

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

Actividad 02: Laboratorio 01.

Desarrollo de Aplicaciones en Red

Profesora: Belén Bermejo

Universidad Internacional de la Rioja

7 de enero de 2020

Tecnologías AJAX

José Antonio Gil Valencia
08862804F

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

Índice

<u>1-Definición de la Actividad.</u>	<u>3</u>
<u>2-Desarrollo de la Actividad.</u>	<u>4</u>
<u>2.1.-Ejercicios JavaScript</u>	<u>5</u>
2.1.1.-Ejercicio 01	6
2.1.2.-Ejercicio 02	7
2.1.3.-Ejercicio 03	8
2.1.4.-Ejercicio 04	10
<u>2.2.-Ejercicios AJAX</u>	<u>13</u>
2.2.1.-Ejercicio 01	15
2.2.2.-Ejercicio 02	16
2.2.3.-Ejercicio 03	17
2.2.4.-Ejercicio 04	19
2.2.5.-Ejercicio 05	19
<u>2.3.-Conclusión</u>	<u>20</u>
<u>3-Referencias. Bibliografía. Webgrafía.</u>	<u>21</u>

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

1.-Definición de la Actividad.

Laboratorio #1: Tecnologías AJAX

Descripción y pautas de elaboración

Para la realización de este laboratorio tendremos que explorar varios usos históricos de la tecnología AJAX.

Entrega

Deberemos de entregar un resumen narrado de todas las actividades llevadas a cabo en el laboratorio. Formato PDF o HTML.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

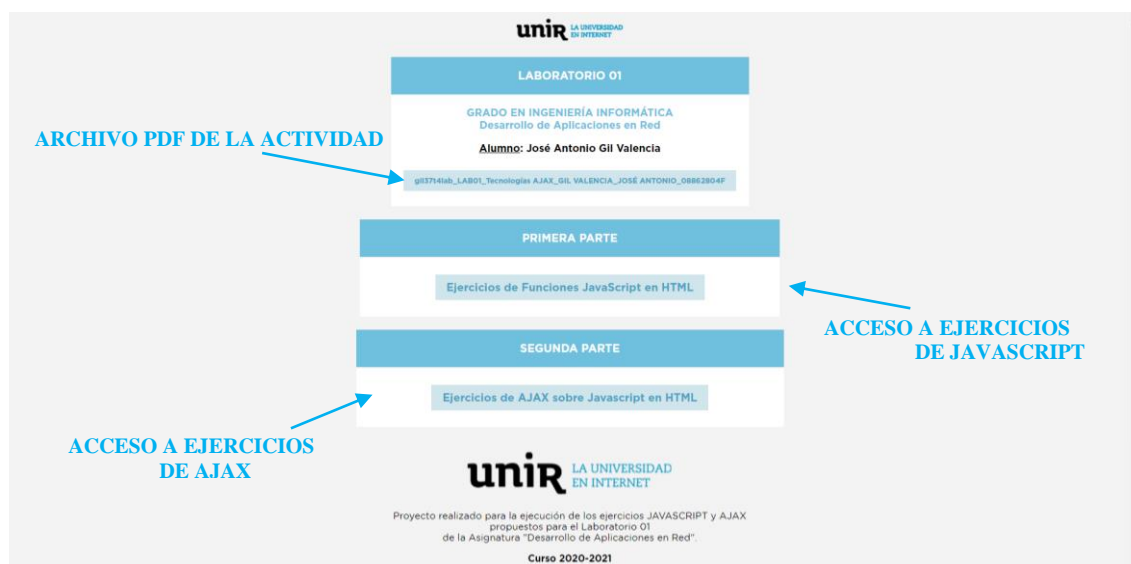
2.-Desarrollo de la Actividad.

A continuación, se muestran los diferentes **ejercicios del laboratorio sobre JavaScript y AJAX**. Aunque los resultados de dichos ejercicios se muestran en el proyecto web presentado en este laboratorio con **enlace en la siguiente página web** (https://gilvalencia.github.io/lab01_DAR/), el análisis y procedimiento de resolución de los ejercicios se presentan en los siguientes apartados.

Como se indicó en clase, desarrollé un proyecto web en *HTML* con una página principal *index.html* a partir de la cual se ofrecían otras **dos páginas con las resoluciones de los ejercicios: una para JavaScript y otra para AJAX**.

Ambas páginas cuentan con un fichero para los **documentos de estilo CSS**, y un fichero **js** para las correspondientes **funciones de JavaScript y AJAX** empleadas en el laboratorio.

Dicho proyecto lo subí a un **repositorio de GitHub**, siguiendo las mismas pautas realizadas para la Actividad 01 de la asignatura, con el fin de que la profesora pueda ver a través de un navegador el resultado del laboratorio. En dicho proyecto también se puede acceder a este documento *PDF*, en la cabecera de la propia página, como se puede ver en la siguiente captura del proyecto.



Aunque disponga del proyecto en este repositorio nombrado, **la carpeta del proyecto también va incluida en los archivos entregados en este trabajo**, en formato **.zip**, para su revisión.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

2.1.-Ejercicios JavaScript.

Para comenzar la resolución de los siguientes ejercicios, previamente realicé los ejercicios prácticos propuestos en el temario de la asignatura sobre el lenguaje *JavaScript*. El resultado de los siguientes ejercicios se muestra a continuación.

De forma general, en los 4 ejercicios, he tenido en cuenta el **contenido que introduce el usuario a través de la función prompt()** de JavaScript. Si no es el dato solicitado (números, cadenas de caracteres o palabras), la implementación de los ejercicios está hecha de tal forma que la propia función de cada ejercicio, **se llama recursivamente** solicitando al usuario el dato correcto. **Hasta que el usuario no introduce lo solicitado, el ejercicio no se ejecuta.**

```
if(numEspacios > 0){
    alert("Introduce una palabra, no una frase.");
    return ejercicio01();
}

else{
    alert(palindromo(texto));
}
```

En este condicional, si se solicita una palabra, para comprobar si es palíndromo, y se introducen espacios en el contenido del prompt(), quiere decir que no se trata de una palabra, pues hay espacios. En ese caso, se vuelve a solicitar al usuario la entrada de datos.

El contenido de estos ejercicios viene especificado tanto en el proyecto .zip entregado, como en la siguiente URL del proyecto web https://gilvalencia.github.io/labo1_DAR/parte01_javascript.html.



LA UNIVERSIDAD EN INTERNET

LABORATORIO 01

GRADO EN INGENIERÍA INFORMÁTICA
Desarrollo de Aplicaciones en Red

Alumno: José Antonio Gil Valencia

gitHub_LAB01_Tecnologías AJAX_DG_VALENCIA_JOSÉ ANTONIO_08828040

Volver a ÍNDICE Ejercicios AJAX

PARTE 01: EJERCICIOS JAVASCRIPT.		
EJERCICIO	ENUNCIADO	SOLUCIÓN
EJERCICIO 01	Detectar si la cadena de entrada es un palíndromo.	EJERCICIO 01: Palíndromo
EJERCICIO 02	Escribe un programa que pida dos números y escriba en la pantalla cuál es el mayor.	EJERCICIO 02: Mayor y menor
EJERCICIO 03	Escribe un programa que pida una frase y escriba las vocales que aparecen.	EJERCICIO 03: Vocales en cadena
EJERCICIO 04	Escribe un programa que pida una frase y escriba cuántas veces aparecen cada una de las vocales.	EJERCICIO 04: Recuento de vocales en cadena



LA UNIVERSIDAD EN INTERNET

Proyecto realizado para la ejecución de los ejercicios JAVASCRIPT y AJAX propuestos para el Laboratorio 01 de la Asignatura "Desarrollo de Aplicaciones en Red".

Curso 2020-2021

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

2.1.1.-Ejercicio 01.

-Enunciado: detectar si la cadena de entrada es un palíndromo.

-Código JavaScript: (src="js/code01.js").

```
function ejercicio01(){
    var texto=prompt("Introduce una palabra.");
    var longitud=texto.length;

    comprobar(texto);

    function comprobar(texto){
        var numEspacios = 0;
        for(i=0; i<longitud+1; i++){
            if(texto[i] == " "){
                numEspacios++;
            }
        }

        if(numEspacios > 0){
            alert("Introduce una palabra, no una frase.");
            return ejercicio01();
        }
        else{
            alert(palindromo(texto));
        }
    }

    function palindromo(texto){
        var mitad=Math.floor(longitud/2);
        for(i=0; i<mitad; i++){
            if(texto[i] !== texto[longitud - 1 - i]){
                return "La cadena \""+texto+"\" no es un palindromo."
            }
        }

        return "La cadena \""+texto+"\" es un palindromo."
    }
}
```

-Solución:

- Se solicita al usuario que introduzca una palabra, que se almacena en la variable texto, a partir del método prompt().
- La variable longitud mide el tamaño de la cadena introducida por el usuario.
- Se usa la función comprobar() para chequear que el dato introducido por el usuario es el correcto (que no tenga espacios).
- Si se trata de una frase en vez de una palabra (pues tendrá espacios), la propia función devuelve un alert() para comunicar al usuario que introduzca una palabra, no una frase, y llama recursivamente a la propia función ejercicio01(), para comenzar de nuevo a ejecutarla.
- Una vez que el usuario ha introducido el dato correctamente, la función palindromo() analiza todos los caracteres de la cadena a través del bucle for, comprobando en la condición del condicional if, si la primera mitad de la

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

palabra coincide con la segunda mitad de la misma en cuanto a contenido de caracteres se refiere.

-Tanto si coincide como si no, el método `alert()` devuelve el resultado por pantalla.

2.1.2.-Ejercicio 02.

-Enunciado: escribe un programa que pida dos números y escriba en la pantalla cuál es el mayor.

-Código JavaScript: (`src="js/code01.js"`).

```
function ejercicio02(){
    var numero1 = comprobarEntrada();
    var numero2 = comprobarEntrada();

    alert(comprobante(numero1, numero2));

    function comprobarEntrada(){
        var valorAceptado = /^[0-9]+$/;
        var numero=prompt("Introduce un n\xfamero.");

        if(numero.match(valorAceptado)){
            return numero;
        }
        else{
            alert("Introduzca un n\xfamero v\xellido, por favor.");
            return comprobarEntrada();
        }
    }

    function comprobante(numero1, numero2){
        if(numero1 > numero2){
            return "El n\xfamero \""+numero1+"\" es mayor que \""+numero2+"\"";
        }
        else if(numero2 > numero1){
            return "El n\xfamero \""+numero2+"\" es mayor que \""+numero1+"\"";
        }
        else{
            return "Los n\xfameros \""+numero1+"\" y \""+numero1+"\" son iguales";
        }
    }
}
```

-Solución:

-Lo primero que realiza el programa es ejecutar la función `comprobarEntrada()`. El resultado que devuelva dicha función, será el contenido de las dos variables `numero1` y `numero2`.

-La función `comprobarEntrada()`, solicita al usuario que introduzca un número. Y dicho dato, lo compara con el contenido de la variable `valorAceptado` dentro del método `.match()`, para comprobar que contiene números y no letras ni espacios.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

-En caso de que contenga números, la función devuelve el número introducido por el usuario y dicho dato lo devuelve a la variable numero1 o numero2. En caso contrario, se llamará recursivamente a la función comprobarEntrada() hasta que el dato sea correcto.

-Una vez son correctos los datos introducidos por el usuario, y las dos variables numero1 y numero2 tienen su contenido, se ejecuta mediante el método alert() la función comprobante(). Esta función compara ambos números (variables), y comprueba cuál de los dos es el mayor o si ambos son iguales.

-Lo que devuelva esta función, será el contenido del aviso alert() que aparezca por pantalla.

2.1.3.-Ejercicio 03.

-Enunciado: escribe un programa que pida una frase y escriba las vocales que aparecen.

-Código JavaScript: (src="js/code01.js").

```
function ejercicio03(){
    var cadena=prompt("Introduce una frase.");
    comprobar(cadena);
    function comprobar(cadena){
        var longitud = cadena.length;
        var numeroVocales = 0;
        var numeroEspacios = 0;
        var numeroConsonantes = 0;

        for(i=0; i<longitud+1; i++){
            if(cadena[i] == "a" || cadena[i] == "A"){
                numeroVocales++;
            }
            else if(cadena[i] == "\xe1" || cadena[i] == "\xc1"){
                numeroVocales++;
            }
            else if(cadena[i] == "e" || cadena[i] == "E"){
                numeroVocales++;
            }
            else if(cadena[i] == "\xe9" || cadena[i] == "\xc9"){
                numeroVocales++;
            }
            else if(cadena[i] == "i" || cadena[i] == "I"){
                numeroVocales++;
            }
            else if(cadena[i] == "\xed" || cadena[i] == "\xcd"){
                numeroVocales++;
            }
            else if(cadena[i] == "o" || cadena[i] == "O"){
                numeroVocales++;
            }
            else if(cadena[i] == "\xf3" || cadena[i] == "\xd3"){
                numeroVocales++;
            }
        }
    }
}
```


Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

```

    }
    else if(cadena[i] == "u" || cadena[i] == "U"){
        numeroVocales++;
    }
    else if(cadena[i] == "\xfa" || cadena[i] == "\xda"){
        numeroVocales++;
    }
    else if(cadena[i] == " "){
        numeroEspacios++;
    }
    else{
        numeroConsonantes++;
    }
}

if(numeroEspacios == 0){
    alert("Introduce una frase, no una palabra.");
    return ejercicio03();
}
else{
    alert(resultado(numeroVocales));
}
}

function resultado(numero){
    if(numero == 0){
        return "La cadena introducida tiene 0 vocales.";
    }
    else if(numero == 1){
        return "La cadena introducida tiene 1 vocal.";
    }
    else{
        return "La cadena introducida tiene "+numero+" vocales.";
    }
}
}

```

-Solución:

- Se solicita al usuario que introduzca una cadena de caracteres mediante el método `prompt()`, cuyo resultado se asigna a la variable `cadena`.
- Una vez introducido, se comprueba si el dato es correcto, mediante la función `comprobar()`.
- Para llevar a cabo la implementación de la comprobación, como se indicó en una de las clases de la asignatura, he tenido en cuenta la importancia de tener en cuenta la mayoría de posibilidades de entrada (vocales con y sin acento, espacios y consonantes).
- Se comprueba la longitud de la cadena introducida (variable `longitud`, mediante método `.length`), para usar dicho dato en el bucle `for` que recorrerá todos los caracteres, comprobando en cada uno si se trata de vocales, espacios o consonantes.
- Una vez analizado, si el número de espacios es igual a 0, quiere decir que lo introducido es una palabra, no una frase. Y así se le notifica al usuario por

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

pantalla, llamando recursivamente a la función `ejercicio03()` para comenzar de nuevo el ejercicio.

-Una vez los datos son los correctos, se ejecuta la función `resultado()`, introduciendo el contenido de la variable `numeroVocales`. En función de si es 0, 1 o varios, la función devuelve una frase diferente dentro del método `alert()`.

2.1.4.-Ejercicio 04.

-Enunciado: escribe un programa que pida una frase y escriba cuántas veces aparecen cada una de las vocales.

-Código JavaScript: (`src="js/code01.js"`).

```
function ejercicio04(){
    var cadena=prompt("Introduce una frase.");

    var longitud=cadena.length;
    var vecesA = 0;
    var vecesE = 0;
    var vecesI = 0;
    var vecesO = 0;
    var vecesU = 0;
    var vecesEspacio = 0;
    var vecesConsonante = 0;

    for(i=0; i<longitud+1; i++){
        if(cadena[i] == "a" || cadena[i] == "A"){
            vecesA++;
        }
        else if(cadena[i] == "\xe1" || cadena[i] == "\xc1"){
            vecesA++;
        }
        else if(cadena[i] == "e" || cadena[i] == "E"){
            vecesE++;
        }
        else if(cadena[i] == "\xe9" || cadena[i] == "\xc9"){
            vecesE++;
        }
        else if(cadena[i] == "i" || cadena[i] == "I"){
            vecesI++;
        }
        else if(cadena[i] == "\xed" || cadena[i] == "\xcd"){
            vecesI++;
        }
    }
}
```

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

```

        else if(cadena[i] == "o" || cadena[i] == "O"){
            vecesO++;
        }
        else if(cadena[i] == "\xf3" || cadena[i] == "\xd3"){
            vecesO++;
        }
        else if(cadena[i] == "u" || cadena[i] == "U"){
            vecesU++;
        }
        else if(cadena[i] == "\xfa" || cadena[i] == "\xda"){
            vecesU++;
        }
        else if(cadena[i] == " "){
            vecesEspacio++;
        }
        else{
            vecesConsonante++;
        }
    }

    if(vecesEspacio == 0){
        alert("Introduce una frase. No una palabra.");
        return ejercicio04();
    }
    else{
        alert("La frase introducida tiene:"+"\n->" + vecesASalida() + "\n->" + vecesESalida() + "\n->" + vecesISalida() + "\n->" + vecesOSalida() + "\n->" + vecesUSalida());
    }

    function vecesASalida(){
        if(vecesA == 1){
            return "Una letra 'A'.";
        }
        else if(vecesA > 1){
            return vecesA + " veces la letra 'A'.";
        }
        else{
            return "Ninguna letra 'A'.";
        }
    }

    function vecesESalida(){
        if(vecesE == 1){
            return "Una letra 'E'.";
        }
        else if(vecesE > 1){
            return vecesE + " veces la letra 'E'.";
        }
    }

```

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

```

        else{
            return "Ninguna letra 'E'.";
        }
    }
    function vecesISalida(){
        if(vecesI == 1){
            return "Una letra 'I'.";
        }
        else if(vecesI > 1){
            return vecesI+" veces la letra 'I'.";
        }
        else{
            return "Ninguna letra 'I'.";
        }
    }
    function vecesOSalida(){
        if(vecesO == 1){
            return "Una letra 'O'.";
        }
        else if(vecesO > 1){
            return vecesO+" veces la letra 'O'.";
        }
        else{
            return "Ninguna letra 'O'.";
        }
    }
    function vecesUSalida(){
        if(vecesU == 1){
            return "Una letra 'U'.";
        }
        else if(vecesU > 1){
            return vecesU+" veces la letra 'U'.";
        }
        else{
            return "Ninguna letra 'U'.";
        }
    }
}

```

-Solución:

- En primer lugar, se solicita al usuario que introduzca una frase mediante el método `prompt()`, cuyo contenido quedará almacenado en la variable `cadena`.
- Una vez introducido, se inicializan las variables `longitud` (cuyo contenido es el número de caracteres introducidos por el usuario, mediante el método `.length`),

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

vecesA, vecesE, vecesI, vecesO, vecesU, vecesEspacio y vecesConsonante.

Estas últimas, recogerán el número de caracteres para cada caso.

-A continuación, un bucle for analiza la cadena introducida por el usuario (repitiendo hasta el valor de la variable longitud-1). En función de la condición if, se va incrementando el valor de las variables previamente inicializadas. Aquí se ha tenido en cuenta que las vocales sean mayúsculas, minúsculas y los acentos en las mismas.

-En el caso de que ninguna de las condiciones se cumpla, por defecto (else) se incrementa el dato en la variable vecesConsonante.

-Siguiendo la ejecución lineal del programa, el siguiente condicional if comprueba que el dato introducido por el usuario (variable cadena) contenga espacios. Para ello, la primera condición es que, si es igual a 0 el valor de la variable vecesEspacio, se informa al usuario de que el dato introducido no es una frase (método alert()), y se llama recursivamente a la función ejercicio04() hasta que inserte el dato correcto.

-Si se cumple la condición, se ejecutan las cinco funciones vecesSalida() dentro del alert(), informando al usuario del número de veces que aparece cada vocal.

-Las funciones vecesSalida(), las he realizado para tener en cuenta el valor del dato. En el caso de que sea 0, 1 o varios, la frase que devuelve es diferente.

-La comprobación en esta función ejercicio04(), la he podido realizar mediante funciones separadas. Pero, para realizarlo de forma diferente al resto de ejercicios, la he implementado de forma lineal dentro del ejercicio.

-De forma general, dentro de los archivos con extensión .js, **el uso de las tildes se ha llevado a cabo a través del código ASCII (Blog Unijimpe, 2008).**

2.2.-Ejercicios AJAX.

Lo primero a tener en cuenta en los ejercicios de AJAX propuestos, es haber realizado los ejercicios del tutorial propuesto en la asignatura (W3Schools, 2020). Sin estos ejercicios previos, hubiera sido casi imposible haber realizado la implementación de estos ejercicios.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

El contenido de estos ejercicios viene especificado tanto en el proyecto .zip entregado, como en la siguiente URL del proyecto web https://gilvalencia.github.io/labo1_DAR/parte02_ajax.html

Los ejercicios de AJAX, aunque vienen definidos por separado, se han realizado sobre un documento *javascript* (*code02.js*), en el que todos los apartados están interrelacionados. El resultado de un ejercicio influye en el siguiente. De tal forma que, aunque se habla en esta resolución de los ejercicios por separado, el resultado es genérico para los 5 enunciados.

Dentro del código, hay que explicar el uso del método *prototype* y la función *unescape()* en *JavaScript*. El código sería el que se muestra a continuación:

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

```
String.prototype.transformaCaracteresEspeciales = function() {
    return unescape(escape(this).
        replace(/%0A/g, '<br/>').
        replace(/%3C/g, '&lt;').
        replace(/%3E/g, '&gt;'));
}
```

Debido a que el contenido que recoge el código en este ejercicio es contenido HTML de la URL, se usa este código para no perder los datos HTML en javascript. La función `unescape()` decodifica una cadena codificada que procede de un contenido HTML. De ahí que se use para decodificar el contenido de la URL añadida en el ejercicio que recibe el contenido en HTML (*W3Schools, 2020 / TJ Crowder, 2018*).

2.2.1.-Ejercicio 01.

-Enunciado: al cargar la página, el cuadro de texto debe mostrar por defecto la URL de la propia página.

-Código JavaScript: (`src="js/code02.js"`).

```
window.onload = function() {
    var recurso = document.getElementById('recurso');
    recurso.value = location.href;

    document.getElementById('cargarContenido').onclick = cargaContenido;
}
```

-Solución:

- La directiva `window.onload` captura un evento. En este caso, cuando la página se carga, se ejecuta la función correspondiente.
- Dentro de esta función, se declara la variable `recurso`, la cual se inicializa al valor que posea el elemento HTML `recurso`. Esta referencia HTML se consigue gracias a la propiedad `document.getElementById`, indicando entre los parámetros el nombre de la etiqueta del elemento HTML (`recurso`).
- El valor de dicha variable `recurso` inicializada obtiene el valor de la URL completa en la que nos encontramos, gracias a la propiedad `location.href` de *javascript*.
- Dentro de esta función, cuando el usuario haga click (se produce otro evento `.onclick`) sobre el elemento `cargaContenido` de HTML, se ejecuta la función `cargaContenido()`, correspondiente ya al ejercicio 02 y 03 de carga y muestra de contenido.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

2.2.2.-Ejercicio 02.

-Enunciado: al pulsar el botón "Mostrar Contenidos", se debe descargar mediante peticiones AJAX el contenido correspondiente a la URL introducida por el usuario. El contenido de la respuesta recibida del servidor se debe mostrar en la zona de "Contenidos del archivo".

-Código JavaScript: (`src="js/code02.js"`).

```
function cargaContenido() {
    document.getElementById('contenidos').innerHTML = "";
    document.getElementById('estados').innerHTML = "";

    if(window.XMLHttpRequest) {
        petition = new XMLHttpRequest();
    }
    else {
        petition = new ActiveXObject("Microsoft.XMLHTTP");
    }

    petition.onreadystatechange = muestraContenido;

    tiempoInicial = new Date();

    var recurso = document.getElementById('recurso').value;

    petition.open('GET', recurso+'?nocache='+Math.random(), true);
    petition.send(null);
}
```

-Solución:

-Dicha etiqueta cargarContenido del ejercicio 01, está atribuida al botón "MostrarContenidos", ejecutando la función cargaContenido().

-Las dos primeras operativas .innerHTML, permiten que, antes de mostrar el contenido, cuando la página se ha cargado, por defecto aparezca el contenido de contenidos y estados vacío, limpio. Esto se debe a que permite reemplazar la sintaxis HTML del elemento por la nueva (MDN Web Docs, 2020), gracias a que se iguala a 0 caracteres.

-Una vez se tiene limpio, la gran utilidad de AJAX, y lo que solicita el ejercicio, es poder leer los datos de un servidor web y actualizar la página (leer una nueva URL y mostrar su contenido) si tener que recargar la página.

-Lo primero de todo, para hacer la actividad, llegados a este punto, tuve que crear un servidor interno en Windows 10, ya que el problema del CORPS en el navegador impedía leer correctamente el contenido.

-Para comenzar, se tiene que inicializar el objeto XMLHttpRequest sobre el que queremos realizar la consulta de datos. El condicional if se introduce para el caso de que el navegador sobre el que se ejecuta el programa no sea muy

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

moderno. Si el navegador es moderno, la variable `peticion` será un objeto `XMLHttpRequest()`. En caso contrario, será un objeto `ActiveXObject()`.

-Una vez instanciado, se implementa la respuesta sobre la variable `peticion`, con la propiedad `onreadystatechange`, igualando al contenido obtenido por la función `muestraContenido()`. Mostrará el contenido cuando el estado del objeto cambie.

-Se realiza la petición tomando como URL el contenido de la variable `recurso`. Se inicializa la variable tiempo `timeStart` al momento exacto en el que se usa el método `Date()`, para tomar el tiempo de inicio de la petición.

-El método `open` especifica la solicitud de la petición, incluyendo GET (tipo de la solicitud), la URL completa (dentro de la variable `recurso`) y se selecciona si es asíncrona la respuesta de la solicitud (`true`).

-Como usamos la opción GET en el tipo de la solicitud, el método `send` de envío de la petición al servidor no incluye ningún string en el parámetro (entre paréntesis). Simplemente se indica `null` o vacío.

2.2.3.-Ejercicio 03.

-Enunciado: en la zona "Estados de la petición" se debe mostrar en todo momento el estado en el que se encuentra la petición (No inicializada, cargando, completada, etc.).

-Código JavaScript: (`src="js/code02.js"`).

```
function muestraContenido() {
    var tiempoFinal = new Date();
    var milisegundos = tiempoFinal - tiempoInicial;

    var estados = document.getElementById('estados');
    estados.innerHTML += "[" + milisegundos + " mseg." + " +
        + estadosPosibles[peticion.readyState] + "<br/>";

    if(peticion.readyState == 4) {
        if(peticion.status == 200) {
            var contenidos = document.getElementById('contenidos');
            contenidos.innerHTML=
                =peticion.responseText.transformaCaracteresEspeciales();
        }
        muestraCabeceras();
        muestraCodigoEstado();
    }
}
```

-Solución:

-La función `muestraContenido()` que se ejecuta dentro de la función `cargaContenido()`, pretende mostrar el contenido de los estados de la

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

petición, el contenido HTML de la propia página y, a su vez, ejecutar las funciones `muestraCabeceras()` y `muestraCodigoEstado()` que nos darán la información de los códigos de estado de la petición (200, 403 o 404) y las cabeceras (método `getAllResponseHeaders()`).

-Para comenzar a desglosar la función de respuesta (`muestraContenido()`), hay que tener en cuenta que el contenido de la página se tiene que ver en el elemento `contenidos` de HTML. Y la información sobre los estados de la petición (`readyState`) tiene que ir en el elemento `estados` del HTML.

-La variable `estados` se inicializa con el elemento devuelto del `getElementById` del HTML (`estados`), y su valor se va modificando con el uso del método `innerHTML`, obteniendo un valor diferente en función del valor del estado `readyState` sobre el recurso `peticion`. Los valores de `estadosPosibles` vienen definidos en matriz en las diferentes posiciones de los estados (0 = “No inicializado”; 1 = “Conexión al Servidor”; 2 = “Solicitud recibida”; 3 = “Procesando”; 4 = “Respuesta finalizada”) (*w3Schools*, 2020).

-Dentro de los estados, se incluye el tiempo de ejecución de cada estado, gracias a la variable `ms`, que resta el valor de la variable `timeEnd` menos el valor de la variable `timeStart`.

-Una vez implementada la salida de los estados en el recurso “estados”, se implementa la muestra del contenido de la URL en el elemento “contenidos”. Si el estado de la solicitud es igual a 4 (Respuesta finalizada) y el status es igual a 200 (sólo en este caso), se indica que el valor del elemento HTML `contenidos`, modifique su contenido a través del método `innerHTML` obteniendo de la `peticion` el contenido de texto de la respuesta del servidor. Aquí es donde se usaría la transformación de caracteres especiales de HTML del servidor adaptándolos a *javascript* (se habló de esta función al inicio del apartado AJAX de este documento).

-Una vez que se tiene el contenido de la URL en el elemento `contenidos` y `estados`, necesitamos ejecutar las funciones `muestraCabeceras()` y `muestraCodigoEstado()` de los ejercicios 04 y 05.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

2.2.4.-Ejercicio 04.

-Enunciado: mostrar el contenido de todas las cabeceras de la respuesta del servidor en la zona "Cabeceras HTTP de la respuesta del servidor".

-Código JavaScript: (**src="js/code02.js"**).

```
function muestraCabeceras() {
    var cabeceras = document.getElementById('cabeceras');
    cabeceras.innerHTML=peticion.getAllResponseHeaders().transformaCaracteresEspeciales();
}
```

-Solución:

- La propiedad `getAllResponseHeaders()` permite obtener el nombre de las cabeceras del servidor introducido. Igual que en la función `muestraContenido()`, los caracteres HTML se tienen que adaptar a javascript con la función `transformaCaracteresEspeciales()`.
- El contenido de dicha variable `cabeceras`, se iguala al elemento `cabeceras` del documento HTML.

2.2.3.-Ejercicio 05.

-Enunciado: mostrar el código y texto de estado de la respuesta del servidor en la zona "Código de estado".

-Código JavaScript: (**src="js/code02.js"**).

```
function muestraCodigoEstado() {
    var codigo = document.getElementById('codigo');
    codigo.innerHTML = peticion.status + "<br/>" + peticion.statusText;
}
```

-Solución:

- Como queremos el status del objeto `XMLHttpRequest` `peticion`, usamos solamente la propiedad `status` y `statusText` de la petición, para que nos devuelva el número (codigo) del status y el nombre del status.
- Esto se atribuye con el `getElementById` al elemento HTML `codigo`, para que dicha información aparezca dentro de dicho elemento.

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	


2.3.-Conclusión.

La actividad no es que haya sido de las más sencillas desde que llevo en esta universidad. Pero ha sido fructífera, en el sentido de que he aprendido bastante. No me lo he tomado como una actividad en la que me centre solamente en lo que solicita el profesor de la asignatura. He intentado aunar todas las tecnologías vistas hasta el momento en la asignatura. De ahí que haya presentado el proyecto dentro de un repositorio (proyecto web), con los correspondientes archivos de HTML, CSS y JS.

Algunas partes del laboratorio han sido bastante tediosas. Sobre todo, las relacionadas con AJAX. El tutorial sobre AJAX que se nos ofreció en la plataforma, permitía conocer la tecnología pero sin comentar la necesidad de crear un servidor, por ejemplo. Esto implicó dedicar más tiempo y recursos (Sheldon, 2020) para el aprendizaje de creación de un servidor propio en el PC. Es algo que se debería de haber comentado. Aunque nunca viene mal aprender cosas nuevas y de forma autodidacta a través de la web.

Por otro lado, las diferentes propiedades y métodos de AJAX vienen especificados en dicho tutorial, aunque de forma sencilla. La resolución de los ejercicios la tuve que ir desarrollando a medida que iba encontrando tutoriales y definiciones de los métodos necesarios para dar solución a los ejercicios. Un ejemplo es el método que nos devuelve los nombres de la cabecera de la URL visitada en la petición.

Por lo demás, he intentado que la presentación del laboratorio sea muy compacta. De ahí que todo lo que se pide en el enunciado del laboratorio lo haya presentado dentro del proyecto web almacenado en el repositorio GitHub (https://gilvalencia.github.io/lab01_DAR/index.html). Incluido este propio documento, con el fin de que la profesora pueda ir chequeando en el mismo navegador la resolución de ejercicios, código y apariencia.

	gilvalencia Delete main.css	3d602bc 13 minutes ago	🕒 52 commits
📁 archive	Delete main.css	2 hours ago	
📁 css	Delete main.css	13 minutes ago	
📁 images	Delete unir2.jpg	1 hour ago	
📁 js	Delete texto_prueba.txt	1 hour ago	
📄 index.html	Add files via upload	1 hour ago	
📄 parte01_javascript.html	Add files via upload	1 hour ago	
📄 parte02_ajax.html	Add files via upload	1 hour ago	

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

3.-Referencias. Bibliografía. Webgrafía.

3.1.-General Actividad.

Sheldon (diciembre de 2020). **TecKangaroo.com**. *Habilitar Servidor en Windows 10*. Recuperado el 12 de diciembre de 2020 de <https://teckangaroo.com/enable-iis-windows10/#:~:text=%20How%20to%20Check%20Internet%20Information%20Services%20%28IIS%29,Internet%20Information%20Services%20manager%20dialog%20box...%20More%20>

3.2.-JavaScript.

Blog Unijimpe (16 de mayo de 2008). **blog.unijimpe.net**. *Tildes en JavaScript*. Recuperado el 14 de diciembre de 2020 de <http://blog.unijimpe.net/tildes-en-javascript/>

W3Schools (2020). **w3schools.com**. *JavaScript Location href Property*. Recuperado el 14 de diciembre de 2020 de https://www.w3schools.com/jsref/prop_loc_href.asp

W3Schools (2020). **w3schools.com**. *JavaScript Input Text value Property*. Recuperado el 14 de diciembre de 2020 de https://www.w3schools.com/jsref/prop_text_value.asp

MDN Web Docs (2020). **developer.mozilla.com**. *JavaScript Document.getElementById*. Recuperado el 14 de diciembre de 2020 de <https://developer.mozilla.org/es/docs/Web/API/Document/getElementById>

MDN Web Docs (2020). **developer.mozilla.com**. *JavaScript element.innerHTML*. Recuperado el 14 de diciembre de 2020 de <https://developer.mozilla.org/es/docs/Web/API/Element/innerHTML>

W3Schools (2020). **w3schools.com**. *JavaScript unescape() Function*. Recuperado el 14 de diciembre de 2020 de https://www.w3schools.com/jsref/jsref_unescape.asp

TJ Crowder (junio 2018). **StackOverFlow.com**. *JavaScript para reemplazar% oA para
 elementos html*. Recuperado el 14 de diciembre de 2020 de

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones en Red	Apellidos: Gil Valencia	7 de enero de 2021
	Nombre: José Antonio	

<https://stackoverflow.com/questions/50820835/javascript-to-replace-oa-for-br-html-elements>

3.3.-AJAX.

W3Schools (2020). *w3schools.com. AJAX Introduction*. Recuperado el 12 de diciembre de 2020 de https://www.w3schools.com/js/js_ajax_intro.asp

FrameWork Television (26 de julio de 2011). *Youtube.com. Simple Ajax Tutorial*. Recuperado el 12 de diciembre de 2020 de <https://www.youtube.com/watch?v=qqRiDlm-SnY>