**BOOTCAMP INTENSIVO DE REACT.JS**

**De Principiante a Profesional**

**Basado en el Roadmap de** [**https://roadmap.sh/react**](https://roadmap.sh/react)

**Academia: AcademyCoder.com**

**Curso Personalizado para:**  
Javier y Naomy

**Duración:** 12 Semanas  
**Horario:** Lunes a Viernes, 8:00 AM - 10:00 AM (2 horas diarias)  
**Fecha de Inicio:** 11 de agosto de 2025  
**Fecha de Finalización:** 31 de octubre de 2025

**Fecha de Elaboración:** 12 de agosto de 2025

**Índice**

1. Introducción al Curso - Página 3
2. Materiales y Recursos Necesarios - Página 4
3. Temario con Explicación de Cada Tema - Página 5  
   3.1 Módulo 1: Fundamentos de React - Página 5  
   3.2 Módulo 2: Conceptos Intermedios - Página 6  
   3.3 Módulo 3: Gestión de Estado y Estilos - Página 7  
   3.4 Módulo 4: Pruebas, Performance y Avanzado - Página 8  
   3.5 Módulo 5: Frameworks y Mejores Prácticas - Página 9  
   3.6 Módulo 6: Proyectos y Carrera - Página 10
4. Cronograma de Estudio - Página 11  
   4.1 Semana 1 - Página 11  
   4.2 Semana 2 - Página 11  
   ...  
   4.12 Semana 12 - Página 16
5. Proyecto Final de Portafolio Profesional - Página 17
6. Incorporación a Upwork y Plataformas Freelance - Página 18
7. Anexos - Página 19  
   7.1 Glosario de Términos - Página 19  
   7.2 Recursos Adicionales - Página 20  
   7.3 Certificado de Finalización (Plantilla) - Página 21

**1. Introducción al Curso**

¡Bienvenidos, Javier y Naomy, al Bootcamp Intensivo de React.js ofrecido por AcademyCoder.com!

Este curso ha sido diseñado de manera personalizada para ustedes dos, fomentando un enfoque colaborativo que incluye sesiones de pair programming (programación en pareja), discusiones grupales diarias y proyectos conjuntos para maximizar el aprendizaje mutuo. React.js es una biblioteca de JavaScript de código abierto desarrollada y mantenida por Meta (anteriormente Facebook), ampliamente utilizada para construir interfaces de usuario (UI) interactivas, eficientes y escalables, especialmente en aplicaciones de una sola página (Single Page Applications o SPA). Este plan de estudios se basa directamente en el roadmap integral de <https://roadmap.sh/react>, que cubre desde los conceptos básicos hasta técnicas avanzadas, asegurando un progreso completo y estructurado de nivel principiante (0) a profesional.

El bootcamp es altamente intensivo, con sesiones diarias de 2 horas de lunes a viernes (8:00 AM - 10:00 AM), combinando explicaciones teóricas, ejercicios prácticos, desafíos de codificación y revisiones de código. Cada semana concluye con un proyecto práctico los viernes para reforzar y aplicar los conocimientos adquiridos. Al finalizar el curso, construirán un portafolio profesional que integre todos los proyectos semanales, y recibirán orientación detallada para incorporarse a plataformas freelance como Upwork y Freelancer.com, facilitando la transición hacia oportunidades laborales como desarrolladores React independientes o en equipo.

**Objetivos Generales del Curso:**

* Dominar los fundamentos y conceptos avanzados de React.js para desarrollar aplicaciones web modernas.
* Construir un portafolio sólido que demuestre habilidades prácticas y proyectos reales.
* Desarrollar habilidades de colaboración, resolución de problemas y mejores prácticas de codificación.
* Prepararse para el mercado laboral freelance, con énfasis en plataformas como Upwork, incluyendo creación de perfiles, redacción de propuestas y búsqueda de clientes.

**Beneficios Específicos para Javier y Naomy:**

* Aprendizaje personalizado: Sesiones adaptadas a su ritmo, con feedback individual y en pareja.
* Enfoque práctico: Más del 50% del tiempo dedicado a codificación en vivo.
* Soporte continuo: Acceso a mentores de AcademyCoder.com vía email o chat durante el curso.
* Certificado: Al finalizar, recibirán un certificado digital personalizado de AcademyCoder.com, válido para LinkedIn y portafolios.

**Requisitos Previos:**  
Conocimientos básicos de HTML, CSS y JavaScript (incluyendo variables, funciones, arrays y objetos). Si es necesario, la primera semana incluirá una revisión rápida de estos temas para alinear sus bases.

**Metodología de Enseñanza:**

* **Estructura Diaria:** 45-60 minutos de teoría y explicación, seguidos de 60-75 minutos de práctica hands-on (ejercicios, coding challenges y pair programming).
* **Evaluación:** Revisiones semanales de proyectos, quizzes rápidos y feedback mutuo entre Javier y Naomy. Progreso medido por completitud de tareas y calidad de código.
* **Tareas Adicionales:** Se recomienda 1-2 horas diarias fuera de clase para revisión de documentación, práctica extra y resolución de desafíos.
* **Herramientas de Colaboración:** Uso de GitHub para repositorios compartidos, Zoom o Google Meet para sesiones en vivo, y Slack para dudas diarias.

Estamos entusiasmados de acompañarlos en este viaje transformador. ¡Con dedicación, se convertirán en desarrolladores React profesionales listos para el mundo laboral!

**2. Materiales y Recursos Necesarios**

Para un aprendizaje óptimo y sin interrupciones, preparen los siguientes materiales y recursos antes del inicio del curso. AcademyCoder.com proporcionará guías de instalación detalladas vía email.

**Hardware Requerido:**

* Computadora personal (laptop o desktop) con procesador Intel Core i5 o equivalente, al menos 8GB de RAM y 256GB de almacenamiento SSD.
* Conexión a internet estable (mínimo 10 Mbps) para sesiones en vivo y descargas.
* Micrófono y cámara web para interacciones en pair programming.

**Software Esencial:**

* **Editor de Código:** Visual Studio Code (VS Code) - Descargar desde <https://code.visualstudio.com/>. Instalar extensiones: ESLint, Prettier, React Developer Tools, Live Share (para colaboración en tiempo real).
* **Entorno de Desarrollo:** Node.js (versión LTS, descargar desde <https://nodejs.org/>) y npm (incluido con Node.js) para gestionar paquetes.
* **Navegador:** Google Chrome con la extensión React Developer Tools instalada (desde Chrome Web Store).
* **Control de Versiones:** Git (instalar desde <https://git-scm.com/>) y cuenta en GitHub (crear en <https://github.com/> para repositorios colaborativos).

**Recursos Educativos Gratuitos:**

* **Documentación Principal:** React.js Official Docs - Referencia completa para todos los temas.
* **Roadmap Base:** <https://roadmap.sh/react> - Mapa visual interactivo para rastrear progreso.
* **Plataformas en Línea:**
  + freeCodeCamp: Sección de Front-End Libraries con React.
  + Codecademy o Udemy: Cursos gratuitos/introductorios de React (buscar "React for Beginners").
  + YouTube: Canales recomendados: Traversy Media ("React Crash Course"), Net Ninja ("React Tutorial for Beginners"), freeCodeCamp ("React Tutorial").
* **Libros y Guías (Gratuitos o Accesibles):**
  + "The Road to React" por Robin Wieruch (versión PDF gratuita en <https://www.roadtoreact.com/>).
  + "React Hooks in Action" por John Larsen (extractos gratuitos en línea).
* **Herramientas Prácticas:**
  + Create React App: Para bootstrapping proyectos (instalar vía npm).
  + CodeSandbox o StackBlitz para prototipado rápido sin instalación local.
  + APIs Gratuitas para Proyectos: JSONPlaceholder (para datos mock), Unsplash API (para imágenes), o PokeAPI (para ejemplos divertidos).

**Recursos Adicionales para Javier y Naomy:**

* Carpeta compartida en Google Drive proporcionada por AcademyCoder.com con plantillas de código, ejercicios resueltos y grabaciones de sesiones.
* Comunidad: Únanse a subreddits como r/reactjs, r/learnreactjs; Stack Overflow para dudas; y el Discord oficial de React para networking.

Todos los recursos son de acceso libre o proporcionados por la academia. Si enfrentan problemas de instalación, contacten a [support@academycoder.com](mailto:support@academycoder.com).

**3. Temario con Explicación de Cada Tema**

El temario está dividido en módulos progresivos, basados en el roadmap de roadmap.sh/react y complementado con fuentes como la documentación oficial de React, guías de MDN Web Docs y experiencias prácticas de desarrolladores profesionales. Cada tema incluye:

* **Explicación Detallada:** Concepto clave, por qué es importante y ejemplos.
* **Objetivos de Aprendizaje:** Qué dominarán al finalizar.
* **Tiempo Estimado:** Integrado en el cronograma, pero con enfoque en práctica.
* **Recursos Específicos:** Enlaces directos para profundizar.

**3.1 Módulo 1: Fundamentos de React (Semanas 1-2)**

Este módulo establece las bases, asumiendo conocimientos básicos de JS. Enfoque en construir componentes simples.

* **Introducción a React y Setup:**  
  Explicación: React es una biblioteca para UI declarativa, usando componentes reutilizables y Virtual DOM para actualizaciones eficientes. Setup incluye instalación de Node.js y creación de apps con npx create-react-app.  
  Objetivos: Crear y ejecutar una app básica, entender el ecosistema.  
  Importancia: Fundamento para todo desarrollo React; evita errores comunes en configuración.  
  Recursos: <https://react.dev/learn/installation>.
* **JSX y Componentes:**  
  Explicación: JSX es una extensión de sintaxis que permite escribir HTML-like en JS. Componentes son funciones o clases que retornan JSX; funcionales son preferidos por simplicidad. Ejemplo: <mycomponent>.<br> Objetivos: Construir y anidar componentes.<br> Importancia: Permite código modular y reutilizable, clave para apps escalables.<br> Recursos: <a href="https://react.dev/learn/writing-markup-with-jsx">https://react.dev/learn/writing-markup-with-jsx</a>.</mycomponent>
* **Props y State:**  
  Explicación: Props pasan datos de padre a hijo (inmutables). State maneja datos internos mutables con setState o useState. Ejemplo: Contador con state.  
  Objetivos: Crear componentes dinámicos.  
  Importancia: Habilita interactividad y flujo de datos unidireccional.  
  Recursos: <https://react.dev/learn/passing-props-to-a-component>.
* **Manejo de Eventos y Formularios:**  
  Explicación: Eventos como onClick usan funciones JS. Formularios controlados vinculan inputs a state. Validación básica con condicionales.  
  Objetivos: Manejar inputs usuario y submits.  
  Importancia: Esencial para apps interactivas como logins o searches.  
  Recursos: <https://react.dev/learn/responding-to-events>.
* **Listas, Keys y Renderizado Condicional:**  
  Explicación: Mapear arrays a elementos JSX con keys únicas para eficiencia. Condicionales con ternary o &&.  
  Objetivos: Renderizar datos dinámicos.  
  Importancia: Optimiza performance en listas grandes, evita bugs de re-render.  
  Recursos: <https://react.dev/learn/rendering-lists>.

**3.2 Módulo 2: Conceptos Intermedios (Semanas 3-5)**

Avance a lógica más compleja, introduciendo hooks para componentes modernos.

* **Ciclo de Vida de Componentes:**  
  Explicación: Métodos como componentDidMount para fetches iniciales; en hooks, useEffect simula esto. Cleanup para evitar leaks.  
  Objetivos: Manejar side effects como API calls.  
  Importancia: Controla el comportamiento temporal de componentes.  
  Recursos: <https://react.dev/learn/lifecycle-of-reactive-effects>.
* **Hooks Básicos (useState, useEffect):**  
  Explicación: useState para estado local; useEffect para efectos (fetches, subscriptions). Dependencias para control.  
  Objetivos: Construir apps funcionales sin clases.  
  Importancia: Estándar moderno; hace código más limpio y legible.  
  Recursos: <https://react.dev/reference/react/useState>.
* **Hooks Avanzados (useContext, useReducer, Custom Hooks):**  
  Explicación: useContext para estado global sin props drilling; useReducer para state complejo como Redux-lite; custom hooks para lógica reutilizable.  
  Objetivos: Apps con estado compartido.  
  Importancia: Mejora escalabilidad y mantenibilidad.  
  Recursos: <https://react.dev/reference/react/useContext>.
* **Enrutamiento con React Router:**  
  Explicación: Biblioteca para navegación en SPA; rutas con <route>, params y hooks como useNavigate.<br> Objetivos: Crear apps multi-vista.<br> Importancia: Fundamental para apps reales como dashboards.<br> Recursos: <a href="https://reactrouter.com/en/main">https://reactrouter.com/en/main</a>.</route>

**3.3 Módulo 3: Gestión de Estado y Estilos (Semanas 6-7)**

Enfoque en estado global y UI estilizada.

* **Gestión de Estado Avanzada (Redux):**  
  Explicación: Patrón Flux con store central, actions, reducers y middleware como thunk para async. Integración con React-Redux.  
  Objetivos: Manejar estado en apps grandes.  
  Importancia: Estándar industry para complejidad; evita caos en props.  
  Recursos: <https://redux.js.org/>.
* **Estilos en React:**  
  Explicación: CSS modules para scoping; styled-components para CSS-in-JS dinámico basado en props.  
  Objetivos: Crear UI responsive y temática.  
  Importancia: Mantiene estilos encapsulados, evita conflictos globales.  
  Recursos: <https://styled-components.com/>.

**3.4 Módulo 4: Pruebas, Performance y Avanzado (Semanas 8-9)**

Técnicas profesionales para código robusto.

* **Pruebas:**  
  Explicación: Jest para assertions; React Testing Library para simular interacciones usuario. Tests unitarios e integración.  
  Objetivos: Escribir tests para componentes.  
  Importancia: Asegura calidad y facilita refactors.  
  Recursos: <https://testing-library.com/docs/react-testing-library/intro/>.
* **Optimización de Performance:**  
  Explicación: Memoization con React.memo/useMemo para evitar re-renders; lazy loading con Suspense.  
  Objetivos: Apps rápidas en datos grandes.  
  Importancia: Mejora UX y reduce carga en producción.  
  Recursos: <https://react.dev/reference/react/memo>.
* **Conceptos Avanzados:**  
  Explicación: HOC para wrappers (ej. withAuth); Render Props para sharing; Error Boundaries para catches; Refs para DOM access.  
  Objetivos: Solucionar patrones complejos.  
  Importancia: Herramientas para edge cases en apps enterprise.  
  Recursos: <https://react.dev/learn/manipulating-the-dom-with-refs>.
* **TypeScript en React:**  
  Explicación: Tipado estático para props/state; interfaces y generics. Integración con Create React App.  
  Objetivos: Código tipado y seguro.  
  Importancia: Reduce errores en equipos grandes; usado en 70% de jobs React.  
  Recursos: <https://www.typescriptlang.org/docs/handbook/react.html>.

**3.5 Módulo 5: Frameworks y Mejores Prácticas (Semanas 10-11)**

Extensión a frameworks y estándares.

* **Server-Side Rendering (Next.js):**  
  Explicación: Framework sobre React para SSR/SSG; getStaticProps para data fetching.  
  Objetivos: Apps con SEO y performance.  
  Importancia: Puente a full-stack; popular en e-commerce.  
  Recursos: <https://nextjs.org/docs>.
* **Mejores Prácticas:**  
  Explicación: Código limpio (DRY, KISS); accesibilidad (ARIA); i18n con react-i18next; security basics.  
  Objetivos: Código production-ready.  
  Importancia: Cumple estándares profesionales y éticos.  
  Recursos: <https://react.dev/learn/keeping-components-pure>.

**3.6 Módulo 6: Proyectos y Carrera (Semana 12)**

Integración y orientación laboral.

* **Proyectos y Portafolio:**  
  Explicación: Construir un sitio que agregue proyectos; deploy en Vercel.  
  Objetivos: Portafolio showcase.  
  Importancia: Evidencia tangible para empleadores.
* **Búsqueda de Empleo:**  
  Explicación: Estrategias para freelance; perfiles optimizados.  
  Objetivos: Primeros jobs.  
  Importancia: Transición práctica al mundo laboral.

**4. Cronograma de Estudio**

Estructura diaria: 8:00-8:45 AM Teoría/Explicación, 8:45-10:00 AM Práctica/Pair Programming. Incluye breaks de 5 min. Tareas: 1-2 hrs extra/día. Proyectos viernes: Desarrollo y revisión.

**4.1 Semana 1: Fundamentos Básicos**

* Lunes: Intro a React y Setup (Instalación, app hello world; práctica: Configurar entorno local).
* Martes: JSX y Componentes Funcionales (Ejemplos, práctica: Componente greeting personalizado).
* Miércoles: Componentes de Clase y Props (Comparación, práctica: Pasar props en familia de componentes).
* Jueves: State y Manejo de Eventos (Ejemplos interactivos, práctica: Botón que actualiza state).
* Viernes: Proyecto 1 - App de Todo List simple (Agregar/eliminar items; revisión en pareja).

**4.2 Semana 2: Interactividad Básica**

* Lunes: Formularios y Renderizado Condicional (Inputs controlados, práctica: Login form con validación).
* Martes: Listas y Keys (Mapping arrays, práctica: Lista dinámica de usuarios).
* Miércoles: Composición vs. Herencia (Patrones, práctica: Componentes compuestos).
* Jueves: Lifting State Up (Compartir state, práctica: App con state compartido).
* Viernes: Proyecto 2 - Formulario de Registro con validación (Email/password, errores condicionales).

**4.3 Semana 3: Ciclo de Vida y Hooks**

* Lunes: Ciclo de Vida de Componentes (Métodos clave, práctica: Fetch en mount).
* Martes: Intro a Hooks (useState en profundidad, práctica: Contador avanzado).
* Miércoles: useEffect para side effects (Dependencias, práctica: Timer con cleanup).
* Jueves: Custom Hooks (Crear uno propio, práctica: Hook para fetch data).
* Viernes: Proyecto 3 - Contador con timers y efectos (Reloj en vivo).

**4.4 Semana 4: Hooks Avanzados y Context**

* Lunes: useReducer (Reducer patterns, práctica: State machine simple).
* Martes: useContext y Context API (Estado global, práctica: Theme switcher).
* Miércoles: Combinar hooks (useEffect + useReducer, práctica: App con múltiples hooks).
* Jueves: Práctica con ejemplos reales (Debugging hooks).
* Viernes: Proyecto 4 - App de Autenticación simple con Context (Login/logout global).

**4.5 Semana 5: Enrutamiento**

* Lunes: Instalación React Router (Setup, práctica: Rutas básicas).
* Martes: Rutas básicas y navegación (Links, práctica: Menú nav).
* Miércoles: Rutas anidadas y params (Dynamic routes, práctica: Perfil usuario).
* Jueves: Protected Routes (Auth guards, práctica: Ruta privada).
* Viernes: Proyecto 5 - Blog multi-página con router (Posts, detalles).

**4.6 Semana 6: Gestión de Estado con Redux**

* Lunes: Intro a Redux (Store/actions, práctica: Setup básico).
* Martes: Reducers y dispatch (Lógica, práctica: Counter con Redux).
* Miércoles: Conectar Redux a React (useSelector/dispatch, práctica: Integración).
* Jueves: Middleware (Thunk para async, práctica: API call con thunk).
* Viernes: Proyecto 6 - App de E-commerce con carrito (Agregar/quitar items).

**4.7 Semana 7: Estilos y Pruebas**

* Lunes: CSS Modules (Scoping, práctica: Estilos locales).
* Martes: Styled-Components (CSS-in-JS, práctica: Componentes temáticos).
* Miércoles: Intro a Jest (Setup/tests, práctica: Test simple).
* Jueves: Testing Library para componentes (Simulaciones, práctica: Test formulario).
* Viernes: Proyecto 7 - Dashboard estilizado con tests básicos (UI + coverage).

**4.8 Semana 8: Performance y Optimización**

* Lunes: React.memo y useMemo (Memoization, práctica: Lista optimizada).
* Martes: useCallback y code splitting (Funcs memo, práctica: Lazy components).
* Miércoles: Lazy loading con Suspense (Práctica: App con carga diferida).
* Jueves: Análisis de performance (Dev tools, práctica: Profiling).
* Viernes: Proyecto 8 - App optimizada con listas grandes (Tabla datos).

**4.9 Semana 9: Conceptos Avanzados**

* Lunes: Refs y Forward Refs (DOM access, práctica: Focus input).
* Martes: Error Boundaries (Catch errors, práctica: Componente seguro).
* Miércoles: HOC y Render Props (Patrones, práctica: HOC auth).
* Jueves: Portals (Modals fuera DOM, práctica: Dialog portal).
* Viernes: Proyecto 9 - Modal avanzado con HOC (Custom modal).

**4.10 Semana 10: TypeScript en React**

* Lunes: Intro a TypeScript (Setup, tipos básicos).
* Martes: Tipado de props y state (Interfaces, práctica: Componente tipado).
* Miércoles: Hooks con TS (useState tipado, práctica: Hooks seguros).
* Jueves: Integración con Redux (Types en actions, práctica: Redux TS).
* Viernes: Proyecto 10 - Refactorizar app anterior a TS (Todo list tipada).

**4.11 Semana 11: SSR y Mejores Prácticas**

* Lunes: Intro a Next.js (Setup, páginas).
* Martes: Páginas y routing en Next.js (Práctica: Blog SSR).
* Miércoles: Data fetching (getStaticProps/getServerSideProps).
* Jueves: Mejores prácticas (Accesibilidad, i18n, seguridad).
* Viernes: Proyecto 11 - Blog SSR con Next.js (Posts dinámicos).

**4.12 Semana 12: Proyectos Finales y Carrera**

* Lunes: Construir Portafolio (Integrar proyectos, Next.js base).
* Martes: Desplegar portafolio (Vercel/Netlify, práctica: Deploy live).
* Miércoles: Crear perfiles en Upwork y Freelancer.com (Guía paso a paso).
* Jueves: Consejos para propuestas, gigs React y búsqueda de clientes (Templates).
* Viernes: Proyecto Final - Portafolio Profesional (Presentación, feedback).

**5. Proyecto Final de Portafolio Profesional**

El culmen del curso es un portafolio web profesional construido en React con Next.js para SSR, integrando todos los proyectos semanales.

**Descripción Detallada:**

* **Estructura:** Página de inicio con bios de Javier y Naomy (fotos, resúmenes). Galería de proyectos (cards con descripciones, links a demos/GitHub). Sección de habilidades (lista React topics). Formulario de contacto (integrado con EmailJS o similar).
* **Integración de Proyectos:** Cada proyecto semanal como demo embebido o enlace (ej. Todo List en iframe).
* **Tecnologías:** React, Next.js, Styled-Components, deploy en Vercel.
* **Objetivos:** Demostrar dominio completo; usable para job applications.
* **Evaluación:** Revisión por mentores; debe ser responsive, accesible y optimizado.
* **Beneficios:** Sirve como CV digital; facilita networking en LinkedIn/Upwork.

Tiempo: Toda la Semana 12, con soporte para debugging.

**6. Incorporación a Upwork y Plataformas Freelance**

Para facilitar la búsqueda profesional de trabajo, esta sección proporciona una guía paso a paso enfocada en React developers.

**Upwork:**

* Crear perfil: Usa foto profesional, título "React.js Developer | Front-End Specialist". Agrega portafolio link, skills (React, Hooks, Redux). Toma tests gratuitos en Upwork (JS, React).
* Búsqueda de Jobs: Filtra por "React", "Front-End". Envía propuestas personalizadas (ej. "Basado en mi experiencia con Redux, puedo optimizar su app en 2 semanas"). Precio inicial: $20-30/hora.
* Consejos: Construye reseñas con gigs pequeños ($50-100). Usa templates de AcademyCoder.com.

**Freelancer.com:**

* Similar a Upwork: Perfil con portafolio, participa en contests React. Ofertas en projects como "Build React Dashboard".

**Otras Plataformas:** Fiverr para gigs fijos, LinkedIn para networking.

**Estrategias Generales:**

* Actualiza CV con certificado AcademyCoder.com.
* Busca clientes: Groups en Facebook/Reddit para React freelancers.
* Metas Iniciales: 1-2 gigs/mes para construir portfolio real.

AcademyCoder.com ofrece revisión de perfiles y propuestas post-curso.

**7. Anexos**

**7.1 Glosario de Términos**

* **Componente:** Bloque reutilizable de UI en React.
* **Hook:** Función que permite usar state/effects en componentes funcionales.
* **Redux:** Librería para gestión de estado global.
* **SSR:** Server-Side Rendering, renderiza en servidor para SEO.
* *(Ampliar con 20+ términos clave para referencia rápida.)*

**7.2 Recursos Adicionales**

* Lista extendida: Podcasts (Syntax.fm), Blogs (Overreacted por Dan Abramov).
* Challenges: LeetCode para JS, HackerRank para React.