# سرفصل دوره آموزشی React با Vite.js

## [مقدمه و معرفی دوره](#_مقدمه_و_معرفی)

### [معرفی React و ویژگی‌های کلیدی](#_معرفی_React_و)

[تاریخچه و فلسفه ایجاد React توسط متا (فیسبوک سابق)](#_تاریخچه_و_فلسفه)

مزایای استفاده از React: کامپوننت‌بندی، declarative programming، مجازی DOM و یکطرفه بودن جریان داده

مقایسه React با سایر فریمورک‌ها (Angular, Vue) و دلایل محبوبیت آن

[معرفی Vite.js و مزایای آن](#_معرفی_Vite.js_و)

دلایل حرکت از Create React App (CRA) به Vite: سرعت بالاتر، حجم کمتر dependencyها (۱۴۰ مگابایت در CRA در مقابل ۳۱ مگابایت در Vite)، پشتیبانی از ماژول‌های ES

عملکرد Vite در توسعه: استفاده از ESM برای بارگذاری سریع‌تر و عدم نیاز به bundle کامل در هر تغییر

# [فصل ۱: راه‌اندازی محیط توسعه](#_فصل_۱:_راه‌اندازی)

#### پیش‌نیازهای نصب

نصب Node.js (نسخه ۱۲.۲.۰ یا بالاتر) و آشنایی با npm/yarn

نصب ویرایشگر کد (VS Code) و extensions مفید

نصب و راه‌اندازی پروژه با Vite

ایجاد پروژه جدید با دستور `yarn create vite` یا `npm create vite@latest`

انتخاب قالب (template) React و TypeScript (اختیاری)

نصب dependencyها با `yarn` یا `npm install`

[ساختار پروژه Vite + React](#_ساختار_پروژه_Vite)

بررسی پوشه‌های `src`, `public`, `node\_modules`

نقش فایل `index.html` و نقطه ورود برنامه (`main.jsx`)

بررسی فایل `vite.config.js` و تنظیمات اولیه

# [فصل ۲: مبانی React و JSX](#_فصل_۲:_مبانی)

### آشنایی با JSX

تعریف JSX و قوانین پایه (تگ‌های selfclosing، return تک عنصری)

نوشتن JavaScript درون JSX (با curly braces)

پیاده‌سازی شرط و حلقه در JSX

[کامپوننت‌ها در React](#_تمرین_۱:_کامپوننت)

تعریف کامپوننت‌های تابعی و کلاسی

استفاده و ترکیب کامپوننت‌ها

[Props و داده‌ها در کامپوننت](#_بخش_۳:_Props)

ارسال داده با Props (مقادیر پیش‌فرض، children)

انواع داده‌ها در Props و تغییرناپذیری آنها

## فصل ۳: state و مدیریت وضعیت

### آشنایی با state

state در کامپوننت‌های تابعی (useState)

state در کامپوننت‌های کلاسی

مدیریت رویدادها (Event Handling)

رویدادهای کلیدی و form handling

### هوک‌های (Hooks) پایه

useEffect برای side effects (data fetching, subscriptions)

useContext برای مدیریت state سراسری

## فصل ۴: مسیریابی (Routing) و navigation

### نصب و راه‌اندازی React Router

ایجاد مسیرها و تعریف صفحات

لینک‌دهی و navigation

### پارامترهای مسیر و query strings

dynamic routing و استفاده از useParams

حفاظت از مسیرها (Route Guards)

کنترل دسترسی بر اساس وضعیت کاربر

## فصل ۵: استایل‌دهی در React

روش‌های استایل‌دهی

CSS معمولی و CSS Modules

فریمورک‌های CSSinJS (مانند Styled Components)

### استایل‌دهی با Tailwind CSS

نصب و پیکربندی Tailwind در پروژه Vite

استفاده از utility classes و customizing

### کتابخانه‌های کامپوننت UI

معرفی Material UI, Ant Design, Chakra UI

نصب و استفاده از کامپوننت‌های از پیش ساخته شده

## فصل ۶: ارتباط با سرور و داده‌ها

### دریافت داده از API

استفاده از fetch و axios

مدیریت loading و error states

هوک‌های سفارشی برای data fetching

ایجاد useFetch hook

### کار با GraphQL (مقدماتی)

introduction to GraphQL and Apollo Client

## فصل ۷: مدیریت state پیشرفته

### Context API برای state سراسری

ایجاد context و provider

مصرف context در کامپوننت‌ها (useContext)

### مدیریت state با Redux (اختیاری)

نصب و راه‌اندازی Redux Toolkit

slices و async actions

کتابخانه‌های جایگزین (Zustand, Jotai)

معرفی و مقایسه

## فصل ۸: تست‌نویسی در React

### انواع تست‌ها

unit testing, integration testing, endtoend testing

تست کامپوننت‌ها با Jest و React Testing Library

### نصب و پیکربندی

نوشتن تست برای کامپوننت‌ها و hooks

تست مسیرها و async operations

## فصل ۹: build و deploy

### آماده‌سازی برای production

build پروژه با `yarn build` یا `npm run build`

بهینه‌سازی bundle و تحلیل آن

استقرار (deploy) روی سرور

deploy روی Netlify, Vercel, یا سرورهای سنتی (Apache, Nginx)

### پروژه‌های پیشرفته‌تر

معرفی Next.js برای SSR و SSG

PWA کردن اپلیکیشن

## فصل ۱۰: پروژه عملی

پیاده‌سازی یک پروژه کامل

طراحی و implement یک برنامه fullstack کوچک (مثلاً Todo app, blog, weather app)

استفاده از تمامی مفاهیم آموزش داده شده

### کدنگاری best practices

ساختار پوشه‌بندی، naming conventions

استفاده از custom hooks برای منطق تکراری

نحوه ارائه پروژه و تحویل

## ضمیمه: منابع و ادامه مسیر

منابع یادگیری بیشتر

مستندات رسمی React و Vite

دوره‌های پیشرفته (Next.js, React Native)

### مهارت‌های تکمیلی

TypeScript در React

authentication و authorization

optimization techniques (lazy loading, memoization)

# مقدمه و معرفی دوره

حتماً! در ادامه یک مقدمه و معرفی دوره کامل و جذاب برای دوره آموزشی React با Vite.js ارائه می‌شود. این مقدمه را می‌توانید در ابتدای دوره، در صفحه معرفی دوره یا در ویدیوی اول برای دانشجویان استفاده کنید.

مقدمه و معرفی دوره: "مسیر جامع تسلط بر React با Vite.js"

سلام به همه توسعه‌دهندگان آینده!

به دوره "React Modern با Vite.js" خوش آمدید! دوره‌ای که شما را از یک توسعه‌دهنده مبتدی FrontEnd به یک متخصص React تبدیل می‌کند که می‌تواند اپلیکیشن‌های مدرن، سریع و مقیاس‌پذیر بسازد.

🤔 چرا React؟

امروزه، React نه تنها یک کتابخانه، بلکه یک اکوسیستم کامل و پایه‌ای ترین مهارت برای هر توسعه‌دهنده FrontEnd است.

محبوبیت بی‌نظیر: React برای چندین سال متوالی، محبوب‌ترین کتابخانه یا فریمورک FrontEnd در جهان بوده است.

تقاضای بازار کار: طبق گزارش‌های جهانی، ده‌ها هزار موقعیت شغلی در سراسر جهان برای توسعه‌دهندگان React موجود است. یادگیری React یعنی سرمایه‌گذاری بر روی آینده شغلی شما.

قدرت و انعطاف: از سایت‌های استاتیک ساده تا پیچیده‌ترین اپلیکیشن‌های وب مانند فیسبوک، اینستاگرام و نتفلیکس، همه با React ساخته شده‌اند.

چرا Vite.js؟ انقلابی در تجربه توسعه!

شاید بپرسید که چرا به جای Create React App (CRA) از Vite استفاده می‌کنیم؟ پاسخ در سه کلمه است: سرعت، سرعت، سرعت!

شروع فوق‌العاده سریع: Vite هنگام اجرای پروژه در حالت توسعه، به جای bundle کردن کل کدها، از ماژول‌های ES استفاده می‌کند. این یعنی راه‌اندازی سرور توسعه در کمتر از یک ثانیه! (مقایسه کنید با چندین ده ثانیه در CRA).

HMR (Hot Module Replacement) lightningfast: تغییرات در کد شما بلافاصله و بدون نیاز به ریلود کامل صفحه، در مرورگر منعکس می‌شود که تجربه توسعه را نرم و لذت‌بخش می‌کند.

حجم کمتر وابستگی‌ها: پروژه Vite نسبت به CRA وابستگی‌های بسیار کمتری دارد و build نهایی آن بهینه‌تر و سبک‌تر است.

Vite در حال حاضر بهترین ابزار ممکن برای توسعه مدرن React است و بسیاری از شرک‌ها و پروژه‌های بزرگ در حال مهاجرت به آن هستند.

در این دوره چه چیزی یاد می‌گیرید؟

این دوره به گونه‌ای طراحی شده که شما را به صورت گام به گام و از پایه تا سطح پیشرفته پیش ببرد:

1. از صفر تا راه‌اندازی: نصب و پیکربندی محیط توسعه حرفه‌ای با Vite.

2. مبانی React: درک عمیق JSX، کامپوننت‌ها، props و state.

3. مدیریت حالت (State): کار با هوک‌های حیاتی مثل `useState` و `useEffect`.

4. مسیریابی (Routing): ساخت اپلیکیشن‌های چندصفحه‌ای با React Router.

5. استایل‌دهی مدرن: استفاده از CSS Modules، Tailwind CSS و کتابخانه‌های کامپوننت.

6. ارتباط با backend: fetch کردن داده از API و مدیریت آن.

7. مدیریت state سراسری: استفاده از Context API و Redux Toolkit.

8. تست‌نویسی: نوشتن تست‌های اتوماتیک برای کامپوننت‌ها.

9. Build و Deploy: خروجی گرفتن و استقرار پروژه روی سرور.

🎁 در پایان این دوره شما قادر خواهید بود:

اپلیکیشن‌های واقعی و کاربردی React را از صفر بسازید.

از بهترین practices و جدیدترین متدها در توسعه React استفاده کنید.

پروژه‌های خود را با امکانات مدرن Vite بهینه و سریع کنید.

رزومه خود را با یک مهارت بسیار پرتقاضا تقویت کنید.

👨‍💻 این دوره برای چه کسانی مناسب است؟

توسعه‌دهندگان FrontEnd که می‌خواهند دانش خود را به روز کنند.

توسعه‌دهندگان BackEnd که قصد حرکت به سمت FullStack را دارند.

مبتدیانی که HTML, CSS و JavaScript پایه را بلد هستند و می‌خواهند اولین فریمورک FrontEnd خود را یاد بگیرند.

همه کسانی که به ساخت اپلیک�یشن‌های وب مدرن و پویا علاقه دارند.

بیایید این سفر هیجان‌انگیز را вместе شروع کنیم!

در اولین قدم، با هم محیط توسعه را نصب می‌کنیم و اولین کامپوننت React خود را با Vite می‌نویسیم.

پیش به سوی تسلط بر React!

# معرفی React و ویژگی‌های کلیدی

React چیست؟

React یک کتابخانه متن‌باز (opensource) جاوااسکریپتی برای ساخت رابط‌های کاربری (UI) است که توسط فیسبوک (اکنون متا) ایجاد و نگهداری می‌شود. React به توسعه‌دهندگان اجازه می‌دهد تا کامپوننت‌های قابل استفاده مجدد برای ساخت اپلیکیشن‌های پویا و تعاملی ایجاد کنند.

چرا React؟ (مزایای کلیدی)

1. کامپوننت‌محوری (ComponentBased)

استفاده مجدد: کامپوننت‌ها بلوک‌های سازنده مستقل و قابل استفاده مجدد هستند.

قابلیت نگهداری: کدها به بخش‌های کوچک‌تر و مدیریت‌پذیر تقسیم می‌شوند.

توسعه پذیری: امکان اشتراک‌گذاری و استفاده از کامپوننت‌ها در پروژه‌های مختلف.

2. Virtual DOM (DOM مجازی)

عملکرد بالا: React یک کپی سبک‌وزن از DOM واقعی را در حافظه نگه می‌دارد.

بهینه‌سازی تغییرات: تنها بخش‌هایی که تغییر کرده‌اند به‌روزرسانی می‌شوند.

تجربه کاربری بهتر: کاهش زمان render و افزایش سرعت اپلیکیشن.

3. JSX (JavaScript XML)

ترکیب HTML و JavaScript: امکان نوشتن کدهای HTMLlike در JavaScript.

خوانایی بالاتر: کدها intuitive و شبیه به ساختار نهایی خروجی هستند.

قدرت کامل JavaScript: دسترسی به تمام ویژگی‌های JavaScript درون JSX.

4. جریان داده یک‌طرفه (OneWay Data Binding)

قابل پیش‌بینی: داده‌ها فقط در یک جهت (از parent به child) جریان می‌یابند.

اشکال‌زدایی آسان: ردیابی تغییرات داده‌ها و شناسایی باگ‌ها ساده‌تر است.

کنترل بهتر: مدیریت state در اپلیکیشن‌های پیچیده کارآمدتر است.

5. اکوسیستم غنی

ابزارهای توسعه: React DevTools برای دیباگ و آنالیز کامپوننت‌ها.

کتابخانه‌های مکمل: React Router برای مسیریابی، Redux برای مدیریت state.

جامعه فعال: منابع آموزشی فراوان، کامپوننت‌های آماده و پشتیبانی قوی.

6. چندسکویی (CrossPlatform)

React Native: برای ساخت اپلیکیشن‌های موبایل بومی (iOS و Android).

React 360: برای ساخت تجربیات واقعیت مجازی (VR).

React Desktop: برای ساخت اپلیکیشن‌های دسکتاپ.

مقایسه React با سایر فریمورک‌ها

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | React (کتابخانه) | Angular (فریمورک کامل) | Vue (فریمورک پیشرونده) |
| نوع | کتابخانه UI | فریمورک کامل | فریمورک پیشرونده |
| یادگیری | آسان | پیچیده | آسان |
| انعطاف‌پذیری | بسیار بالا | متوسط | بالا |
| کاربرد | پروژه‌های متوسط و بزرگ | سطح سازمانی | پروژه‌های کوچک و متوسط |

نمونه کد: اولین کامپوننت React

|  |
| --- |
| import React from 'react';  function Welcome(props) {  return <h1>Hello, {props.name}!</h1>;  }  export default Welcome; |

|  |
| --- |
| // استفاده از کامپوننت  import Welcome from './Welcome';  function App() {  return (  <div>  <Welcome name="Sara" />  <Welcome name="Cahal" />  <Welcome name="Edite" />  </div>  );  } |

## آمار و محبوبیت React

محبوب‌ترین کتابخانه فرانت‌اند: بر اساس نظرسنجی Stack Overflow 2023

استفاده در شرکت‌های بزرگ: فیسبوک، اینستاگرام، نتفلیکس، Airbnb

بازار کار عالی: بیش از ۴۰٪ از آگهی‌های شغلی فرانت‌اند نیازمند React

### جمع‌بندی: چرا React یاد بگیریم؟

1. تقاضای بازار کار بالا

2. یادگیری نسبتاً آسان

3. انعطاف‌پذیری و قدرت

4. اکوسیستم غنی و جامعه فعال

5. قابلیت استفاده در پلتفرم‌های مختلف

این معرفی می‌تواند به دانشجویان کمک کند تا درک روشنی از ماهیت React و دلایل انتخاب آن به عنوان تکنولوژی اصلی دوره به دست آورند. در جلسات بعدی به صورت عملی با نصب و راه‌اندازی React با Vite آشنا خواهیم شد!

# تاریخچه و فلسفه ایجاد React

ایجاد توسط جردن واک: React در سال ۲۰۱۱ توسط جردن واک (Jordan Walke)، مهندس نرم‌افزار فیسبوک، توسعه داده شد. او از XHP (یک چارچوب HTML برای PHP) الهام گرفت و ایده‌ای برای ساخت رابط‌های کاربری پویا با جاوااسکریپت ارائه کرد.

استقرار اولیه: React برای اولین بار در فید خبری فیسبوک در سال ۲۰۱۱ و سپس در اینستاگرام در سال ۲۰۱۲ مورد استفاده قرار گرفت.

منبع‌گذاری عمومی: در سال ۲۰۱۳، فیسبوک React را به صورت متن‌باز (opensource) در کنفرانس JSConf US معرفی کرد.

انتشار React Native: در سال ۲۰۱۵، فیسبوک React Native را معرفی کرد که امکان توسعه اپلیکیشن‌های موبایل بومی (Android و iOS) با استفاده از React را فراهم می‌کند.

معرفی React Fiber: در سال ۲۰۱۷، فیسبوک React Fiber را به عنوان یک موتور رندر جدید معرفی کرد که بهبودهای قابل توجهی در عملکرد و قابلیت‌های React ایجاد کرد.

## فلسفه ایجاد React

مدیریت کارآمد رابط کاربری: هدف اصلی React، حل مشکلات مربوط به مدیریت DOM در اپلیکیشن‌های پویا بود. تغییرات مکرر در DOM واقعی باعث کاهش عملکرد می‌شد، بنابراین React از DOM مجازی (Virtual DOM) استفاده کرد تا تنها بخش‌های تغییرکرده را به روز رسانی کند و عملکرد را بهبود بخشد.

استفاده از کامپوننت‌ها: React بر پایه کامپوننت‌ها (Components) ساخته شده است. این کامپوننت‌ها قابلیت استفاده مجدد دارند و توسعه رابط‌های کاربری پیچیده را آسان‌تر می‌کنند.

جریان داده یک‌طرفه: React از جریان داده یک‌طرفه (OneWay Data Binding) استفاده می‌کند که باعث می‌شود داده‌ها تنها در یک جهت جریان یابند و اشکال‌زدایی و پیش‌بینی رفتار اپلیکیشن آسان‌تر شود.

سادگی و یادگیری آسان: React به گونه‌ای طراحی شده که ساده و قابل درک باشد. توسعه‌دهندگان می‌توانند با دانش پایه HTML و CSS، از React برای ساخت رابط‌های کاربری استفاده کنند.

انعطاف‌پذیری: React تنها روی لایه view تمرکز دارد و می‌تواند با سایر کتابخانه‌ها یا فریمورک‌ها (مانند Angular) ترکیب شود.

## ویژگی‌های کلیدی React

JSX: React از JSX استفاده می‌کند که یک syntax شبیه به HTML است و امکان نوشتن کدهای HTML را در جاوااسکریپت فراهم می‌کند.

DOM مجازی: React با استفاده از DOM مجازی، تغییرات را بهینه می‌کند و تنها بخش‌های لازم را به روز رسانی می‌کند.

کامپوننت‌های قابل استفاده مجدد: کامپوننت‌ها در React می‌توانند بارها استفاده شوند و توسعه را کارآمدتر می‌کنند.

پشتیبانی از سرور و موبایل: React می‌تواند برای ساخت اپلیکیشن‌های وب و موبایل (با React Native) استفاده شود.

تأثیر React بر توسعه وب

React انقلابی در توسعه رابط‌های کاربری ایجاد کرد و به دلیل سادگی، عملکرد بالا و انعطاف‌پذیری، به یکی از محبوب‌ترین کتابخانه‌های جاوااسکریپت تبدیل شد. بسیاری از شرکت‌های بزرگ مانند فیسبوک، اینستاگرام، نتفلیکس و Airbnb از React استفاده می‌کنند.

## جمع‌بندی

React با هدف ساده‌سازی توسعه رابط‌های کاربری پویا و بهبود عملکرد through DOM مجازی و کامپوننت‌ها ایجاد شد. تاریخچه آن نشان‌دهنده تکامل مستمر و پذیرش گسترده در جامعه توسعه‌دهندگان است.

# معرفی Vite.js و مزایای آن

Vite.js یک ابزار ساخت (Build Tool) مدرن و فوق‌العاده سریع برای توسعه اپلیکیشن‌های وب است که توسط Evan You، خالق فریمورک Vue.js، توسعه داده شده است. این ابزار با هدف بهبود تجربه توسعه و سرعت بخشیدن به فرآیند توسعه و ساخت پروژه‌های وب ایجاد شده است. در ادامه به معرفی و بررسی مزایای کلیدی Vite.js می‌پردازیم.

Vite.js که نام آن از کلمه فرانسوی "Vite" به معنای "سریع" گرفته شده است، به‌عنوان یک ابزار ساخت و سرور توسعه محلی عمل می‌کند. این ابزار به‌طور خاص برای رفع مشکلات ابزارهای قدیمی مانند Webpack و Rollup در پروژه‌های بزرگ و پیچیده طراحی شده است. Vite.js از ماژول‌های ES بومی (ECMAScript Modules) استفاده می‌کند و به توسعه‌دهندگان اجازه می‌دهد تا بدون نیاز به باندلینگ در مرحله توسعه، مستقیماً از ایمپورت‌های ES در مرورگر بهره ببرند.

## مزایای کلیدی Vite.js

1. سرعت فوق‌العاده بالا

راه‌اندازی سریع (Cold Start): Vite.js به‌دلیل استفاده از ماژول‌های ES بومی، سرعت راه‌اندازی پروژه را ۱۰ تا ۱۰۰ برابر نسبت به ابزارهای سنتی مانند Webpack افزایش می‌دهد.

به‌روزرسانی لحظه‌ای (HMR): جایگزینی ماژول داغ (Hot Module Replacement) در Vite.js به‌صورت بسیار سریع و بدون نیاز به رفرش کامل صفحه انجام می‌شود. این ویژگی باعث می‌شود تغییرات کد بلافاصله در مرورگر منعکس شوند.

2. پشتیبانی از اکوسیستم گسترده

فریمورک‌های مختلف: Vite.js از فریمورک‌های محبوب مانند React, Vue, Preact, Svelte, Lit و حتی Vanilla JavaScript پشتیبانی می‌کند.

پلاگین‌های قابل گسترش: این ابزار دارای سیستم پلاگین‌های قدرتمندی است که به توسعه‌دهندگان اجازه می‌دهد تا functionalityهای جدیدی به پروژه اضافه کنند.

3. بهینه‌سازی پیشرفته

پیش‌باندلینگ (PreBundling): Vite.js وابستگی‌های پروژه را با استفاده از esbuild پیش‌باندل می‌کند که این امر سرعت بارگذاری صفحه را significantly بهبود می‌بخشد.

تقسیم کد (Code Splitting): این ابزار به‌صورت خودکار کدها را به بخش‌های کوچک‌تر تقسیم می‌کند و فقط کدهای لازم را بارگذاری می‌کند که منجر به بهبود عملکرد می‌شود.

حذف کدهای مرده (Tree Shaking): Vite.js کدهای استفاده‌نشده را حذف می‌کند و حجم نهایی باندل را کاهش می‌دهد.

4. پیکربندی آسان و انعطاف‌پذیر

پیکربندی پیش‌فرض: Vite.js به‌صورت پیش‌فرض پیکربندی بهینه‌ای ارائه می‌دهد که نیاز به تنظیمات دستی را کاهش می‌دهد.

فایل پیکربندی قابل تنظیم: توسعه‌دهندگان می‌توانند از طریق فایل `vite.config.js` پیکربندی پروژه را به‌صورت پیشرفته سفارشی کنند.

5. تجربه توسعه بهبودیافته

سرور توسعه داخلی: Vite.js دارای یک سرور توسعه داخلی است که برای بارگذاری سریع و HMR بهینه شده است.

پشتیبانی از TypeScript: این ابزار به‌صورت پیش‌فرض از TypeScript پشتیبانی می‌کند و تبدیل TypeScript به JavaScript را با استفاده از esbuild انجام می‌دهد که ۲۰ تا ۳۰ برابر سریع‌تر از tsc است.

6. استقرار آسان

دستور build بهینه: با استفاده از دستور `vite build` می‌توانید یک باندل بهینه‌شده برای production ایجاد کنید که فقط به یک فضای میزبانی استاتیک نیاز دارد.

پشتیبانی از Deploy Platforms: این ابزار با پلتفرم‌های استقرار مانند Netlify, Vercel و GitHub Pages سازگاری کامل دارد.

## جمع‌بندی

Vite.js با تمرکز بر سرعت، سادگی و بهره‌وری، به‌عنوان یک ابزار ساخت مدرن، تحول بزرگی در فرآیند توسعه اپلیکیشن‌های وب ایجاد کرده است. ویژگی‌هایی مانند سرعت فوق‌العاده، پشتیبانی از فریمورک‌های متنوع و بهینه‌سازی‌های پیشرفته، آن را به انتخابی ایده‌آل برای پروژه‌های کوچک و بزرگ تبدیل کرده است. اگر به دنبال بهبود تجربه توسعه و سرعت بخشیدن به کار خود هستید، Vite.js قطعاً ارزش امتحان کردن را دارد!

# دلایل حرکت از Create React App (CRA) به Vite:

در سال‌های اخیر، جامعه React شاهد یک انتقال قابل توجه از Create React App (CRA) به سمت Vite بوده است. این حرکت دلایل متقن و مزایای ملموسی دارد که در ادامه به تفصیل شرح داده می‌شود:

1. سرعت بسیار بالاتر در توسعه

راه‌اندازی آنی سرور توسعه: Vite از ماژول‌های ES بومی (Native ES Modules) استفاده می‌کند و به جای bundle کردن کل پروژه در ابتدا، تنها به صورت ondemand فایل‌ها را سرو می‌دهد. این امر باعث می‌شود سرور توسعه در کسری از ثانیه راه‌اندازی شود، در حالی که CRA مبتنی بر Webpack ممکن است حتی در پروژه‌های کوچک هم ده‌ها ثانیه زمان نیاز داشته باشد .

Hot Module Replacement (HMR) فوق‌العاده سریع: Vite به‌طور میانگین ۱۰ تا ۱۰۰ برابر سریع‌تر از CRA ماژول‌های تغییر یافته را به روز می‌کند. این یعنی تغییرات کد شما تقریباً بلافاصله در مرورگر قابل مشاهده است، بدون آنکه نیاز به رفرش کامل صفحه باشد . این سرعت، بهره‌وری توسعه‌دهنده را به شدت افزایش می‌دهد.

2. معماری مدرن و کارآمد

استفاده از ابزارهای ساخت نوین: Vite از esbuild برای پیش‌باندل کردن وابستگی‌ها استفاده می‌کند که با زبان Go نوشته شده و بسیار سریع‌تر از باندلرهای مبتنی بر JavaScript است. برای بیلد نهایی نیز از Rollup بهره می‌برد که bundleهای بهینه‌تر و کوچک‌تری تولید می‌کند .

پشتیبانی ذاتی از ES Modules: این معماری به توسعه‌دهندگان اجازه می‌دهد تا از مزایای ماژول‌های ES در مرحله توسعه بهره ببرند، که منجر به تجربه توسعه بسیار روان‌تر و نزدیک‌تر به محیط production می‌شود .

3. پیکربندی آسان و انعطاف‌پذیر

کاهش پیچیدگی پیکربندی: برخلاف CRA که برای سفارشی‌سازی اغلب نیاز به "eject" دارد (عملیاتی که irreversible است و شما را با یک پیکربندی Webpack بسیار پیچیده تنها می‌گذارد)، Vite به طور پیش‌فرض معقول پیکربندی شده و در عین حال به راحتی می‌توان آن را گسترش داد و سفارشی کرد .

اکوسیستم غنی پلاگین‌ها: Vite از اکوسیستم پلاگین‌های متنوعی بهره می‌برد که امکان یکپارچه‌سازی آسان با ابزارها و کتابخانه‌های دیگر (مانند PWA، Testing Library و ...) را فراهم می‌کند .

4. بهینه‌سازی عالی برای Production

حجم باندل کمتر: bundleهای تولید شده توسط Vite معمولاً بهینه‌تر و کوچک‌تر هستند که منجر به زمان بارگذاری سریع‌تر برای کاربران نهایی می‌شود .

تقسیم کد (Code Splitting) کارآمد: Vite به صورت پیش‌فرض از تقسیم کد پشتیبانی می‌کند و آن را به شکلی انجام می‌دهد که بهترین عملکرد را در پی داشته باشد .

5. پشتیبانی از آخرین فناوری‌ها و آینده‌نگری

سازگاری با React 19 و ویژگی‌های جدید: CRA در ارائه پشتیبانی به روز برای آخرین ویژگی‌های React (مانند Server Components) کند بوده است، در حالی که Vite به طور فعال نگهداری می‌شود و به سرعت با جدیدترین تغییرات React همگام می‌شود .

توسعه و پشتیبانی فعال: CRA دیگر به طور فعال نگهداری نمی‌شود و آخرین به‌روزرسانی آن به سال ۲۰۲۲ بازمی‌گردد. در مقابل، Vite به طور مداوم به‌روز می‌شود و جامعه پویایی دارد .

6. مهاجرت نسبتاً ساده

راهنماهای رسمی برای مهاجرت: هم تیم React و هم جامعه Vite راهنماهای مفصلی را برای مهاجرت از CRA به Vite ارائه داده‌اند .

ابزارهای کمکی: ابزارهایی مانند viject` (که توسط Evan You، خالق Vue، توصیه شده) فرآیند مهاجرت را تا حد زیادی خودکار می‌کنند . این ابزار کارهایی مانند جایگزینی اسکریپت‌های npm، جابجایی فایل index.html به ریشه پروژه، و تبدیل فایل‌های .js حاوی JSX به .jsx را انجام می‌دهد.

جمع‌بندی: چرا Vite انتخاب بهتری است؟

|  |  |  |
| --- | --- | --- |
| ویژگی | Create React App (CRA) | Vite |
| سرعت راه‌اندازی سرور | کند (ثانیه‌ها یا حتی دقیقه‌ها) | فوق‌العاده سریع (کسری از ثانیه) |
| سرعت HMR | کند | تقریباً آنی |
| پیکربندی | پیچیده یا نیاز به Eject | ساده و انعطاف‌پذیر |
| حجم باندل Production | بزرگ‌تر | کوچک‌تر و بهینه‌تر |
| پشتیبانی و اکوسیستم | غیرفعال | فعال و در حال رشد |
| پشتیبانی از جدیدترین ویژگی‌ها | کند | سریع |

نتیجه‌گیری

انتقال از CRA به Vite یک ارتقاء استراتژیک است که بهره‌وری توسعه‌دهندگان را به دلیل سرعت فوق‌العاده، تجربه کاربری نهایی را به دلیل bundleهای بهینه‌تر و قابلیت نگهداری پروژه را به دلیل معماری مدرن و پشتیبانی فعال، به طور قابل توجهی بهبود می‌بخشد. با توجه به توصیه رسمی React برای کنار گذاشتن CRA، این انتقال نه تنها یک انتخاب، بلکه یک ضرورت برای پروژه‌های مدرن به نظر می‌رسد .

برای شروع یک پروژه جدید با Vite، می‌توانید از دستور زیر استفاده کنید:+

|  |
| --- |
| npm create vite@latest myapp template react |

# فصل ۱: راه‌اندازی محیط توسعه

نصب و راه‌اندازی پروژه با Vite

|  |
| --- |
| npm init y  npm create vite@latest myapp template react  cd myapp  npm install  npm run dev |

1. npm create: این دستور، یک دستور shorthand در npm است. وقتی شما npm create  را اجرا می‌کنید، npm به طور خودکار بسته create<name>  را پیدا و اجرا می‌کند. در این مورد، vite  برابر است با createvite.
2. vite@latest: با اضافه کردن @latest مطمئن می‌شوید که همیشه آخرین نسخه پایدار Vite استفاده می‌شود.
3. myapp: این نام پوشه‌ای است که پروژه جدید در آن ایجاد خواهد شد.
4. template react: دو خط تیره () به این معنی است که آرگومان‌های بعدی به اسکریپت createvite  پاس داده می‌شوند. در اینجا به آن می‌گوییم که از template ( قالب) React استفاده کند.

پس از اجرای این دستور، مراحل زیر به صورت خودکار انجام می‌شود:

1. npm آخرین نسخه createvite را به صورت موقت (بدون نیاز به نصب global) دانلود می‌کند.
2. اسکریپت createvite اجرا شده و یک پروژه جدید با نام myapp ایجاد می‌کند.
3. ساختار اولیه پروژه با تمپلیت React در پوشه myapp قرار داده می‌شود.

در نهایت، برای نصب dependencyها (کتابخانه‌های مورد نیاز پروژه) باید دستورات زیر را اجرا کنید:

|  |
| --- |
| cd myapp  npm install |

# ساختار پروژه Vite + React

یک پروژه Vite + React معمولاً ساختاری شبیه به زیر دارد:

|  |
| --- |
| myreactapp/  ├── node\_modules/ وابستگی‌های نصب شده پروژه  ├── public/ فایل‌های استاتیک  │ ├── favicon.ico آیکون وب‌سایت  │ ├── index.html فایل HTML اصلی (در Vite این فایل در ریشه قرار می‌گیرد)  │ └── otherstaticfiles سایر فایل‌های استاتیک  ├── src/ کد منبع اصلی پروژه  │ ├── assets/ منابعی مانند تصاویر، فونت‌ها و...  │ ├── components/ کامپوننت‌های ری‌اکت  │ ├── hooks/ هوک‌های سفارشی  │ ├── pages/ صفحات مختلف برنامه  │ ├── styles/ فایل‌های استایل  │ ├── utils/ ابزارها و توابع کمکی  │ ├── App.jsx کامپوننت اصلی برنامه  │ ├── App.css استایل‌های کامپوننت اصلی  │ ├── main.jsx نقطه ورود برنامه  │ └── index.css استایل‌های全局  ├── .gitignore فایل‌های نادیده گرفته شده توسط Git  ├── index.html فایل HTML اصلی (در Vite این فایل در ریشه قرار دارد) :cite[3]:cite[8]  ├── package.json اطلاعات پروژه و وابستگی‌ها  ├── vite.config.js پیکربندی Vite (اختیاری) :cite[8]:cite[9]  ├── eslintrc.cjs پیکربندی ESLint (اختیاری)  └── README.md مستندات پروژه |

توضیح جزئیات ساختار پروژه

1. فایل‌های ریشه پروژه

index.html`:

نقش: نقطه ورود اصلی برنامه

ویژگی در Vite: برخلاف Create React App که این فایل در پوشه `public` قرار دارد، در Vite این فایل مستقیماً در ریشه پروژه قرار می‌گیرد . Vite این فایل را به عنوان بخشی از گراف ماژول در نظر می‌گیرد و تگ‌های اسکریپت را پردازش می‌کند .

محتوا: شامل یک `div` با `id="root"` که برنامه React در آن render می‌شود و یک تگ `script` که به `src/main.jsx` اشاره می‌کند.

vite.config.js`:

نقش: فایل پیکربندی Vite برای سفارشی‌سازی تنظیمات

محتوا: می‌تواند شامل تنظیمات سرور، پلاگین‌ها، alias مسیرها و سایر تنظیمات build باشد.

package.json`:

نقش: شامل metadata پروژه، وابستگی‌ها و اسکریپت‌ها

اسکریپت‌های معمول:

|  |
| --- |
| {  "scripts": {  "dev": "vite", // شروع سرور توسعه  "build": "vite build", // ساخت برای production  "preview": "vite preview" // پیش‌نمایش build ساخته شده  }  } |

2. پوشه `public/`

نقش: شامل فایل‌های استاتیکی که بدون پردازش مستقیماً کپی می‌شوند

محتوای معمول:

`favicon.ico`: آیکون وب‌سایت

سایر فایل‌های استاتیک مانند تصاویر یا فونت‌هایی که نیاز به پردازش ندارند

1. پوشه `src/` (قلب پروژه)

|  |
| --- |
| import React from 'react'  import ReactDOM from 'reactdom/client'  import App from './App.jsx'  import './index.css'  ReactDOM.createRoot(document.getElementById('root')).render(  <React.StrictMode>  <App />  </React.StrictMode>,  ) |

App.jsx`:

نقش: کامپوننت اصلی و ریشه برنامه

محتوا: شامل ساختار اولیه برنامه و مسیریابی اصلی

index.css`:

نقش: استایل‌های فراگیر برنامه

پوشه `components/`:

نقش: شامل کامپوننت‌های قابل استفاده مجدد

ساختار پیشنهادی: سازماندهی بر اساس ویژگی یا عملکرد (مانند `Button/`, `Card/`, `Modal/`)

پوشه `pages/`:

نقش: شامل کامپوننت‌های صفحه که معمولاً مسیرهای مختلف برنامه را نشان می‌دهند

پوشه `assets/`:

نقش: شامل منابعی مانند تصاویر، فونت‌ها، آیکون‌ها

طریقه استفاده:

|  |
| --- |
| import logo from './assets/logo.png'  function App() {  return <img src={logo} alt="Logo" />  } |

پوشه `hooks/`:

نقش: شامل هوک‌های سفارشی React برای منطق قابل استفاده مجدد

پوشه `utils/`:

نقش: شامل توابع کمکی، constants و سایر کدهای utility

## ویژگی‌های خاص Vite در ساختار پروژه

1. سرویس دادن به `index.html` از ریشه:

در Vite، فایل `index.html` در ریشه پروژه قرار دارد نه در پوشه `public` .

Vite این فایل را به عنوان منبع و بخشی از گراف ماژول در نظر می‌گیرد .

2. پشتیبانی از ES Modules بومی:

Vite در مرحله توسعه از ES Modules بومی استفاده می‌کند که منجر به راه‌اندازی سریع سرور و به روزرسانی‌های فوری می‌شود .

3. Hot Module Replacement (HMR) سریع:

تغییرات در کد بلافاصله بدون نیاز به رفرش کامل صفحه در مرورگر منعکس می‌شوند .

4. پشتیبانی از انواع assetها:

می‌توانید مستقیماً انواع فایل‌ها (مانند تصاویر، CSS، فونت‌ها) را import کنید .

سفارشی‌سازی ساختار

می‌توانید ساختار پروژه را بر اساس نیازهای خود سفارشی کنید:

ایجاد پوشه‌های اضافی: مانند `services/` برای API calls، `context/` برای مدیریت state سراسری، یا `constants/` برای مقادیر ثابت

استفاده از absolute imports: با تنظیم `vite.config.js` می‌توانید از importهای مطلق استفاده کنید:

|  |
| --- |
| // vite.config.js  import { defineConfig } from 'vite'  import react from '@vitejs/pluginreact'  export default defineConfig({  plugins: [react()],  resolve: {  alias: {  '@': '/src',  },  },  }) |

جمع‌بندی

ساختار پروژه Vite + React طراحی شده تا:

ساده و قابل درک باشد برای توسعه‌دهندگان

مقیاس‌پذیر باشد برای پروژه‌های کوچک و بزرگ

منعطف باشد برای سفارشی‌سازی بر اساس نیازهای خاص

بهینه باشد برای عملکرد و تجربه توسعه

این ساختار با ترکیب بهترین روش‌های توسعه React و مزایای عملکردی Vite، پایه‌ای محکم برای ساخت برنامه‌های وب مدرن و کارآمد فراهم می‌کند.

# فصل ۲: مبانی React و JSX

هدف فصل

در این فصل با مفاهیم پایه‌ای React و JSX آشنا خواهیم شد که سنگ بنای توسعه اپلیکیشن‌های React هستند.

بخش ۱: معرفی JSX

۱.۱ JSX چیست؟

JSX (JavaScript XML) یک extension سینتکس برای JavaScript است که به شما امکان می‌دهد کدهای شبیه به HTML را در JavaScript بنویسید.

|  |
| --- |
| // مثال ساده JSX  const element = <h1>Hello, World!</h1>; |

۱.۲ ویژگی‌های کلیدی JSX

۱.۲.۱ تگ‌های selfclosing

|  |
| --- |
| // تگ‌های self-closing باید حتما بسته شوند  const image = <img src="image.jpg" alt="Example" />;  const input = <input type="text" />; |

۱.۲.۲ return تک عنصری

|  |
| --- |
| // ❌ اشتباه - بازگشت چندین عنصر  function WrongComponent() {  return (  <h1>Title</h1>  <p>Content</p>  );  }  // ✅ صحیح - استفاده از Fragment  function CorrectComponent() {  return (  <>  <h1>Title</h1>  <p>Content</p>  </>  );  } |

۱.۲.۳ نوشتن JavaScript در JSX

|  |
| --- |
| function UserInfo() {  const name = "John Doe";  const age = 25;  const isActive = true;  return (  <div>  <h2>{name}</h2>  <p>Age: {age}</p>  <p>Status: {isActive ? 'Active' : 'Inactive'}</p>  <p>Current year: {new Date().getFullYear()}</p>  </div>  );  } |

بخش ۲: کامپوننت‌ها در React

۲.۱ انواع کامپوننت‌ها

۲.۲.۱ کامپوننت‌های تابعی (Functional Components)

|  |
| --- |
| // کامپوننت تابعی ساده  function Welcome() {  return <h1>Welcome to React!</h1>;  }  // کامپوننت تابعی با Arrow Function  const Welcome = () => {  return <h1>Welcome to React!</h1>;  };  // خلاصه‌تر  const Welcome = () => <h1>Welcome to React!</h1>; |

۲.۲.۲ کامپوننت‌های کلاسی (Class Components)

|  |
| --- |
| // کامپوننت کلاسی  class Welcome extends React.Component {  render() {  return <h1>Welcome to React!</h1>;  }  } |

۲.۲ استفاده و ترکیب کامپوننت‌ها

|  |
| --- |
| // کامپوننت Button  const Button = () => {  return <button>Click me</button>;  };  // کامپوننت Header  const Header = () => {  return (  <header>  <h1>My App</h1>  <nav>  <a href="/home">Home</a>  <a href="/about">About</a>  </nav>  </header>  );  };  // کامپوننت اصلی که سایر کامپوننت‌ها را استفاده می‌کند  function App() {  return (  <div>  <Header />  <main>  <h2>Main Content</h2>  <Button />  </main>  </div>  );  } |

### بخش ۳: Props و داده‌ها در کامپوننت

۳.۱ ارسال داده با Props

|  |
| --- |
| // کامپوننت UserCard  const UserCard = (props) => {  return (  <div className="user-card">  <h3>{props.name}</h3>  <p>Email: {props.email}</p>  <p>Age: {props.age}</p>  </div>  );  };  // استفاده از کامپوننت با props مختلف  function App() {  return (  <div>  <UserCard name="Alice" email="alice@email.com" age={25} />  <UserCard name="Bob" email="bob@email.com" age={30} />  <UserCard name="Charlie" email="charlie@email.com" age={35} />  </div>  );  } |

۳.۲ Destructuring Props

|  |
| --- |
| // روش بهتر با destructuring  const UserCard = ({ name, email, age }) => {  return (  <div className="user-card">  <h3>{name}</h3>  <p>Email: {email}</p>  <p>Age: {age}</p>  </div>  );  }; |

۳.۳ مقادیر پیش‌فرض برای Props

|  |
| --- |
| // تعیین مقادیر پیش‌فرض  const UserCard = ({ name, email, age = 18 }) => {  return (  <div className="user-card">  <h3>{name}</h3>  <p>Email: {email}</p>  <p>Age: {age}</p>  </div>  );  }; |

۳.۴ children Prop

|  |
| --- |
| // کامپوننت Container با children  const Container = ({ children, title }) => {  return (  <div className="container">  <h2>{title}</h2>  <div className="content">  {children}  </div>  </div>  );  };  // استفاده از children  function App() {  return (  <Container title="User List">  <UserCard name="Alice" email="alice@email.com" />  <UserCard name="Bob" email="bob@email.com" />  <UserCard name="Charlie" email="charlie@email.com" />  </Container>  );  } |

بخش ۴: کاربردهای عملی JSX

۴.۱ شرط‌ها در JSX

|  |
| --- |
| const WelcomeMessage = ({ isLoggedIn, username }) => {  return (  <div>  {isLoggedIn ? (  <h1>Welcome back, {username}!</h1>  ) : (  <h1>Please log in to continue</h1>  )}  </div>  );  };  // یا با استفاده از && operator  const Notification = ({ hasNewMessages }) => {  return (  <div>  {hasNewMessages && (  <div className="notification">  You have new messages!  </div>  )}  </div>  );  }; |

۴.۲ حلقه‌ها در JSX

|  |
| --- |
| const UserList = ({ users }) => {  return (  <ul>  {users.map((user) => (  <li key={user.id}>  {user.name} - {user.email}  </li>  ))}  </ul>  );  };  // استفاده از کامپوننت  function App() {  const users = [  { id: 1, name: 'Alice', email: 'alice@email.com' },  { id: 2, name: 'Bob', email: 'bob@email.com' },  { id: 3, name: 'Charlie', email: 'charlie@email.com' }  ];  return <UserList users={users} />;  } |

۴.۳ استایل‌دهی در JSX

|  |
| --- |
| const StyledButton = () => {  // استایل inline  const buttonStyle = {  backgroundColor: 'blue',  color: 'white',  padding: '10px 20px',  border: 'none',  borderRadius: '5px',  cursor: 'pointer'  };  return (  <button style={buttonStyle}>  Click me  </button>  );  };  // استفاده از کلاس‌های CSS  const Card = () => {  return (  <div className="card">  <h3 className="card-title">Title</h3>  <p className="card-content">Content goes here</p>  </div>  );  }; |

تمرین‌های عملی

تمرین ۱: کامپوننت ProductCard

|  |
| --- |
| // یک کامپوننت ProductCard ایجاد کنید که شامل:  // - تصویر محصول  // - نام محصول  // - قیمت  // - دکمه "Add to Cart"  const ProductCard = ({ image, name, price }) => {  return (  <div className="product-card">  <img src={image} alt={name} />  <h3>{name}</h3>  <p>Price: ${price}</p>  <button>Add to Cart</button>  </div>  );  }; |

تمرین ۲: کامپوننت Navigation

|  |
| --- |
| // یک کامپوننت Navigation ایجاد کنید که:  // - لیستی از لینک‌ها را نمایش دهد  // - از props برای دریافت آیتم‌های منو استفاده کند  const Navigation = ({ items }) => {  return (  <nav>  <ul>  {items.map((item) => (  <li key={item.id}>  <a href={item.url}>{item.text}</a>  </li>  ))}  </ul>  </nav>  );  }; |

خلاصه فصل

مفاهیم کلیدی:

1. JSX: سینتکس شبیه HTML برای نوشتن کامپوننت‌های React

2. کامپوننت‌ها: بلوک‌های سازنده قابل استفاده مجدد

3. Props: روش انتقال داده از parent به child components

4. Conditional Rendering: نمایش شرطی محتوا

5. Lists: نمایش لیست‌ها با استفاده از map()

بهترین روش‌ها:

استفاده از Fragment (`<> </>`) برای بازگشت چندین عنصر

استفاده از destructuring برای props

دادن مقادیر پیش‌فرض به props

استفاده از `key` prop هنگام نمایش لیست‌ها

سازماندهی کد با کامپوننت‌های کوچک و متمرکز

این فصل پایه‌ای اساسی برای کار با React فراهم می‌کند. در فصل بعدی با state و مدیریت وضعیت در کامپوننت‌ها آشنا خواهیم شد.