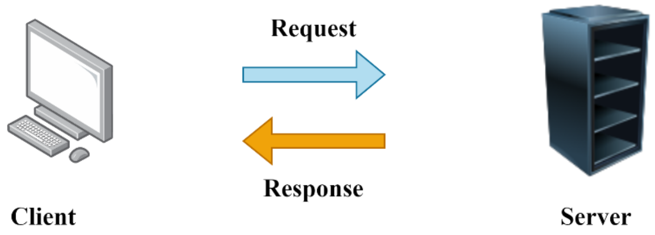
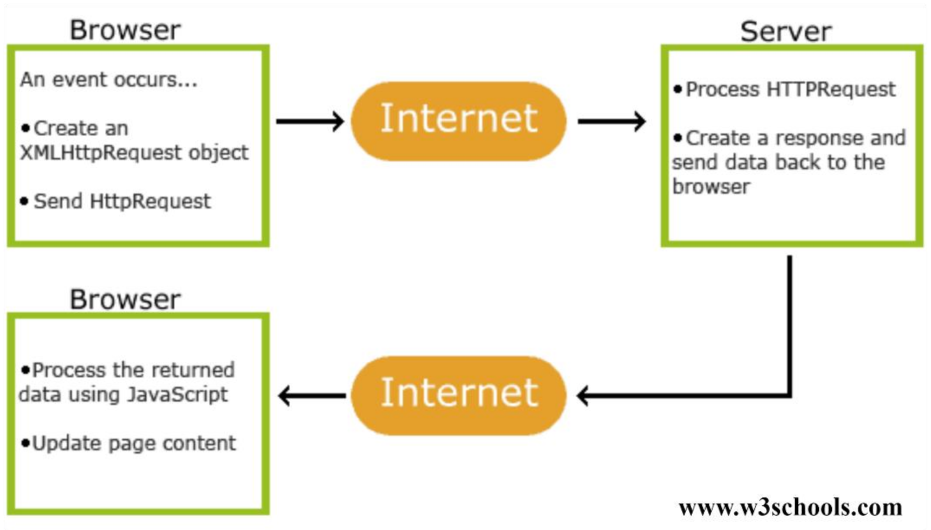
Ajax

- **Overview**
  - Standard web-Applications
    - Traditional server-based model



  - AJAX enables browser-based applications.
  - Static, empty HTML page as application shell
  - JavaScript requests data from web-service
  - Use JavaScript to create HTML content on the fly.
  - Parameters from URL or from events

- **Web applications and Ajax**
  - Ajax: Asynchronous JavaScript and XML.
  - Asynchronous: the exchange of the data to and from the server is done in the background (without refreshing the page)
  - It is an API in the form of an object.
  - It can be used with other protocols than HTTP.
  - It works with different data types: XML, JSON, and plain text.
  - It is not a programming language. It is a technique using JavaScript.
  - It avoids the "click-wait-refresh" pattern.
  - It allows dynamically updating a page without refreshing the browser.
  - How does Ajax work?



- **XMLHttpRequest**
  - XMLHttpRequest is a built-in browser object that allows making HTTP requests in JavaScript.
  - Despite having the word “XML” in its name, it can operate on any data, not only in XML format.
  - It can upload/download files, track progress, and much more.
  - It updates a web page without reloading the page.
  - It requests and receives data from a server after the page has loaded.
  - It sends data to a server in the background.
  - Right now, there’s another, more modern method fetch, that somewhat deprecates XMLHttpRequest.
  - In modern web development XMLHttpRequest is used for three reasons:
    - Historical reasons: we need to support existing scripts with XMLHttpRequest.
    - We need to support old.
    - Working with XMLHttpRequest:

```
// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onload = function() {
  // Here you can use the Data
}
or
xhttp.onreadystatechange = function(){
   // Callback function body
}

// Send a request
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

- XMLHttpRequest has two modes of operation:
  - synchronous
  - asynchronous.
- To do the request, we need 3 steps:
  1. Create XMLHttpRequest:

     ```
     let xhr = new XMLHttpRequest();
     ```

     Note that the constructor has no arguments.

  2. Initialize it, usually right after new XMLHttpRequest:

     ```
     xhr.open(method, URL, [async, user, password])
     ```

     Where

     Method:
     HTTP-method. Usually "GET" or "POST".
     URL:
     The URL to request, a string, can be URL object.
     Async:
     If it is set to false, then the request is synchronous.
     User, password:
     It is used for basic HTTP auth (if required).
  3. Send it out.

     ```
     xhr.send([body])
     ```

     It opens the connection and sends the request to the server.
     Body contains the request body.
     Get requests do not have a body.
     POST request use the body to send the data to the server.

- **XMLHttpRequest Object Properties**

Property	Description
onload	Defines a function to be called when the request is received (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes

readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

- **Example**
  - Example 1: Change the content of a page
    - Files:
      - uniport.jpg
      - genbank.jpg
      - omim.jpg
      - ajax_example_1.html
      - icons.html
      - ajax_info.txt

```
<!DOCTYPE html>
<html>

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

```
function loadDoc1() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo1").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.open("GET", "icons.html", true);
  xhttp.send();
}
</script>
<body>

<div id="demo" >
  <h2 >Let AJAX change this text</h2>
  <button type="button"  style="background-color: rgb(255, 234, 0);"
onclick="loadDoc()">Update The Page</button>
</div>
<div id="demo1">
    <h2>Another Update</h2>
    <button type="button" style="background-color: aqua;"   onclick="loadDoc1()">Update The
Page</button>
</div>

</body>
</html>
```

- Example 2: Explore UniProt database
  - Files:
    - mystyle.css
    - uniprot_2.json
    - ajax_proteins_app.js
    - ajax_uniprot_app.html

```
<html>
<head>
    <link rel="stylesheet" href="mystyle.css">
    <script  src="ajax_proteins_app.js"></script>
</head>
<body>

    <h1 align="center"> Proteins Details - Ajax Application</h1>
    <div align="center">
     <button type="button"  class="button" onclick="loadJSON();">
```

```
        Update Details
        </button>
        <div id="container"/>
    </div>

   </body>
</html>
```

- ajax_proteins_app.js

```
function loadJSON() {
   //The file to process
   var data_file = "uniprot_2.json";

   // Create an XMLHttpRequest object
   http_request = new XMLHttpRequest();

   //Callback function: Process the response
   http_request.onreadystatechange = function() {
      if (http_request.readyState == 4  && http_request.status == 200) {
       var table = "<table border=3 bgcolor=lighblue width=60% align=center>";
       table += "<tr class=th><th>Entry</th><th>Organism</th><th>Gene
Name</th><th>OMIM</tr>";

       // Javascript function JSON.parse to parse JSON data
       var jsonObj = JSON.parse(http_request.responseText);
       //check if everything is ok.
       console.log(jsonObj);
       let size = jsonObj.docs.length;
         // jsonObj variable now contains the data structure and can
         // be accessed as jsonObj keys.
       for (var i = 0; i < size; i++ ) {
           table +="<tr>";
           table += "<td>" +  jsonObj.docs[i].Entry; + "</td>";
           table += "<td>" +  jsonObj.docs[i]["Entry name"]; + "</td>";
           table += "<td>" +  jsonObj.docs[i]["Gene names"]; + "</td>";
           table += "<td>" +  jsonObj.docs[i].OMIM; + "</td>";
           table +="</tr>";
       }
       table += "</table>"
       document.getElementById('container').innerHTML = table;
      }
   }
   //open asynchronous since we specify true
   http_request.open("GET", data_file, true);
   //send request
   http_request.send();
}
```

```
<html>
<head>
    <link rel="stylesheet" href="mystyle.css">
    <script type = "application/javascript">
         function loadJSON() {
            //The file to process
            var data_file = "uniprot_2.json";

            // Create an XMLHttpRequest object
            http_request = new XMLHttpRequest();

            //Callback function: Process the response
            http_request.onreadystatechange = function() {
               if (http_request.readyState == 4  && http_request.status == 200) {
                var table = "<table border=3 bgcolor=lighblue width=60% align=center>";
                table += "<tr class=th><th>Entry</th><th>Organism</th><th>Gene
Name</th><th>OMIM</tr>";

                // Javascript function JSON.parse to parse JSON data
                var jsonObj = JSON.parse(http_request.responseText);
                //check if everything is ok.
                console.log(jsonObj);
                let size = jsonObj.docs.length;
                  // jsonObj variable now contains the data structure and can
                  // be accessed as jsonObj keys.
                for (var i = 0; i < size; i++ ) {
                    table +="<tr>";
                    table += "<td>" +  jsonObj.docs[i].Entry; + "</td>";
                    table += "<td>" +  jsonObj.docs[i]["Entry name"]; + "</td>";
                    table += "<td>" +  jsonObj.docs[i]["Gene names"]; + "</td>";
                    table += "<td>" +  jsonObj.docs[i].OMIM; + "</td>";
                    table +="</tr>";
                }
                table += "</table>"
                document.getElementById('container').innerHTML = table;
               }
            }
            //open asynchronous since we specify true
            http_request.open("GET", data_file, true);
            //send request
            http_request.send();
         }
    </script>

</head>
<body>

    <h1 align="center"> Proteins Details - Ajax Application</h1>
    <div align="center">
     <button type="button"  class="button" onclick="loadJSON();">
        Update Details
        </button>
        <div id="container"/>
    </div>

   </body>
</html>
```