JAVASCRIPT

AJAX and AJAJ

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

VIDEO

https://www.youtube.com/watch?v=RDo3hBL1rfA

AJAX – OVERVIEW

- Asynchronous JavaScript and XML (AJAX)

- Is a technique and not a technology or programming language

AJAX – SYNCHRONICITY

Events are "meaningful coincidences" if they occur with no causal relationship yet seem to be meaningfully related.

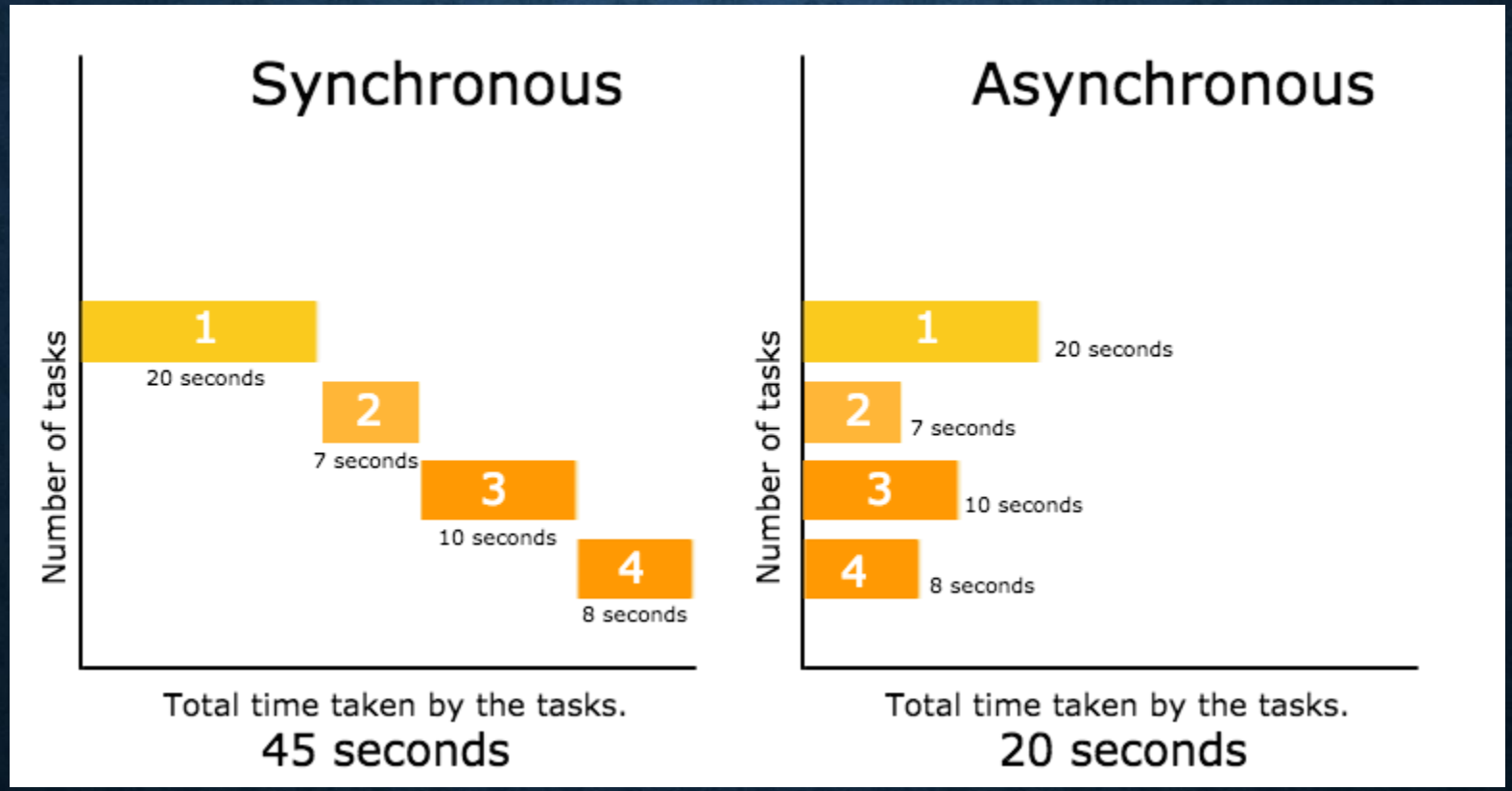
- Carl Jung (German Analytical Psychologist)

AJAX – SYNCHRONICITY

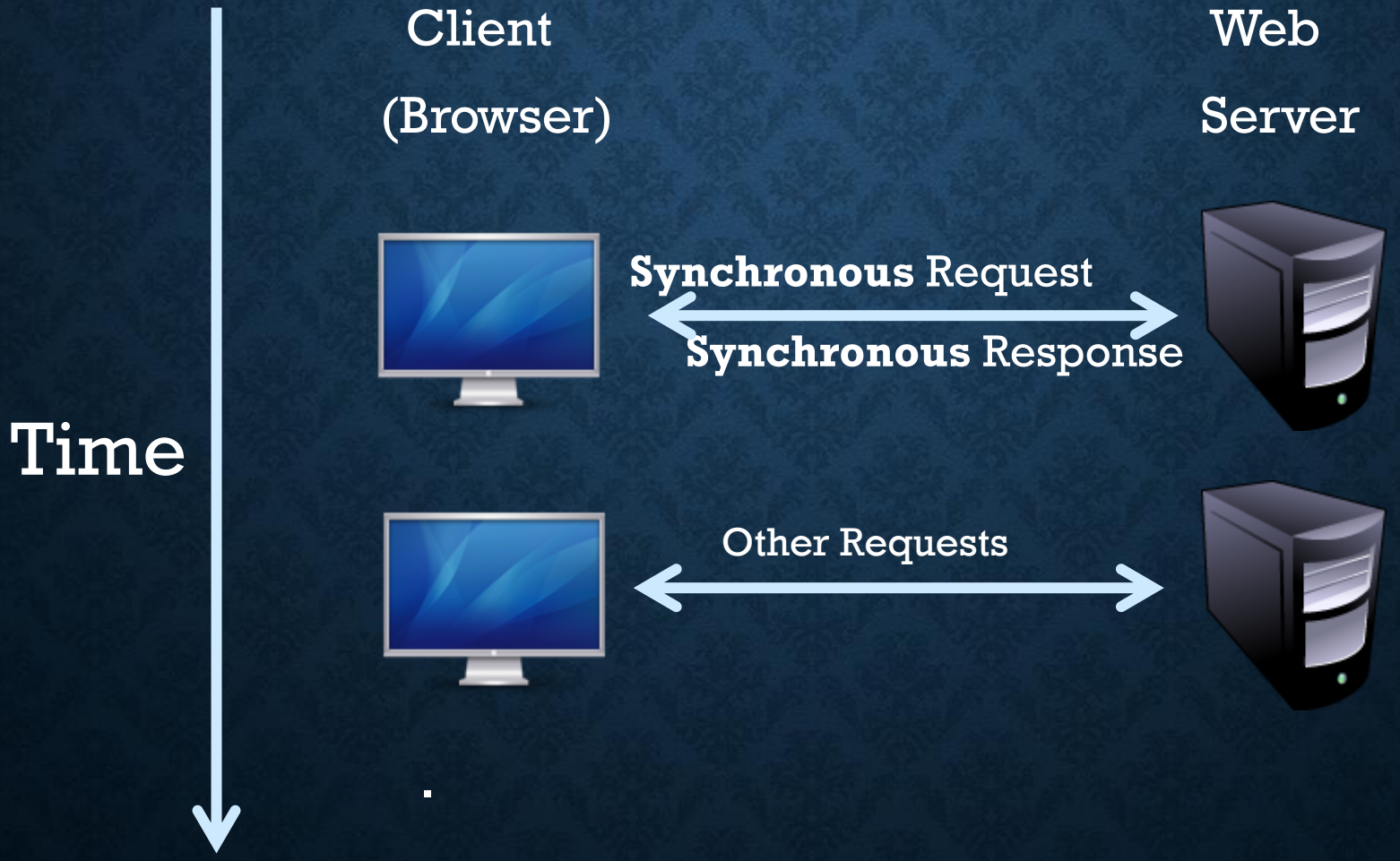
- Synchronous calls
  - Blocking
  - No parallel execution
  - Sends a request and waits/blocks until a reply comes back
  - No other code is executed while waiting for the reply

- Asynchronous calls
  - Non-blocking
  - Parallel execution
  - Sends a request and does not wait/block until a reply comes back
  - Continue executing other code while waiting for the reply

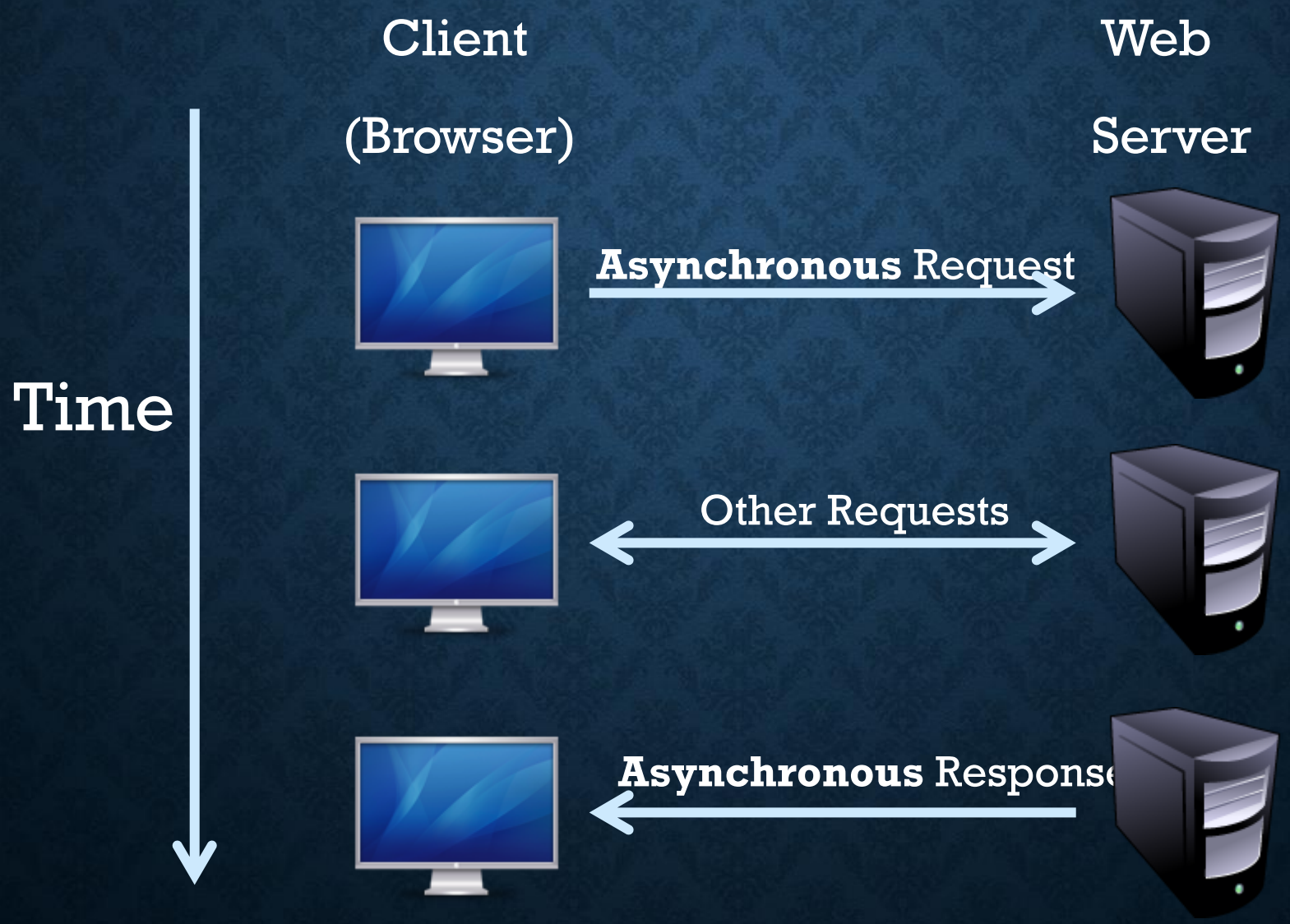
AJAX – SYNCHRONICITY



AJAX – SYNCHRONOUS CALLS



AJAX – ASYNCHRONOUS CALLS



AJAX – IFRAME

- iframe was the first asynchronous request mechanism in HTML
- While the iframe loads its content, the remainder of the website continues execution

- Browsers load all website content (HTML, JS, CSS, images, videos, etc) asynchronously
  - If a page is opened, all resources are requested in parallel
  - If resources would be requested sequentially, pages would load very slowly

AJAX – AJAJ

- Asynchronous calls can be used to request any kind of data
- AJAX was specifically created to retrieve XML asynchronously

- Most modern AJAX calls request JSON data, not XML
- Asynchronous JavaScript and JSON (AJAJ)
- Using JSON instead of XML saves bandwidth
- The AJAX term is often used to refer to AJAJ

AJAX – HISTORY

- In the 1990s if a page's content changed, the entire page had to be reloaded

- In 1996 the iframe was introduced in HTML
  - Pages inside the iframe could be loaded/refreshed asynchronously
  - Allowed to create static menus, headers, and footers that do not change
  - Only refreshable parts had to be put inside an iframe

AJAX – HISTORY

- In 1998 Microsoft implemented the first component as a client script (XMLHTTP)

- In 1999 Microsoft introduced it officially as ActiveX in Internet Explorer 5
- ActiveX was mostly restricted to Internet Explorer and Windows

AJAX – HISTORY

- Mozilla, Safari and Opera adopted the idea and used the JavaScript **XMLHttpRequest** object

- Microsoft adopted the same standard since Internet Explorer 7
  - Although ActiveX is still supported

AJAX – HISTORY

- In 2004/2005 Google developed cross-browser AJAX for Gmail and GoogleMaps
- The name AJAX was publicly stated in 2005 by Jesse James Garrett
- In 2006 W3C released a first draft of XMLHttpRequest to create an official web standard

AJAX – PROCESS

1. Browser creates request object
2. Browser sends request to server
3. Server processes request
4. Server sends response to browser
5. Browser processes response
6. Browser updates the webpage

AJAX – CREATE REQUEST

- Create a new XMLHttpRequect object

```
var req = new XMLHttpRequest();
```

AJAX – OPEN REQUEST

- Specify the parameters for the request
  1. HTTP request method (commonly POST or GET, other methods are also possible)
  2. URL to the requested resource
  3. If the request should be asynchronous (true) or synchronous (false)

```
req.open("GET", "http://ajax.com/api", true);
```

AJAX – HTTP METHOD

- When to use GET or POST
  - Depends on the API/server the request is sent to
  - Depends on which data is sent and if encryption is important

AJAX – HTTP METHOD

- GET
  - Slightly faster
  - No encryption of parameters (do not use for sensitive information) over HTTP
  - Allows only little data to be sent (4-8kb)
  - **Use for receiving data from the server**
- POST
  - Slightly slower
  - Allows encryption of data when SSL/TLS certificates are available
  - Allows to send larger amounts of data (still limited by server)
  - **Use for adding/updating data on the server**

AJAX – SEND REQUEST

- Send the request
  1. Specify optional POST parameters
     - Not for GET requests
     - GET parameters should be added to the URL when opening a request

```
req.send(); // GET or POST without parameters
// or
req.send("fname=John&lname=Lennon"); // POST with parameters
```

AJAX – HANDLE RESPONSE

- Handle state changes, errors, and responses from the server

```
req.onreadystatechange = function()
{
        if(req.readyState == 4 && req.status == 200)
        {
                var res = req.responseText;
                // Do something with the result
        }
}
```

AJAX – READY STATES

- Possible ready states
  - **0:** Request not initialized
  - **1:** Server connection established
  - **2:** Request received
  - **3:** Processing request
  - **4:** Request finished and response ready

AJAX – STATUS CODES

- **200:** OK (everything is fine)
- **400:** Bad request (eg: invalid syntax)
- **403:** Forbidden (eg: password-protected)
- **404:** Not found (eg: non-existing resource or invalid URL)
- **408:** Timeout (eg: response took too long)
- **500:** Internal server error (eg: generic error or server problem)
- **501:** Not implemented (eg: invalid method call)
- **503:** Service unavailable (eg: server unavailable)
- Many more

AJAX – EXAMPLE

```
var req = new XMLHttpRequest();
req.onreadystatechange = function()
{
        if(req.readyState == 4 && req.status == 200)
        {
                var res = req.responseText;
        }
};
req.open("GET", "resourceURL", true);
req.send();
```

AJAX – ORDER

- The order of calls with XMLHttpRequest matters
  1. Create the XMLHttpRequest object
  2. Set event handlers
  3. Open the request
  4. Send the request

AJAX – ACTIVEX

- Microsoft had their own AJAX implementation called ActiveX
  - Only needed for Internet Explorer 6 and prior
  - New versions of Internet Explorer and Edge use the official standard now
- Required to create a special object for Internet Explorer and another one for all other browsers

```
if(window.XMLHttpRequest) // code for modern browsers
{
        req = new XMLHttpRequest();
}
else // code for old Internet Explorer
{
        req= new ActiveXObject("Microsoft.XMLHTTP");
}
```

AJAX – JSON

- Data returned by an asynchronous call is a string
- Has to be parsed/converted depending on the return data type
- A JSON string can be parsed to a JS object

```
req.onreadystatechange = function()
{
        if(this.readyState == 4 && this.status == 200)
        {
                var object = JSON.parse(this.responseText);
                document.getElementById("entry").innerHTML = object.name;
        }
};
```

AJAX – EXAMPLE

- Assume that an REST API exists that returns the following JSON on a simple GET request

```
[
        {
                "name" : "Bitcoin",
                "price" : 10523
        },
        {
                "name" : "Ethereum",
                "price" : 862
        }
]
```

AJAX – EXAMPLE

- Furthermore, assume that we have the following empty HTML table

```
<!DOCTYPE html>
<html>
        <body>
                <table id="prices" border="1">
                        <tr>
                                <th>Name</th>
                                <th>Price</th>
                        </tr>
                </table>
        </body>
</html>
```

Name	Price

AJAX – EXAMPLE

```
var req = new XMLHttpRequest();
req.onreadystatechange = function()
{
        if(req.readyState == 4 && req.status == 200)
        {
                var values = JSON.parse(req.responseText);
                var table = document.getElementById("prices");
                for(var i = 0; i < values.length; ++i)
                {
                        table.innerHTML +=
                                "<tr><td>" + values[i].name + "</td><td>"
                                + parseInt(values[i].price) + "</td></tr>";
                }
        }
};
req.open("GET", "resourceURL", true);
req.send();
```

AJAX – EXAMPLE

- The table after the AJAX call successfully executed

Name	Price
Bitcoin	10523
Ethereum	862


JavaScript
JSON
AJAX