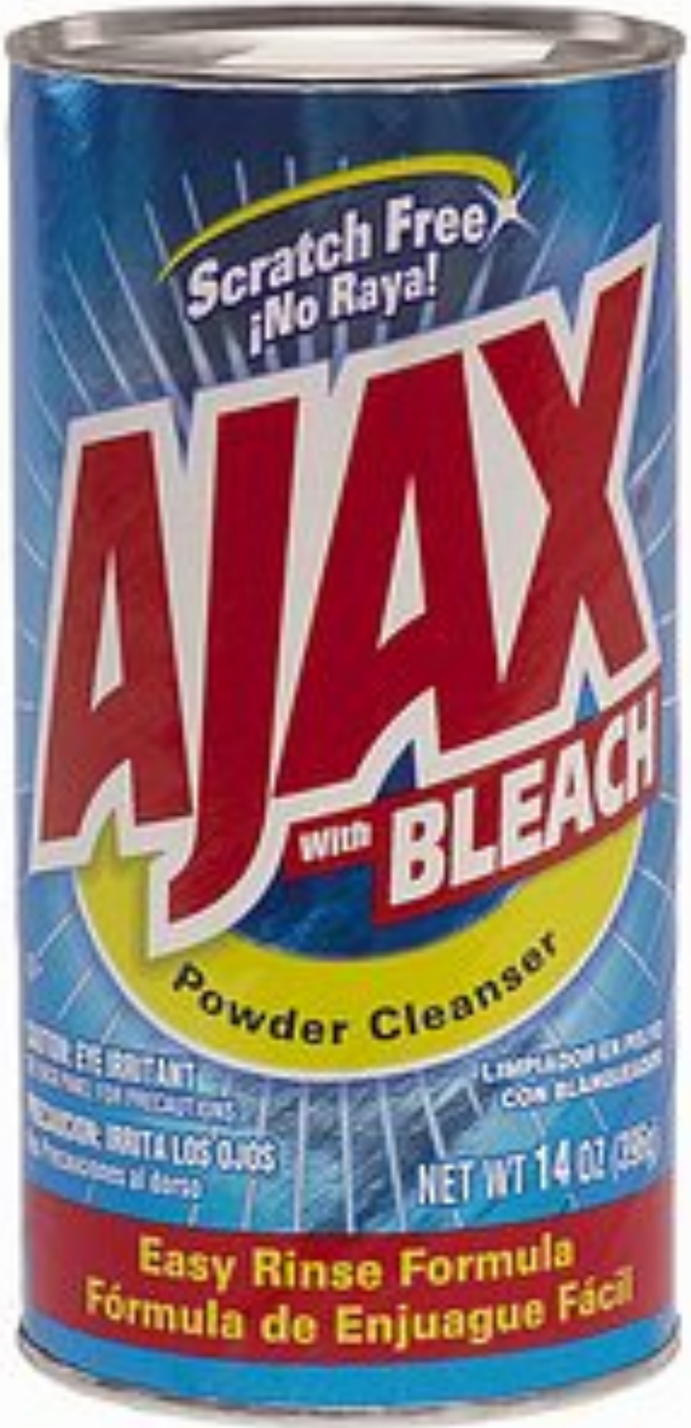
AJAX

Asynchronous
Javascript and XML



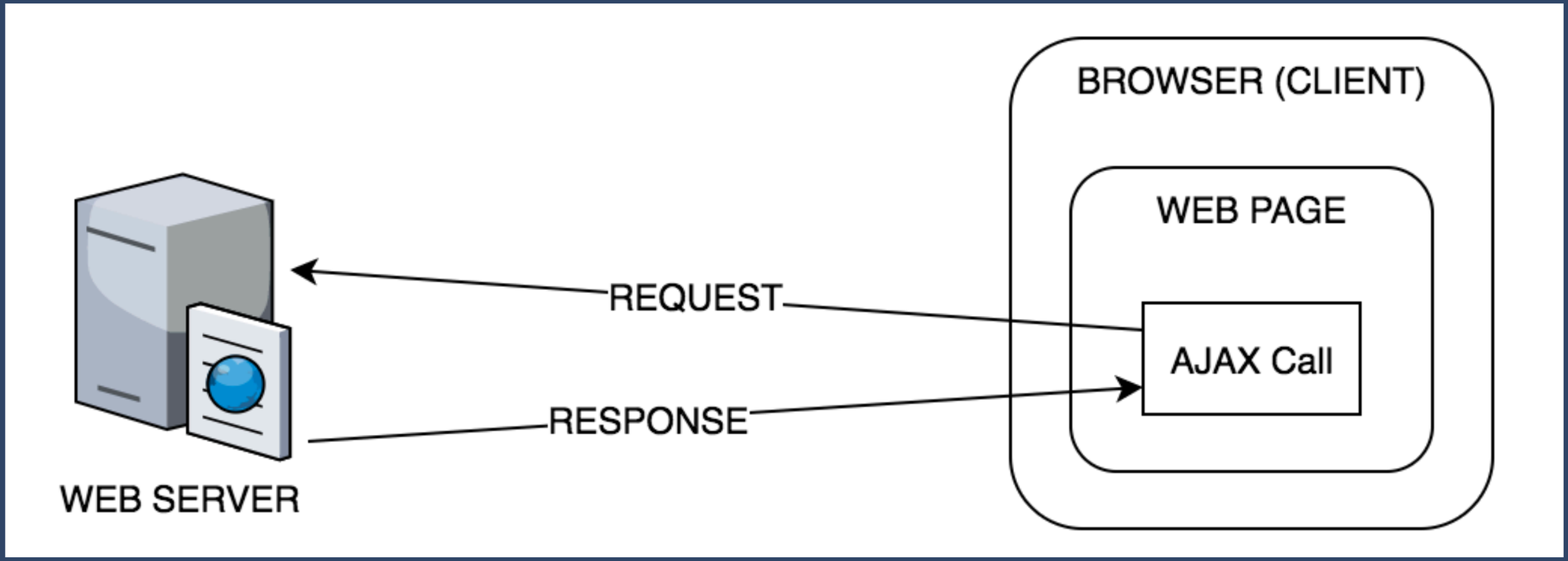
Objectives

1. Explain what AJAX is.
2. Explain why AJAX primarily transfers JSON data now.
3. Use AJAX to retrieve data from a server.
4. Handle CORS issues caused by Ajax requests.
5. Handle race conditions caused by Ajax requests.

1. Explain what AJAX is.

What is AJAX?

A set of techniques used to send data to and retrieve data from a server asynchronously



Technologies used for AJAX technique

- HTML (structure) and CSS (presentation)
- The DOM for dynamic changes to HTML/CSS
- The Javascript XMLHttpRequest object
- JSON or XML as a data format

Why use Ajax?

To exchange data with a server without blocking the other functionality of our app: asynchronous requests

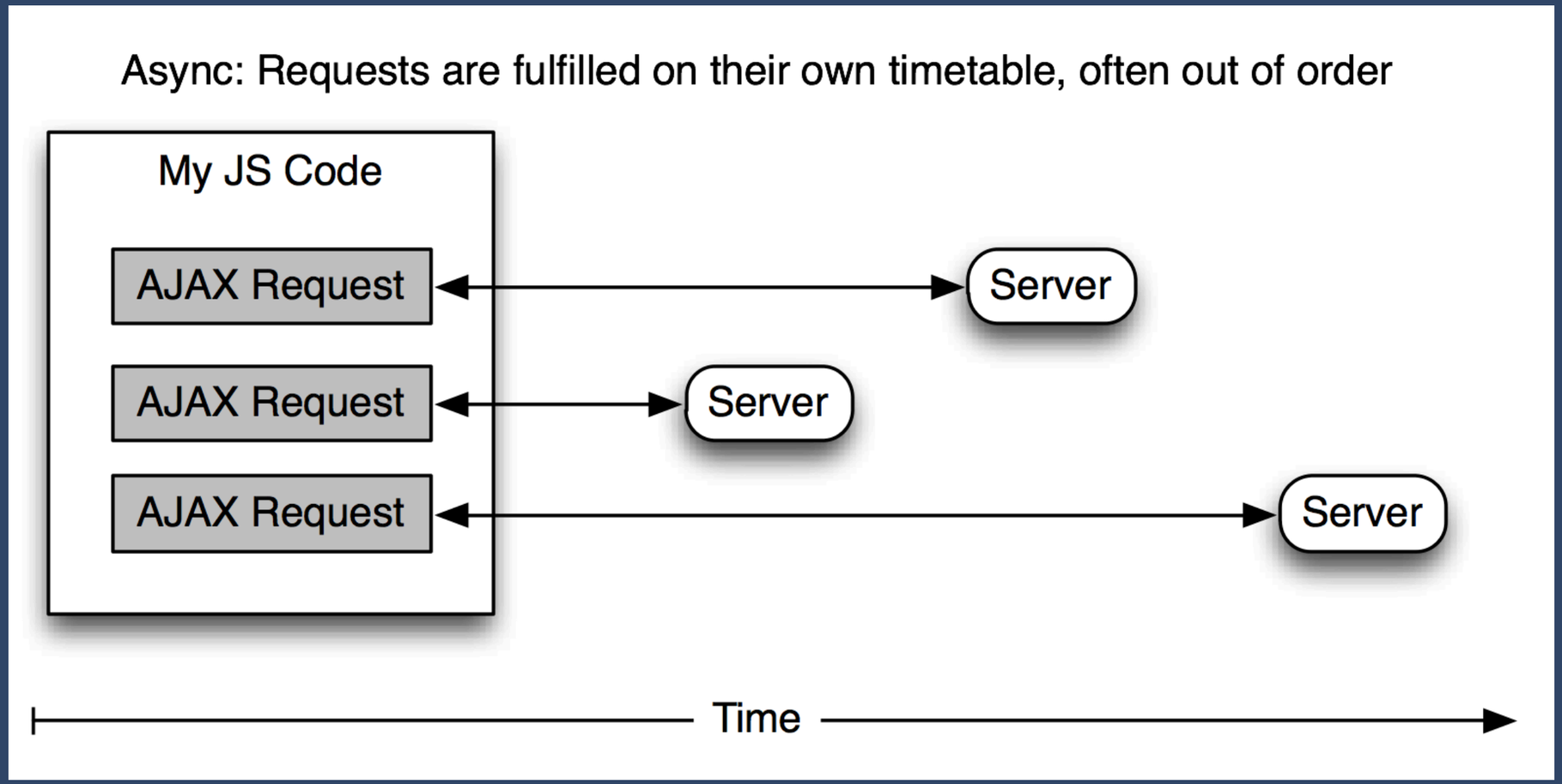
No one likes to wait in line

Asynchronous vs Synchronous

Two competing philosophies in programming.

- **Sync:** The standard way you expect code to execute: The next line executes only once the previous has finished execution
- **Async:** Set up listeners/triggers that are event-driven. This allows for tasks to occur in parallel, independent of each other.

Parallel and independent



Two use cases for Ajax:

- using Ajax while pages are loading
- using Ajax when pages have loaded

Where are we sending these requests?

- To an Application Programming Interface (API)
- A set of functions that are exposed on an application in order for other applications to interact with it
- The interface that you send your HTTP requests to

2. Explain why Ajax primarily transfers JSON data now.

XML is nasty to parse for humans, and it's ***a disaster to parse even for computers.*** *There's just no reason for that horrible crap to exist.*

– Linus Torvalds

```json
{
  "firstName": "Luke",
  "lastName": "Skywalker",
  "isAlive": true,
  "age": 64,
  "address": {
    "estate": "Lars Homestead",
    "planet": "Tatooine"
  },
  "bullseyes": ["Womp Rats", "Death Star"]
}
```

```xml
<person>
    <firstName>Luke</firstName>
    <lastName>Skywalker</lastName>
    <isAlive>true</isAlive>
    <age>64</age>
    <address>
        <estate>Lars Homestead</estate>
        <planet>Tatooine</planet>
    </address>
    <bullseyes>
        <color>Womp Rats</color>
        <color>Death Star</color>
    </bullseyes>
</person>
```

3. Use AJAX to retrieve data from a server.

XMLHttpRequest

- Introduced in 2002
  - Revolutionized the web
  - Websites could be dynamic for the first time
- Used to make requests to other servers without navigating away from the current webpage.

Vanilla JS AJAX

```
// Make a new object
var xhr = new XMLHttpRequest();

// Define callback to trigger when the "load" (async) event happens
xhr.addEventListener('load', function() {
    // If the response has status "200", that indicates success
    if (xhr.status !== 200) {
        return;
    }
    // Log raw response if not status 200
    var data = JSON.parse(xhr.responseText);
    console.log(data);
});

// Tell it what URL to load data from, using what HTTP method (GET/POST etc)
xhr.open('GET', 'https://www.omdbapi.com/?t=Serpico');
xhr.send(); // Send it off!
```

There must be an easier way. jQuery?

Using jQuery

```
var $xhr = $.getJSON('https://www.omdbapi.com/?t=Serpico');

$xhr.done(function(data) {
    if ($xhr.status !== 200) {
        return;
    }

    console.log(data);
});
```

You can even use it to manipulate the DOM

```
$.get("http://www.reddit.com/r/aww.json", function(data) {
    var title = data.data.children[0].title;
    $(".result").append(‘<h1>’ + title + </h1>);
});
```

The .ajax() method

```
$.ajax({
  url: "example.com/api",
  type: "get",
  data: {"first": "Brendan, "last": "Haskins"},
  success: function(response) {
    //Do Something
  },
  error: function(xhr) {
    //Do Something to handle error
  }
});
```

4. Handle CORS issues caused by Ajax requests.

Same Origin Policy (SOP)

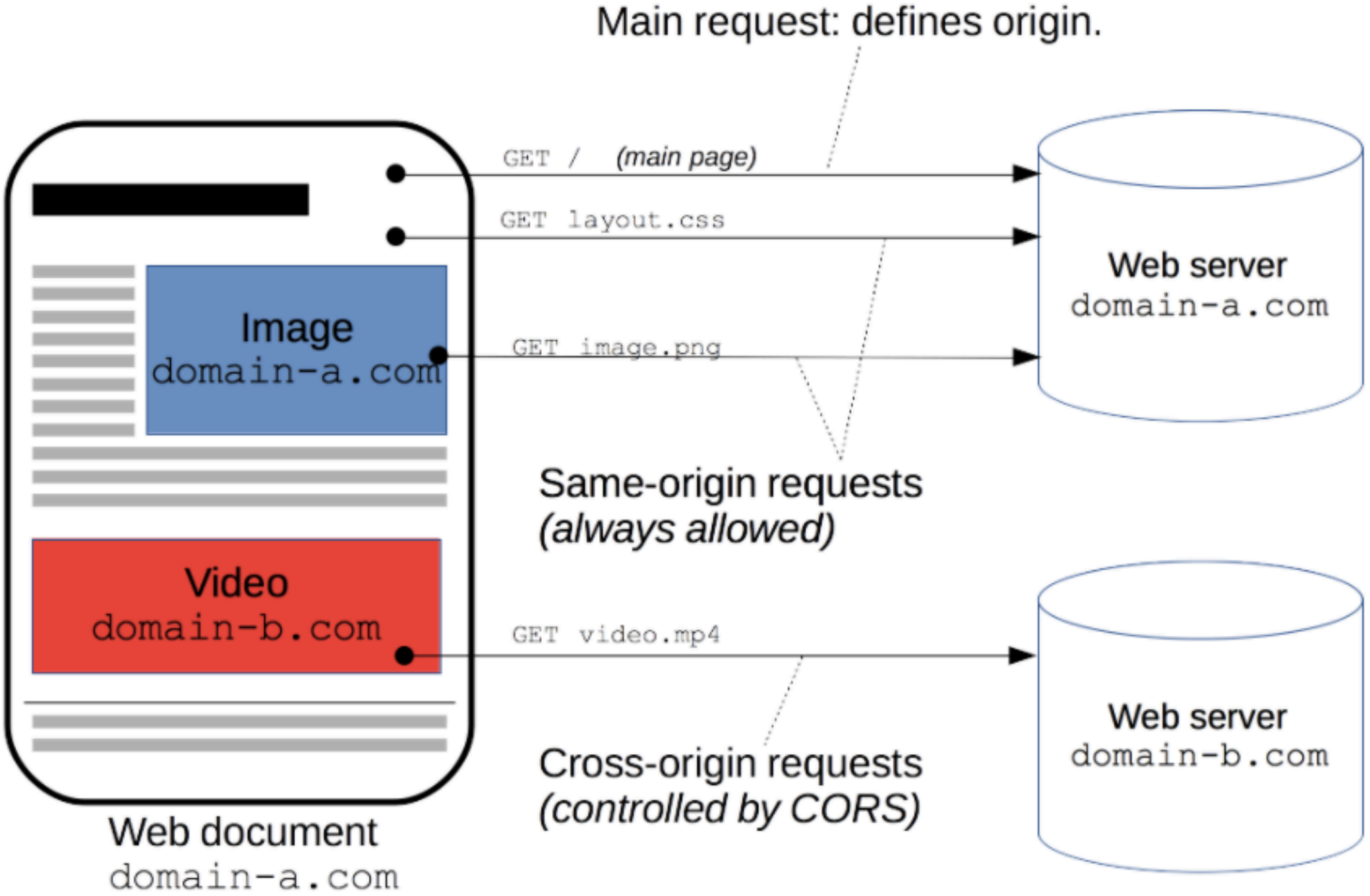
- The Same Origin Policy permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin.
- An origin is defined as a combination of URI scheme, hostname, and port number.
- This policy prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's DOM.

The SOP can pose a problem

Certain "cross-domain" requests, notably AJAX requests, however are forbidden by default by the same-origin security policy.

CORS Restriction example

```
XMLHttpRequest cannot load http://example.com/.
No 'Access-Control-Allow-Origin' header is present
on the requested resource. Origin 'http://example.net/'
is therefore not allowed access.
```

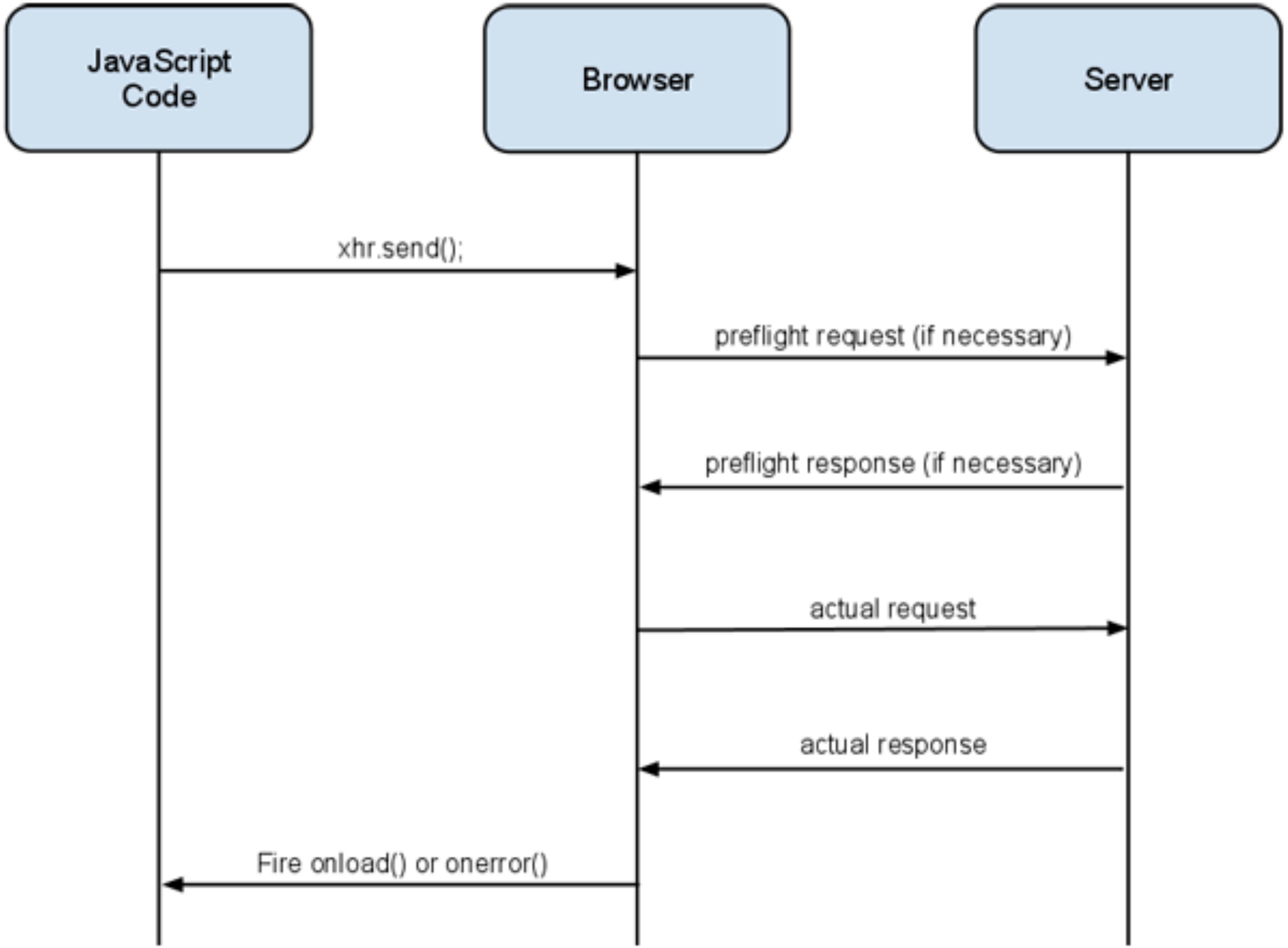

Main request: defines origin.
GET / (main page)
GET layout.css
GET image.png
Image
domain-a.com
Video
domain-b.com
Web server
domain-a.com
Same-origin requests
(always allowed)
GET video.mp4
Web server
domain-b.com
Cross-origin requests
(controlled by CORS)
Web document
domain-a.com

Cross-Origin Resource Sharing

- A mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the resource originated
- Extends HTTP with Origin request header and Access-Control-Allow-Origin response header

Cross-Origin Resource Sharing

- Allows servers to use a header to explicitly list origins that may request a file or to use a wildcard and allow a file to be requested by any site
- Get around CORS restrictions with a proxy (a server, rather than a browser, that grabs and relays a request along)


JavaScript Code
Browser
Server
xhr.send();
preflight request (if necessary)
preflight response (if necessary)
actual request
actual response
Fire onload() or onerror()

5. Handle race conditions caused by Ajax requests.

```
console.log('BEFORE THE AJAX');

var $xhr = $.getJSON('https://www.omdbapi.com/?t=Gods and Monsters');

$xhr.done(function(data) {
    if ($xhr.status !== 200) {
        return;
    }

    console.log(data.Title);
});

console.log('AFTER THE AJAX');
```

Review

1. Explain what Ajax is.
2. Explain why Ajax primarily transfers JSON data now.
3. Use Ajax to retrieve data from a server.
4. Handle CORS issues caused by Ajax requests.
5. Handle race conditions caused by Ajax requests.