



UNIVERSIDAD
CAECE

Clase 3

Práctica guiada nro. 2

Arquitectura Web

Dr. Miguel Méndez Garabetti, Ing. Eduardo Piray

Contenido

1. Introducción al desarrollo web	3
2. Hoja de ruta de la clase 3	4
3. JavaScript	4
4. Practicas guiadas básicas para HTML, CSS y JavaScript: Potenciando nuestro sitio web de ejemplo	5
4.1. Primera parte	7
4.2. Segunda parte	9
5. Desplegar la aplicación en Netlify	14
6. Referencias	19

1. Introducción al desarrollo web

Para introducirse en el desarrollo web es necesario adquirir y formar bases de conocimientos. Estas bases se centran en aprender tres campos del conocimiento en el siguiente orden:

1. **HTML**, y como usarlo para maquetar sitios web. Es la tecnología fundamental que se utiliza para definir la estructura de una página web. HTML se utiliza para especificar si el contenido de una web se debe reconocer como un párrafo, lista, encabezado, enlace, imagen, reproductor multimedia, formulario o de uno de los tantos elementos disponibles o incluso un nuevo elemento que se defina.
2. **CSS**, y cómo usarlo para estilizar HTML (por ejemplo, modificar el tamaño del texto y los tipos de letra utilizados, agregar bordes y sombras, modelar tu página con varias columnas, agregar animaciones y otros efectos visuales).
3. **JavaScript**, y cómo usarlo para agregar funcionalidad dinámica a las páginas web (por ejemplo, encontrar tu ubicación y trazarla en un mapa, hacer que los elementos de la Interfaz de Usuario (IU) aparezcan/desaparezcan cuando se alterna un botón, guardar los datos del usuario localmente en su computadora y mucho más).

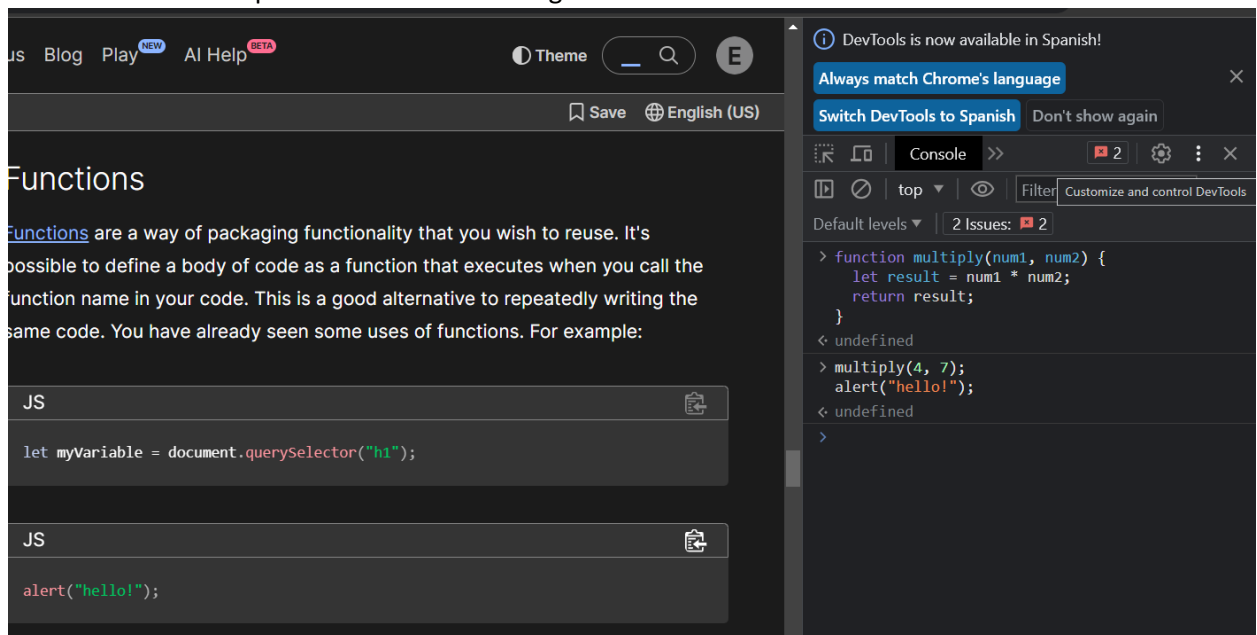
2. Hoja de ruta de la clase 3

1. Puesta en común del ejercicio de planetas.
2. Introducción a JavaScript.
3. Realización de ejercicio de práctica como continuación del ejercicio de la clase 2.
4. Despliegue en Netlify.

3. JavaScript

Trataremos lo siguiente:

1. ¡Hola Mundo! En JavaScript.
2. Variables.
3. Operadores.
4. Comentarios.
5. Condicionales.
6. Ventanas de alerta.
7. Pruebas con JavaScript en la consola del navegador.



8. Funciones.
9. Eventos.
10. Funciones anónimas o de flecha.

4. Practicas guiadas básicas para HTML, CSS y JavaScript: Potenciando nuestro sitio web de ejemplo

4.1. Introducción DOM

Antes de comenzar la práctica hablemos un poco del DOM

El Modelo de Objetos del Documento (DOM) es una interfaz multiplataforma e independiente del lenguaje que permite a programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de una página web. Se utiliza para conectar páginas web a scripts o lenguajes de programación. El DOM también conecta el contenido dinámico de las páginas web con la comunicación entre las aplicaciones del lado del cliente y el servidor.

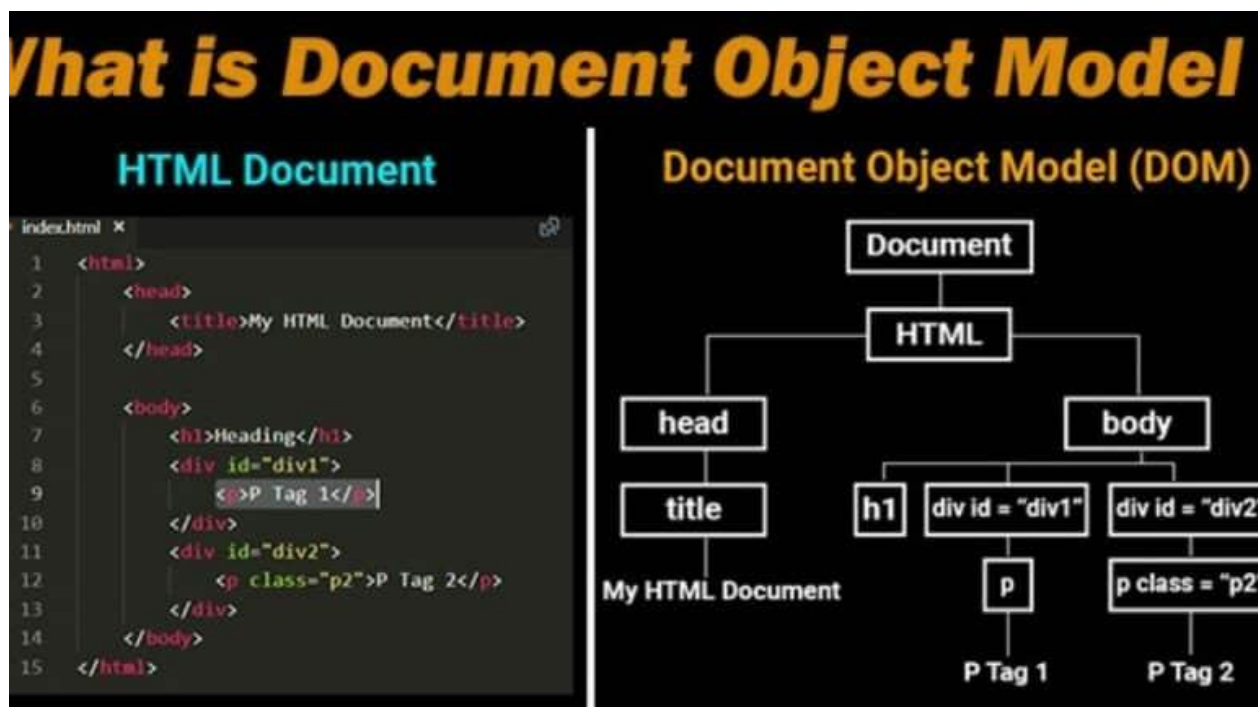
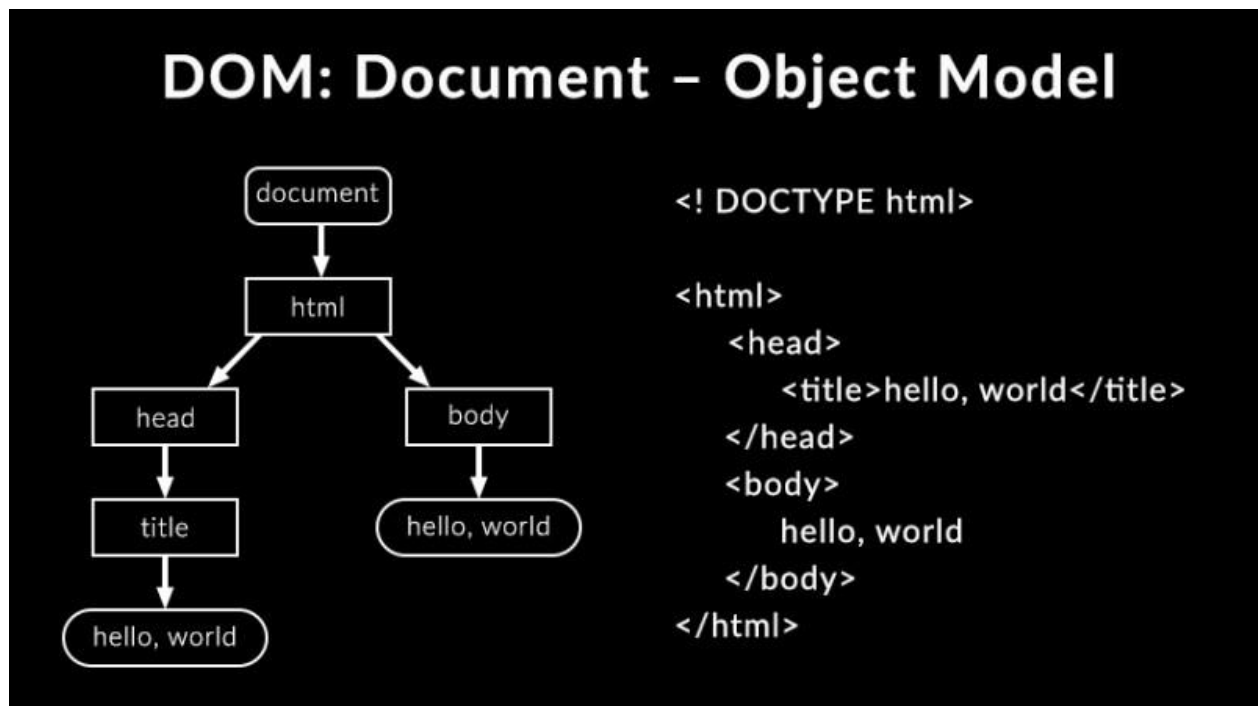
El modelo de objetos de documento (DOM) conecta páginas web con scripts o lenguajes de programación representando la estructura de un documento (como el HTML que representa una página web) en la memoria. Por lo general, se refiere a JavaScript, aunque modelar documentos HTML, SVG o XML como objetos no es parte del lenguaje principal de JavaScript¹.

El DOM representa un documento con un árbol lógico. Cada rama del árbol termina en un nodo y cada nodo contiene objetos. Los métodos DOM permiten el acceso programático al árbol. Con ellos se puede cambiar la estructura, el estilo o el contenido del documento.

El DOM se basa en un modelo de datos orientado a objetos al que se accede a través de una API, a menudo denominada árbol DOM. Se puede acceder a los nodos de este árbol para manipular tanto el contenido como la estructura de un documento. Este árbol de documentos se puede recorrer, buscar, modificar y manipular, y los cambios realizados se reflejan inmediatamente en la representación del árbol DOM de la estructura del documento.

¹ https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

En las siguientes figuras observamos la correspondencia entre HTML y el DOM.



El DOM se introdujo en 1996 y lo mantiene el Consorcio World Wide Web (W3C). Es compatible con la mayoría de los navegadores web modernos.

El DOM es una parte importante del desarrollo web moderno porque permite a los desarrolladores web crear páginas web elaboradas con interacciones entre el navegador y el servidor. También constituye la

base de muchas bibliotecas y marcos de desarrollo web, como React y Angular. Además, la manipulación del DOM se utiliza a menudo en ciberseguridad para identificar y responder a ataques maliciosos de secuencias de comandos en sitios cruzados (XSS).

4.2. Primera parte de la práctica

Aprender como trabajar con JavaScript y manipular el DOM (Document Object Model) para alternar una imagen entre dos imágenes de la página HTML de manera dinámica. Es decir que desarrollaremos un “cambiador de imágenes”. Para realizar esta tarea tomamos como base la página web que venimos desarrollando desde la clase anterior (Clase 2).

Hay que disponer de una imagen similar en dimensiones en pixeles a la imagen que ya teníamos disponible en la clase anterior, y denominaremos a la nueva imagen `firefox2.png`².

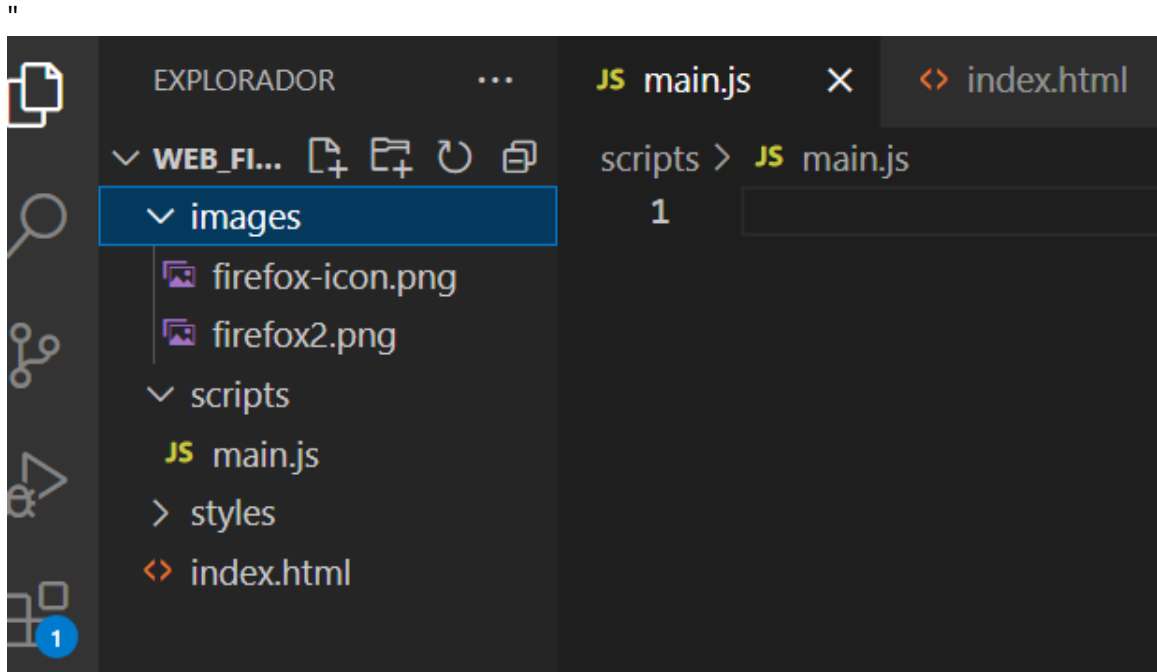
Proceso de trabajo:

1. Tomar como base la página web que venimos desarrollando.



2. Alojar la nueva imagen `firefox2.png` en la carpeta `images`.
3. Crear una carpeta en el directorio principal de la página web llamada `script`.
4. Dentro de la carpeta `script` crear un archivo JavaScript llamado `main.js`. En la siguiente imagen se observa como organizar la estructura de la página web.

² Se comparte por chat de Whatsapp la imagen.



5. Codificar el archivo main.js para crear una función que nos permite realizar el “cambiador de imágenes”.

```
const myImage = document.querySelector("img");

myImage.onclick = () => {
  const mySrc = myImage.getAttribute("src");
  if (mySrc === "images/firefox-icon.png") {
    myImage.setAttribute("src", "images/firefox2.png");
  } else {
    myImage.setAttribute("src", "images/firefox-icon.png");
  }
};
```

Resultado final: al hacer clic en la imagen alternamos una con la otra.



4.3. Segunda parte de la práctica

Cambiar el título de la página a un mensaje de bienvenida personalizado cuando el usuario visite el sitio por primera vez. Este mensaje de bienvenida persistirá. Si el usuario abandona el sitio y regresa más tarde, guardaremos el mensaje utilizando la API de Web Storage³. También incluiremos una opción para cambiar el usuario, y por lo tanto, el mensaje de bienvenida.

El proceso de desarrollo es:

1. Añadir un botón al documento HTML, justo antes de la etiqueta `<script>`

```
<button>Change user</button>
```

2. En `main.js`, codificar el código al final del archivo, exactamente como está escrito. Esto toma referencias al nuevo botón y al encabezado, almacenando cada uno de ellos dentro de las variables:

```
let myButton = document.querySelector("button");
let myHeading = document.querySelector("h1");
```

3. Codificar una función personalizada para definir la funcionalidad pretendida.

```
function setUsername() {
  const myName = prompt("Please enter your name.");
  localStorage.setItem("name", myName);
  myHeading.textContent = `Mozilla is cool, ${myName}`;
}
```

Este código se explica de la siguiente manera: La función `setUsername()` contiene una función `prompt()`, que muestra un cuadro de diálogo similar a `alert()`. Esta función de aviso hace más que `alert()`, solicita al usuario que ingrese datos y los almacena en una variable después de que el usuario hace clic en Aceptar. En este caso, le pedimos al usuario que ingrese un nombre. A continuación, el código llama a una API `localStorage`, que nos permite almacenar datos en el

³ La API de Web Storage (almacenamiento web) proporciona mecanismos mediante los cuales los navegadores pueden almacenar pares clave/valor (key/value), de una manera mucho más intuitiva que el uso de cookies.

navegador y recuperarlos más tarde. Usamos la función `setItem()` de `localStorage` para crear y almacenar un elemento de datos llamado 'name', estableciendo su valor en la variable `myName` que contiene la entrada del usuario para el nombre. Finalmente, configuramos el contenido de texto del encabezado en una cadena, concatenando el nombre recién almacenado del usuario en `myName`.

4. Agregar el siguiente bloque de condición. Podríamos llamar a esto un código de inicialización, ya que estructura la aplicación cuando se carga por primera vez.

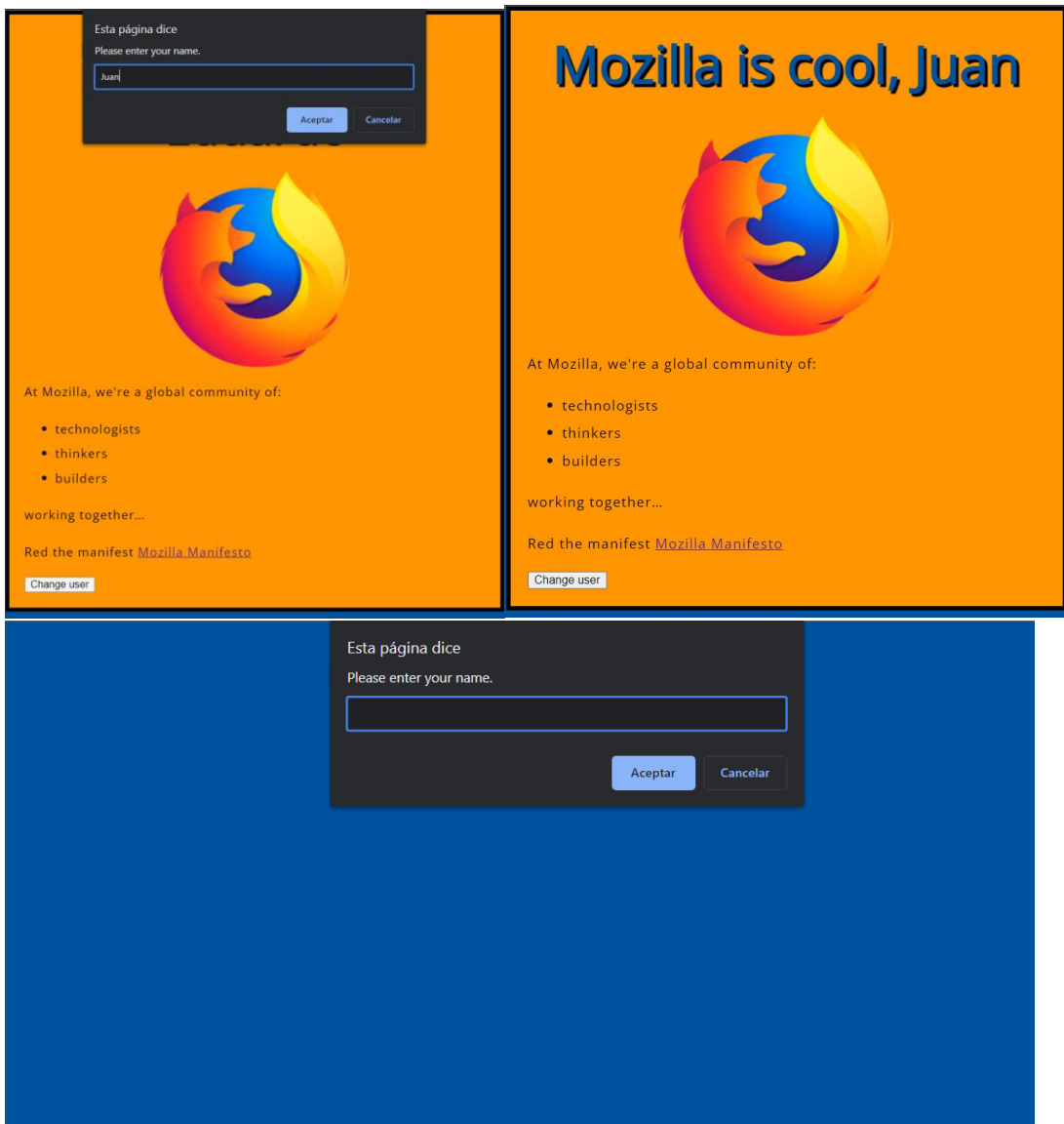
```
if (!localStorage.getItem("name")) {  
  setUsername();  
} else {  
  const storedName = localStorage.getItem("name");  
  myHeading.textContent = `Mozilla is cool, ${storedName}`;  
}
```

Esta primera línea de este bloque utiliza el operador de negación (NO lógico, representado por `!`) para verificar si los datos del nombre existen. Si no existen datos del usuario, se ejecuta la función `setUsername()` para crearlo. Si existe (es decir, el usuario estableció un nombre de usuario durante una visita anterior), recuperamos el nombre almacenado usando `getItem()` y configuramos el contenido de texto del encabezado en una cadena, concatenando el nombre del usuario, como lo hicimos dentro de `setUsername()`.

5. Manejar el evento cuando el usuario haga clic en el botón del HTML. El evento lo manejamos llamando a la función que creamos hace un momento, la función `setUsername`. Colocar abajo del botón un manejador o controlador de eventos (event handler), este manejador en este caso es `onclick`. Cuando se hace clic, se ejecuta la función `setUsername()`. Esto permite al usuario ingresar un nombre diferente presionando el botón.

```
myButton.onclick = () => {  
  setUsername();  
};
```

El resultado final debería ser:



6. Manejo de null. Puede darse la siguiente situación.



Cuando ejecute el ejemplo y aparezca el cuadro de diálogo que le solicita que ingrese su nombre de usuario, intente presionar el botón Cancelar. Deberías terminar con un título que diga Mozilla is coll, null. Esto sucede porque, cuando cancela el mensaje, el valor se establece como nulo. Nulo es un valor especial en JavaScript que se refiere a la ausencia de un valor. Además, intente hacer clic en Aceptar en el cuadro de dialogo sin ingresar un nombre. Deberías terminar con un título que diga "Mozilla is coll," ya que no hay un nombre ingresado.

Para evitar estos problemas, es posible verificar que el usuario no haya ingresado un nombre en blanco. Actualizar la función setUsername() a esto:

```
function setUsername() {  
  const myName = prompt("Please enter your name.");  
  if (!myName) {  
    setUsername();  
  } else {  
    localStorage.setItem("name", myName);  
    myHeading.textContent = `Mozilla is cool, ${myName}`;  
  }  
}
```

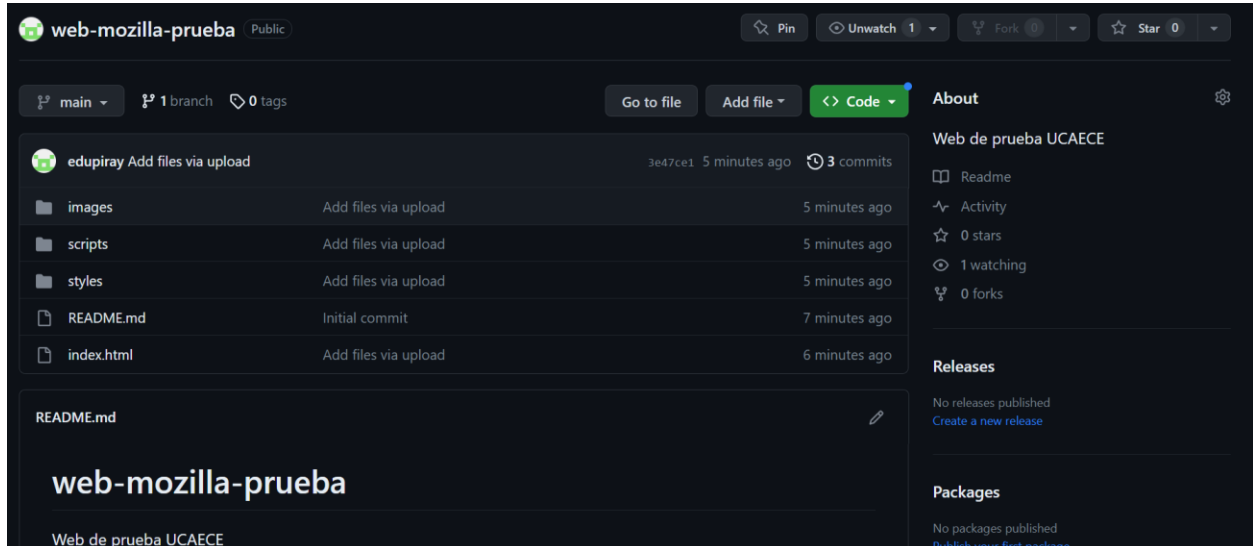
Este código hace lo siguiente: si myName no tiene valor, ejecute setUsername() nuevamente desde el principio. Si tiene un valor (si la afirmación anterior no es cierta), almacene el valor en localStorage y configúrelo como el texto del encabezado.

7. Como paso final borramos el localStorage para que cuando se recargue la página todo el proceso se reinicie. Para ello añadimos la instrucción localStorage.clear() antes de la llave de cierre de la función setUsername(). Por lo tanto, dicha función completa será:

```
function setUsername() {  
  const myName = prompt("Please enter your name.");  
  if (!myName) {  
    setUsername();  
  } else {  
    localStorage.setItem("name", myName);  
    myHeading.textContent = `Mozilla is cool, ${myName}`;  
  }  
  localStorage.clear();  
}
```

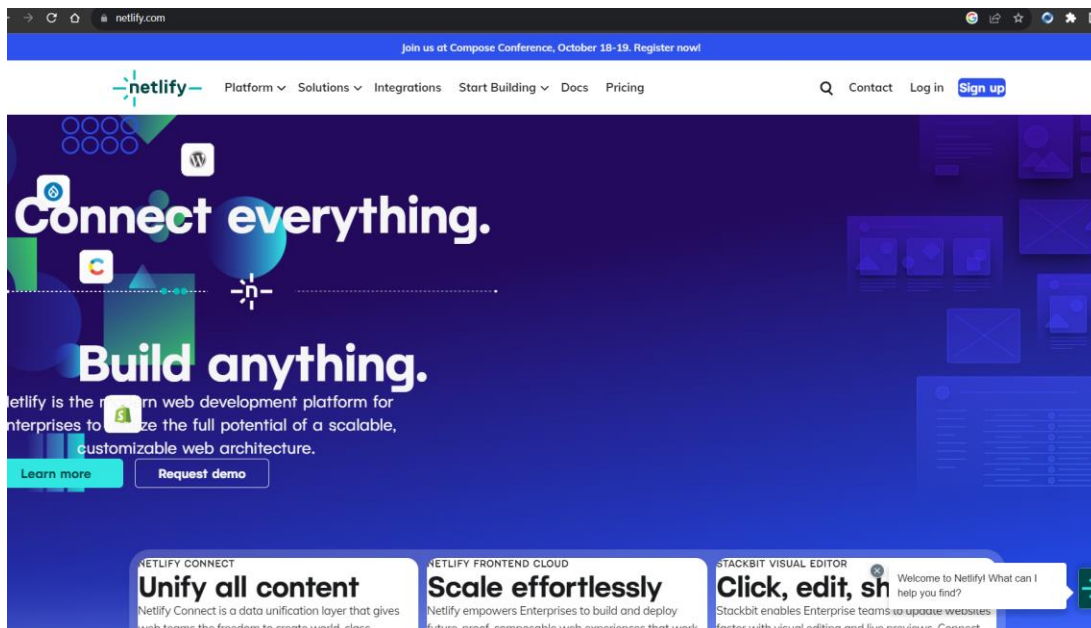
5. Desplegar la aplicación en Netlify

Es necesario crear un repositorio en nuestro servicio de GIT que más utilicemos (en la imagen siguiente se utiliza GitHub). De no tener una cuenta, es necesario crearse una para aprovechar la práctica.

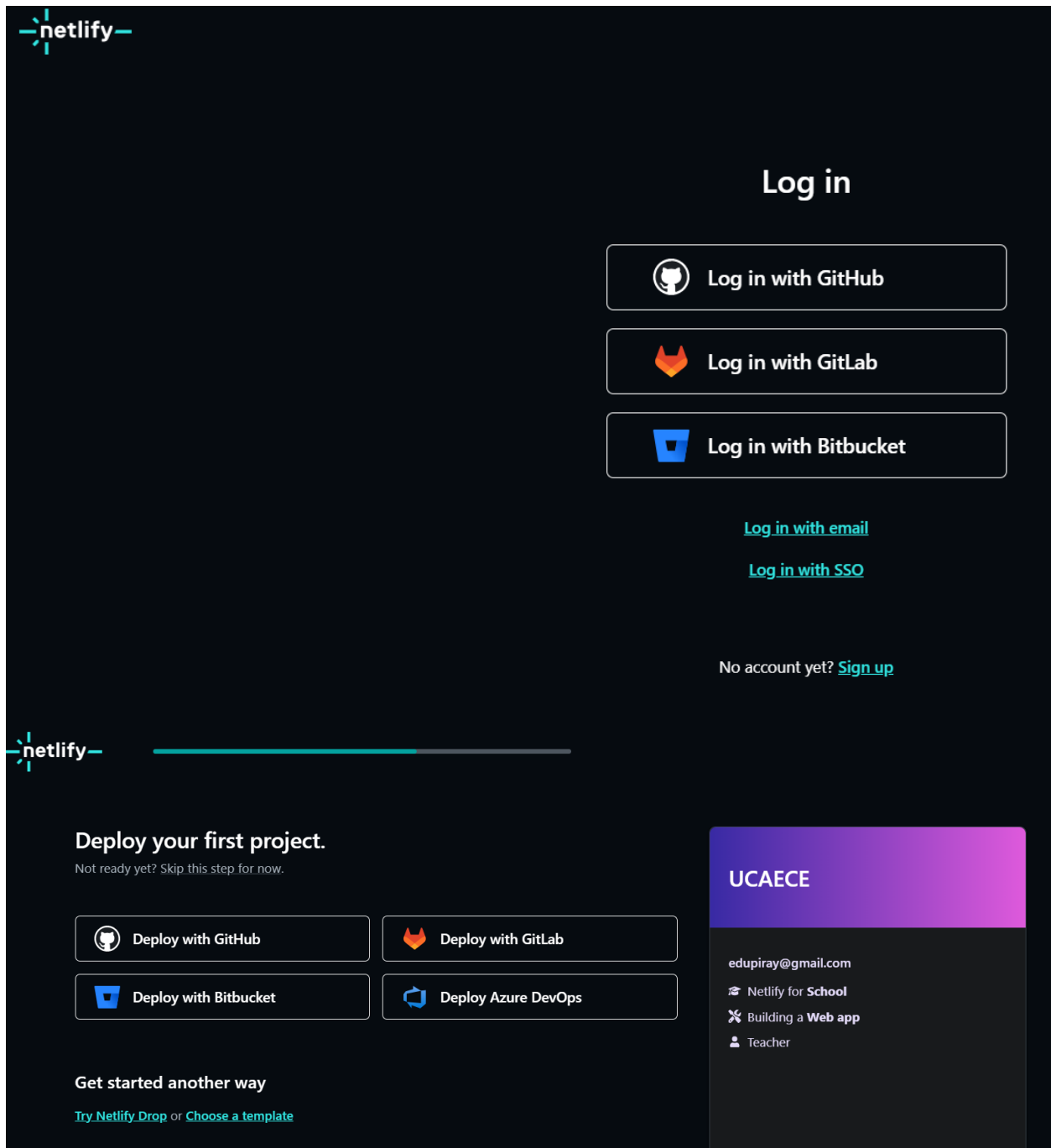


Recordar que los archivos del sitio deben estar en la rama principal. Se pueden subir los archivos a GitHub manualmente haciendo drag and drop, o utilizando el cliente Git que tengamos instalado en nuestra PC.

Ir al sitio de Netlify: <https://www.netlify.com/>





Iniciar sesión en Netlify con alguna cuenta de sistema de control de versiones GIT:




The image shows the Netlify login and deployment interface. At the top left is the Netlify logo. The main heading is "Log in". Below it are three buttons for logging in with GitHub, GitLab, and Bitbucket. There are also links for "Log in with email" and "Log in with SSO". At the bottom right, there is a "Sign up" link for users without an account. Below the login section, there is a section titled "Deploy your first project." with a subtext "Not ready yet? Skip this step for now." and four buttons for deploying with GitHub, GitLab, Bitbucket, and Azure DevOps. At the bottom left, there is a section titled "Get started another way" with links for "Try Netlify Drop" and "Choose a template". On the right side, there is a purple header for "UCAECE" and a list of items: "edupiray@gmail.com", "Netlify for School", "Building a Web app", and "Teacher".

Log in

 Log in with GitHub

 Log in with GitLab


 Log in with Bitbucket


[Log in with email](#)


[Log in with SSO](#)


No account yet? [Sign up](#)

Deploy your first project.
Not ready yet? [Skip this step for now.](#)

 Deploy with GitHub

 Deploy with GitLab

 Deploy with Bitbucket




 Deploy Azure DevOps

Get started another way

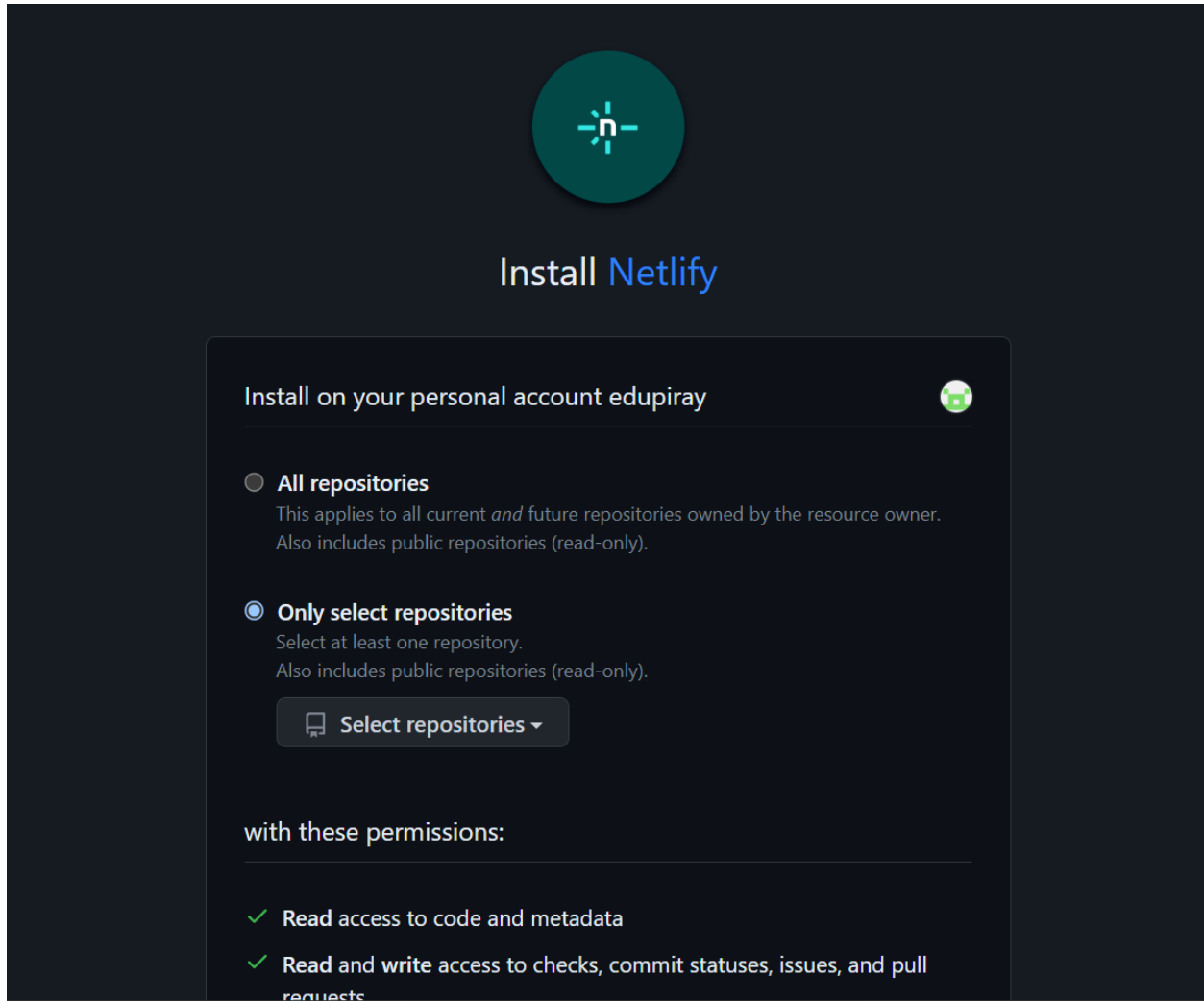
[Try Netlify Drop](#) or [Choose a template](#)

UCAECE

edupiray@gmail.com

-  Netlify for School
-  Building a Web app
-  Teacher

Seleccionar el repositorio:



Seguir los pasos. Puede requerir alguna autenticación.

The screenshot shows the Netlify dashboard for the 'UCAECE' team. The left sidebar contains navigation links: Team overview, Sites, Builds, Integrations, Domains, Members, Audit log, Billing, and Team settings. The main content area is titled 'Let's deploy your project.' and shows the configuration for deploying the 'web-mozilla-prueba' repository. The configuration steps are: 1. Connect to Git provider, 2. Select repository, and 3. Configure site and deploy. The 'Review configuration for web-mozilla-prueba' section shows the team as 'UCAECE' and the branch to deploy as 'main'. The 'Build settings' section includes a link to 'Learn more in the docs' and a 'Base directory' input field. A 'Deploy web-mozilla-prueba to Netlify...' button is visible at the bottom of the configuration panel. Below the configuration panel, there is a large section titled 'Let's deploy your project.' with a preview of the 'web-mozilla-prueba' repository and a 'Deploy web-mozilla-prueba to Netlify...' button. A link to 'Choose a different repo' is also present.

UCAECE

Search anything...

News Support

Team overview
Sites
Builds
Integrations
Domains
Members
Audit log
Billing
Team settings

1. Connect to Git provider 2. Select repository 3. Configure site and deploy

Let's deploy your project.

Review configuration for web-mozilla-prueba

Deploy as **edupiray** on **UCAECE** team from **main** branch

Team
UCAECE

Branch to deploy
main

Build settings

Specify how Netlify will build your site.

[Learn more in the docs](#)

Base directory

The directory where Netlify installs dependencies and runs your build command.

[Upgrade](#)

netlify

Let's deploy your project.

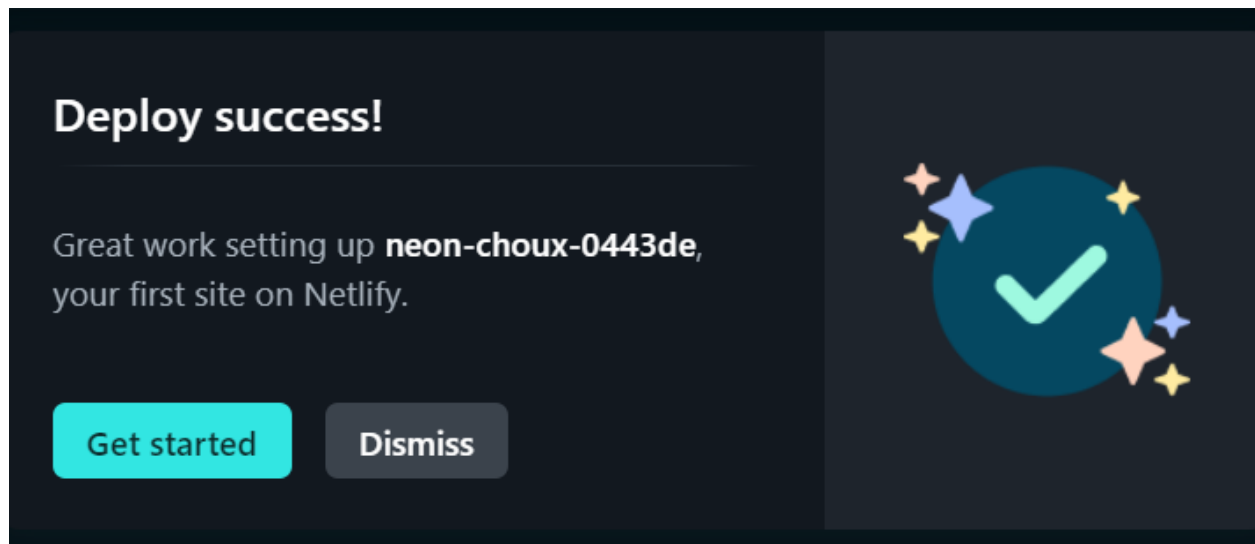
web-mozilla-prueba

Deploy as **edupiray** on **UCAECE** team from **main** branch

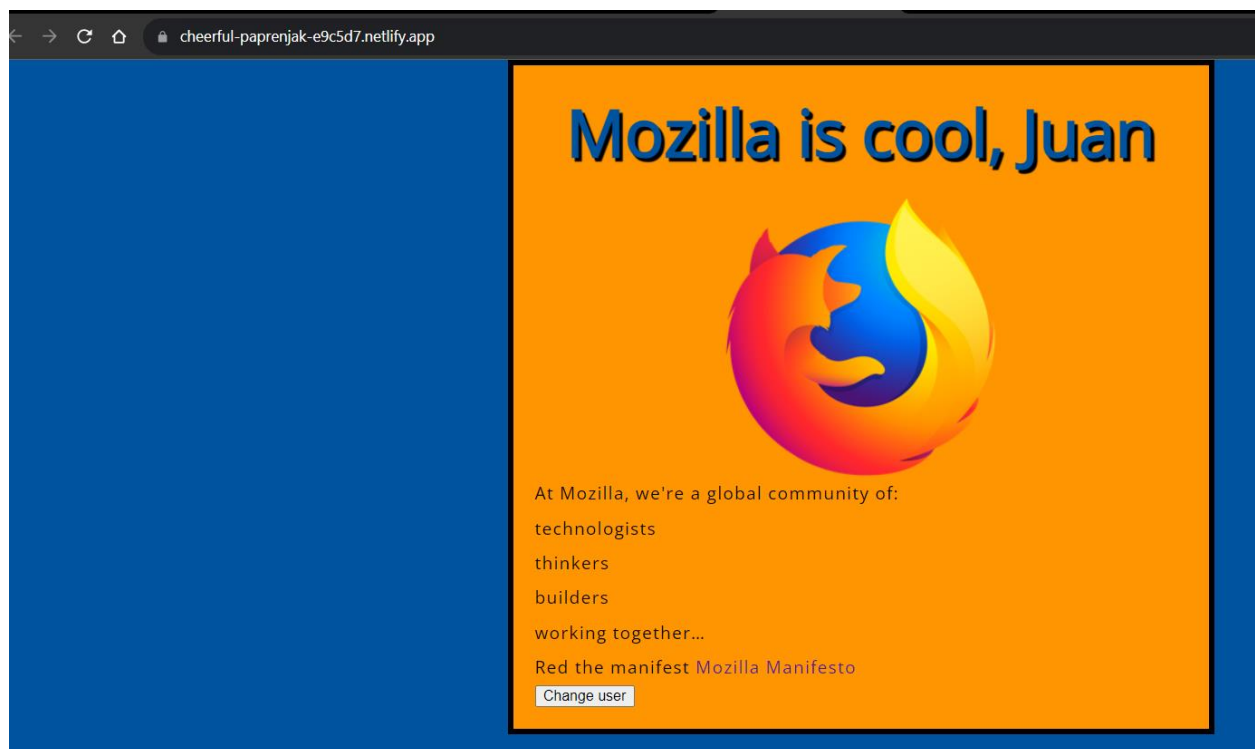
Need to be more specific? [Edit build settings](#)

[Deploy web-mozilla-prueba to Netlify...](#)

[Choose a different repo](#)



Puede observarse en la siguiente imagen un ejemplo del sitio ya desplegado en <https://cheerful-paprenjak-e9c5d7.netlify.app/>
Se puede personalizar la URL ya que el Netlify genera un dominio aleatorio.



6. Referencias

- MDN de JavaScript: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics
- Starge en el navegador: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- DOM: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model