AJAX (Asynchronous Javascript And XML)

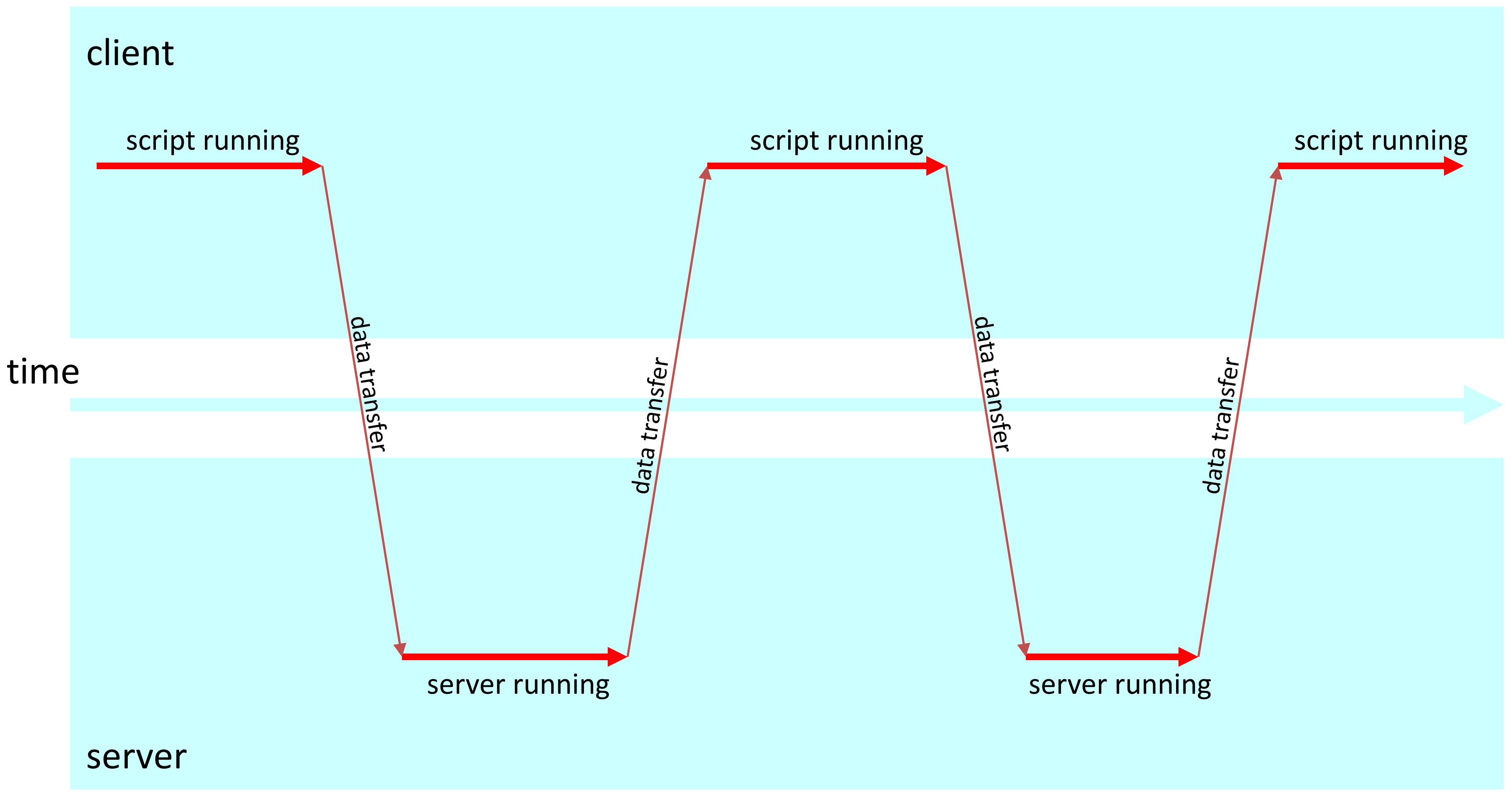
What is AJAX?

- A recent (2005) technique for *partial updating* of pages *in the background*
  - Speed-up in interactions (whole page loading avoided)
  - Gradual downloading in the background avoids blocking the user when waiting for responses from the server
- Useful when:
  - only a small part of the page (or some data) needs to be updated without total reloading
  - local operations on the browser need to be allowed while waiting for the server
- Based on Javascript/DOM 1, HTML, CSS, XML
  - can work even without XML

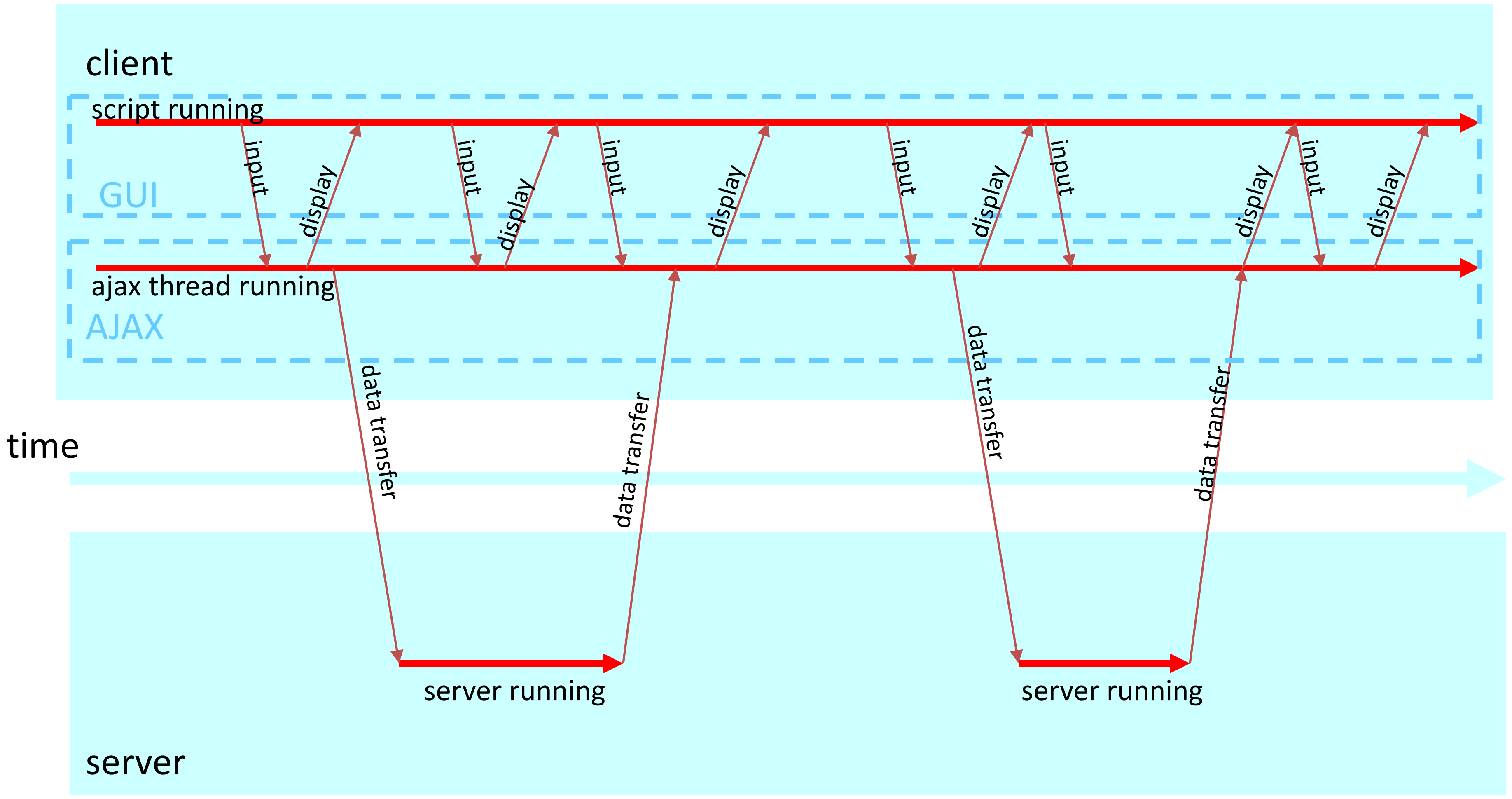
AJAX Flow

- A (Javascript) script running on the browser issues a request to the web server
  - using the standard **`XMLHttpRequest`** object
- A separate thread waits for and downloads the response
- The script intercepts the response and updates the page being displayed
- Main advantages:
  - The response is normally much smaller than the whole page
  - The user is not blocked when waiting for the response

Normal HTTP Interactions



AJAX Interactions



Some Applications that use AJAX

- Google Maps
- Gmail
- Youtube
- Facebook
- Google suggest
  - Fist application (2005)
  - While the user is writing in the Google search text field
  - each character is sent to the server in the background
  - a list of "suggestions“ is returned by the server and displayed by the script

The `XMLHttpRequest` Object

- Standard object available in most browsers
  - replaced by **`ActiveXObject`** in IE 6 and previous versions
- It is the representation of a request to the server
- The response is associated to the same object
- How to create the object: browser-dependent
  - Most browsers: **`req = new XMLHttpRequest()`**
  - IE5: **`req = new ActiveXObject(“Microsoft.XMLHTTP”)`**
  - IE5: **`req= new ActiveXObject(“Microsoft.XMLHTTP”)`**
  - IE6+: **`req= new ActiveXObject(“Msxml2.XMLHTTP”)`**

A Browser-Independent Creation Function

```
function ajaxRequest() {
   try { // Non IE Browser?
         var request = new XMLHttpRequest()
   } catch(e1){ // No
         try { // IE 6+?
               request = new ActiveXObject("Msxml2.XMLHTTP")
         } catch(e2){ // No
              try { // IE 5?
                    request = new ActiveXObject("Microsoft.XMLHTTP")
              } catch(e3){ // No AJAX Support
                   request = false
              }
         }
    }
    return request
}
```

How to use AJAX

- Write a Javascript script that
  - prepares a request (GET or POST)
  - sends the prepared request
  - intercepts the response and
    - checks for errors
    - updates the page as necessary
- Possible choices
  - use of HTTP headers (e.g. cache control)
  - synchronous vs asynchronous request

    synchronous:
    - the script blocks until a response is received

    asynchronous:
    - the script can get on while the request is being executed
    - a handler function is associated with the request object

Request Properties (1)

- **`readystate`**
  - an integer that specified the status of the request

0	uninitialized
1	initialized (uploading request)
2	loaded (request received by server)
3	interactive (request being executed on server)
4	completed (response received and available)

- **`onreadystatechange`**
  - event handling function called whenever **`readystate`** changes (typically set just after creation)

Request Properties (2)

- **`status`**
  - the HTTP status code returned by the server (e.g. 200, 404, …)
  - initialized when a response has been received
- **`statusText`**
  - the HTTP status text returned by the server (e.g. OK, PAGE NOT FOUND,…)
- **`responseText`**
  - the HTTP response returned by the server in text (or HTML) format
- **`responseXML`**
  - the HTTP response returned by the server in XML format

Preparing a Request

- A request can be prepared by calling the method:

  `open(`*`<method>, <url>, <async>)`*

  - *<method>***:** GET or POST
  - *<url>* **:** the request target URL
  - *<async>* true if request is asynchronous *(*boolean)

- Example:

```
AjaxReq=new XMLHttpRequest();
AjaxReq.open("GET", "ex1.php", true);
```

same path as current page

Sending a Request

- A request can be sent by calling the method:
- **send(**<*args*>**)**
  - <*args*> list of URL-encoded arguments name1=val1&name2=val2&…
- Example:

```
AjaxReq=new XMLHttpRequest();
AjaxReq.open("POST","ex1.php", true);
AjaxReq.setRequestHeader("Content-type",
  "application/x-www-form-urlencoded");
AjaxReq.send("fname=Mario&lname=Rossi");
```

Example

- Change something in the current page
- HTML:

```
<html>
<body>
  <div id=tochange>
    <h1 text-align:center>
      Text to be changed
    </h1> <br>
    <button type=button onclick=startAjax()>Change
    </button>
  </div>
  …
</body>
</html>
```

Javascript

Example (II)

- Javascript:

```
<script type="text/javascript">
var req;

function ajaxRequest() {…}

// Handler definition
function f(){
  if (req.readyState==4 &&
     (req.status== 0 || req.status==200)) {
         document.getElementById("tochange").
         innerHTML=req.responseText;
  };
}
```

success status code in case of local file

Example (III)

- Javascript (contd):

```
function startAjax() {
   req = ajaxRequest();
   req.onreadystatechange = f;
   req.open("GET","ajax.txt", true);
   req.send();
}
</script>
```

- File ajax.txt

```
<img src=“Caravaggio.jpg" height=500 width=500>
```

ajax1.html

Remarks

- The request can be issued to a dynamic page (server-side script)
- The URI of the AJAX request must have the same prefix of the original page
- This limitation can be bypassed:
  - Write a server-side script
  - Send the request to the script
  - The server-side script sends the request and forwards the response back to the client

Example

- Load a web page into a div using the server as a sort of proxy

`urlpost.html`

`urlget.html`

Other Example: Hints

- Make form with text field
- Give hint while the user is typing in the field

`hints.html`

Concurrent Requests

- The script is constantly running

  => several concurrent requests can be issued

  => Different **`XMLHttpRequest`** objects are necessary

- With concurrent requests,
  - no guarantee about order of responses received
  - the browser may limit the maximum number of concurrent requests that can be created

Example

- Using a parameterized function for creating new AJAX requests

ajax2.html

Other Example:
Robin’s Nest Signup with AJAX

AJAX and JQuery

- Some JQuery functions simplify the use of AJAX
- Example: load a file (URL) in the selected element

`load(`<*url*>**`[,`** <*args*>**`][,`** <*callback*>**`])`**

– <*url*>	the URL to be loaded
– <*args*>	same meaning as the **send** argument (POST). If absent, GET is used
– <*callback*>	function to be called when operation terminated

Change Current Page (with JQuery)

```
<html>
<head>
<script type="text/javascript“ src="jquery.js"></script>
<body>
<script type="text/javascript">
$("document").ready(function(){
   $("button").click(function(){
       $("#tochange").load("ajax.txt");})
;})
</script>
<div id=tochange>
<h1> Text to be changed </h1> <br>
<button type=button> Change </button>
</div>
</body</html>
```

ajax3.html

Hints (with JQuery)

```
<html><head>
<script type=text/javascript src=jquery.js></script>
<script type="text/javascript">
$("document").ready(function(){
   $("#txt1").keyup(function(){
       var str=document.getElementById("txt1").value;
       str="gethint.php?q="+str;
       $("#txtHint").load(str);});
});
</script>
</head>
<body>
<h3>Start entering name:</h3>
<form action=""> Name: <input type="text" id="txt1" >
</form> <p>Hints: <span id="txtHint"></span></p>
</body></html>
```