



Formularios, Eventos, DOM, React

Clase 9

KOICA

IGU HANDONG GLOBAL
UNIVERSITY



UNA

OBJETIVOS DE LA CLASE 9

- Manejar los conceptos de programación orientada a objetos en JavaScript, formularios, eventos y DOM.
- Implementar los códigos de ejemplos propuestos en clase.



Formularios y Eventos

- El uso de Javascript en los formularios HTML se hace fundamentalmente con el objetivo de validar los datos ingresados. Se hace esta actividad en el cliente (navegador) para desligar de esta actividad al servidor que recibirá los datos ingresados por el usuario.
- Esta posibilidad de hacer pequeños programas que se ejecutan en el navegador, evitan intercambios innecesarios entre el cliente y el servidor (navegador y sitio web).

Formularios y Eventos

- Suponemos que conoce las etiquetas HTML para la creación de formularios en una página web:

form <form> ... </form>

text <input type="text">

password <input type="password">

textarea <textarea> ... </textarea>

button <input type="button">

submit <input type="submit">

reset <input type="reset">

checkbox <input type="checkbox">

radio <input type="radio">

select <select> ... </select>

hidden <input type="hidden">

Formularios y Eventos

- El navegador crea un objeto por cada control visual que aparece dentro de la página. Nosotros podemos acceder posteriormente desde JavaScript a dichos objetos.
- El objeto principal es el FORM que contendrá todos los otros objetos: TEXT (editor de líneas), TEXTAREA (editor de varias líneas), etc.
- Nuestra actividad en JavaScript es procesar los eventos que generan estos controles (un evento es una acción que se dispara, por ejemplo, si se presiona un botón).

Ejemplo

— — —

Dispondremos un botón y cada vez que se presione, mostraremos un contador:

```
<form>
    <input type="button" onClick="incrementar()"
value="incrementar">
</form>

<script>
    let contador = 0;

    function incrementar() {
        contador++;
        alert('El contador ahora vale :' + contador);
    }
</script>
```

Formularios y Eventos

- A los eventos de los objetos HTML se les asocia una función, dicha función se ejecuta cuando se dispara el evento respectivo. En este caso cada vez que presionamos el botón, se llama a la función incrementar, en la misma incrementamos la variable contador en uno. Hay que tener en cuenta que a la variable contador la definimos fuera de la función para que no se inicialice cada vez que se dispara el evento.
- La función alert crea una ventana que puede mostrar un mensaje.

Problema

— — —

Crear un formulario con tres botones con las leyendas "1", "2" y "3". Mostrar un mensaje indicando qué botón se presionó.

```
<form>

  <input type="button" onClick="presion1()" value="Boton 1">
  <input type="button" onClick="presion2()" value="Boton 2">
  <input type="button" onClick="presion3()" value="Boton 3">

</form>

<script>
  function presion1() {
    alert('Se presionó el botón 1');
  }

  function presion2() {
    alert('Se presionó el botón 2');
  }

  function presion3() {
    alert('Se presionó el botón 3');
  }
</script>
```


Programación orientada a objetos en JavaScript

- El lenguaje JavaScript no es un lenguaje orientado a objetos completo, pero permite definir clases con sus atributos y responsabilidades.
- Luego nos permite definir objetos de estas clases.
- Pero el otro pilar de la programación orientada a objetos, es decir la herencia, recién se incorporó en las últimas versiones (lo veremos más adelante).
- Veremos la sintaxis para la declaración de una clase y la posterior definición de objetos de la misma.
- Desarrollaremos una clase que represente un cliente de un banco.

Programación orientada a objetos en JavaScript

- La clase Cliente tiene como atributos:
nombre
saldo
- y las responsabilidades o métodos de la clase son:
Constructor (inicializamos los atributos del objeto)
depositar
extraer

Programación orientada a objetos en JavaScript

- Luego debemos implementar los siguientes métodos (normalmente en el constructor se utiliza el carácter mayúscula):

```
function Cliente(nombre, saldo) {  
    this.nombre = nombre;  
    this.saldo = saldo;  
    this.depositar = depositar;  
    this.extraer = extraer;  
}
```

```
function depositar(dinero) {  
    this.saldo = this.saldo + dinero;  
}  
  
function extraer(dinero) {  
    this.saldo = this.saldo - dinero;  
}
```

Programación orientada a objetos en JavaScript

El nombre de la clase coincide con el nombre de la función principal que implementamos (también llamado constructor de la clase):

```
function Cliente(nombre, saldo) {  
    this.nombre = nombre;  
    this.saldo = saldo;  
    this.depositar = depositar;  
    this.extraer = extraer;  
}
```

Programación orientada a objetos en JavaScript

A ésta función llegan como parámetro los valores con que queremos inicializar los atributos. Con la palabra clave 'this' diferenciamos los atributos de los parámetros (los atributos deben llevar la palabra clave this)

```
this.nombre = nombre;  
this.saldo = saldo;
```

Programación orientada a objetos en JavaScript

También en el constructor inicializamos la referencia a todos los métodos que contendrá la clase (esto es muy importante y necesario para entender por qué las otras dos funciones pertenecen a esta clase):

```
this.depositar = depositar;  
this.extraer = extraer;
```

Programación orientada a objetos en JavaScript

Por último, implementamos todos los métodos de la clase:

```
function depositar(dinero) {  
    this.saldo = this.saldo + dinero;  
}  
  
function extraer(dinero) {  
    this.saldo = this.saldo - dinero;  
}
```

Programación orientada a objetos en JavaScript

De nuevo recordemos que diferenciamos los atributos de la clase por la palabra clave `this`.

Ahora veamos el archivo HTML completo donde además definiremos un objeto de la clase planteada:

Programación orientada a objetos en JavaScript

```
<script>

    function Cliente(nombre,
saldo) {

        this.nombre = nombre;
        this.saldo = saldo;
        this.depositar =
depositar;
        this.extraer = extraer;
    }

    function depositar(dinero) {
        this.saldo = this.saldo
+ dinero;
    }


```

```
function extraer(dinero) {
    this.saldo = this.saldo - dinero;
}

let cliente1;
cliente1 = new Cliente('Ivan', 1200);
document.write('Nombre del cliente:' +
cliente1.nombre + '<br>');
document.write('Saldo actual:' +
cliente1.saldo + '<br>');
cliente1.depositar(120);
document.write('Saldo luego de depositar
$120---->' + cliente1.saldo + '<br>');
cliente1.extraer(1000);
document.write('Saldo luego de extraer
$1000---->' + cliente1.saldo + '<br>');
```

Programación orientada a objetos en JavaScript

Para definir un objeto de la clase Cliente tenemos:

```
let cliente1;  
cliente1 = new Cliente('Ivan', 1200);
```

Programación orientada a objetos en JavaScript

Luego a las llamadas a métodos le antecedemos el nombre del objeto llamado cliente1:

```
document.write('Nombre del cliente:' + cliente1.nombre + '<br>');  
document.write('Saldo actual:' + cliente1.saldo + '<br>');  
cliente1.depositar(120);  
document.write('Saldo luego de depositar $120---->' + cliente1.saldo + '<br>');  
cliente1.extraer(1000);  
document.write('Saldo luego de extraer $1000---->' + cliente1.saldo + '<br>');
```

Programación orientada a objetos en JavaScript

Podemos decir que la ventaja que podemos obtener con el planteo de clases es hacer nuestros programas mucho más organizados, entendibles y fundamentalmente poder reutilizar clases en distintos proyectos.

Programación orientada a objetos en JavaScript

Podemos directamente dentro del constructor implementar los dos métodos con la siguiente sintaxis:

```
function Cliente(nombre, saldo) {  
    this.nombre = nombre;  
    this.saldo = saldo;  
    this.depositar = function(dinero) {  
        this.saldo = this.saldo + dinero;  
    }  
    this.extraer = function(dinero) {  
        this.saldo = this.saldo - dinero;  
    }  
}
```

Programación orientada a objetos en JavaScript

Luego no hay cambios cuando creamos un objeto de la clase Cliente.

A lo largo de la historia de JavaScript han aparecido varias formas de plantear las clases, más adelante veremos la última sintaxis propuesta utilizando directamente la palabra clave `class` para declarar una clase.

Problema

Confeccionar una clase llamada suma, que contenga dos atributos (valor1, valor2) y tres métodos: cargarvalor1, cargarvalor2 y retornarresultado. Implementar la clase suma.

La definición de un objeto de la clase que deben plantear es:

```
let s=new suma();
```

```
s.primerValor(10);
```

```
s.segundoValor(20);
```

```
document.write('La suma de los dos  
valores es:'+s.retornarResultado());
```

```
function Suma(valor1, valor2) {  
    this.valor1 = valor1;  
    this.valor2 = valor2;  
    this.primerValor = function(valor1) {  
        this.valor1 = valor1;  
    }  
    this.segundoValor = function(valor2) {  
        this.valor2 = valor2;  
    }  
    this.retornarResultado = function() {  
        return this.valor1 + this.valor2;  
    }  
}  
  
let suma1;  
  
suma1 = new Suma(5, 10);  
document.write('La suma de 5 y 10 es:' +  
suma1.retornarResultado() + '<br>');  
  
suma1.primerValor(70);  
suma1.segundoValor(30);  
  
document.write('La suma de 70 y 30 es:' +  
suma1.retornarResultado() + '<br>');
```

DOM (Document Object Model - Modelo Objeto Documento)

El DOM es otra forma para representar una página web. Para entender que es el DOM veamos tres tipos de personas frente a una página web:

1 - Un usuario de internet que conoce muy poco sobre el mundo de la computación, ve una página web como una ventana que contiene texto con distintos tamaños y colores, imágenes, enlaces etc. esto es lo que ve un usuario común de internet.

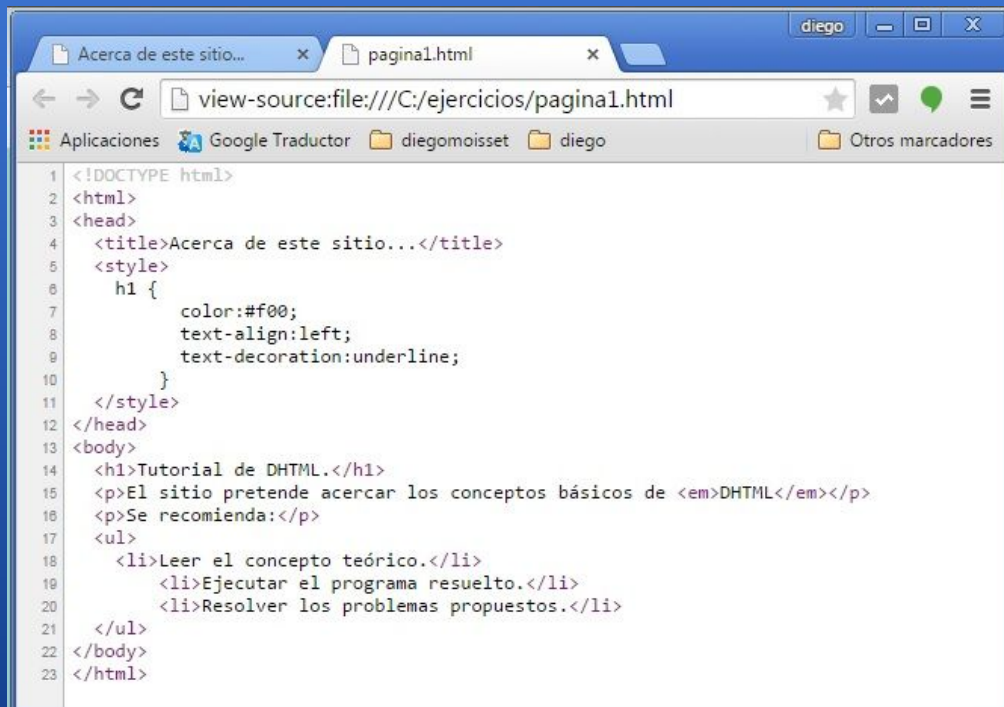
DOM (Document Object Model - Modelo Objeto Documento)



DOM (Document Object Model - Modelo Objeto Documento)

2 - Ahora veamos cómo ve una página web una persona que se ha iniciado hace muy poco en el mundo del diseño de páginas web, este usuario ve un poco más allá de lo que ve un usuario común, puede identificar qué texto dispuso en la marca title del documento, diferencia las distintas marcas HTML para la estructura de la página, si ha definido estilos, si incorporó algún archivo externo css, Javascript etc. Este segundo tipo de personas puede imaginar el contenido HTML y CSS de la página.

DOM (Document Object Model - Modelo Objeto Documento)



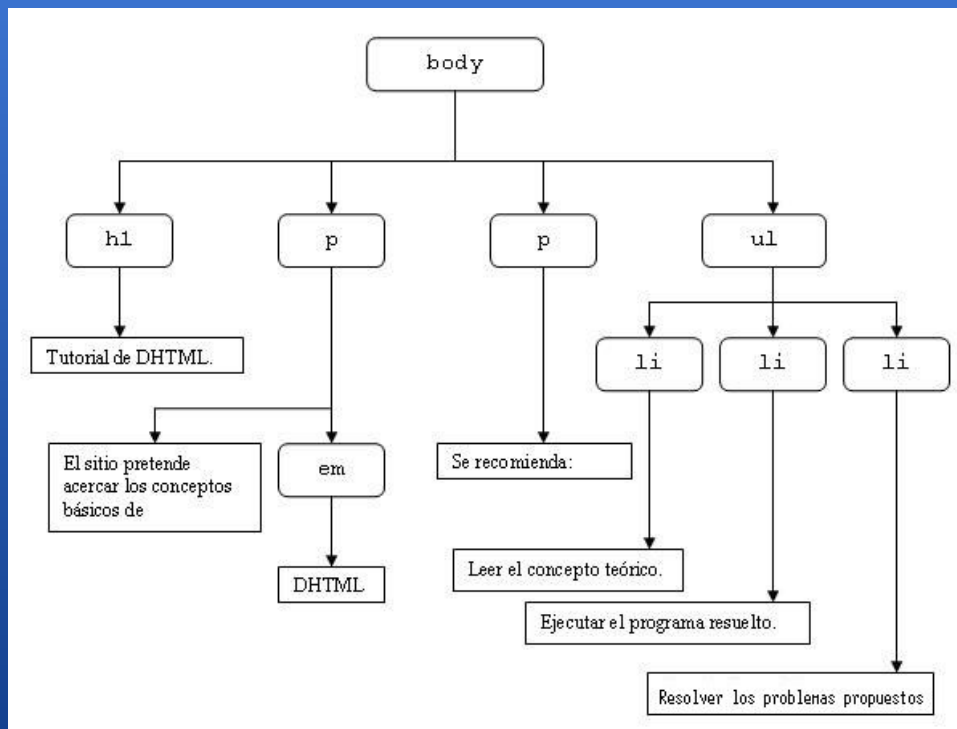
The screenshot shows a web browser window with the title 'diego'. The address bar displays 'view-source:file:///C:/ejercicios/pagina1.html'. The browser's toolbar includes buttons for back, forward, and refresh, as well as a search icon. Below the toolbar, there are several bookmarks: 'Aplicaciones', 'Google Traductor', 'diegomoisset', 'diego', and 'Otros marcadores'. The main content area displays the source code of the HTML file, which is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Acerca de este sitio...</title>
5   <style>
6     h1 {
7       color:#f00;
8       text-align:left;
9       text-decoration:underline;
10    }
11  </style>
12 </head>
13 <body>
14   <h1>Tutorial de DHTML.</h1>
15   <p>El sitio pretende acercar los conceptos básicos de <em>DHTML</em></p>
16   <p>Se recomienda:</p>
17   <ul>
18     <li>Leer el concepto teórico.</li>
19     <li>Ejecutar el programa resuelto.</li>
20     <li>Resolver los problemas propuestos.</li>
21   </ul>
22 </body>
23 </html>
```

DOM (Document Object Model - Modelo Objeto Documento)

3 - Ahora por último, para un programador de sitios web tiene una visión mucho más profunda, puede decir si la página según su interactividad tiene apliques de programación.

DOM (Document Object Model - Modelo Objeto Documento)



DOM (Document Object Model - Modelo Objeto Documento)

Este último tipo de persona puede identificar más allá del código HTML y CSS de la página, intuye perfectamente si hay programación por detrás del esqueleto HTML y la definición de las hojas de estilo.

Este tercer tipo de persona debe conocer perfectamente el DOM.

¿Entonces qué es el DOM?

El DOM es otra forma de ver el contenido de la página. Con el DOM, todos los elementos HTML se insertan en un árbol cuyos nodos son las marcas HTML y las hojas, los valores propiamente dichos de las marcas. Por medio de JavaScript podemos acceder y modificar este árbol de marcas y hacer que la página varíe luego que ya se haya mostrado en el navegador. De aquí el nombre de esta tecnología DHTML es decir Dynamic Hyper Text Markup Language.

¿Entonces qué es el DOM?

Mediante el DOM podemos acceder al contenido y estilo de cada marca del documento y modificarlo de acuerdo a algún evento.

Mediante el DOM podemos insertar, borrar, modificar marcas HTML. Podemos acceder a la hoja de estilo definida a la página y dinámicamente agregar, modificar o borrar propiedades. Todo esto sin tener que recargar la página del servidor, es decir todo se hace en el cliente (navegador) mediante JavaScript.

¿Qué es React?

React es una librería de Javascript para la generación de interfaces visuales.

Está pensado para generar componentes y ser reutilizadas en proyectos medianos y grandes.

Sus principales competidores son Vue (proyecto iniciado por Evan You en 2014) y Angular (proyecto de Google que data de 2010)

¿Qué es React?

React es un proyecto desarrollado por la empresa Facebook y tiene sus inicios en el año 2013.

React nos facilita la implementación de páginas SPA (Single-Page Application) o aplicaciones de página única en un sitio web, este tipo de aplicaciones tiene por objetivo dar al visitante una experiencia más parecida a las aplicaciones de escritorio.

Utiliza la metodología de programación orientada a componentes en forma similar a sus competidores Vue y Angular.

Para desarrollar en forma efectiva una aplicación debemos instalar al menos dos herramientas básicas:

— — —

1. Node.js
2. create-react-app

Instalación de Node.js

La primera herramienta a instalar será Node.js, esto debido a que gran cantidad de programas para el desarrollo en React.js están implementadas en Node.

Debemos Descargar e instalar la última versión estable de Node.js:

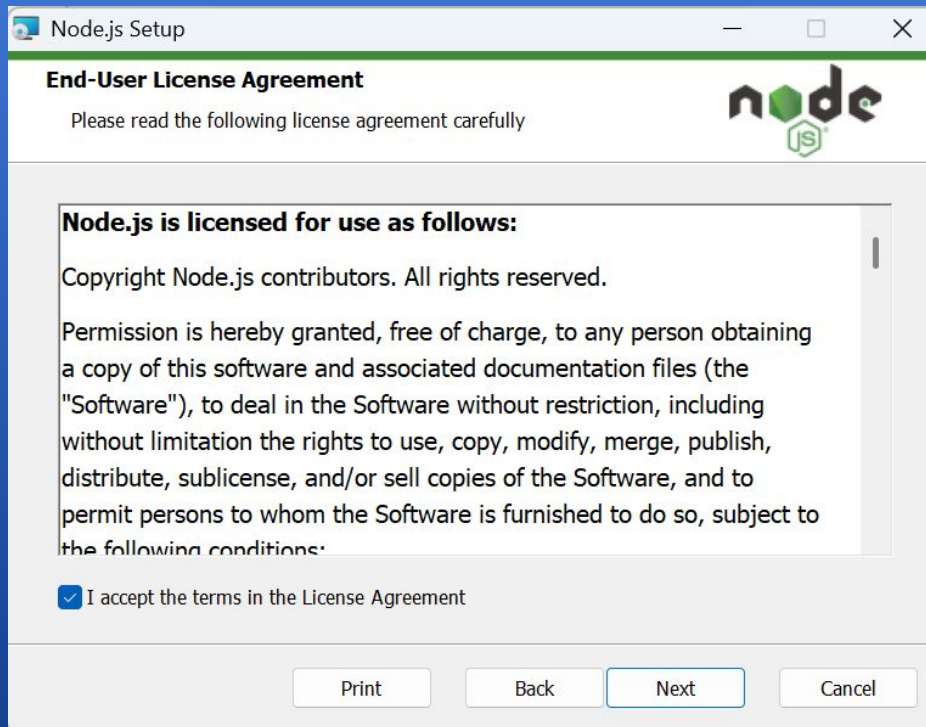
Node.js® es un entorno de ejecución de JavaScript gratuito, de código abierto y multiplataforma que permite a los desarrolladores crear servidores, aplicaciones web, herramientas de línea de comandos y scripts.

Instalación de Node.js

— — —

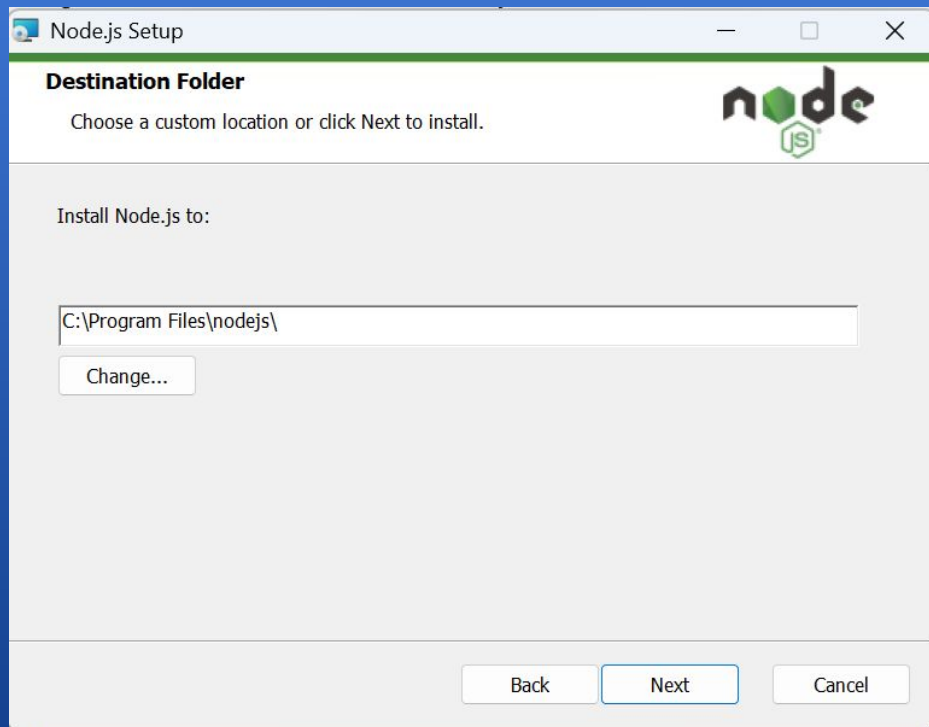


Instalación de Node.js



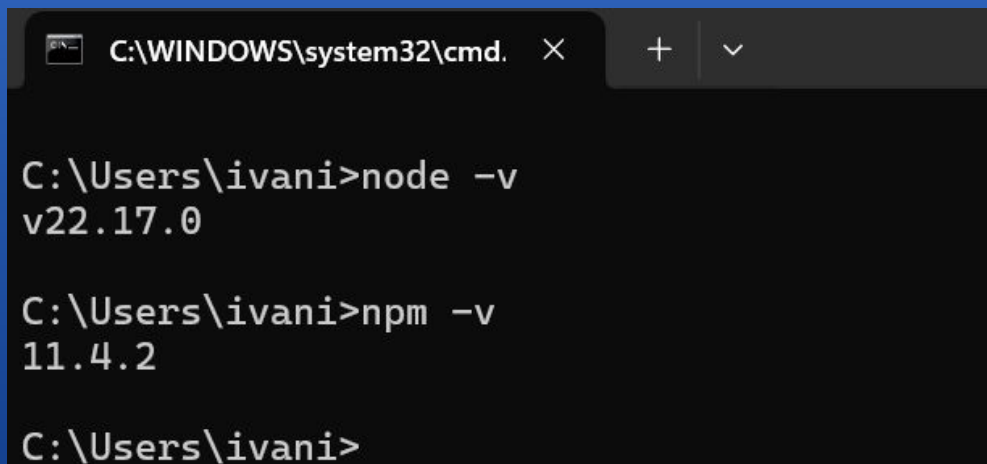
Instalación de Node.js

— — —

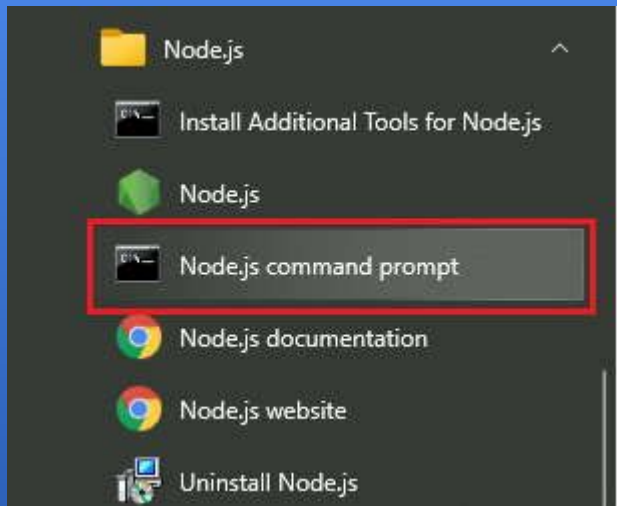


Instalación de Node.js

Una vez instalado Node.js desde la línea de comandos del mismo podemos comprobar su correcto funcionamiento averiguando su versión:



```
C:\WINDOWS\system32\cmd. x + v  
  
C:\Users\ivani>node -v  
v22.17.0  
  
C:\Users\ivani>npm -v  
11.4.2  
  
C:\Users\ivani>
```



Creación del primer proyecto empleando React

En el concepto anterior presentamos qué es React y la instalación de Node.js, veamos ahora los pasos que debemos dar para crear el primer proyecto.

La aplicación create-react-app nos automatiza la generación de todos los archivos y carpetas básicas del proyecto y nos instala Webpack, Babel, ESLint etc. más adelante veremos el objetivo de cada uno de estos programas.

Creación del primer proyecto empleando React

Debemos abrir la consola de comandos de Node y proceder a llamar al programa `npx create-react-app` pasando como parámetro el nombre del proyecto a crear:

```
npx create-react-app proyecto001
```

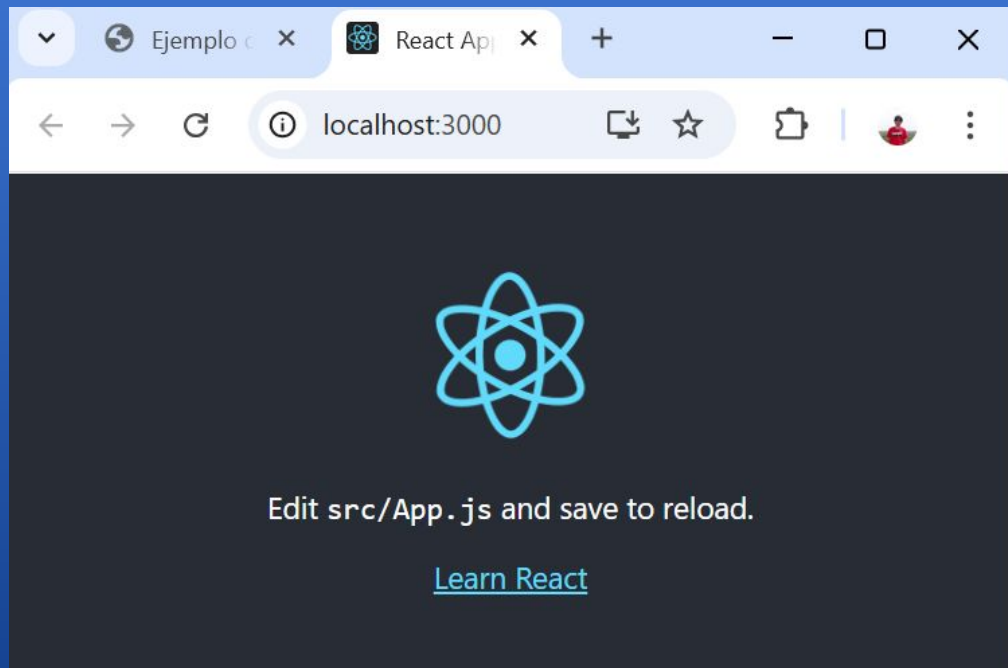
Creación del primer proyecto empleando React

'npx' es una aplicación que descarga la última versión del programa 'create-react-app', dicha aplicación es la que crea el esqueleto de nuestra aplicación mínima en React. Requiere varios segundos para que se creen las carpetas, archivos básicos y descarguen los programas que necesita luego la aplicación. Luego de esto podemos probar la aplicación mínima sin hacer cambios descendiendo desde la línea de comandos a la carpeta 'proyecto001' y ejecutando npm start:

```
cd proyecto001
```

```
npm start
```

Al cabo de unos segundos se abre el navegador y aparece la aplicación funcionando:



Probemos de hacer unos cambios mínimos en el archivo 'App.js' que tiene actualmente el contenido:

```
function App() {  
  return (  
    <h1>Hola mundo</h1>  
  );  
}
```



**¡MUCHAS
GRACIAS!**