jQuery.JS and AJAX

By Eman Fathi

The old-fashioned approach

DESCRIPTION:
Don't call this knotted headdress a hair net—it's actually a snood. In the late 1850s and early 1860s, it was in vogue for young women to hold their hair back with close-fitting, bag-shaped caps they'd woven from velvet, lace, yarn, or other materials. The trend faded by the 1870s, but the snood made a comeback in the 1940s after female factory workers realized it added both practicality and panache to their work ensembles. The hat fell to the wayside once more after World War II ended and women returned to the domestic sphere.

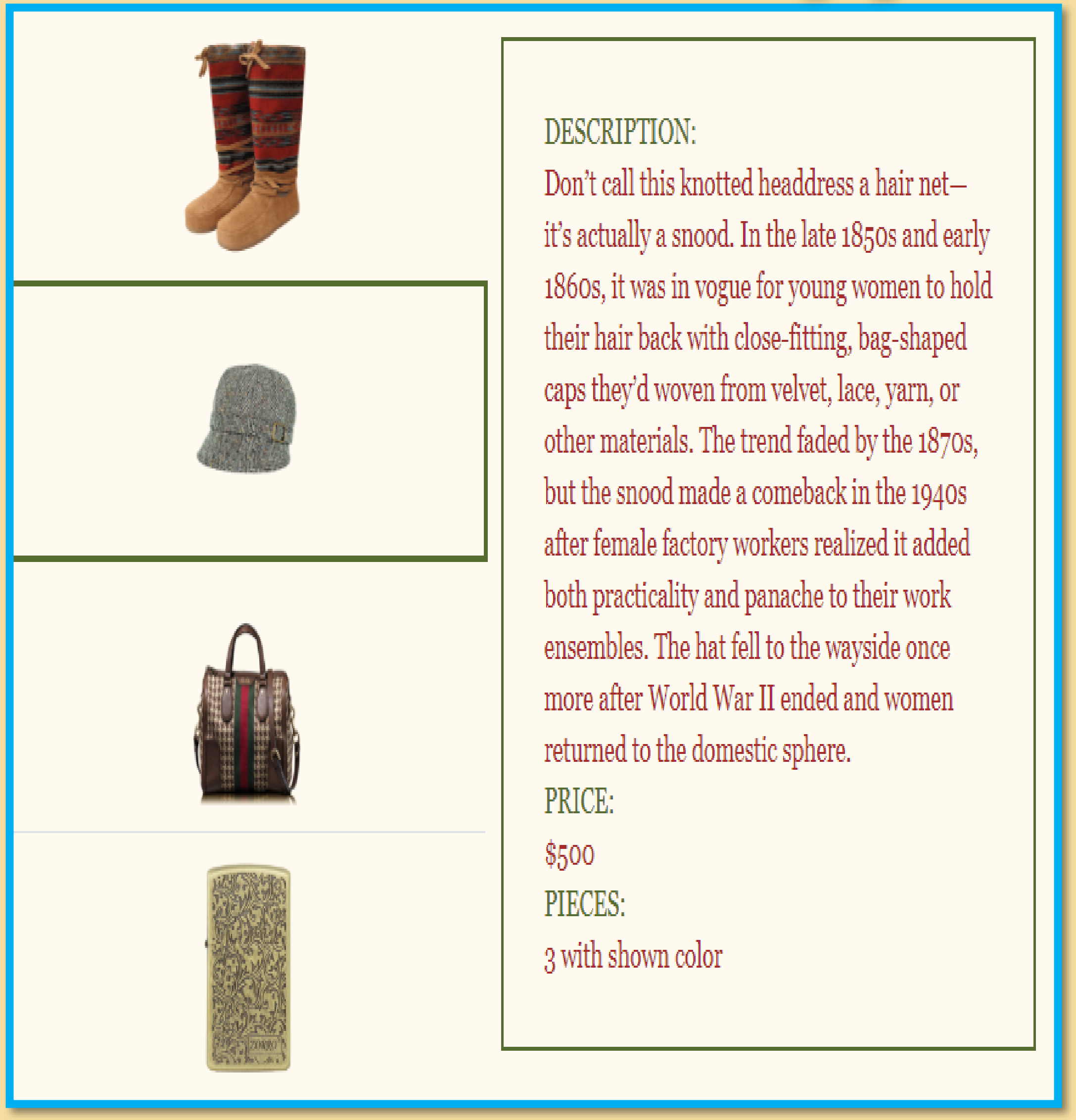
PRICE:
$500

PIECES:
3 with shown color

The old-fashioned approach

Client

When user Clicks on Image



DESCRIPTION:

Don't call this knotted headdress a hair net–it's actually a snood. In the late 1850s and early 1860s, it was in vogue for young women to hold their hair back with close-fitting, bag-shaped caps they'd woven from velvet, lace, yarn, or other materials. The trend faded by the 1870s, but the snood made a comeback in the 1940s after female factory workers realized it added both practicality and panache to their work ensembles. The hat fell to the wayside once more after World War II ended and women returned to the domestic sphere.

PRICE:

$500

PIECES:

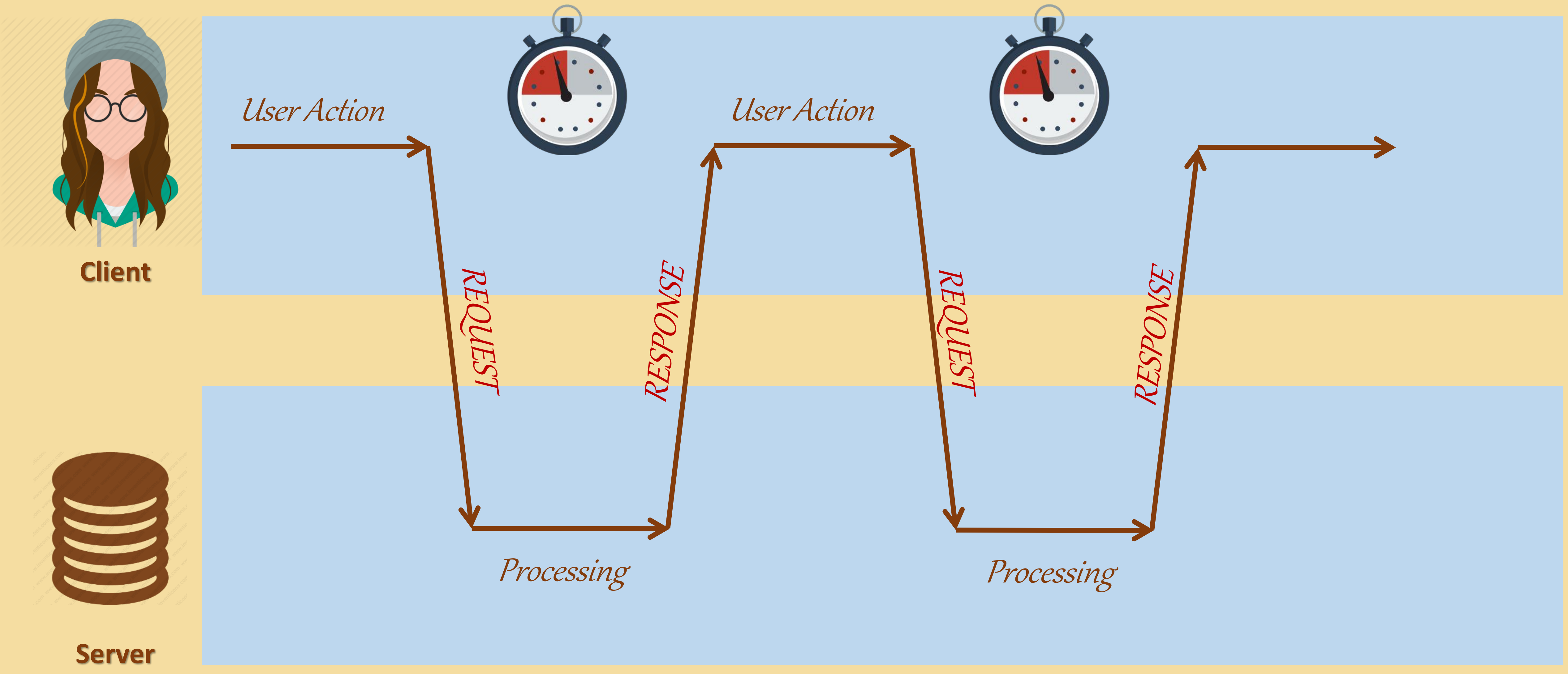
3 with shown color

Browser sends request to server

Server sends back the whole new page

Server

The old-fashioned approach



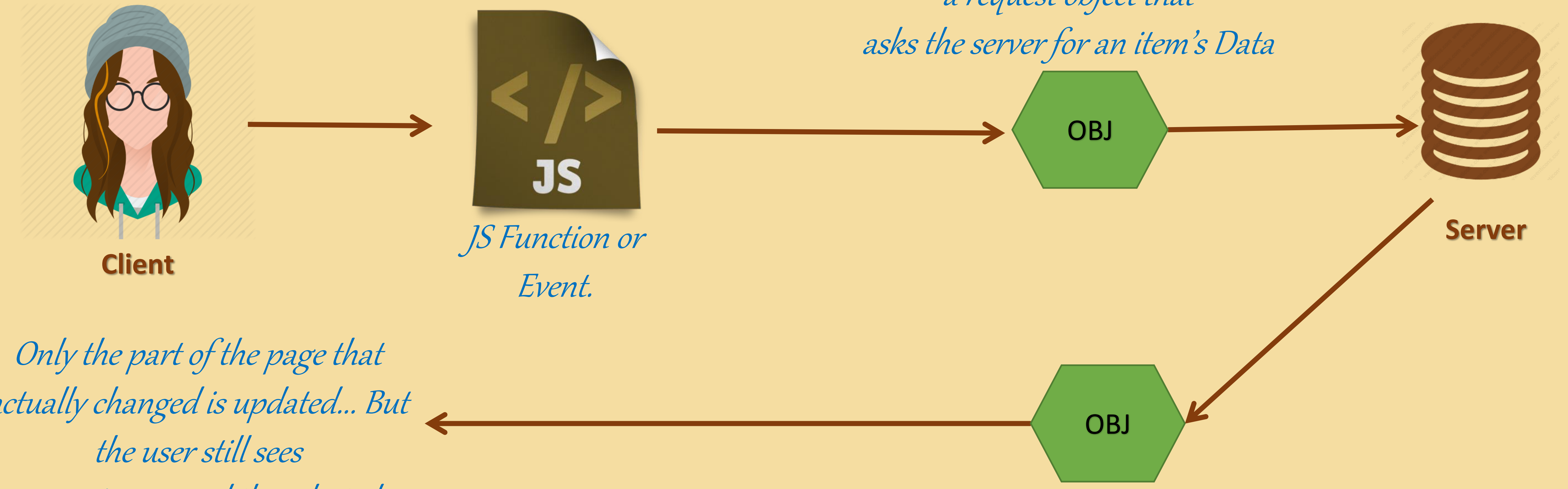
The old-fashioned approach

- ✓Typical browsing behavior consists of **loading a web page**, then selecting some action that we want to do, filling out a form, submitting the information, etc.
- ✓We work in this **sequential** manner, requesting one page at a time, and have to **wait for the server to respond**, **loading a whole new web page** before we continue.
- ✓This is also one of the limitations of web pages, where transmitting information between a client and server generally requires a new page to be loaded.
- ✓**JavaScript** is one way to cut down on (some of) the client-server response time, by using it to **verify form (or other) information before it's submitted to a server**.
- ✓Another drawback to this usual sequential access method is that there are many situations where you load a new page that shares lots of the **same parts** as the old (consider the case where you have a "menu bar" on the top or side of the page that doesn't change from page to page).

Ajax

Asynchronous JavaScript and XML

How does AJAX work?



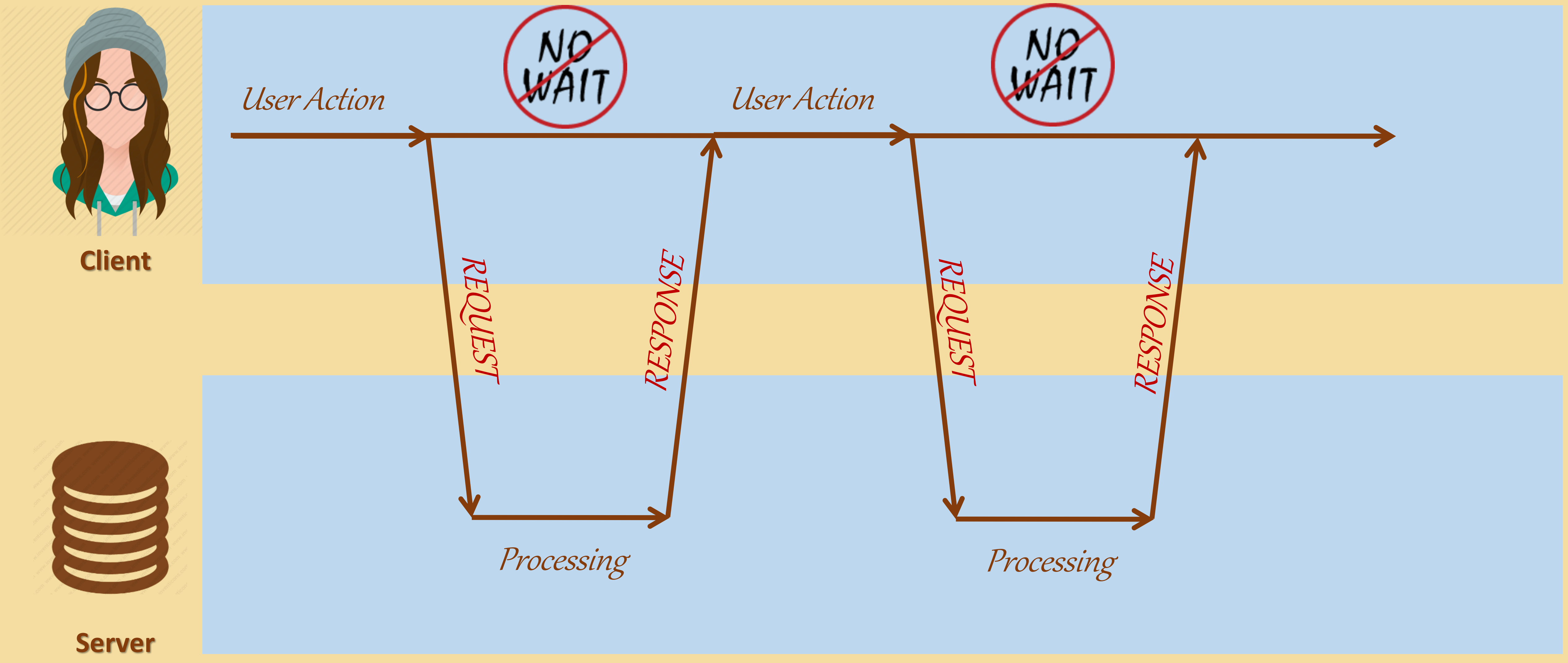
JS Function or Event.

The function creates a request object that asks the server for an item's Data

Server sends object back but updated with new details

Only the part of the page that actually changed is updated... But the user still sees a new image and the selected item's description

The AJAX approach



What is Ajax ?

Ajax is a way of designing and building web pages that are as interactive and responsive as desktop applications. Your pages make ***asynchronous*** requests that allow the user to keep working instead of waiting for a response. You only update the things on your pages that actually change. And best of all, an Ajax page is built using standard Internet technologies, things you probably already know how to use, like:

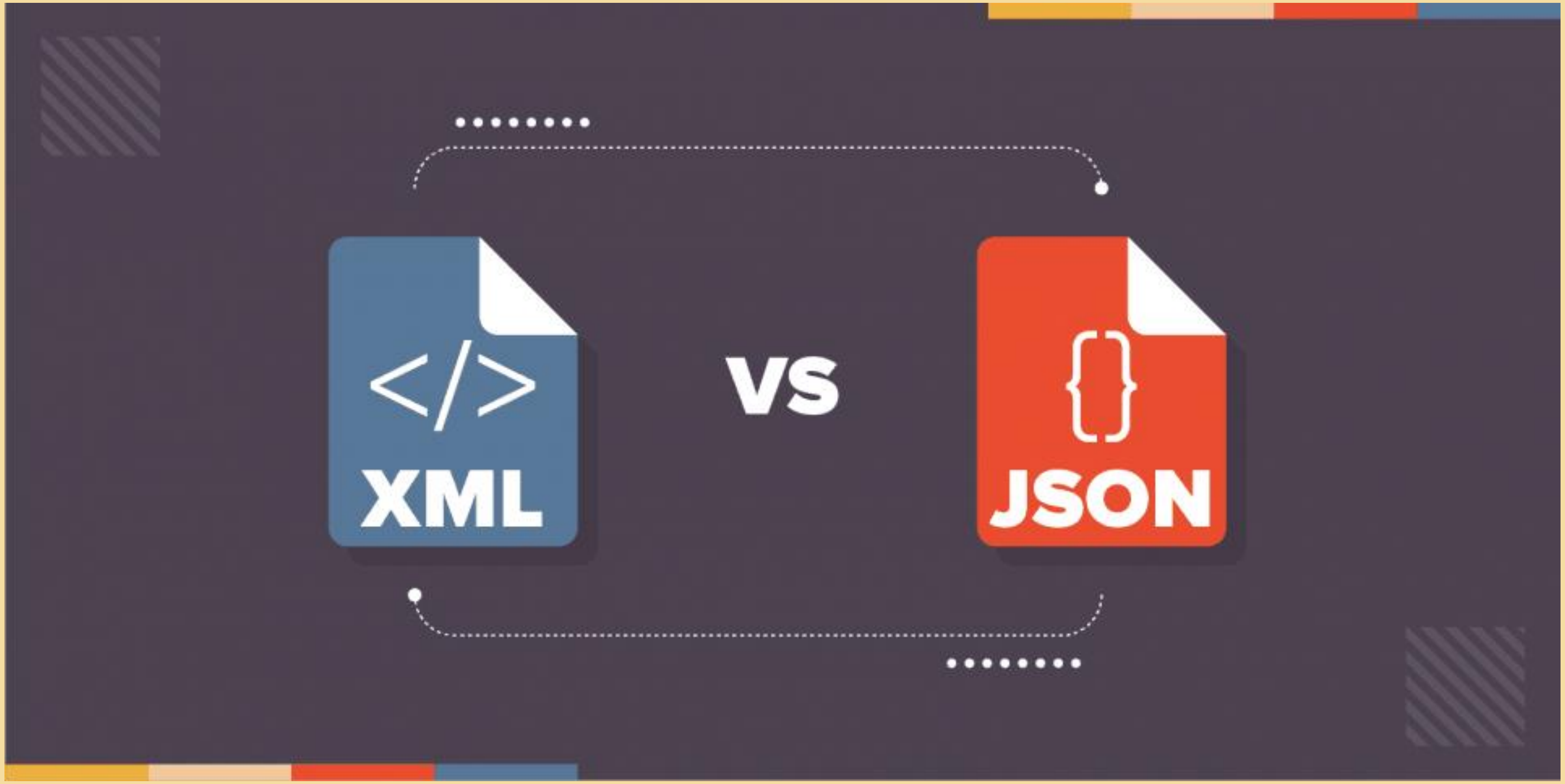
✓*XHTML*

✓*Cascading Style Sheets*

✓*JavaScript*

Why Ajax ?

- ✓ javascript can sumbit request to the server and receive response then update current page while user would never know that web page was even transmitted a request to the server.
- ✓User can continue to use the application while web page sends and receives information from the server in the background

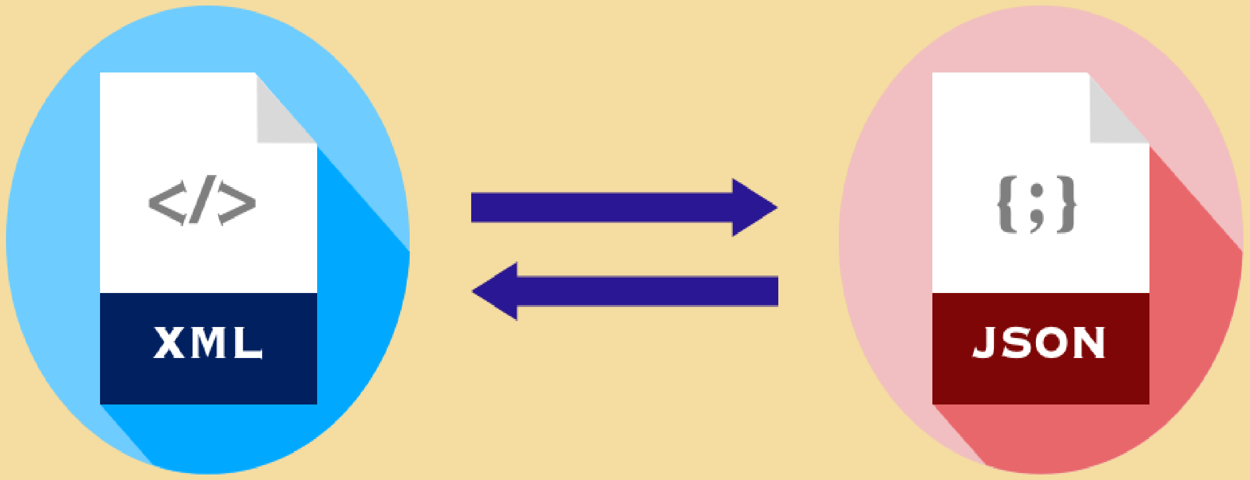

XML
VS
JSON

XML

```
<empinfo>
    <employees>
        <employee>
            <name>James Kirk</name>
            <age>40></age>
        </employee>
        <employee>
            <name>Jean-Luc Picard</name>
            <age>45</age>
        </employee>
        <employee>
            <name>Wesley Crusher</name>
            <age>27</age>
        </employee>
    </employees>
</empinfo>
```

JSON

```
{   “empinfo” :
        {
            “employees” :  [
            {
                “name” : “James Kirk”,
                “age” : 40,
            } ,
            {
                “name” : “Jean-Luc Picard”,
                “age” : 45,
            } ,
            {
                “name” : “Wesley Crusher”,
                “age” : 27,
            }
                           ]
        }
    }
```



JSON	XML
Text based format (Not a Language)	Markup Language
Free to define anything	Has some rules
Smaller Size	Big in Size due to markups
JSON is similar to Java script Objects literals. Browser read faster.	Browser need parsers to handle XML. Slow processing.

Javascript Ajax

XMLHttpRequest

XMLHttpRequest Object

- The XMLHttpRequest object is the backbone of every Ajax method. Each application requires the creation of one of these objects. So how do we do it?
- As with most things in web programming, this depends upon the web browser that the client is using because of the different ways in which the object has been implemented in the browsers.

- Firefox, Safari, Opera, and some other browsers can create one of these objects simply using the “new” keyword.

```
<script type="text/javascript">
       ajaxRequest = new XMLHttpRequest();
</script>
```

-

XMLHttpRequest Object

- ✓Microsoft Internet Explorer implements this object using its proprietary ActiveX technology. This requires a different syntax for creating the object (and can also **depend upon the particular version** of Internet Explorer being used).
- ✓ try { ... } catch (err) { ... }
- ✓To handle different types of browsers, we use the
- ✓format. The "try" section attempts to execute some JavaScipt code. If an error occurs, the "catch" section is used to intervene before the error crashes the JavaScript (either to indicate an error has happened, or to attempt something else).
- ✓To create one of these objects we can use a sequence of try. . . catch blocks, attempting different ways to create an XMLHttpRequest object.

```
function getXMLHttpRequest() { /* This function attempts to get an Ajax request
object by  trying a few different methods for different browsers. */
var request, err;
try {   request = new XMLHttpRequest(); // Firefox, Safari, Opera, etc.
    }
catch (err) {
               try { // first attempt for Internet Explorer
               request = new ActiveXObject("MSXML2.XMLHttp.6.0");}
               catch (err) {
                      try { // second attempt for Internet Explorer
                             request = new ActiveXObject("MSXML2.XMLHttp.3.0");
                          }
                      catch (err) { request = false; // oops, can’t create one! }
                  }
}
return request;
```

XMLHttpRequest Object

- ✓As with any object in JavaScript (and other programming languages), the **XMLHttpRequest** object contains various **properties** and **methods**.
- ✓The main idea is that the properties are set after the object is created to **specify information to be sent to the server**, as well as how to **handle the response received from the server**.
- ✓Some properties **will be updated** to hold status information about whether the request finished **successfully**.
- ✓The **methods are used to send** the request to the server, **and to monitor the progress** of the request as it is executed (and to determine if it was completed successfully).

XMLHttpRequest Object Properties

```
readyState              //An integer from 0 to 4.
                        //(0 means the call is uninitialized, 4 means that the call is complete)
onreadystatechange      //Determines the function called when the objects readyState changes.
responseText            //Data returned from the server as a text string (read-only).
responseXML             //Data returned from the server as an XML document object(readonly).
status                  //HTTP status code returned by the server
statusText              //HTTP status phrase returned by the server
```

We use the **readyState** to determine when the request has been completed, and then check the status to see if it executed without an error.

XMLHttpRequest Object Properties

```
open('method', 'URL', asyn)  //Specifies the
                             //HTTP method to be used (GET or POST) as a string
                             //The target URL
                             //(asyn should be true or false, if omitted, true is assumed).
send(content)                //Sends the data for a POST request and starts the request, if GET
                             is used you should call send(null).
setRequestHeader('x','y')    //Sets a parameter and value pair x=y and assigns it to the header
                             to be sent with the request.
getAllResponseHeaders()      //Returns all headers as a string.
getResponseHeader(x)         //Returns header x as a string.
abort()
```

✓The **open** object method is used to set up the request,

✓The **send** method starts the request by sending it to the server (with data for the server if the POST method is used).

```
<script type="text/javascript">
// ***** include the getXMLHttpRequest function defined before
var ajaxRequest = getXMLHttpRequest();
if (ajaxRequest) { // if the object was created successfully
       ajaxRequest.onreadystatechange = ajaxResponse;
       ajaxRequest.open("GET", "search.aspx?query=Badr");
       ajaxRequest.send(null);
}
function ajaxResponse() { //This gets called when the readyState changes.
       if (ajaxRequest.readyState != 4) // check to see if we’re done
       { return; }
       else {
              if (ajaxRequest.status == 200) // check to see if successful
              { // process server data here. . . }
              else {
              alert("Request failed: " + ajaxRequest.statusText);
              }
       }
}
</script>
```

Fetch API in Jaavscript

JS

Javascript

Request
Response
Headers
Body

XmlHttpRequest

Fetch API

JQuery Ajax

Ajax Object Properties

```
$.ajax({
            url:"http://localhost:8080/loadData",
            method:"POST",
            contentType:"application/json",
            data:"{'name:'Anonymouse'}",
            dataType:"json",
            success:function(result){ /*on success */},
            error:function(error){/*on error */}
});
```

Page URL

Request Type

Type of data sended to the server

Data sended to the server

Type of data recievded from server

Function called when ajax success with result

Function called when ajax failed

Thank You ☺