



**¡Les damos la
bienvenida!**

¿Comenzamos?

Esta clase va a ser

- grabada



COMISIÓN N°43315

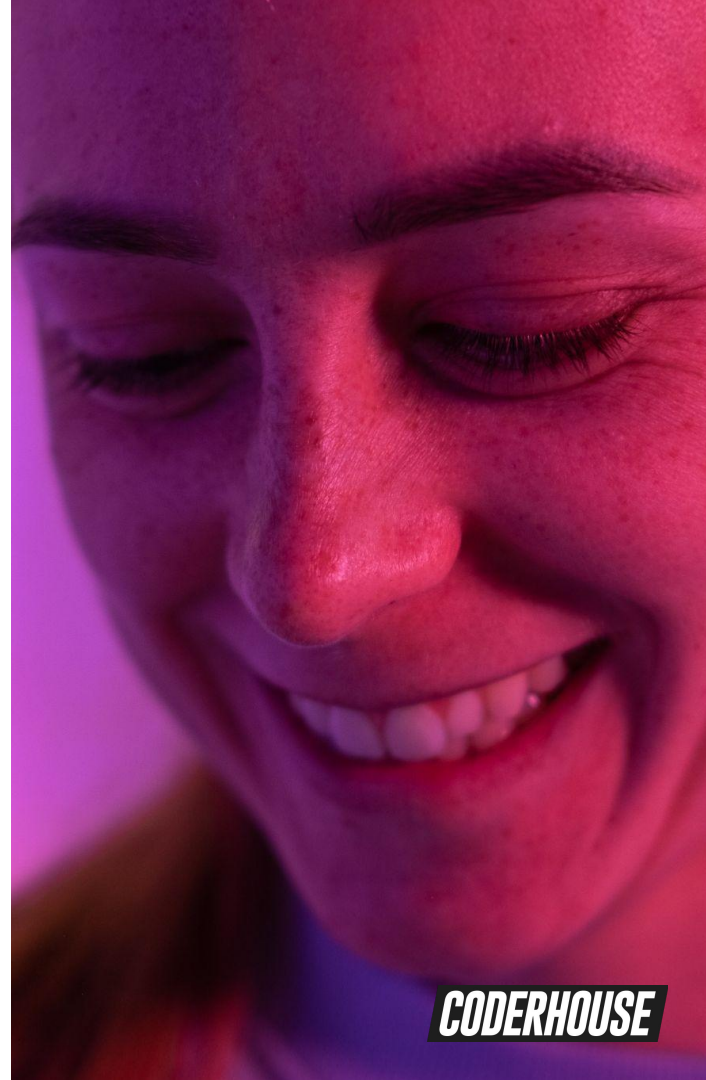
Presentación del equipo

- ✓ Profesor/a responsable: Juan Dahl
- ✓ Coordinador/a:
- ✓ Tutores y tutoras:
 - Ailin Nakaganeku
 - Juan Manuel Maroñas
 - Matias Cantora
 - Fredy Lezama

Presentación de estudiantes

Por encuestas de Zoom

1. País
2. Conocimientos previos
3. ¿Por qué elegiste este curso?



CODERHOUSE

¿Dudas sobre el onboarding?

Mira los vídeos de
Onboarding en la
Plataforma



Lo que debes saber
antes de empezar

Acuerdos y compromisos

ACUERDOS Y COMPROMISOS

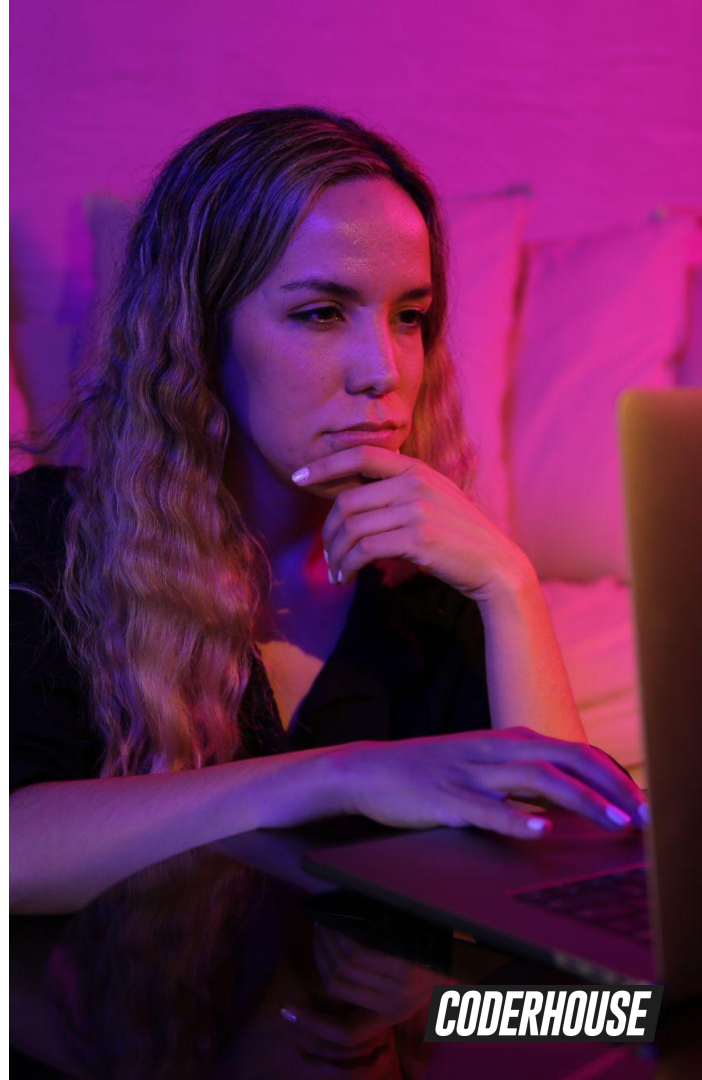
Convivencia

- ✓ Conoce aquí nuestro [código de conducta](#) y ayúdanos a generar un ambiente de clases súper ameno.
- ✓ Durante las clases, emplea los medios de comunicación oficiales para canalizar tus dudas, consultas y/o comentarios: **chat Zoom público y privado, y por el chat de la plataforma.**
- ✓ Ten en cuenta [las normas del buen hablante y del buen oyente](#), que nunca están de más.
- ✓ Verifica el estado de **la cámara y/o el micrófono** (on/off) de manera que esto no afecte la dinámica de la clase.

ACUERDOS Y COMPROMISOS

Distractores

- ✓ Encuentra tu espacio y crea el momento oportuno para **disfrutar de aprender**
- ✓ Evita dispositivos y aplicaciones que puedan **robar tu atención**
- ✓ Mantén la **mente abierta y flexible**, los prejuicios y paradigmas no están invitados



CODERHOUSE

ACUERDOS Y COMPROMISOS

Herramientas

- ✓ Mantén a tu alcance **agua, mate o café**
- ✓ Si lo necesitas, ten a mano lápiz y papel para que no se escapen las ideas. Pero recuerda que **en Google Drive tienes archivos que te ayudarán a repasar, incluidas las presentaciones.**
- ✓ Conéctate desde algún equipo (laptop, tablet) que te permita **realizar las actividades** sin complicaciones.
- ✓ Todas las clases quedarán grabadas y serán compartidas tanto en la **plataforma de Coderhouse como por Google Drive.**



ACUERDOS Y COMPROMISOS

Equipo

- ✓ **¡Participa de los After Class!** Son un gran espacio para atender dudas y mostrar avances.
- ✓ Intercambia ideas por **el chat de la plataforma.**
- ✓ Siempre **interactúa respetuosamente.**
- ✓ No te olvides de **valorar tu experiencia** educativa y de contarnos cómo te va.

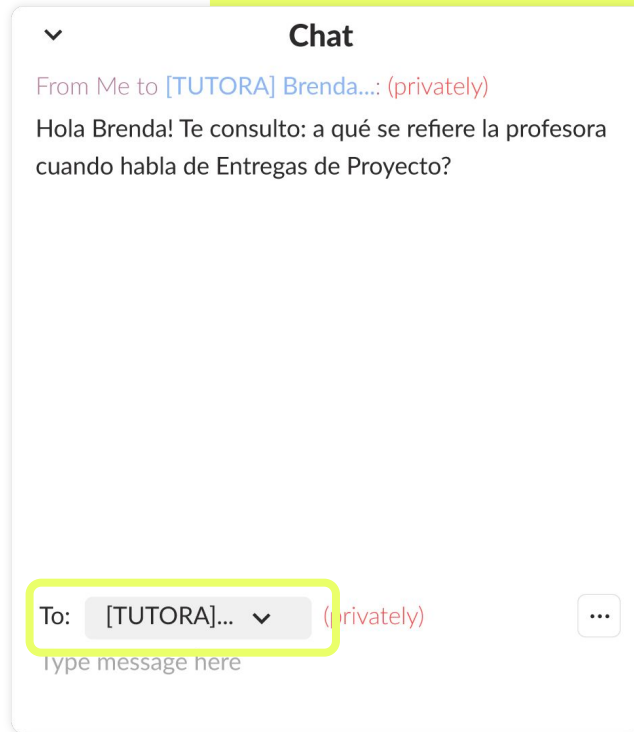
Interacciones en clase

INTERACCIONES EN CLASE

Mientras el profesor explica

Para mantener una comunicación clara y fluida a lo largo de la clase, te proponemos mantener 2 reglas:

1. Si tienes dudas durante la explicación, debes consultarle directamente por privado a tu tutor por el chat de Zoom.

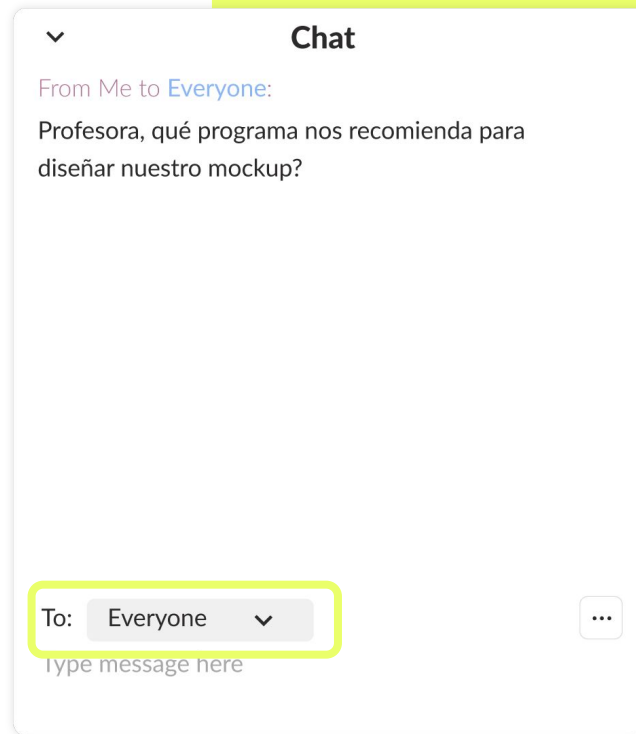


Espacios para consultas

2. Entre contenido y contenido, se abrirán breves espacios de consulta. Allí puedes escribir en el chat tu pregunta.

¡Tu duda puede ayudar a otras personas!

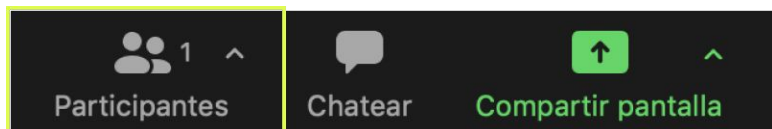
No olvides seleccionar “todos” para que todos puedan leerte (y no solo tu tutor).



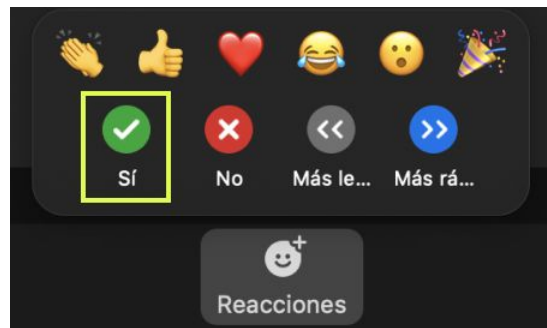
INTERACCIONES EN CLASE

Funcionalidades

Para **evitar saturar el chat de mensajes**, utiliza los signos que figuran en el apartado **Participantes**, dentro de Zoom.**



Por ejemplo: si se pregunta si se escucha correctamente, debes seleccionar la opción "Sí" o "No".



**Para quitar el signo, presiona el mismo botón nuevamente o la opción "clear all".

After Class

AFTER CLASS

¿Qué son?

Te acompañamos para resolver tus consultas sobre el contenido en estos espacios.

Si hay **temas que no se entendieron o necesitan refuerzo** se trabajarán en una clase de 1 hs que opera como **espacio de consulta**.

No son obligatorias ni se toma asistencia, pero son el espacio uno a uno con tu profesor/a** para responder dudas puntuales o reforzar conceptos.

Tu profesor/a está comprometido con tu educación, por lo tanto:

- ✓ Se responderán **dudas puntuales** que hayan quedado sobre los temas dados. ¡Vení preparado, queremos escucharte!
- ✓ Se verán temas de **conocimientos básicos** para la nivelación de saberes.
- ✓ Se retomarán las **actividades extra** propuestas en la Guía de ejercicios complementarios.

Instancias prácticas

Instancias prácticas



Actividades de clase

Ayudan a poner en práctica los conceptos y la teoría vista en clase. No deben ser subidas a la plataforma y se desarrollan en la clase sincrónica.



Guía de actividades para el Proyecto Final

Actividades relacionadas con el Proyecto Final. No son entregables ni obligatorias, pero su resolución es muy importante para llegar con mayor nivel de avance a las entregas obligatorias. Se desarrollan de forma asincrónica.

Instancias prácticas



Guía de ejercicios complementarios

Propuestas de ejercitación práctica complementaria, basadas en problemáticas comunes del desarrollo de aplicaciones. Se desarrollan en los AfterClass.



Pre-entregas

Entregas obligatorias con el estado de avance de tu proyecto final que deberás subir a la plataforma a lo largo del curso y hasta 7 días luego de la clase, para ser corregidas por tu tutor/a.

Instancias prácticas



Dentro de tu carpeta de cursada encontrarás el archivo de “Hoja de ruta”, este espacio fue creado para que puedan visualizar en un mismo lugar, de manera rápida y ágil, todas las pre-entregas y entrega del Proyecto Final.

Te recomendamos utilizar esta herramienta para organizar la cursada y la construcción de tu proyecto final.

ALERTAS

¿Qué son y cuándo aparecen?



CoderAlert

Son avisos creados para comunicar cuándo los temas de una clase están directamente relacionados con alguna **pre-entrega** y con el **proyecto final** de modo que puedas ir construyendo con antelación parte de la consigna. Lo conseguirás usualmente al final de la presentación de la clase.



Coder Training

Son alertas que te indicarán que el contenido de una clase puede ser ejercitado mediante a través de actividades presentes en el Workbook. Son totalmente opcionales y cumplen la función de espacio práctico asincrónico.

NOTA: En ambos casos podrás usar el **workbook** para practicar.

¿Qué son?


























Son actividades o ejercicios que se realizan durante el curso, que ayudan a materializar los contenidos trabajados en las clases mediante la práctica y que suman a la construcción del proyecto final.



Pre-entregas del Proyecto final

Entregas con el estado de avance de tu **proyecto final** que deberás subir a la plataforma a lo largo del curso y hasta 7 días luego de la clase, para ser corregidas por tu docente o tutor/a.

TABLERO DE CLASES

Clase 1 Nivelación 	Clase 2 Instalación y configuración del entorno 	Clase 3 JSX y transpiling 	Clase 4 Componentes I  
Clase 5 Componentes II  	Clase 6 Promises, asincronía y MAP  	Clase 7 Consumiendo API's  	Clase 8 Workshop: Hooks, Children y Patrones
Clase 9 Routing y navegación  	Clase 10 Eventos  	Clase 11 Context  	Clase 12 Técnicas de rendering  
Clase 13 Firebase I  	Clase 14 Firebase II  	Clase 15 Workshop  	

¡Importante!

Las pre entregas del proyecto final se deben cargar hasta **siete días** después de finalizada la clase. Y contarás con **10 días** una vez finalizado el curso para entregar tu proyecto final.
Te sugerimos llevarlos al día.



MARTES 25/01 19:00HS - VALORACIÓN REQUERIDA

2. Metodologías de diseño y UX research

DESAFÍO - EXPIRA EL MARTES 01/02/2022 23:59HS

Definiendo el problema, objetivo y solución

Se bloqueará en 7 días y 11:48hs luego no se podrá entregar.

↑ Entregar



PARA RECORDAR

A tener en cuenta

Este curso cuenta con una [valija de recursos nivelatorios](#).

Podrás encontrar tutoriales, contenido audiovisual y un glosario.

¡Anímate a descubrirla!





**¿Cuál es nuestro
Proyecto final?**

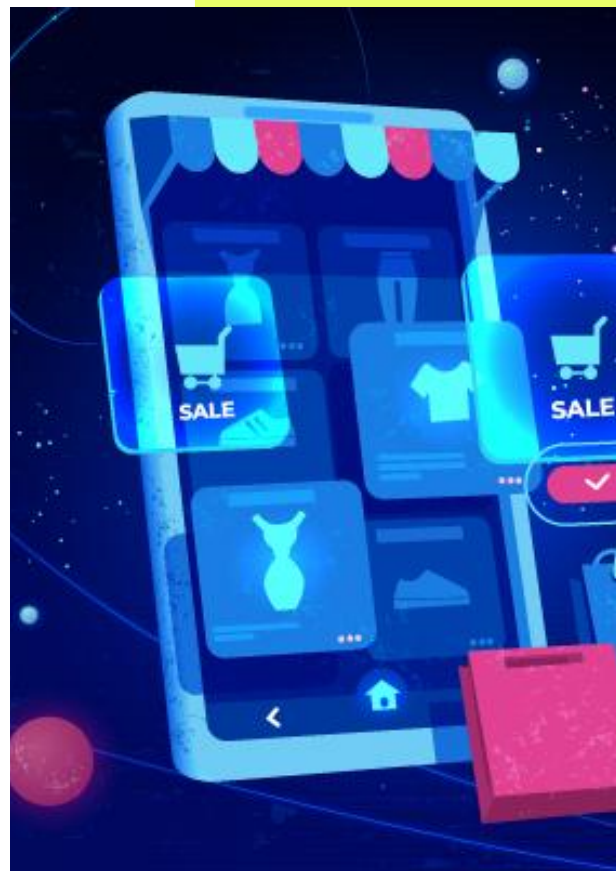


PROYECTO FINAL

Crea tu propia tienda E-commerce

Consigna:

Desarrollarás el front end de una tienda online con carrito de compras, utilizando los componentes de React y Firebase como servidor en la nube. Crearás así una experiencia de usuario amigable con actualizaciones visuales instantáneas y código escalable.



CODERHOUSE



PROYECTO FINAL

Proyectos de nuestros estudiantes

A.M. Florist:

<https://unruffled-hermann-20d922.netlify.app/>

Refugio Tienda Deco:

<https://refugio-tiadecodeco-marceloluismoreno.netlify.app/>

Del Campe:

<https://cocky-bose-3d69f9.netlify.app/>

¡Esperamos que les resulten inspiradores!





PROYECTO FINAL

Entrega	Requisito	Fecha
1° entrega	Durante esta entrega, se hará una revisión integral del estado actual de avance de tu proyecto.	Clase N° 4 (04/07/2023)
2° entrega	Durante esta entrega, se hará una revisión integral del estado actual de avance de tu proyecto.	Clase N° 9 (20/07/2023)
Proyecto Final	Deberás subir a la plataforma el link a tu app de e-commerce completamente funcional.	Clase N° 15 (10/08/2023)

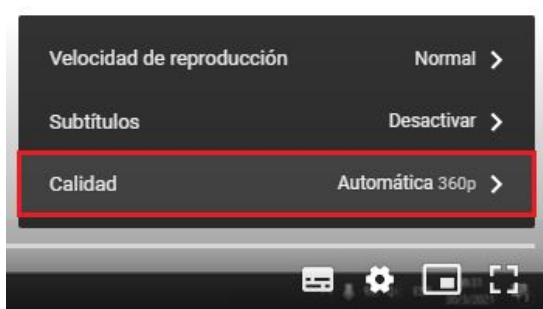
Tutoriales para instalaciones

¿Cómo instalar programas y softwares?

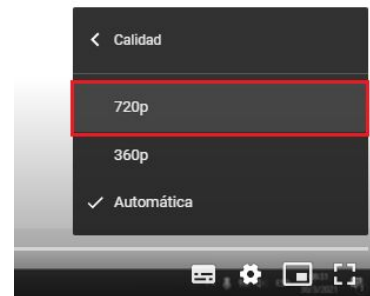
Dentro de la carpeta ["Tutoriales para instalaciones"](#) encontrarás videos donde te explicamos cómo instalar los softwares y programas que utilizarás a lo largo del curso. Para optimizar la calidad de imagen de los mismos al reproducirlos, deberás seguir los siguientes pasos:



Haz clic en configuración



Luego en calidad



Y por último selecciona 720p

Clase 01. REACT JS

Introducción a React Js

Temario

01

Introducción a React Js

- ✓ ¿Qué es React?
- ✓ ¿Qué es un componente?
- ✓ Código declarativo VS Código imperativo
- ✓ ¿Qué es una expresión?
- ✓ ¿Qué es una función?

02

Instalación y configuración del entorno

- ✓ Funcionamiento de React Js
- ✓ ¿Qué es virtual DOM?
- ✓ ¿Qué es NODE?
- ✓ NODE JS ¿Qué es NPM?
- ✓ Crear una aplicación utilizando el CLI

03

JSX y Transpiling

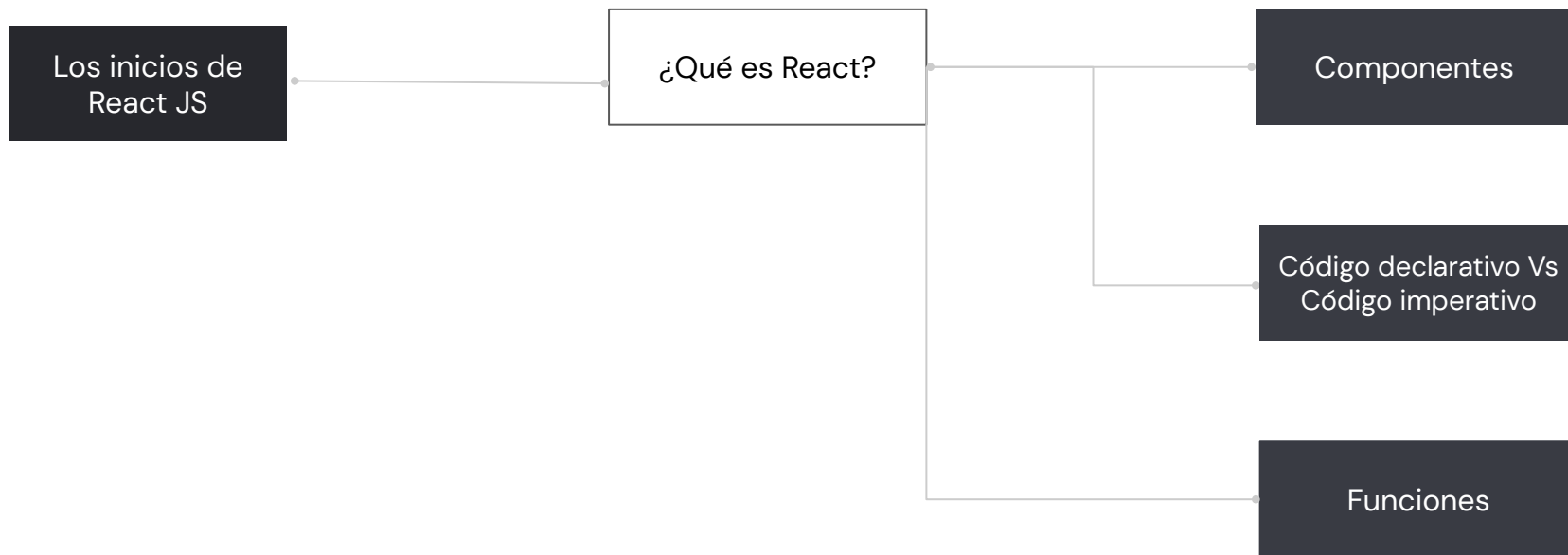
- ✓ Sugar Syntax
- ✓ Polyfills y la retrocompatibilidad
- ✓ Bundling con Webpack
- ✓ Transpiling
- ✓ Jsx

Objetivos de la clase

- **Introducirnos** en los conceptos básicos de React
- **Comprender** de qué manera React disponibiliza sus herramientas.
- **Conocer** a React en acción en su forma más pura.



MAPA DE CONCEPTOS



React Js

Los inicios

React JS fue creada por Jordan Walke, un ingeniero de software en Facebook, inspirado por los problemas que tenía la compañía con el mantenimiento del código de su plataforma, que no paraba de crecer.

Sitios basados en React Js



NETFLIX



y más...

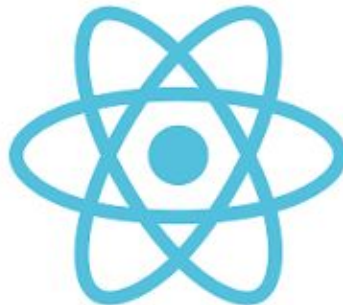
¿Qué es React Js?

React Js

Es una librería de Javascript para construir interfaces de usuario. Las interfaces son la forma que tiene el usuario para interactuar con el sistema.

React Js nos permite crear aplicaciones web y aplicaciones móviles.

En este curso veremos interfaces de usuario para aplicaciones web. 😎



¿En qué se diferencia de Bootstrap?

Bootstrap y React.Js son librerías front-end utilizadas para construir interfaces de usuario para aplicaciones web (entre otras).

Facilitan el proceso de desarrollo proporcionando herramientas para tareas comunes en el desarrollo de UI 's.

Sin embargo, la **diferencia principal** es que React Js es una biblioteca de JavaScript para construir interfaces de usuarios, mientras que Bootstrap es una biblioteca de CSS y JavaScript para el diseño y estilo de la UI.



**Basado en
componentes**

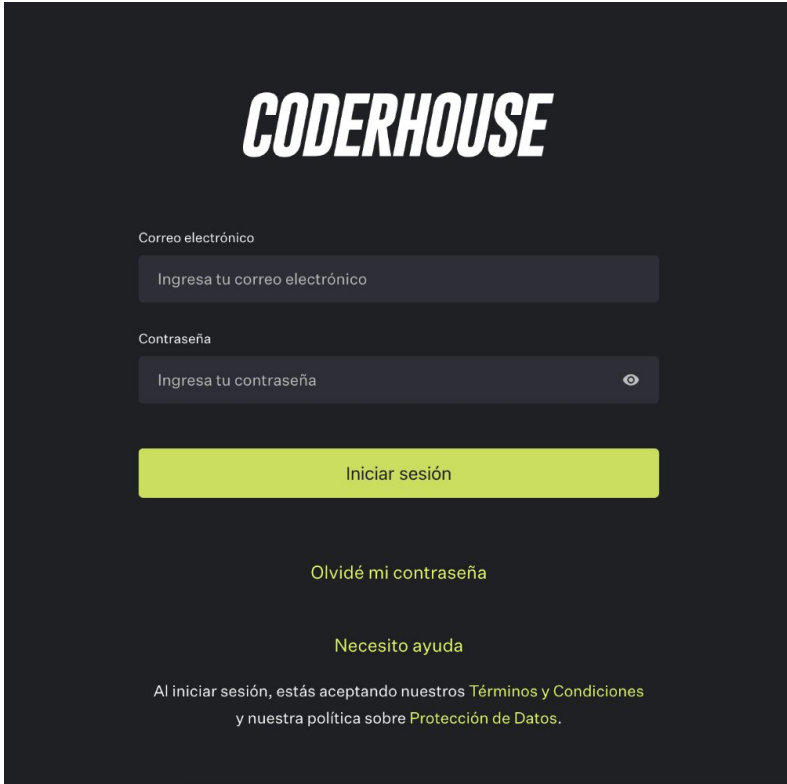
Componentes

Las interfaces de usuario que crearemos están basadas en componentes.

Un componente es un conjunto de elementos que cumplen una función específica. Por ejemplo, un botón.

Esto quiere decir que puedo tener componentes compuestos por otros componentes. Una interfaz de usuario está compuesta por componentes padres y componentes hijos.

Componentes. Ejemplo:




The image shows a login form for 'CODERHOUSE' on a dark background. The form includes two input fields for email and password, a green login button, and links for password recovery and help. At the bottom, there is a disclaimer about terms and conditions.

CODERHOUSE

Correo electrónico

Contraseña



[Iniciar sesión](#)

[Olvidé mi contraseña](#)

[Necesito ayuda](#)

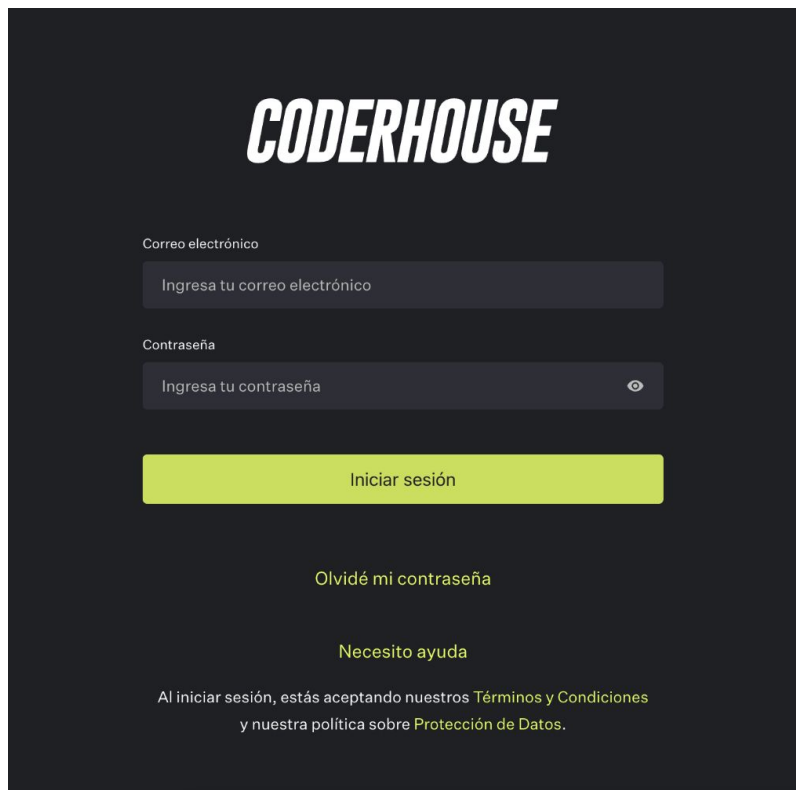
Al iniciar sesión, estás aceptando nuestros [Términos y Condiciones](#) y nuestra política sobre [Protección de Datos](#).



Break

¡10 minutos y volvemos!

Componentes. Ejemplo:



The image shows a login form for Coderhouse. At the top is the 'CODERHOUSE' logo in a stylized, italicized font. Below it, there are two input fields: one for 'Correo electrónico' (Email) and one for 'Contraseña' (Password). The email field has a placeholder text 'Ingresa tu correo electrónico'. The password field has a placeholder text 'Ingresa tu contraseña' and a small eye icon to toggle visibility. Below these fields is a large, bright green button labeled 'Iniciar sesión' (Log in). Underneath the button, there are two links: 'Olvidé mi contraseña' (Forgot my password) and 'Necesito ayuda' (I need help). At the bottom, there is a small line of text: 'Al iniciar sesión, estás aceptando nuestros [Términos y Condiciones](#) y nuestra política sobre [Protección de Datos](#).'

- Input
 - Correo
 - Contraseña
- Botón
- Link
 - Olvidé contraseña
 - Necesito Ayuda
- Entre otros

Componentes padres e hijos

Un componente padre es un componente que contiene a otros componentes llamados componentes hijos. Por ejemplo, un navbar podría ser un componente padre que contiene botones.

Esto permite una organización más clara y modular del código, dándonos muchos beneficios a la hora de escribir código.

En resumen, **los componentes padres e hijos forman una jerarquía de componentes** que permite la creación de interfaces de usuario complejas y modulares en aplicaciones React Js.

Código declarativo

React utiliza código declarativo, pero ¿qué significa? 🤔

Es un estilo de programación en el que se declaran las intenciones y la lógica detrás de un componente o aplicación, en vez de especificar cómo se deben realizar los pasos para lograr el resultado deseado.

En otras palabras, **se enfoca en qué se quiere lograr, en lugar de cómo se logra.**

Este enfoque permite una mayor legibilidad del código y separar la lógica de cada componente en un solo lugar para poder reutilizarla y mantenerla de forma mucho más sencilla.

Diferencia con código imperativo

Por el contrario, el código imperativo es un estilo de programación en el que se especifican los pasos detallados que se deben realizar para lograr un resultado específico. Es decir, **se enfoca en cómo se logra un resultado**.

Puede ser menos legible y requerir más código, ya que se detallan todos los pasos necesarios para lograr un resultado, incluyendo validaciones, bucles y control de flujo.

Un ejemplo de este estilo, es la creación de una UI utilizando Vanilla JS donde somos nosotros quienes debemos indicar paso a paso cómo generar cada elemento en el DOM y dónde.

¿Entonces?

Como primer análisis, podemos decir que React Js contiene todos los pasos y lógica para la manipulación del DOM de manera óptima.

Esto nos brinda la **posibilidad de escribir nuestro código de forma declarativa** centrándonos en lo que se quiere lograr en lugar del cómo. 🖊️

Expresiones

¿Que es una expresión?

Las expresiones son preguntas que JavaScript puede responder. JavaScript responde a las expresiones de la única manera que conoce: con valores.

Por ejemplo, si le “pregunto” la expresión `2 + 2`, JavaScript me responderá 4.

```
console.log(2 + 2)
```

```
4
```

¿Que es una expresión?

Otro ejemplo de expresión puede ser “preguntarle” a JavaScript el tipo de un valor con `typeof`.

```
console.log(typeof(4))    'number'
```

Y JavaScript nos “responderá” con un valor de tipo String “number”.

Funciones

Las funciones son valores

Como React nos va a brindar funciones para que nosotros construyamos nuestras interfaces, es importante que sepamos qué son.

Definimos funciones para poder llamarlas más tarde y ejecutar el código dentro de ellas.

Sin embargo, para realmente entender las funciones en JavaScript, necesitamos olvidar (por un segundo) por qué son útiles. En su lugar, **pensaremos en las funciones como otro tipo de valor: un número, un objeto, una función.**

Un ejemplo de función

```
let contarManzanas = function() { return 7; };  
let manzanas = contarManzanas;  
console.log(manzanas);
```

```
f contarManzanas()
```

Creamos un nuevo valor de función con una expresión **function() { }**, y apuntamos la variable **contarManzanas** a este valor.

Apuntamos la variable **manzanas** al valor al que apunta **contarManzanas**, que es el mismo valor de función. Mostramos en consola el valor al que apunta **manzanas** en este momento.

Un ejemplo de función

Como resultado, tanto contarManzanas como manzanas apuntan al mismo valor, que resulta ser una función. **Así comprobamos que las funciones son valores.**

Podemos apuntar variables a ellas, al igual que podemos hacerlo con números u objetos.

¿Y si llamo a la función?

```
let contarManzanas = function() { return 7; };  
let manzanas = contarManzanas();  
console.log(manzanas);
```

7

Añadir () cambia el significado de nuestro código:

let manzanas = contarManzanas significa "apuntar manzanas al valor al que apunta contarManzanas en este momento".

let manzanas = contarManzanas() significa "apuntar manzanas al valor que RETORNA la función a la que apunta contarManzanas en este momento".

¿Y si llamo a la función?

Como vemos, **contarManzanas()** también es una expresión.

Se conoce como una expresión de llamada, para "responder" a una expresión de llamada, JavaScript ejecuta el código dentro de nuestra función y nos brinda el valor RETORNADO como resultado (en este ejemplo, es 7).

Vale aclarar que si no se especifica explícitamente un valor de retorno en la función, esta responderá con undefined.



Ejemplo en vivo

Creación de componente con HTML.



Ejemplo en vivo

Creación de componente con Vanilla JavaScript.



Ejemplo en vivo

Creación de componente con React en su forma más pura.

[Guía de prueba de React](#)

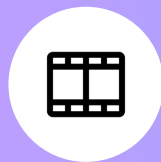
¿Preguntas?



**Completa esta clase
con los siguientes
CoderTips**

Videos y Podcasts

- ✓ [Aprende Programación Web y construye el futuro de nuestra humanidad](#) | Coderhouse
- ✓ [Desarrollo freelance](#) | Coderhouse
- ✓ [Desarrollo profesional](#) | Coderhouse
- ✓ [CoderNews](#) | Coderhouse
- ✓ [Serie de Branding](#) | Coderhouse
- ✓ [Serie para Emprendedores](#) | Coderhouse
- ✓ [Serie Aprende a Usar TikTok](#) | Coderhouse
- ✓ [Serie Finanzas Personales](#) | Coderhouse
- ✓ [CoderConf](#) | Coderhouse



¿Quieres saber más?
**Te dejamos material
ampliado de la clase**



MATERIAL AMPLIADO

Recursos multimedia

- ✓ [HTML](#) | Wikipedia
- ✓ [HTML](#) | w3schools.com
- ✓ [CSS](#) | w3schools.com
- ✓ [Page Inspector](#) | MDN web docs
- ✓ [API](#) | MDN web docs

Disponible en nuestro repositorio.

¿Sabías que
premiamos a nuestros estudiantes
por su dedicación?

Conoce los [beneficios](#) del Top 10

Resumen de la clase hoy

- ✓ Conceptos básicos de React Js
- ✓ Componentes
- ✓ Código declarativo
- ✓ Expresiones
- ✓ Funciones

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación