Web

Comunicação Cliente-Servidor

Catarina Oliveira

DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

CONTEÚDO

1. Contexto
2. XML
    1. Exemplo XML
3. JSON
4. JSON vs. XML
5. AJAX
6. XMLHttpRequest
7. Modelo clássico vs Modelo AJAX
8. Same-Origin Policy (SOP)
    1. DOM access
    2. AJAX Requests
    3. Data Storage
    4. Cookies
9. Contornar restrições do Same-Origin Policy: CORS/HTML5
    1. CORS: Requisições com credenciais / Requests with credentials
    2. CORS: Requisições simples / simple requests
    3. CORS: Requisições com pré-envio / Preflighted requests

Contexto

- Comunicação cliente/servidor assenta no protocolo HTTP
- Necessita de um formato standard para troca de informação. Exemplos:
  - *Extensible Markup Language* (**XML**): https://www.w3schools.com/xml
  - *JavaScript Object Notation* (**JSON**): https://www.w3schools.com/js/js_json_intro.asp
- **API** (*Application Programming Interface*): forma flexível de comunicação com o servidor web. Exemplo:
  - *Asynchronous JavaScript and XML* (**AJAX**): https://www.w3schools.com/js/js_ajax_intro.asp
    - *XMLHttpRequest* (**XHR**) https://www.w3schools.com/xml/xml_http.asp

XML

- Markup language, tal como o HTML, mas sem tags pré-definidas
- Standard, recomendação W3C.
- Exemplo

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

- Extensível

```
<note>
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

Note

To: Tove

From: Jani

Date: 2015-09-01 08:30

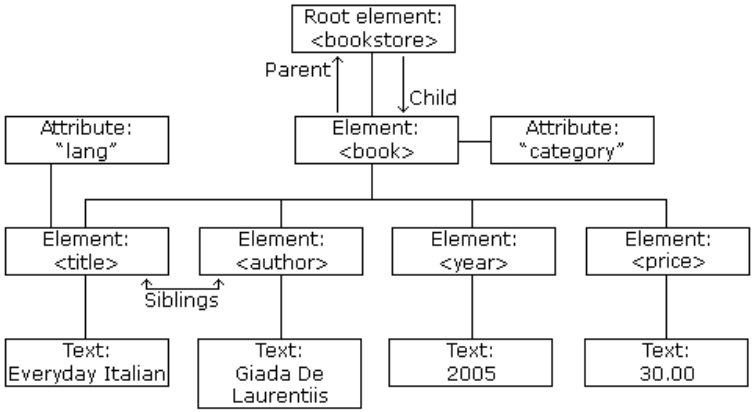
Don't forget me this weekend!

Exemplo XML: Livros

Books.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>

  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>

</bookstore>
```

Árvore do documento:



Title	Author
Everyday Italian	Giada De Laurentiis
Harry Potter	J K. Rowling
XQuery Kick Start	James McGovern
Learning XML	Erik T. Ray

JSON

- Pode guardar o mesmo tipo de informação que o XML

data1.json

```
let data = {
    "note" : {
        "to" : "Tove",
        "from" : "Jani",
        "heading" : "Reminder",
        "body" : "Don't forget me this weekend!"
    }
}
```

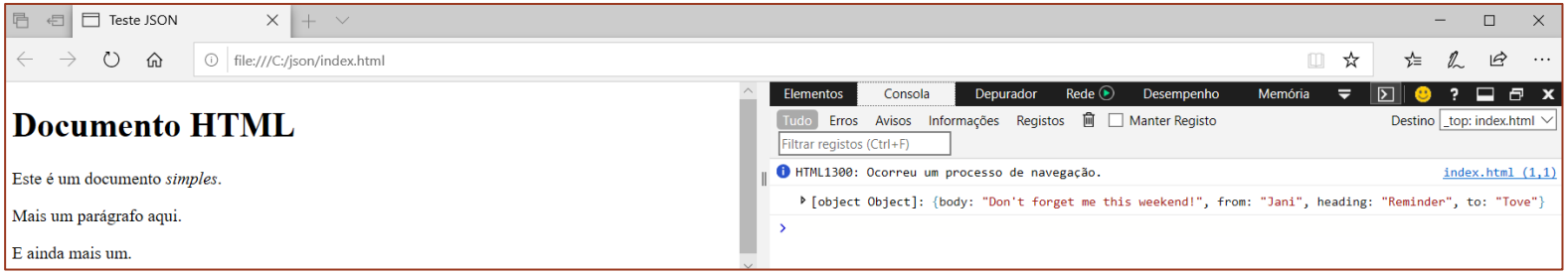
data2.json

```
let data = {
    "note1" : {
        "to" : "Tove",
        "from" : "Jani",
        "heading" : "Reminder",
        "body" : "Don't forget me this weekend!"
    },
    "note2" : {
        "to" : "Jani",
        "from" : "Tove",
        "heading" : "Reminded",
        "body" : "I won't forget!"
    }
}
```

- Facilmente e diretamente conversível para JavaScript

script1.js

```
let obj = JSON.parse('{"to" : "Tove","from" : "Jani","heading" : "Reminder","body" : "Don\'t forget me this weekend!"}')
console.log(obj)
```



- Verificar estrutura: https://jsoneditoronline.org/

JSON vs. XML

JSON

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

Semelhanças

- Self-describing
- Hierárquicos
- Parsing em múltiplas linguagens
- Permitem usar XMLHttpRequest

Diferenças

- JSON não tem end-tag
- JSON é mais curto
- JSON é mais rápido em leitura e escrita
- JSON permite utilização de arrays

AJAX

Asynchronous JavaScript and XML

- Conjunto de tecnologias para potenciar a comunicação assíncrona
- Inclui o XHR
- Permite que os dados sejam passados (por XHR) entre o navegar e o servidor
    - Sem necessidade de recarregamento da página
- Pedidos acionados via JavaScript
- Problema: diferentes browsers implementam AJAX de formas diferentes

XMLHttpRequest

- Pode ser utilizado para pedir dados a um servidor

- Permite:
  - Atualizar a página sem necessidade de recarregar
  - Trocar (pedir / receber / enviar) dados com um servidor depois do carregamento da página

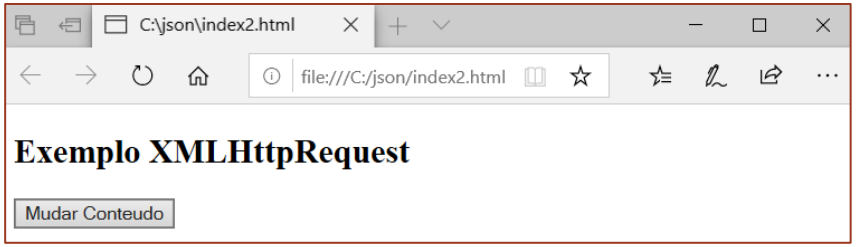
- Exemplo:

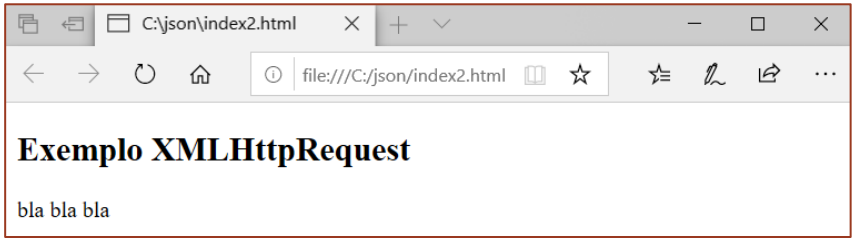
index2.html

```
<!DOCTYPE html>
<html>
<body>

<h2>Exemplo XMLHttpRequest</h2>

<div id="demo">
<button type="button" onclick="loadXMLDoc()">Mudar Conteudo</button>
</div>

<script>
function loadXMLDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "xmlhttp_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

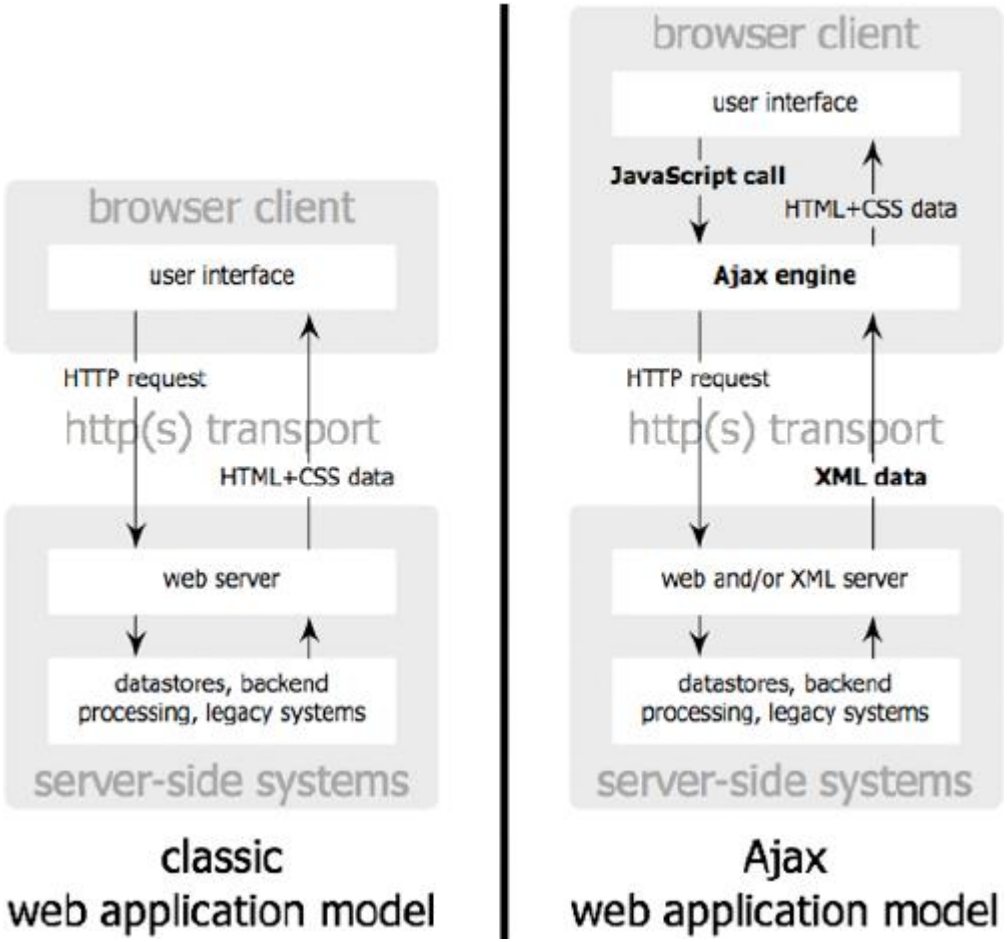
xmlhttp_info.txt

```
bla bla bla
```





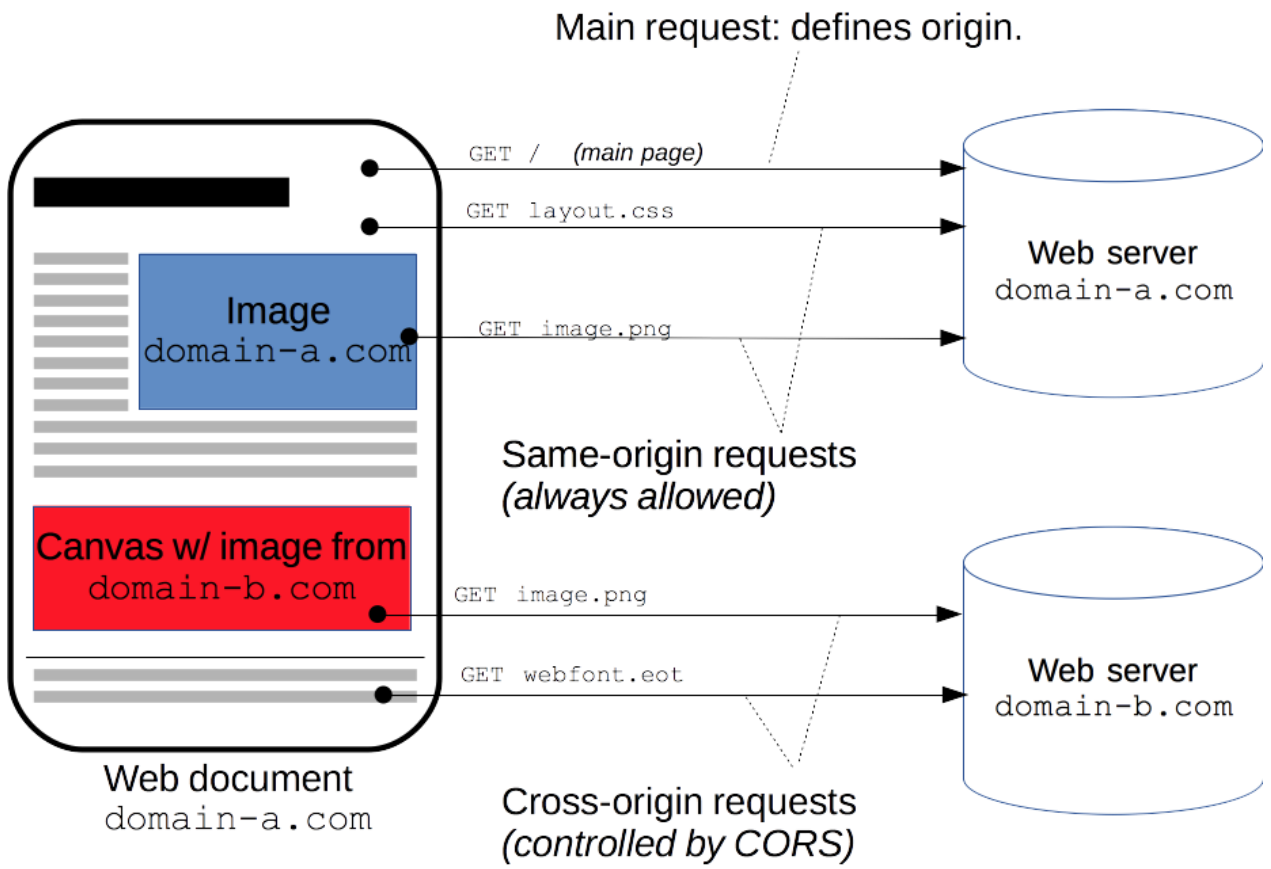
Modelo clássico vs Modelo AJAX



Garrett, J. J. (2005). Ajax: A new approach to web applications.

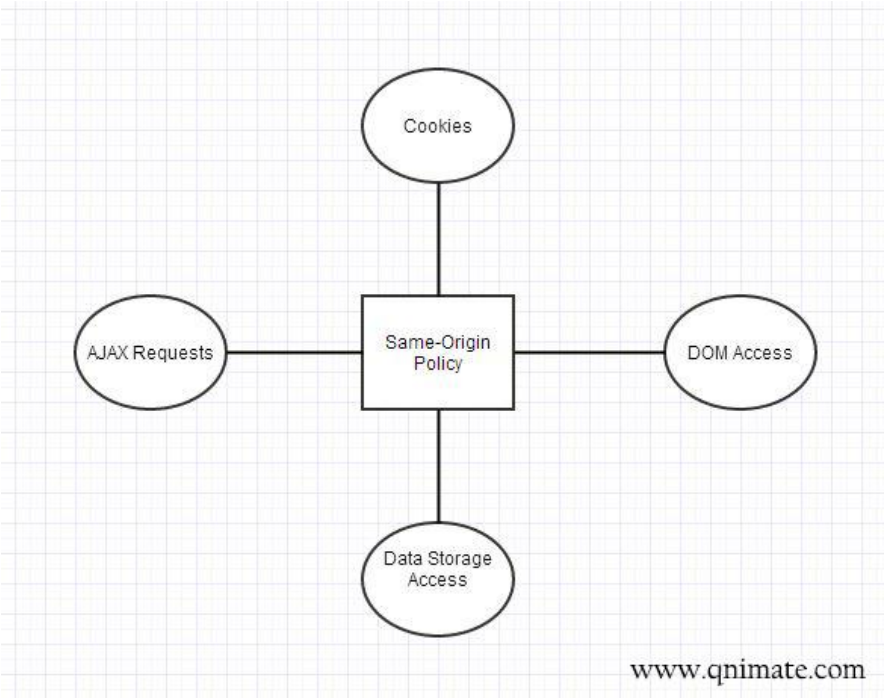
Same-Origin Policy (SOP)

Mesma orígem: mesmo protocolo, domínio e porta

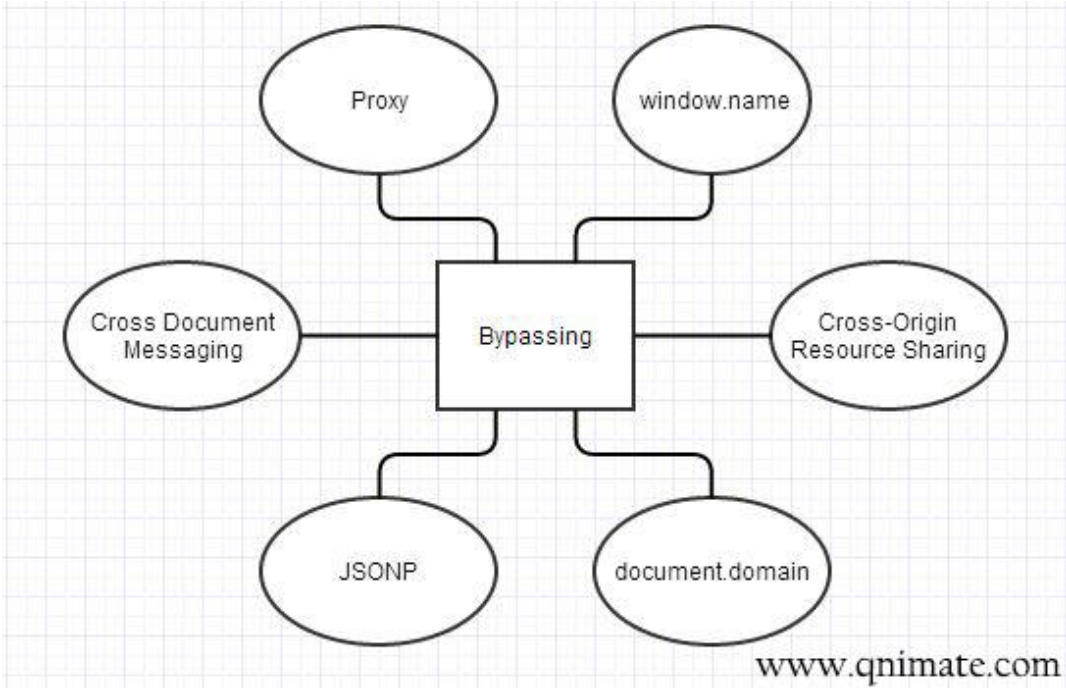


SOP

Restrições



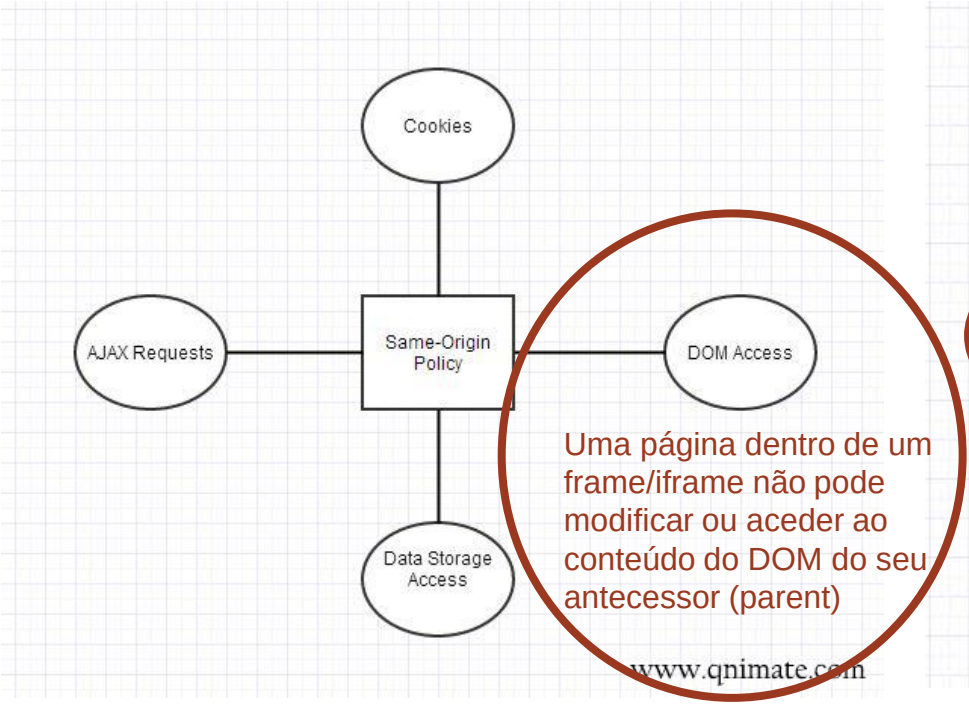
Contornar as restrições



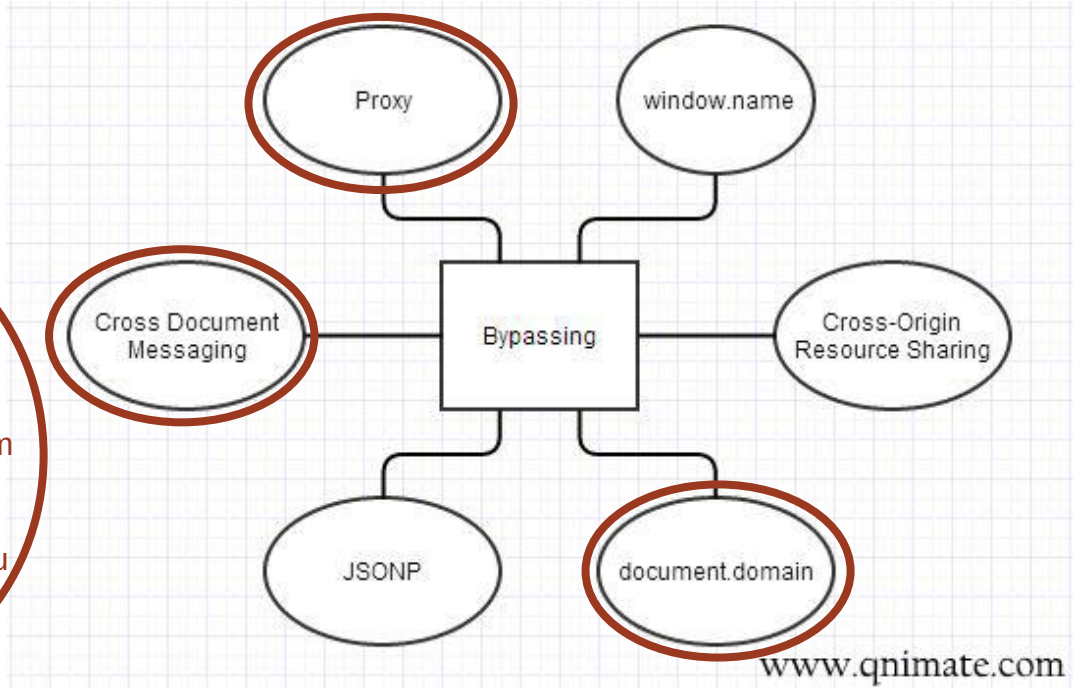
Mais informação: http://qnimate.com/same-origin-policy-in-nutshell/

SOP: DOM access

Restrições

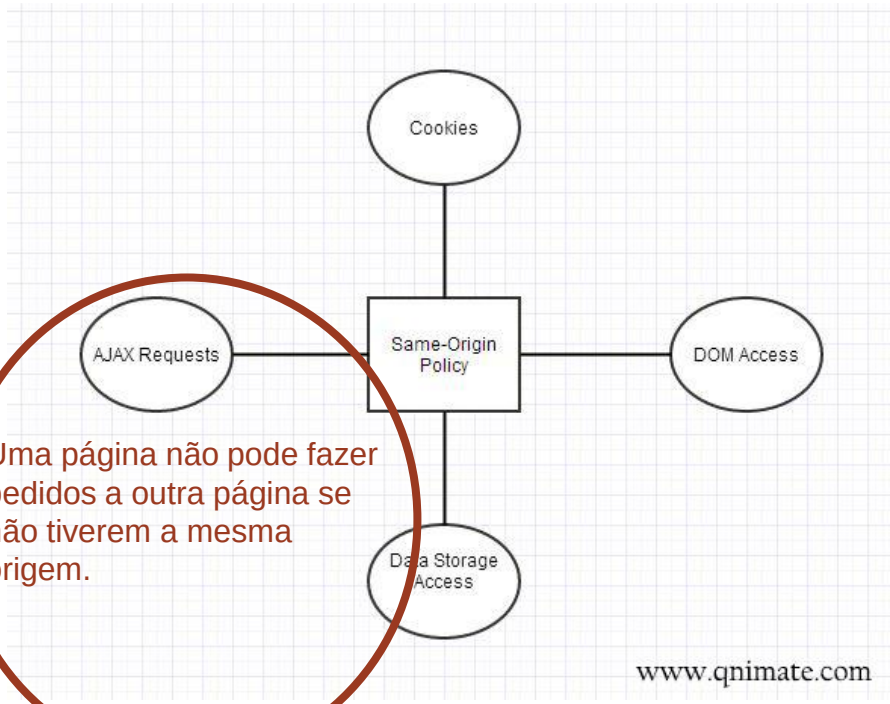


Contornar as restrições



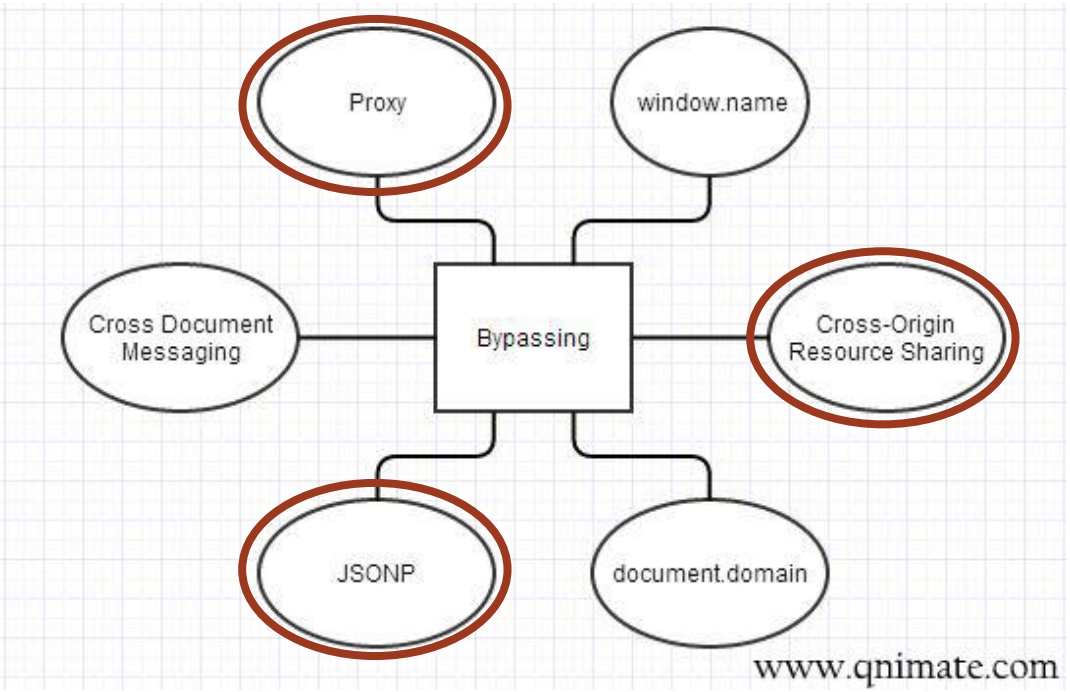
SOP: AJAX Requests

Restrições



Uma página não pode fazer pedidos a outra página se não tiverem a mesma origem.

Contornar as restrições

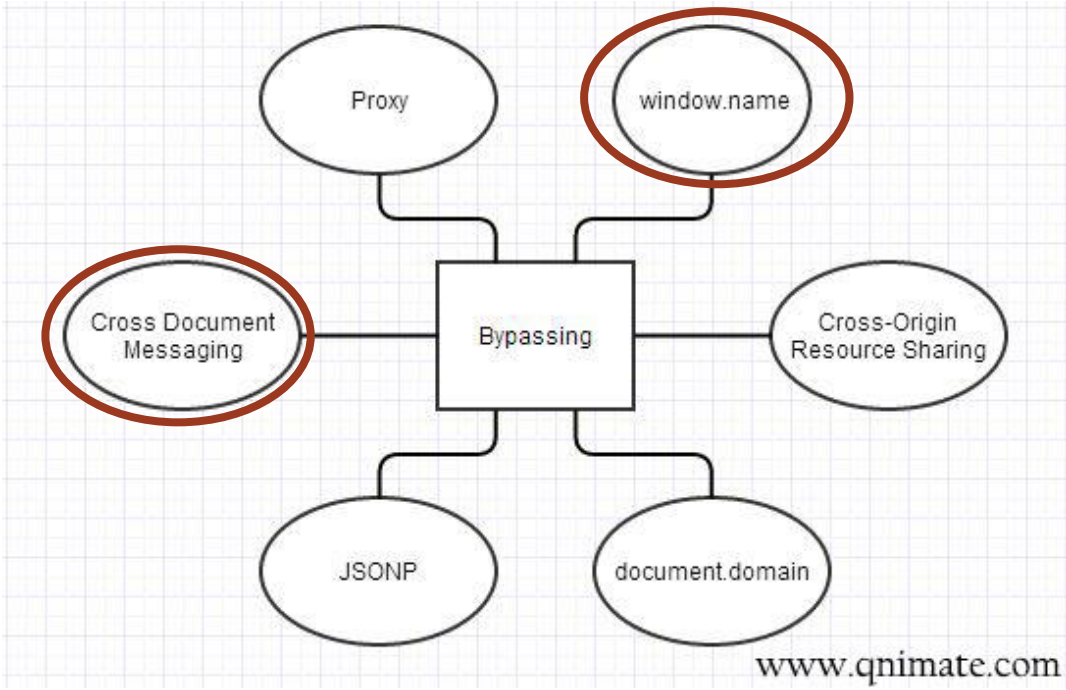


SOP: Data Storage

Restrições

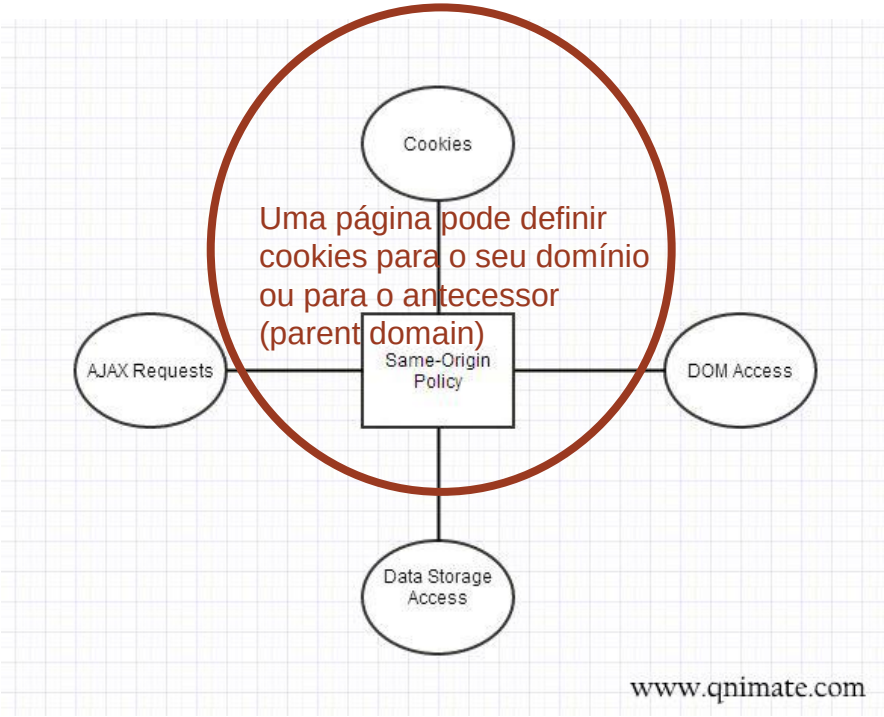
Dados guardados no browser apenas podem ser acedidos a partir da mesma origem de onde foram guardados

Contornar as restrições

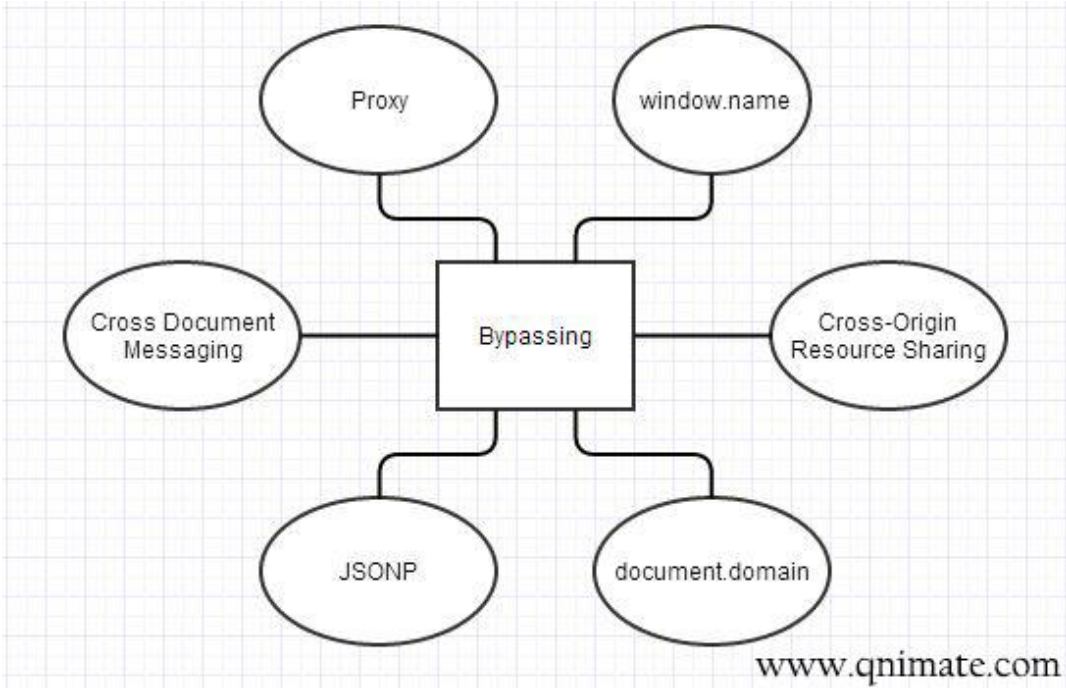


SOP: Cookies

Restrições



Contornar as restrições



Contornar restrições do Same-Origin Policy: CORS/HTML5

- **CORS** (*Cross-Origin Resource Sharing*) – Partilha de recursos com origens diferentes

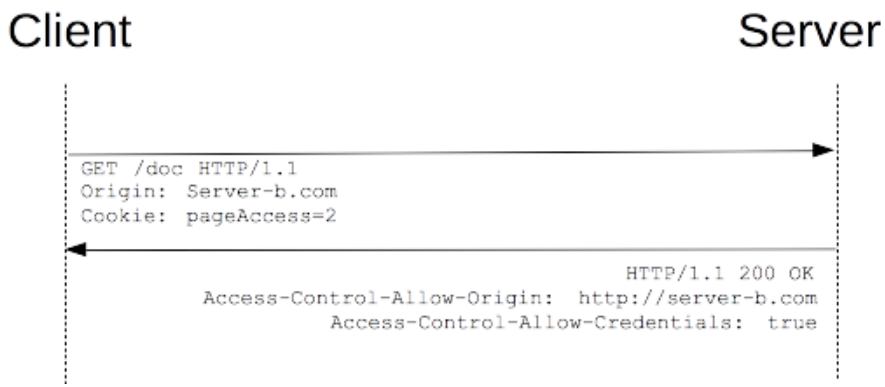
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

- Mecanismo que usa cabeçalhos adicionais HTTP para informar o navegador de que deve permitir que uma aplicação Web seja executada numa origem (domínio) com permissão para aceder a recursos de um servidor numa origem diferente.
- Por razões de segurança, os navegadores restringem requisições cross-origin HTTP a partir de scripts.

- Por exemplo, XMLHttpRequest segue a política de mesma origem (*same-origin policy* - **SOP**).
    - Uma aplicação só poderá fazer pedidos HTTP à mesma origem da qual foi carregada, a não ser que a resposta da outra origem inclua os cabeçalhos CORS corretos.

CORS: Requisições com credenciais / *Requests with credentials*

- Por defeito, os browsers não enviam credenciais. Para que enviem, tem de ser definido explicitamente
- Neste caso, são enviadas as credenciais sob a forma de cookies
- A resposta do servidor tem de incluir a origem no campo Access-Control-Allow-Origin, em vez da wildcard *



CORS: Requisições simples / *simple requests*

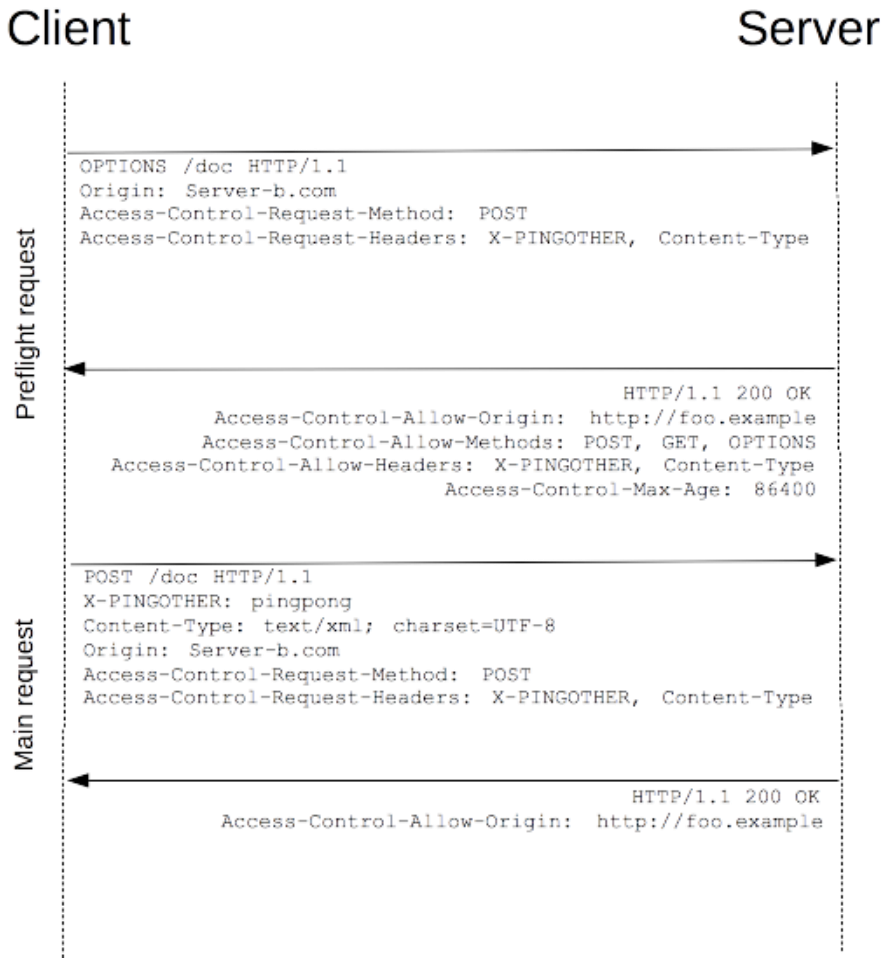
- Não acionam um pré-envio CORS
- **Métodos** permitidos: GET, HEAD, POST
- Além dos **cabeçalhos** definidos automaticamente pelo agente, podem ser definidos manualmente: Accept, Accept-Language, Content-Language, Content-Type (porém observe os requisitos adicionais abaixo), DPR, Downlink, Save-Data, Viewport-Width, Width
- Valores permitidos para o **Content-Type**: application/x-www-form-urlencoded, multipart/form-data, text/plain
- Nenhum ***event listener*** é registado em qualquer objeto XMLHttpRequestUpload usado no pedido (acedidos com XMLHttpRequest.upload)
- Nenhum **objeto** ReadableStream é usado no pedido

CORS: Requisições com pré-envio / *Preflighted requests*

- Enviam um pedido HTTP através do método OPTIONS para obter um recurso noutro domínio, para determinar se o recurso é seguro para envio

- Se usa um dos **métodos**: PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

- Não respeitam o apresentado para simple request em termos de **cabeçalhos**, **Content-type**, ***event listeners*** e **objetos**



UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.