<epam>

Excellence in Software Engineering

BROWSER-TO-SERVER COMMUNICATION. AJAX

THEORY & PRACTICE

Author: Viktor Pishuk

DAILY STAND-UP

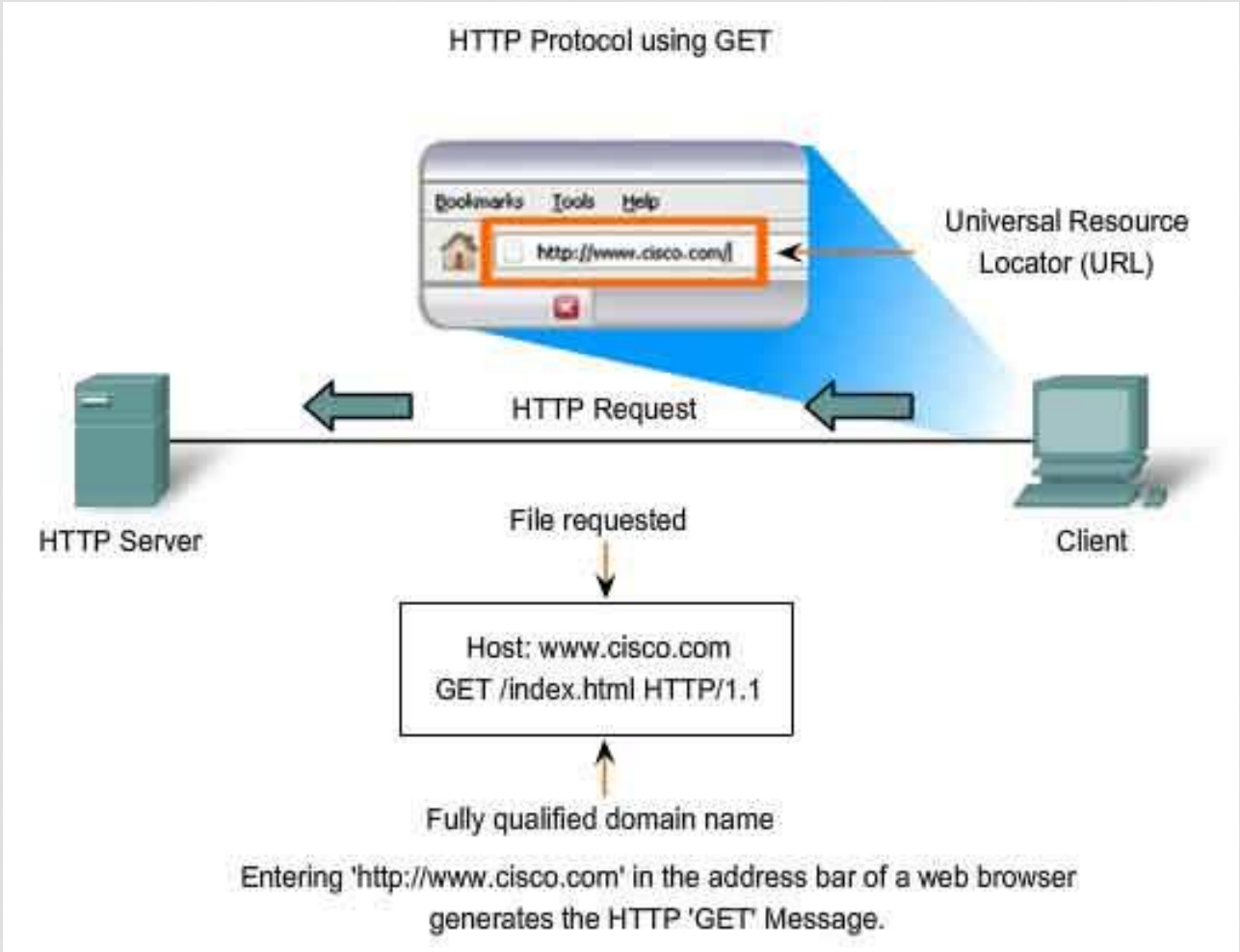
IT IS TIME TO LET EVERYBODY KNOW ABOUT YOUR PROGRESS

AGENDA

- Theory
  - HyperText Transfer Protocol (HTTP)
  - Asynchronous JavaScript and XML (AJAX)
  - JavaScript Object Notation (JSON)
  - Same-origin Policy
  - Cross-origin resource sharing (CORS)
  - JavaScript Object Notation with Padding (JSONP)
  - XMLHttpRequest
  - Promise
- Practice
- Q&A

THEORY

HYPERTEXT TRANSFER PROTOCOL



HYPERTEXT TRANSFER PROTOCOL

Persistence connection to servers was born in 1999

HTTP STRUCTURE

```
GET /conf-2009.avi HTTP/1.0
Host: example.org
Accept: */*
User-Agent: Mozilla/4.0
Referer: http://example.org/
```

HTTP METHODS

GET
POST

PUT, OPTIONS, HEAD, DELETE,
PATCH, TRACE, CONNECT

HTTP STATUS CODES

1xx - Informational

2xx - Success

3xx - Redirection

4xx – Error with Client

5xx – Error with Server

AJAX

Asynchronous JavaScript and XML.

HOW TO REALIZE AJAX

- Through XMLHttpRequest object
- Through iframes
- Through dynamic <script> elements

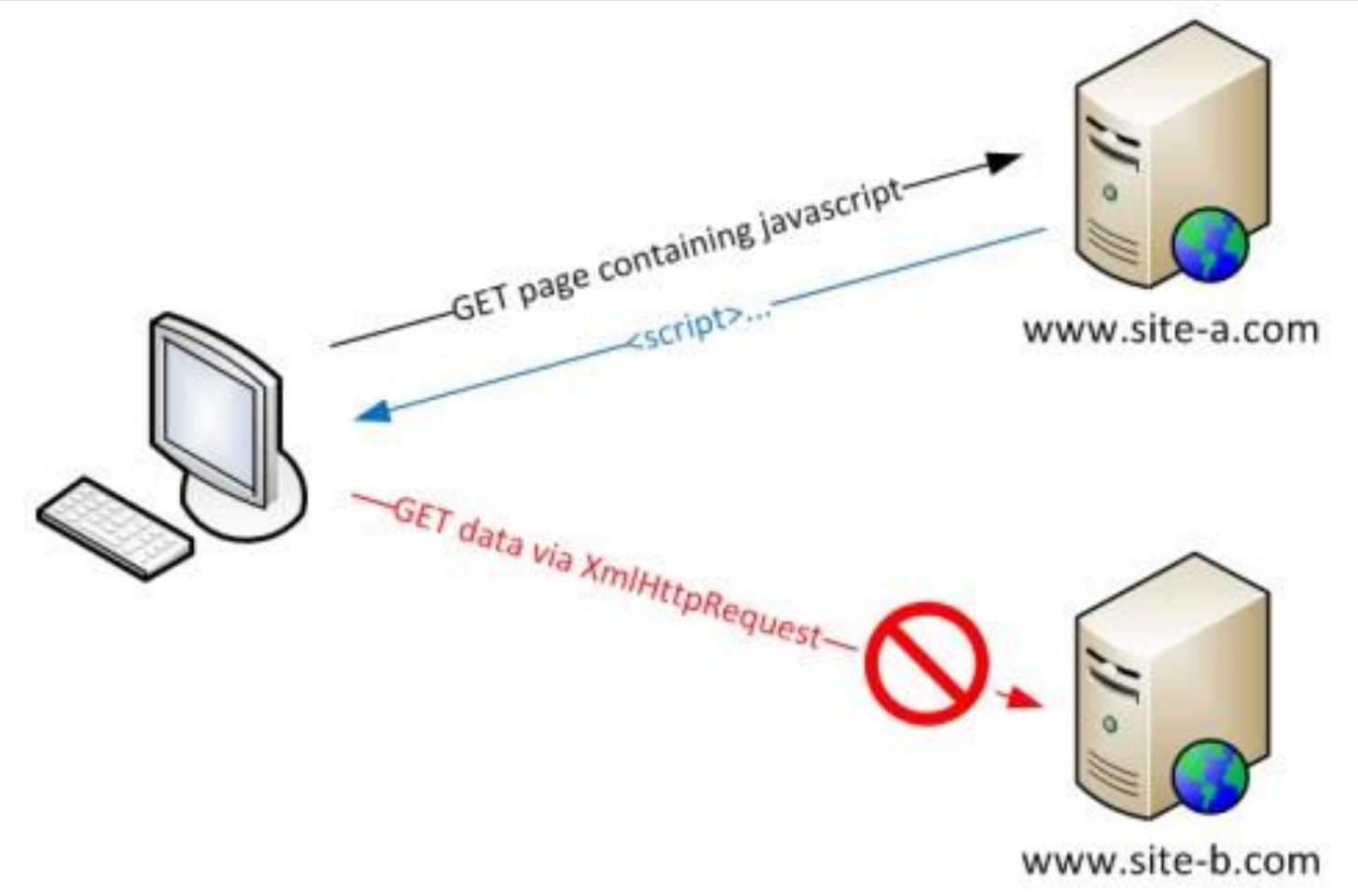
STRENGTH

- Less amount of traffic
- Reducing load on a server
- UI become faster
- UI become more interactive

JSON FORMAT

```
{
    "firstName": "Ivan",
    "lastName": "Ivanov",
    "address": {
        "streetAddress": "Kudryashova",
        "city": "Kyiv",
        "postalCode": 101101
    },
    "phoneNumbers": [
        "050 589 78 59"
    ]
}
```

SAME ORIGIN POLICY



CROSS-ORIGIN NETWORK ACCESS

- Cross-origin writes are typically allowed.
  Examples: form submissions.
- Cross-origin embedding is typically allowed.
  Examples: script, img, video, CSS, etc.
- Cross-origin reads are typically **not allowed**

JSONP FORMAT

```
GET /service?callback=myCallback HTTP/1.0
Host: example.org
Accept: */*
User-Agent: Mozilla/4.0
Referer: http://example.org/
```

```
myCallback({
    "firstName": "Ivan",
    "lastName": "Ivanov"
})
```

CROSS-ORIGIN RESOURCE SHARING

```
GET /cors.txt HTTP/1.1
Host: www.b.com
Origin: www.a.com
```

To let origin a access to origin b add header into response:

```
Access-Control-Allow-Origin: http://www.a.com
```

To open access to origin a from any origin use symbol "*":

```
Access-Control-Allow-Origin: *
```

CORS SUPPORTED BY

v. >= 3.5

v. >= 3

v. >= 4

v. 12

v. >= 8
partially
XDomainRequest

XMLHTTPREQUEST

```
var the_object;
var http_request = new XMLHttpRequest();
http_request.onreadystatechange = function () {
    if (http_request.readyState == 4 ) {
        if ( http_request.status == 200 ) {
            the_object = JSON.parse(http_request.responseText);
        } else {
            console.log( "There was a problem with the URL." );
        }
        http_request = null;
    }
};
http_request.open( "GET", url, true );
http_request.send(null);
```

WHAT ABOUT MULTIPLE CALLS

If 1.000 AJAX calls depend on each other,
how will you organize call flow to see the final result?

```
$.ajax({…}).success(function () {
    $.ajax({…}).success(function () {
        $.ajax({…}).success(function () {
        });
    });
});
```

Do you like this approach?

PROMISE

```
var p1 = new Promise(function(resolve, reject) {
    setTimeout(resolve, 500, "one");
});

var p2 = new Promise(function(resolve, reject) {
    setTimeout(resolve, 100, "two");
});

Promise.all([p1, p2]).then(function(value) {
  // value === "two"
});
```

PROMISE BASIC SUPPORT

v. 29

v. 32

Not Supported

v. 19

Not supported

HOW TO USE PROMISES

Use libraries , frameworks like:

1. jQuery
2. AngularJS
3. WinJS
4. Etc.

PRACTICE

Load any content via AJAX

Task:

Create a function to obtain HTML-file from a web-service, using XMLHttpRequest object.

Log information about progress into console.

Log information about successful or not successful result into console.

Tips & Triks:

XMLHttpRequest has an event onreadystatechange
XMLHttpRequest has a property readyState
XMLHttpRequest has a property status

ANSWER

```
function getHTML(contentUrl) {
    var req = new XMLHttpRequest()
    req.open('GET', contentUrl, true);
    req.onreadystatechange = function() {
        switch (req.readyState) {
            case 1: console.log('Loading the data...'); break; //useless
            case 2: console.log('Data is loaded'); break; //useless
            case 3: console.log('Interactive...'); break; //useless
            case 4:
                if(/^[231]/.test(req.status) { // RegExp is better
                    console.log('Request finished successfully!');
                } else {
                    console.log('Request finished successfully!');
                }
                break;
            default: console.log('Uninitialized'); break; //useless
        }
    };
    req.send(null);
}
```

Return result of Request

Task:

Update code from previous task:

getHtml function shall return promise which you are able to use to bind callbacks.

Tips & Triks:

Think about how to resolve or reject promise in getHtml function

ANSWER

```
function getHTML(url) {
    var resolveRequest, rejectRequest;
    var promise = new Promise(function (resolve, reject) {
        resolveRequest = resolve;
        rejectRequest = reject;
    });

    //...
            case 4:
                if(/^[231]/.test(req.status)) {
                    //...
                    resolveRequest(req.responseText);
                } else {
                    //...
                    rejectRequest("ERROR " + req.status);
                }
                break;
    //...
    return promise;
}
```

The Chain

Task:

Update code from prev. task to add chaining like:

```
getHtml('yourUrl').
    success(onSuccess).failure(onFailure);
```

Tips & Triks:

Perhaps it make sense to store promise and return an object from getHtml function

ANSWER

```
function getHTML(url) {
    //...
    function aChain(_promise) {
        var p = _promise;
        var self = this;
        var successCallback, failureCallback;
        this.success = function (callback) {
            successCallback = callback;
            Promise.all([promise]).then(function (values) {
                successCallback.apply(window, values);
            });
            return self;
        };
        this.failure = function (callback) {
            p.catch(callback);
            return self;
        };
    }
    var chain = new aChain(promise);
    req.send(null);
    return chain;
}
```

JSONP

Task:

Update code from prev. task to add ability to load data in JSONP format.

HOMEWORK

Make you web-pages dynamic:

- Use AJAX approach to load dynamic content
- Use promises (native) and deferred objects (jQuery) to organize callbacks

CONTACTS

Author:

Viktor Pishuk

Senior Software Engineer at "EPAM Systems Ukraine"

E-mail: Viktor_Pishuk@epam.com

Skype: victor.pishuk

USEFUL LINKS

- **http://jsfiddle.net/vmuha/933U9/**
- http://xmlhttprequest.ru/
- https://developer.mozilla.org/en-US/docs/AJAX
- https://ru.wikipedia.org/wiki/HTTP
- https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise
- http://api.jquery.com/category/deferred-object/
- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy