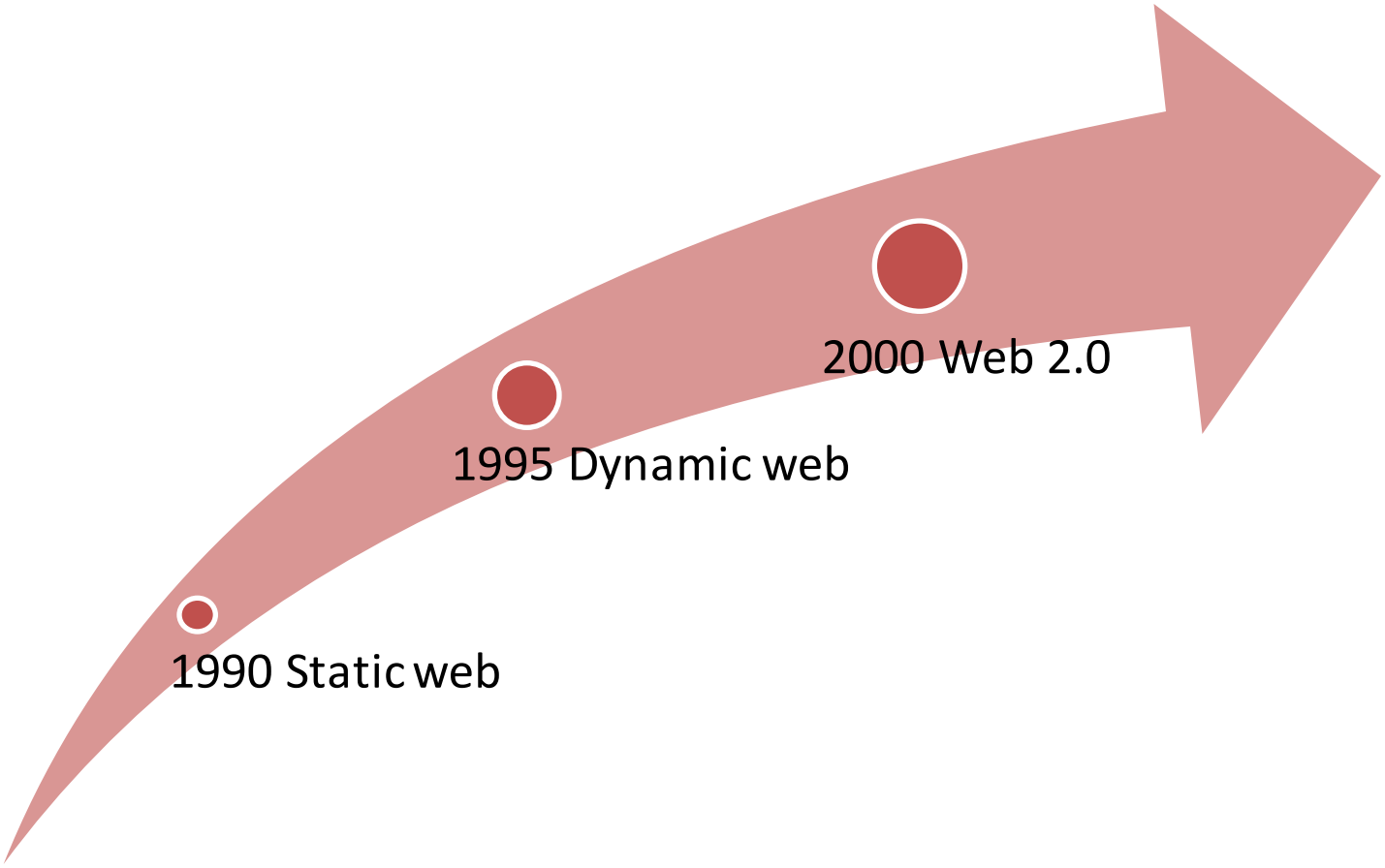
Web Privacy course

University of Trento

Today's class

- What is Web browser and how does it work?
- Web application architecture
- Cookies and JavaScript
- Basic browser security mechanisms
  - Same Origin Policy

Web evolution



See more http://www.evolutionoftheweb.com

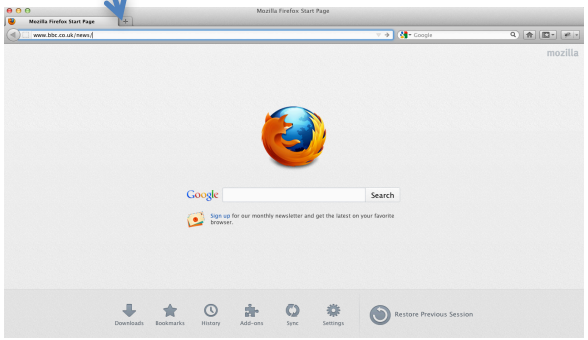
Web Applications are everywhere

- Users generate data while using applications
  - Identity
  - Preferences, tastes
  - Financial situation
  - Social life

HTTP: HyperText Transfer Protocol

www.bbc.co.uk/news/

Web browser



URL path: bbc.co.uk/news
Parameters
Method: GET
…

HTTP request

HTTP response

Status: 200 OK
Content: HTML page
…

Web server

- Rendering pages
- Executing scripts/plugins (JavaScript)
- Launching new HTTP requests

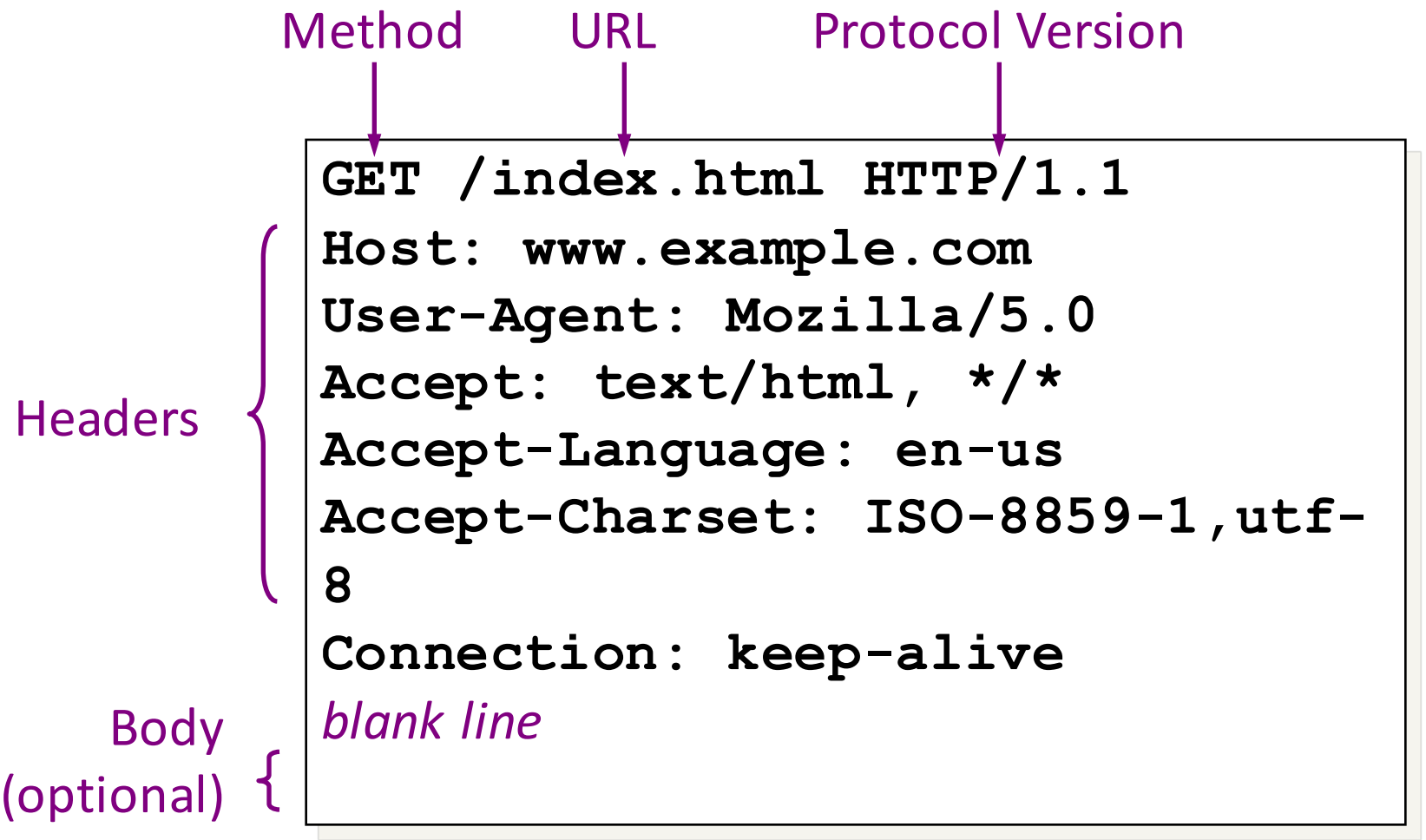
- Mapping requests to apps
- Contacting DBs
- Constructing responses

HTTP: HyperText Transfer Protocol

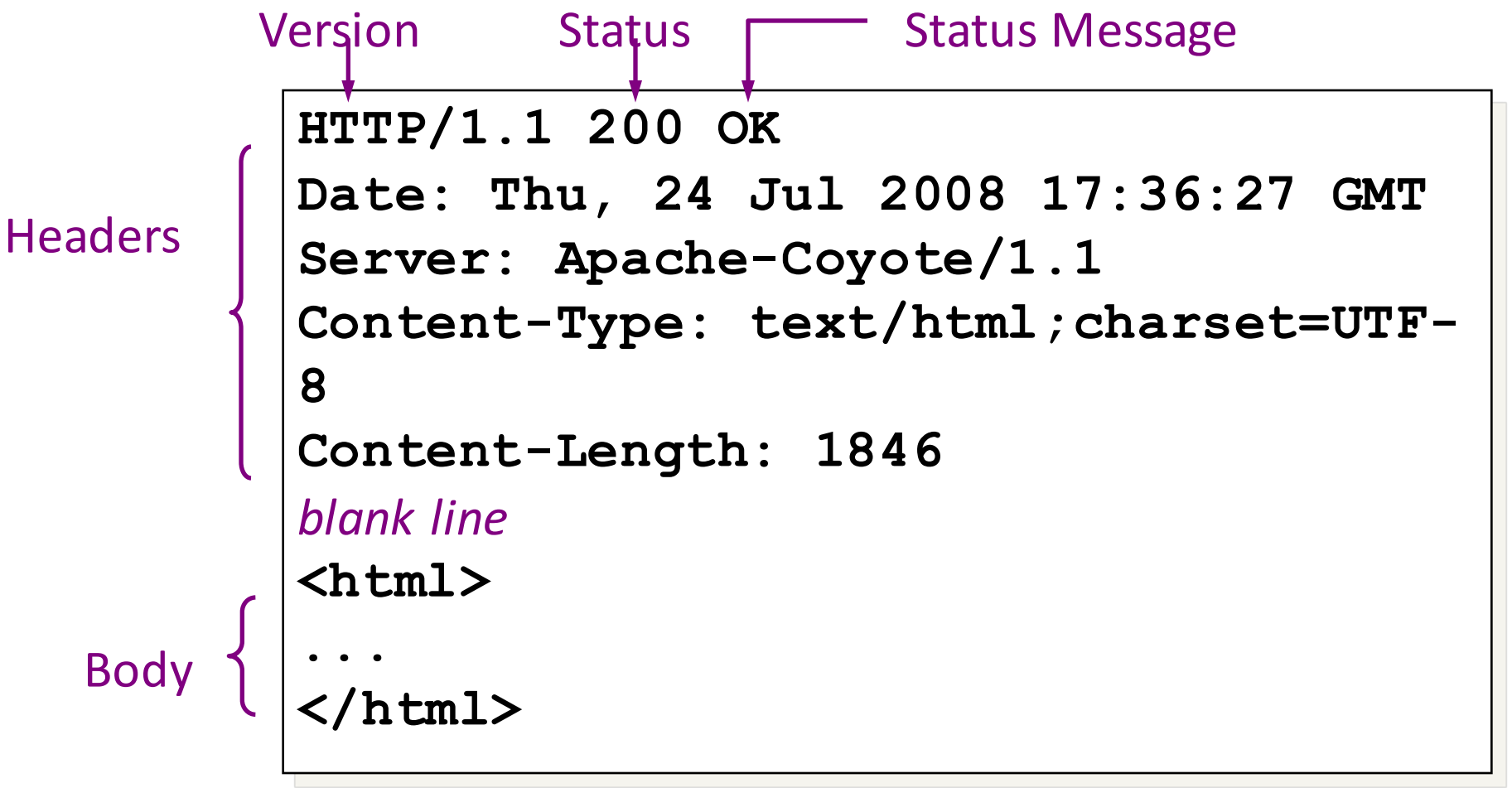
- HTTP important characteristic: no State request/response - each request is independent

- HTTP header: header section of requests and responses, parameters of the HTTP transaction

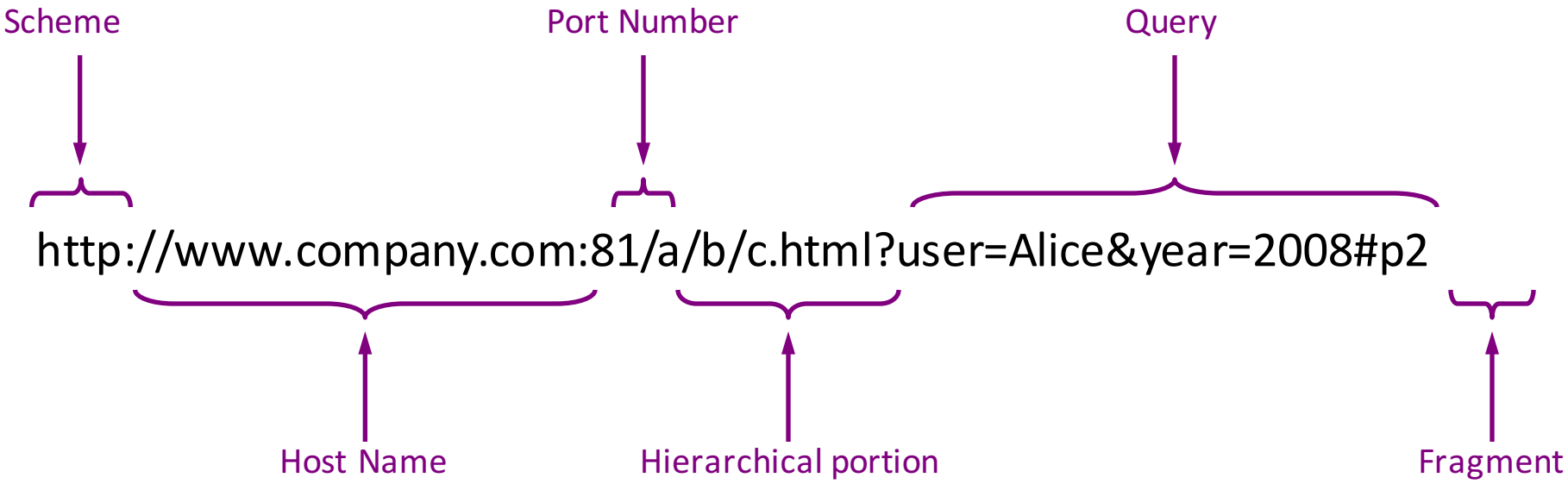
HTTP Request



HTTP Response



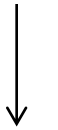
Uniform Resource Locators (URLs)



How to keep state in Web applications?

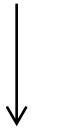
HTTP: Session in URL Example

http://www.buy.com



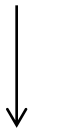
see catalog

http://www.buy.com/shopping.cfm?pID=269



select item

http://www.buy.com/shopping.cfm?pID=269&item=40002



buy item

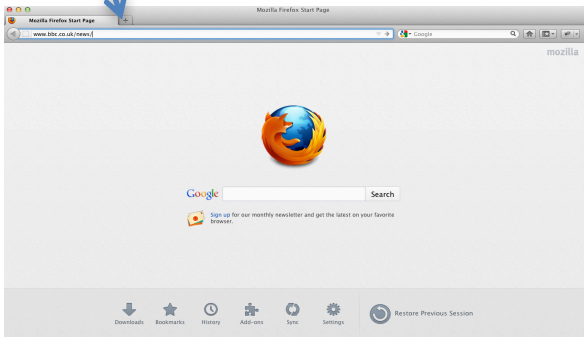
http://www.buy.com/checkout.cfm?pID=269&item=40002

Since HTTP is stateless all session information is saved in the URL!

Slide of Tamara Rezk

HTTP: Session in cookies

Web browser



URL path: bbc.co.uk/news
Parameters
Method: GET
…

HTTP request

HTTP response

Status: 200 OK
Content: HTML page
Set-cookies: session-id=**20827872011** & …

…

Web server

HTTP: Session in cookies

Web browser

URL path: bbc.co.uk/news
Parameters
Method: GET
…

HTTP request

HTTP response

Status: 200 OK
Content: HTML page
Set-cookies: session-id=**20827872011** & …

…

Web server

Cookie Database

bbc.co.uk/news:
session-id=20827872011

HTTP: Session in cookies

Web browser

URL path: bbc.co.uk/news…
Method: GET
Cookies: session-id=**2082787201l** & …
…

HTTP request

Web server

Cookie Database

bbc.co.uk/news:
session-id=2082787201l

Cookies

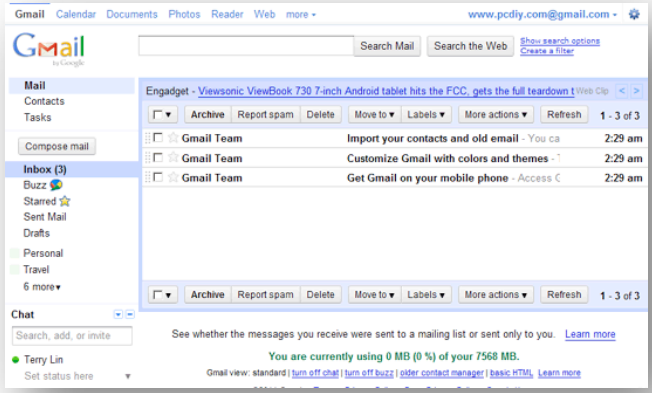
What is a cookie?

- A small piece of data, sent by the HTTP server in an HTTP response, stored by the client, and sent back by the client to the server in all further responses.

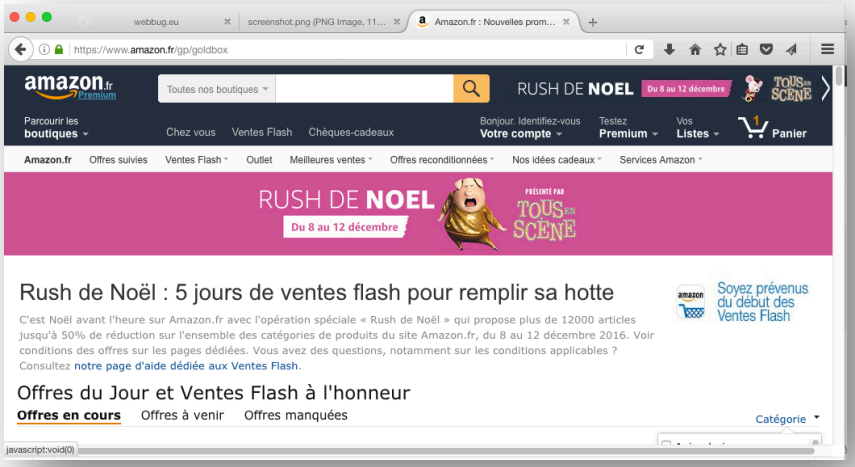
- A cookie may also be set and read directly in the client by some JavaScript code.

What's the original use of cookies?

- Keep the session through different windows/tabs



- Shopping basket



Profitable uses of cookies

- **Personalization:** remember the information about the user who has visited a website in order to show relevant content in the future

- **Tracking:** following the user during a session or across multiple visits.

Structure of a Cookie

- A name,
- A value,
- An expiry date,
- A domain and a path the cookie is settled for,
- Whether we need a secure connection (HTTPS) for the cookie,
- Whether the cookie can be accessed through other means than HTTP (i.e. JavaScript).

From slides of Vincent Simonet

Types of cookies

- **Session cookie:** cookie without expiry date. Disappears when the browser is closed.
- **Persistent cookie:** cookie with an expiry date. Remains until this date, even if the browser is closed.
- **Secure cookie:** sent only in HTTPS requests.
- **HttpOnly cookie:** non-accessible from JavaScript.
- **Third-party cookie:** a cookie from another domain than the domain that is shown in the browser's address bar.

Example of Cookie in the HTTP Protocol

- **1st HTTP request (client):**

```
GET /index.html HTTP/1.1
```

- **1st HTTP response (server):**

```
HTTP/1.0 200 OK
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed,
09 Jun 2021 10:18:14 GMT
```

- **2nd HTTP request (client):**

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
```

Example of cookies with domain and path

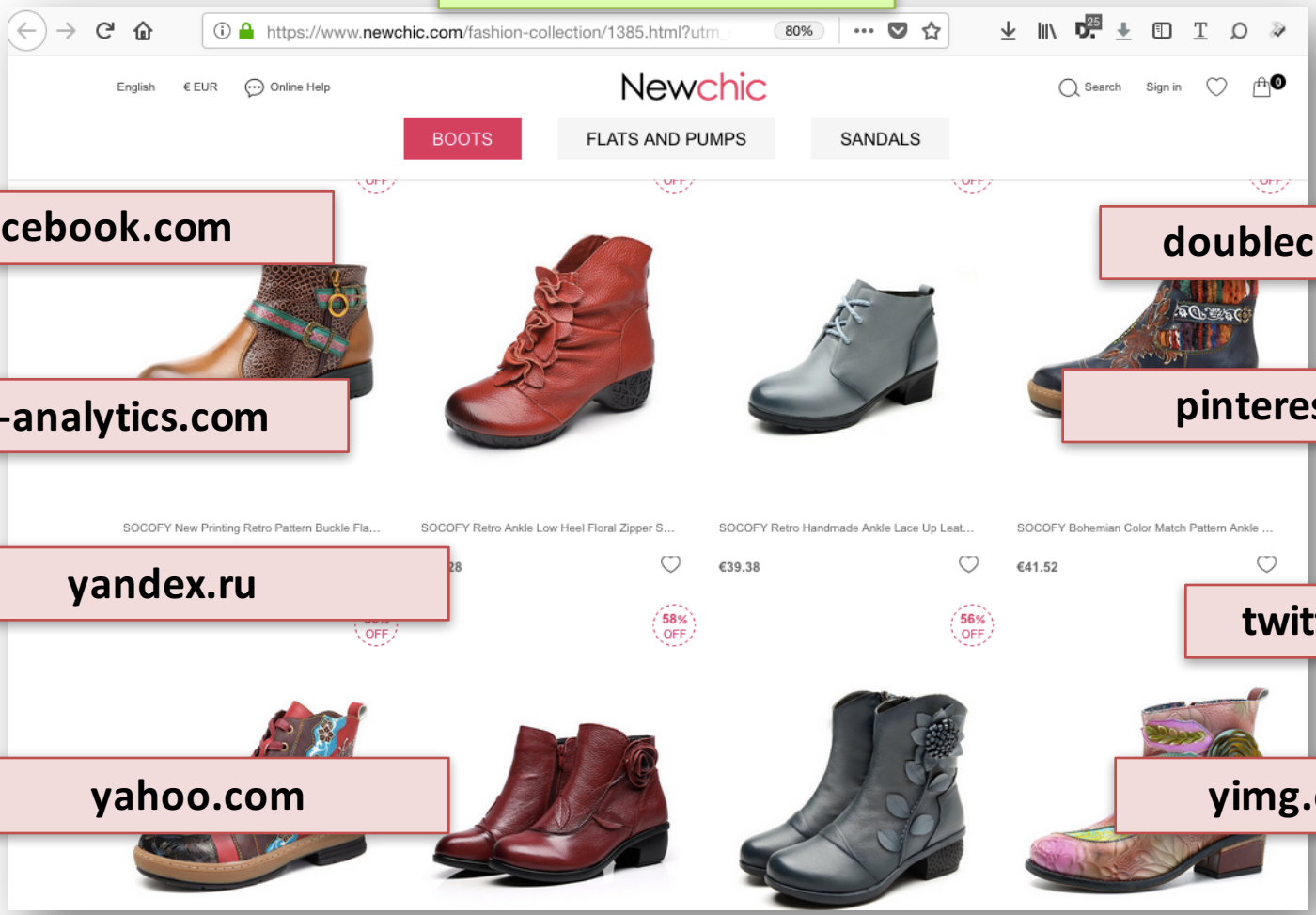
- **Set-Cookie:** LSID=DQAAAK...Eaem_vYg; Domain=docs.foo.com; Path=/accounts; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly

- **Set-Cookie:** HSID=AYQEVn....DKrdst; Domain=. foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; HttpOnly

- If not specified, they default to the domain and path of the object that was requested.
- Cookies can only be set on the top domain and its sub domains

Basic browser security: Same Origin Policy

newchic.com



facebook.com

doubleclick.net

google-analytics.com

pinterest.com

yandex.ru

twitter.com

yahoo.com

yimg.com

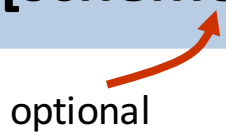
Same origin policy: high level

Same Origin Policy (SOP) for DOM:

- Origin A can access origin B's DOM if match on **(scheme, domain, port)**

Same Original Policy (SOP) for cookies:

- Generally speaking, based on: **([scheme], domain, *path*)**



scheme://domain:port/path?params

URL1 and URL2 are same-origin?

URL1: http://www.example.com/dir/page.html

Compared URL2	Outcome	Reason
http://www.example.com/dir/page.html	✔	Same protocol and host
http://www.example.com/dir2/other.html	✔	Same protocol and host
http://www.example.com:81/dir/page.html	✘	Same protocol and host but different port
https://www.example.com/dir/page.html	✘	Different protocol
http://example.com/dir/page.html	✘	Different host
http://v2.www.example.com/dir/page.html	✘	Different host

In what origin each script is running?

a.com

a.com

```
<script src=b.com/script.js>
```

JavaScript 1

```
<iframe src=b.com/main.html>
```

Html page +

```
<script src=c.com/script.js>
```

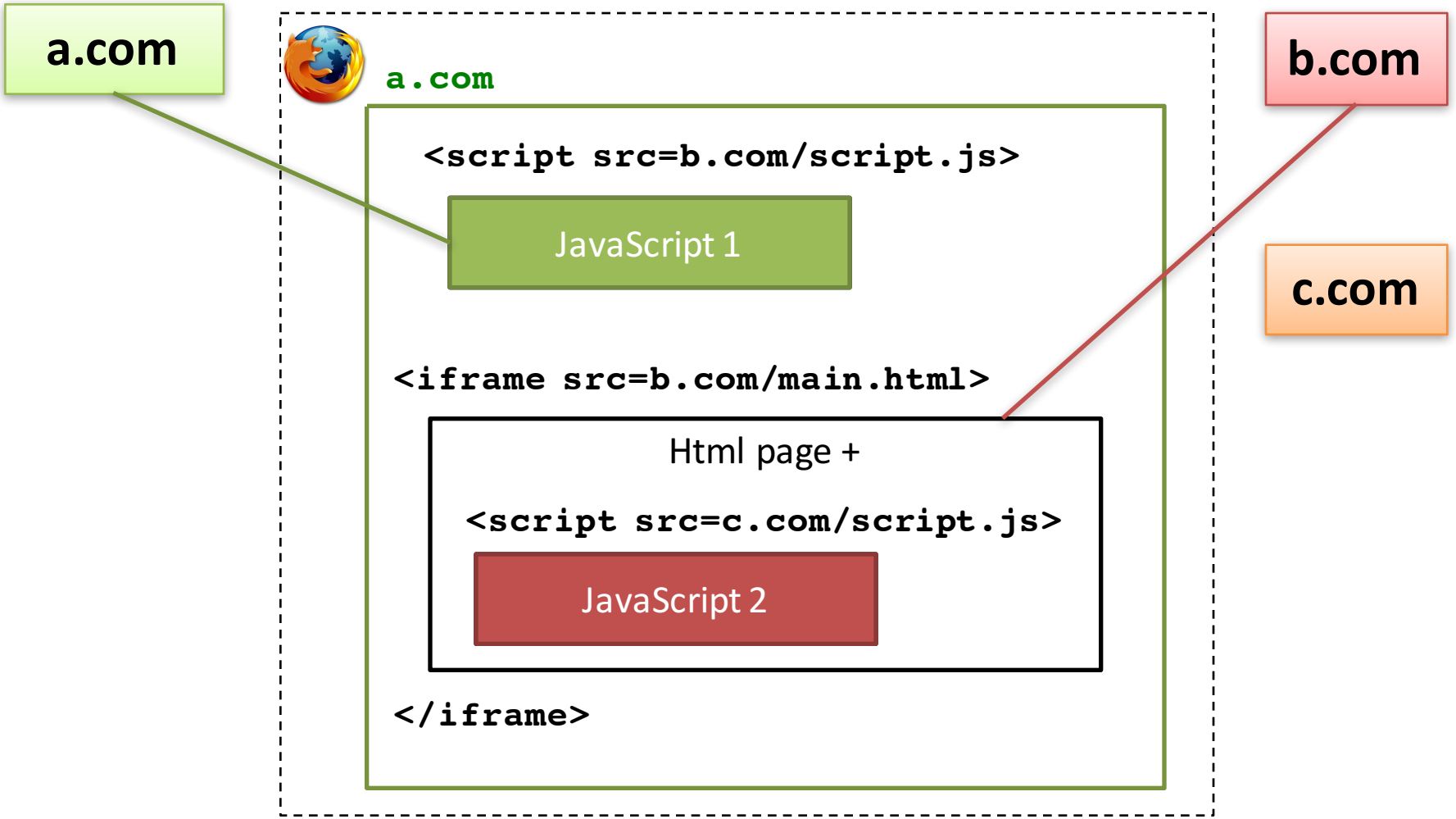
JavaScript 2

```
</iframe>
```

b.com

c.com

In what origin each script is running?



Two ways to access cookies

- Via HTTP header
  - Set/get cookies associated with **(domain, path)** of the requested object
- Via JavaScript: `document.cookie` API
  - Access with respect to SOP : **(scheme, domain, port)**
    - the change of an effective origin by `document.domain` DOM API doesn't affect the cookie access

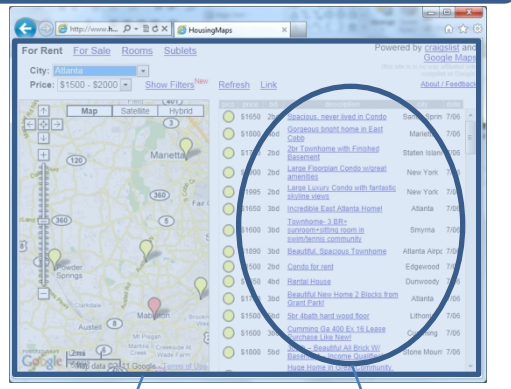
Same Origin Policy

- Scripts running on pages from the same origin can access each other's DOM without restriction.
- Scripts running on pages from different origins cannot access each other's DOM.

- The Same Origin Policy does not apply to `<img>`, `<script>` or `<object>` tags.

Same Origin Policy for DOM

Using `<script>` tag

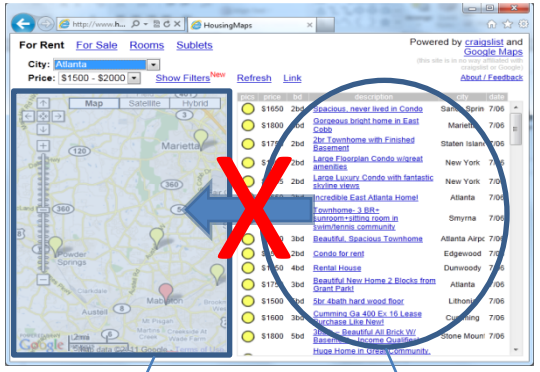


Google Maps Gadget

Integrator's Housing Data

- Full sharing (JS Env.)
- Running as integrator
- Gadget trusted

Using `<iframe>` frame



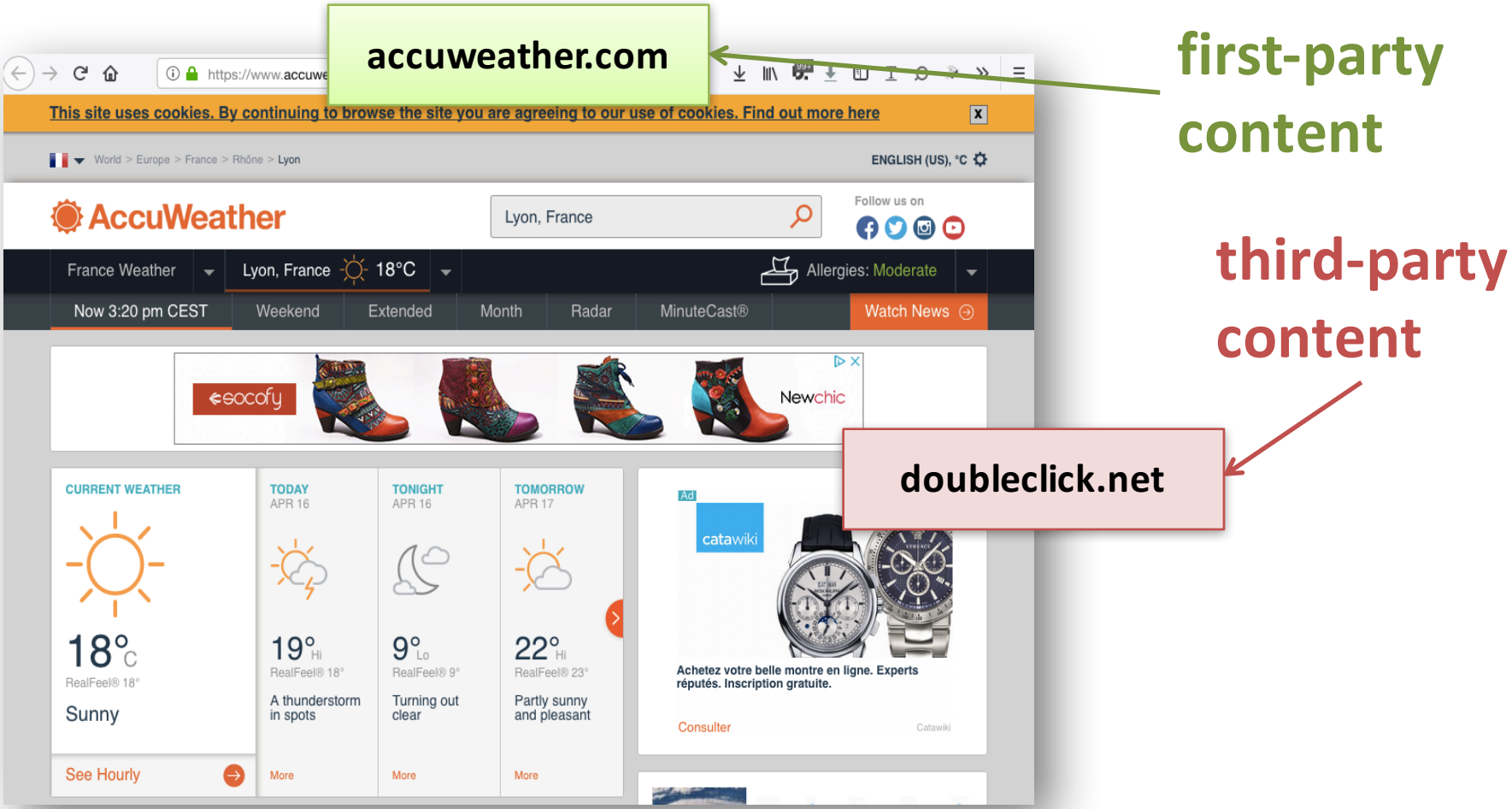
Google Maps Gadget

Integrator's Housing Data

- Full isolation (by SOP)
- Running as gadget
- Limited sharing
  - Frame identifier
  - PostMessage

Cookies: first- and third-party

Content: first- and third-party



Cookies: first- and third-party

first-party cookie

Cookie Database

accuweather.com: id=123

doubleclick.com: id=456

third-party cookie

Third-party cookies

- The same origin policy does not apply to <img>, <script> or <object> tags. This allows a web page to triggers a GET request with cookies to a third-party site.

- Purpose of third-party cookies? **Tracking!**

Cookies: first- & third-party