ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

React JS

Δαμιανός Μαρινάτος

React JS

- Η **React JS** είναι μία **βιβλιοθήκη** JavaScript που ξεκίνησε από τη Meta (ex Facebook) το 2013
- Χρησιμοποιείται για την ανάπτυξη **διαδραστικών διεπαφών χρήστη (UI)** σε ιστοσελίδες και εφαρμογές.
- Βασίζεται σε μια **component-based αρχιτεκτονική**, δηλαδή μια φιλοσοφία ανάπτυξης όπου η διεπαφή «χτίζεται» από **επαναχρησιμοποιήσιμα κομμάτια κώδικα** που ονομάζονται **Components**.

React 19 (1)

- Με την έκδοση 19, η React υποστηρίζει πλέον **επίσημα** Server-Side Rendering (SSR), κάνοντάς την ιδανική επιλογή για εφαρμογές που απαιτούν καλύτερη απόδοση και SEO.
- Η υποστήριξη SSR βασίζεται σε νέα χαρακτηριστικά όπως το **streaming rendering** και το **concurrent rendering**, που βελτιώνουν σημαντικά την εμπειρία του χρήστη.

React 19 (2)

- **Streaming Rendering**: Η σελίδα ξεκινά να αποστέλλεται στον χρήστη τμηματικά, καθώς τα δεδομένα γίνονται διαθέσιμα. Αυτό μειώνει τον χρόνο αναμονής και επιταχύνει την εμφάνιση περιεχομένου.
- **Concurrent Rendering**: Η React μπορεί πλέον να διαχειρίζεται πολλαπλές εργασίες ταυτόχρονα, αποδίδοντας προτεραιότητα στις πιο σημαντικές. Αυτό κάνει τις διεπαφές πιο ομαλές και αποδοτικές, ακόμα και σε απαιτητικές εφαρμογές.

Virtual DOM

- Το Virtual DOM είναι μία σημαντική δομή στην React που επιτρέπει
  - Τη δυνατότητα να αλλάζει (updates) **συγκεκριμένα σημεία** του πραγματικού DOM
  - Να εφαρμόζει **batch updates** σε ένα πέρασμα και όχι μία-μία τις αλλαγές
  - Να αποκρύπτεται το φυσικό επίπεδο που μπορεί να είναι το πραγματικό (Real) DOM ή και άλλες τεχνολογίες όπως mobile (React Native)

Typescript

- Η React υποστηρίζει και **TypeScript**, δηλαδή μια παραλλαγή της JavaScript με αυστηρό **type checking** κατά τη μεταγλώττιση (compile time).

- Αυτό βοηθά στον εντοπισμό σφαλμάτων πριν καν εκτελεστεί ο κώδικας.

JSX

- Η React επίσης έχει δώσει τη δυνατότητα συγγραφής Templates με HTML και interpolation JavaScript με ένα ειδικό syntax που ονομάζεται **JSX** (JavaScript XML) η οποία επιτρέπει την εισαγωγή HTML μέσα σε ένα αρχείο JavaScript.
- Όταν χρησιμοποιούμε **JSX σε TypeScript**, η κατάληξη του αρχείου είναι **.tsx** αντί για **.jsx**.

Κανόνες JSX (1)

- **className αντί για class**
  Επειδή το class είναι δεσμευμένη λέξη στη JavaScript, στη JSX χρησιμοποιούμε className για να ορίσουμε CSS κλάσεις.

```
return <div className="container">Hello</div>;
```

Κανόνες JSX (2)

- **Μόνο ένα Element στο return**

Κάθε JSX block πρέπει να επιστρέφει **ένα μόνο root element**.

Αν θέλουμε να επιστρέψουμε περισσότερα, τα τοποθετούμε σε ή σε **React Fragments** (<> ... </>)

```
return (
  <>
    <h1 className="text-center">Hello</h1>
    <PublicPage />
  </>
);
```

Κανόνες JSX (3)

- **Οι HTML ιδιότητες γράφονται σε camelCase**

  Παραδείγματα:

  - onClick αντί για onclick
  - htmlFor αντί για for

```
<button onClick={handleClick}>Click me</button>
<label htmlFor="email">Email</label>
```

Κανόνες JSX (4)

- **Ενσωμάτωση JavaScript με {}**

  Μπορούμε να χρησιμοποιήσουμε JavaScript εκφράσεις μέσα σε JSX με αγκύλες:

```
const lesson = "React JS";
return <h1>Welcome to {lesson} lesson!</h1>;
```

Naming Conventions (1)

- **PascalCase**

  Κάθε λέξη ξεκινά με κεφαλαίο γράμμα

  Χρησιμοποιείται σε ονόματα Components και Classes

  Παράδειγμα: **UserProfileCard**

Naming Conventions (2)

- **camelCase**

  Η πρώτη λέξη με μικρό, οι επόμενες με κεφαλαίο αρχικό

  Χρησιμοποιείται σε μεταβλητές, συναρτήσεις, props

  Παράδειγμα: **isActive**

Naming Conventions (3)

- **kebab-case**

  Οι λέξεις χωρίζονται με παύλες (-)

  Χρησιμοποιείται κυρίως σε ονόματα αρχείων ή CSS classes

  Παράδειγμα: user-profile

React Ecosystem (1)

- Η React παρέχει τη βιβλιοθήκη (**React API**) για τη δημιουργία και διαχείριση των UI components.
- **Utilities και Starter Templates** για την γρήγορη εκκίνηση των project (π.χ. με create-react-app ή Vite + React).
- Επίσης παρέχονται εργαλεία για components, dependencies, config, κλπ.

React Ecosystem (2)

- **Χρήσιμα εργαλεία στο οικοσύστημα:**
  - **React Router**: Navigation σε Single Page Applications (SPA).
  - **Redux**: Για διαχείριση σύνθετης (global) κατάστασης της εφαρμογής.
  - **React Query / Zustand / Recoil**: Εναλλακτικές για state management.
  - **Jest / React Testing Library**: Για unit testing των components.
  - **Vite / Webpack**: Για ανάπτυξη, bundling, και HMR (hot module replacement).
  - **Tailwind CSS / Styled Components**: Για styling των components.
  - **Next.js:** Για full-stack apps.

Setting up a React Project

- Στο παρελθόν ο βασικός τρόπος για την δημιουργία ενός React project ήταν η εντολή create-react-app (CRA)

```
dammarin@Mac Desktop % npx create-react-app my-app
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y
```

- Τον **Μάρτιο του ‘23 το CRA έφυγε** από το επίσημο repo της React στο GitHub https://github.com/reactjs/react.dev (ήδη και πιο παλιά δεν υποστηριζόταν το CRA από την ομάδα του Facebook)

Vite

- Το **Vite** (French word, pronounced /vit/) είναι ο νέος αντικαταστάτης του Webpack/Babel το οποίο:
  - δημιουργεί ένα **γρήγορο περιβάλλον development**
  - στήνει έναν **τοπικό server με HMR** (Hot Module Replacement)
  - κάνει **build** την εφαρμογή για παραγωγή (production)

Vite / Rollup

- Το Vite είναι ένα build tool με ESM (ECMAScript Modules) support που διαχειρίζεται τα dependencies ενός React project και δημιουργεί τη δομή των φακέλων.
- Χρησιμοποιεί το **Rollup**(https://rollupjs.org/) ως bundler (αντί του Webpack). Είναι πιο γρήγορο από το Webpack/Babel.

New Project with Vite


```
dammarin@Mac cf % npm create vite@latest cf7-react -- --template react-ts

> npx
> create-vite cf7-react --template react-ts

◇  Scaffolding project in /Users/dammarin/Desktop/cf/cf7-react...

   Done. Now run:

  cd cf7-react
  npm install
  npm run dev
```

npm create	Λέμε στο npm να τρέξει ένα εργαλείο δημιουργίας project (όπως vite).
vite@latest	Custom React hooks
cf7-react	Είναι το όνομα του φακέλου/project που θα δημιουργηθεί.
--	Χωρίζει τις παραμέτρους του npm από τις παραμέτρους του vite. (npm version 7+)
--template react-ts	Ζητάμε να χρησιμοποιηθεί το template για React με TypeScript.

IDEs για React.js

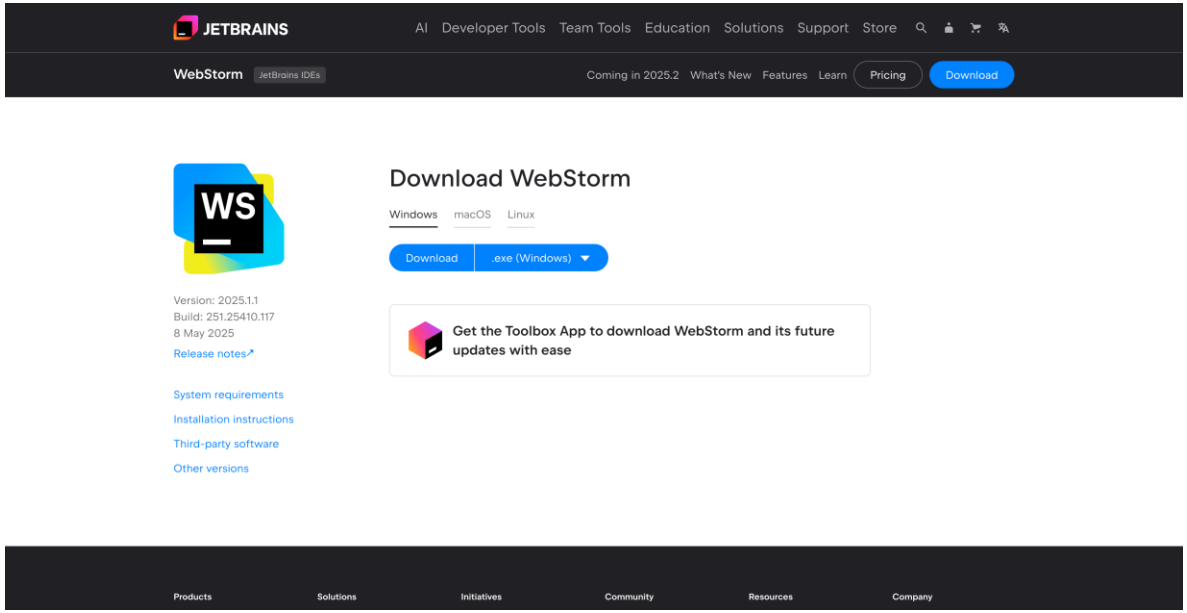
- **WebStorm**
  - Πλήρως παραμετροποιήσιμο
  - Ενσωματωμένη υποστήριξη για React, TypeScript, Git
  - Έξυπνο refactoring & debugging
  - Ιδανικό για επαγγελματική χρήση
  - Απαιτεί άδεια χρήσης (εμπορικό)
- **Visual Studio Code (VS Code)**
  - Δωρεάν & ανοιχτού κώδικα
  - Ελαφρύ & επεκτάσιμο με extensions
  - Υποστηρίζει React μέσω plugins (π.χ. ES7+ snippets)
  - Πολύ δημοφιλές στην κοινότητα
- **StackBlitz / CodeSandbox**
  - IDE στον browser
  - Ιδανικά για demo, παρουσιάσεις ή δοκιμές χωρίς εγκατάσταση
  - Μειονεκτούν σε μεγάλα projects

WebStorm

- Το **WebStorm** είναι ένα επαγγελματικό περιβάλλον ανάπτυξης (IDE) που κατασκευάζεται από την **JetBrains** και είναι ειδικά σχεδιασμένο για **JavaScript**, **TypeScript**, και σύγχρονες τεχνολογίες web.
  - Υποστηρίζει τεχνολογίες όπως: React, Angular, Vue, Next, Nest, Node, Express.js κ.α.
  - Παρέχει αυτόματη συμπλήρωση κώδικα, debugging, testing, και refactoring
  - Προσφέρει υψηλή παραγωγικότητα και σταθερότητα στην ανάπτυξη web εφαρμογών

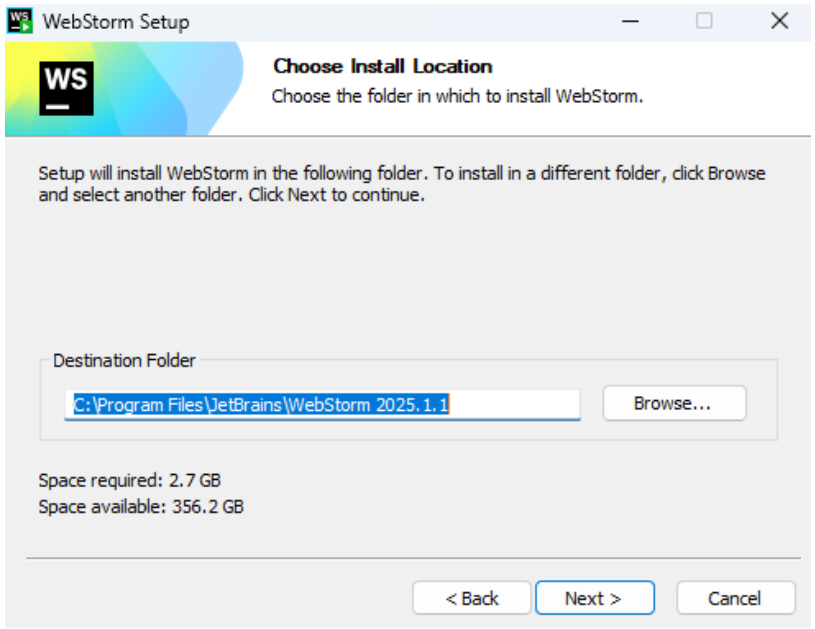
Εγκατάσταση WebStorm (1)

- Μεταβείτε στο https://www.jetbrains.com/webstorm/download
- Πατήστε **Download** ανάλογα με το λειτουργικό σας σύστημα (Windows / macOS / Linux)



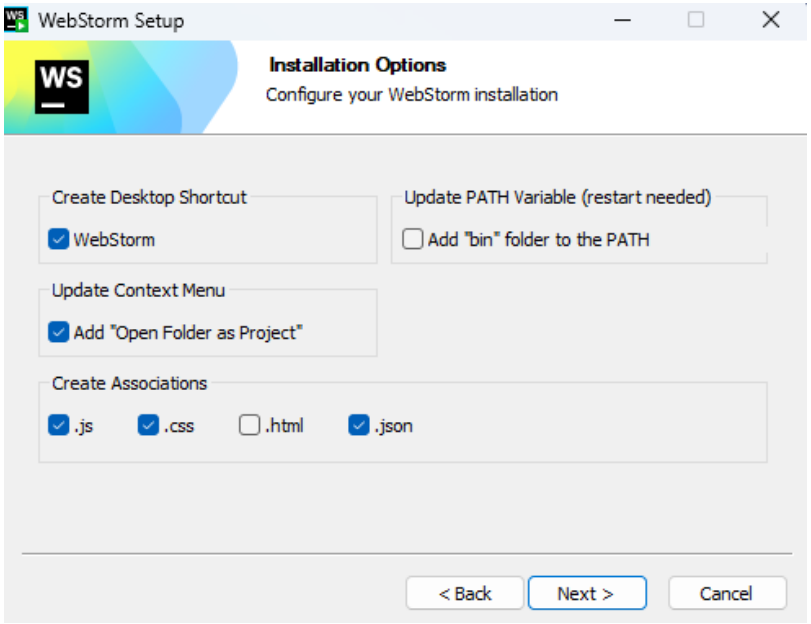
Εγκατάσταση WebStorm (2)

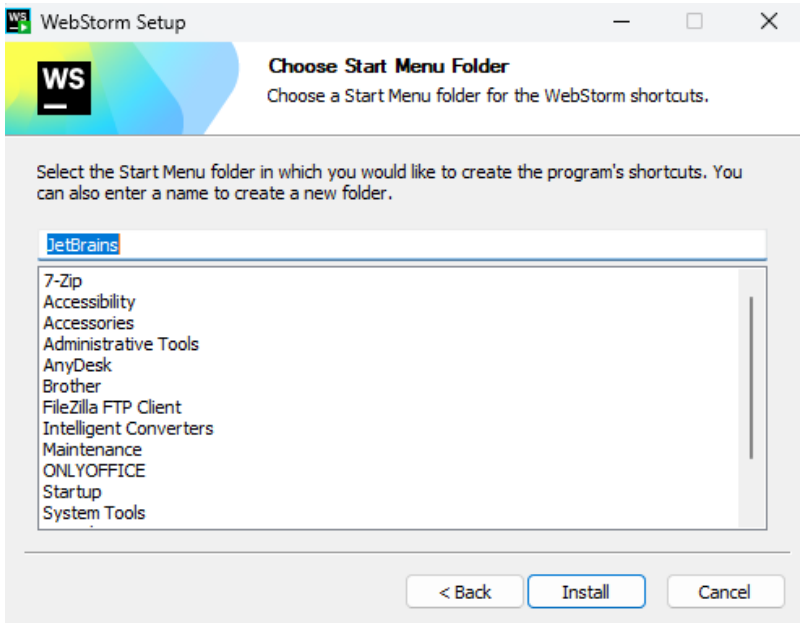
- Κάνουμε διπλό κλικ στο αρχείο .exe που κατεβάσαμε.
- Στο παράθυρο υποδοχής πατάμε "Next" για να ξεκινήσει η διαδικασία εγκατάστασης.
- Επιλέγουμε το φάκελο στον οποίο θέλουμε να εγκατασταθεί το WebStorm.
- Πατάμε "Next" για να συνεχίσουμε.



Εγκατάσταση WebStorm (3)

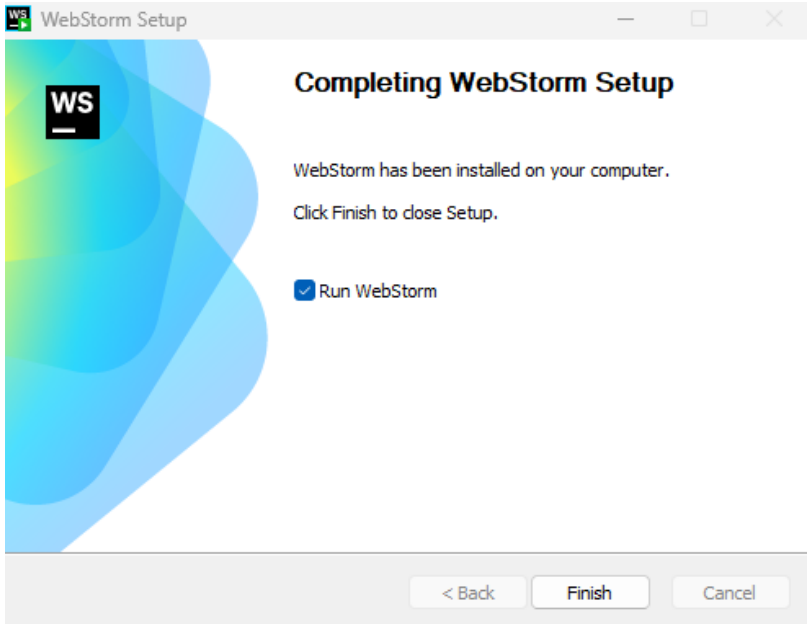
- Στην οθόνη με τις επιλογές εγκατάστασης μπορούμε να ενεργοποιήσουμε:
  - Δημιουργία συντόμευσης στην επιφάνεια εργασίας (Create Desktop Shortcut)
  - Προσθήκη στο δεξί κλικ "Άνοιγμα φακέλου ως Project" (Open Folder as Project)
  - Συσχετισμούς αρχείων με WebStorm (.js, .css, .html, .json)
- Αφού κάνουμε τις επιλογές μας, πατάμε “Next”.





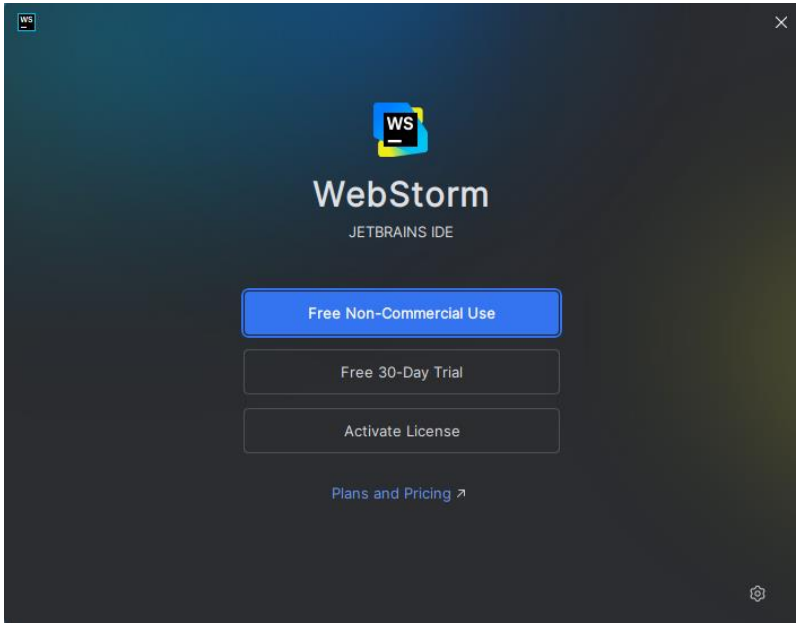
Εγκατάσταση WebStorm (4)

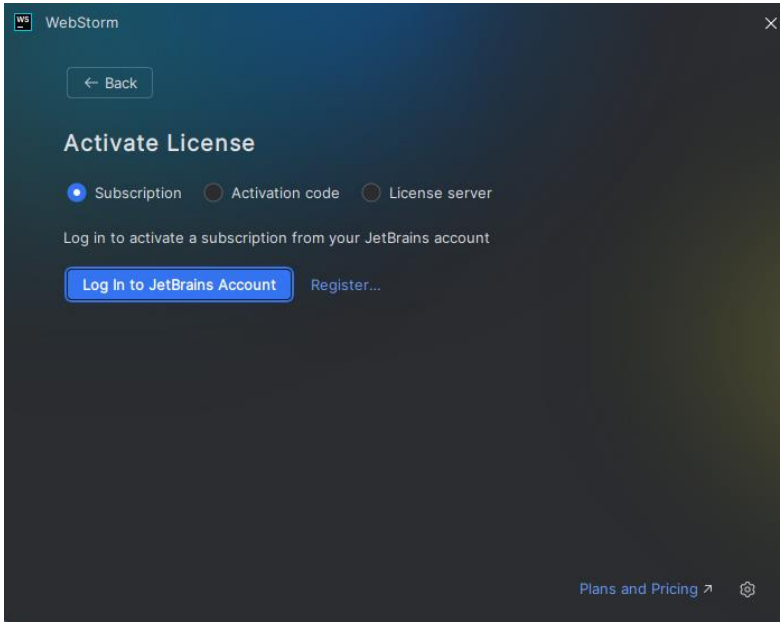
- Μετά την ολοκλήρωση της εγκατάστασης, τσεκάρουμε την επιλογή "Run WebStorm" για να ξεκινήσει αυτόματα η εφαρμογή.
- Πατάμε "Finish" για να κλείσουμε τον οδηγό εγκατάστασης.



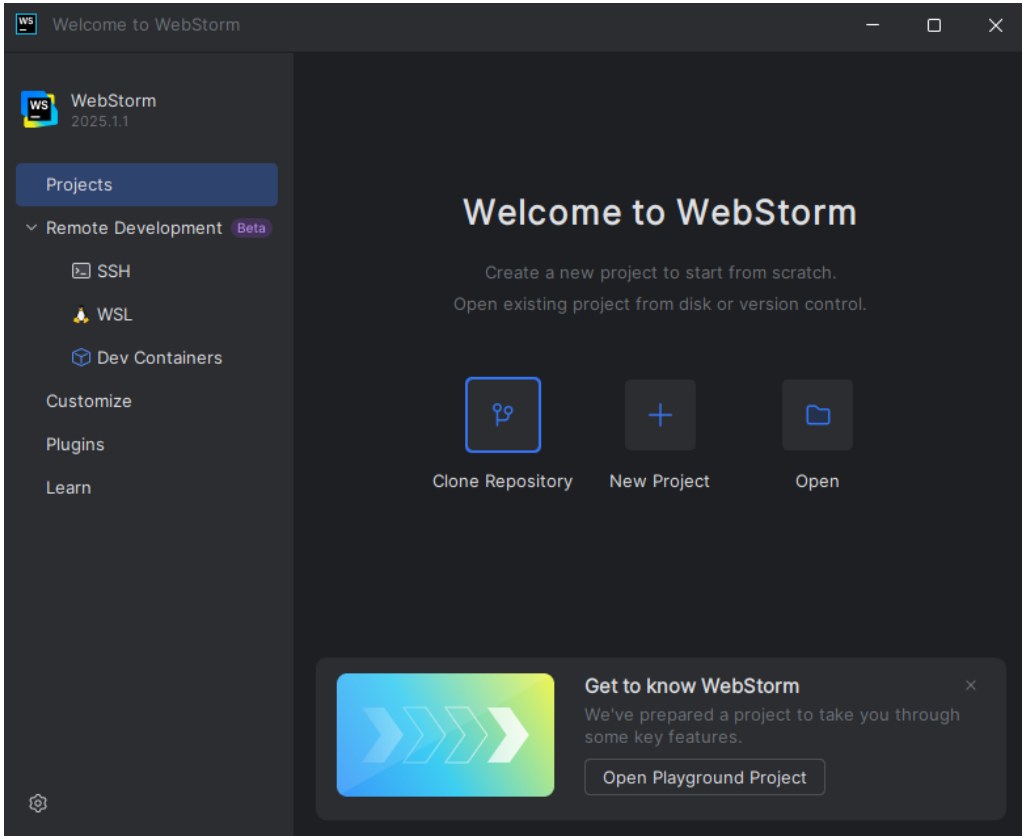
Εγκατάσταση WebStorm (5)

- Εφόσον αποδεχτούμε τους Όρους Χρήσης και επιλέξουμε αν θέλουμε να μοιραζόμαστε ανώνυμα δεδομένα χρήσης με την JetBrains, στην πρώτη οθόνη επιλέγουμε "Activate License".
- Και στη συνέχεια επιλέγουμε "Login with JetBrains Account".





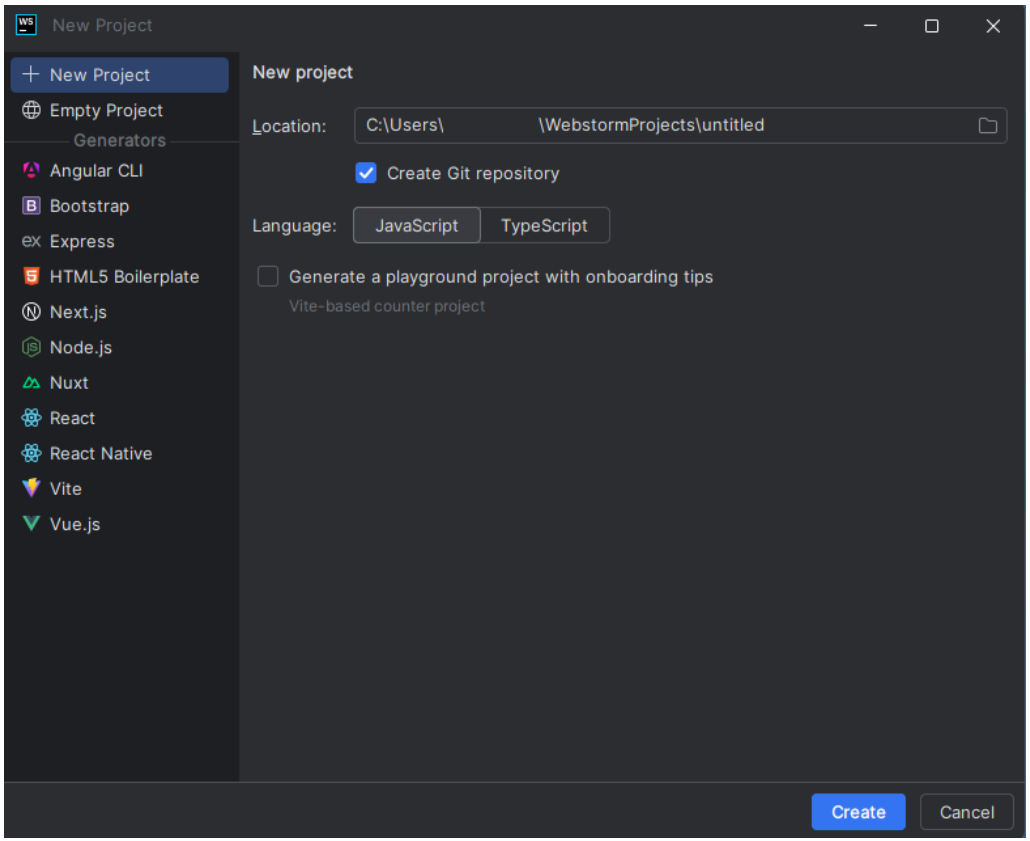
Έναρξη WebStorm (1)



Στην αρχική οθόνη του WebStorm, επιλέγουμε αν θέλουμε:

- Να ανοίξουμε ένα υπάρχον project
- Να φτιάξουμε καινούργιο
- Ή να κάνουμε clone ένα repository από το GitHub ή το GitLab

Έναρξη WebStorm (2)

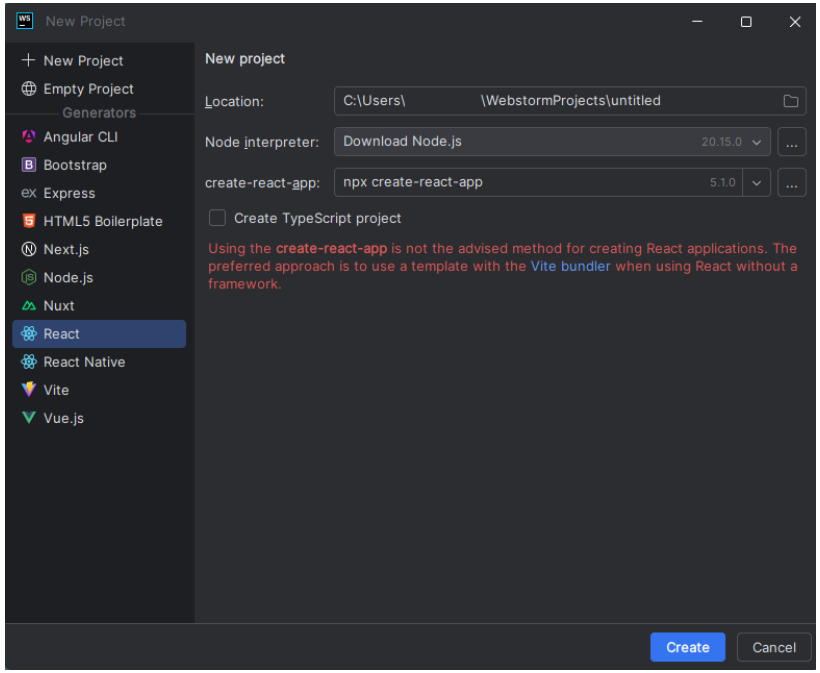


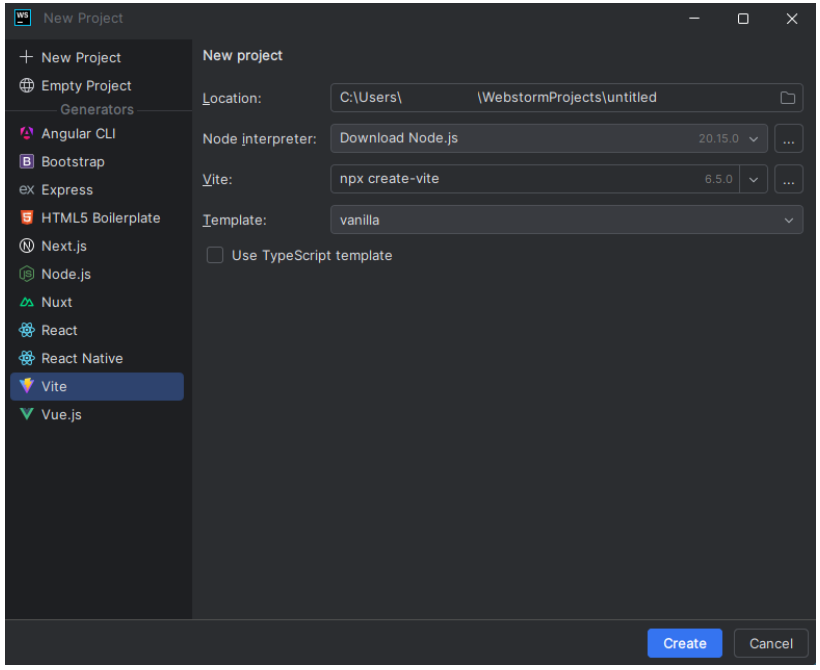
Στην αρχική οθόνη του WebStorm, επιλέγουμε αν θέλουμε:

- Να ανοίξουμε ένα υπάρχον project
- Να φτιάξουμε καινούργιο
- Ή να κάνουμε clone ένα repository από το GitHub ή το GitLab

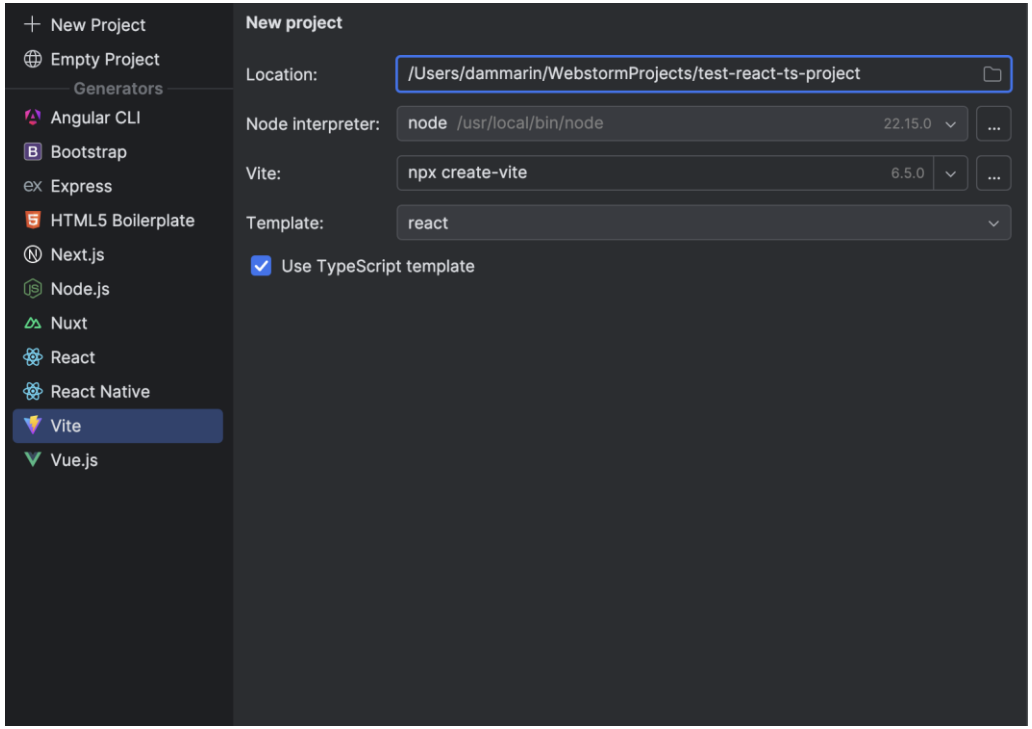
Έναρξη WebStorm (3)

- Δεν χρησιμοποιούμε την επιλογή "React", καθώς βασίζεται στο Create React App, το οποίο έχει καταργηθεί.
- Αντί για αυτό, επιλέγουμε το "Vite", το οποίο είναι ο προτεινόμενος τρόπος για να ξεκινήσουμε ένα νέο React Project.





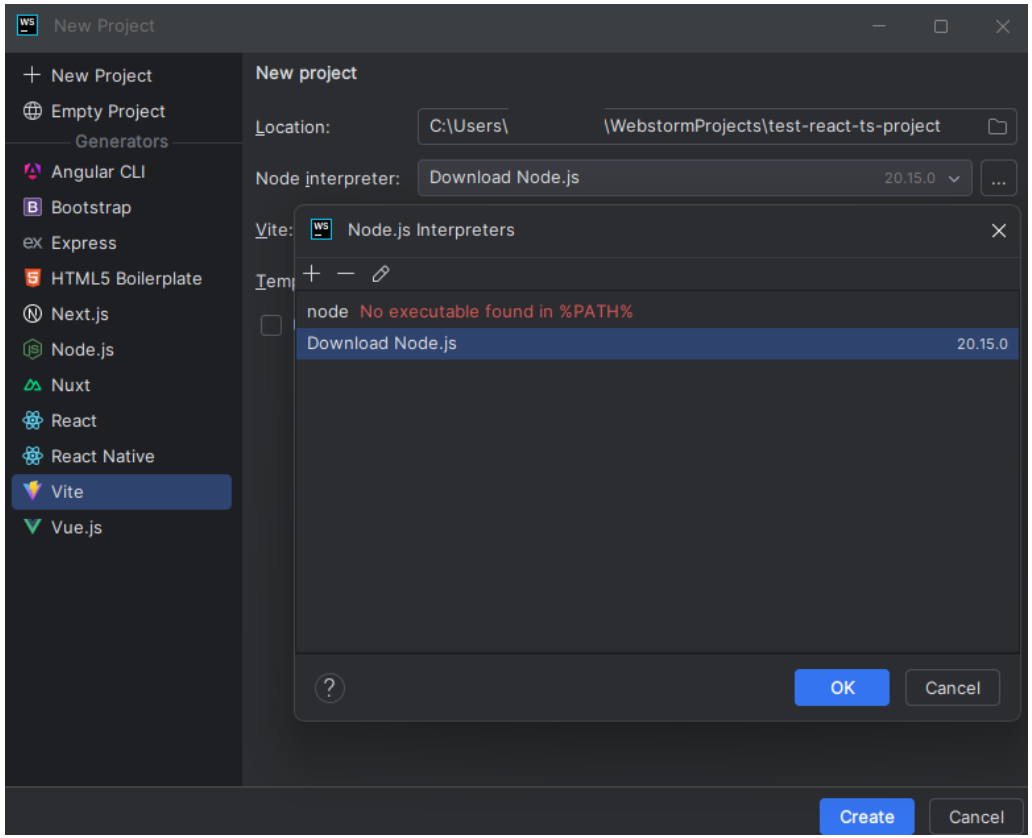
Έναρξη WebStorm (4)



Επιλέγουμε:

- Τον φάκελο στον οποίο θα δημιουργηθεί το project
- Την έκδοση του Node.js που επιθυμούμε να χρησιμοποιήσουμε
- React
- TypeScript

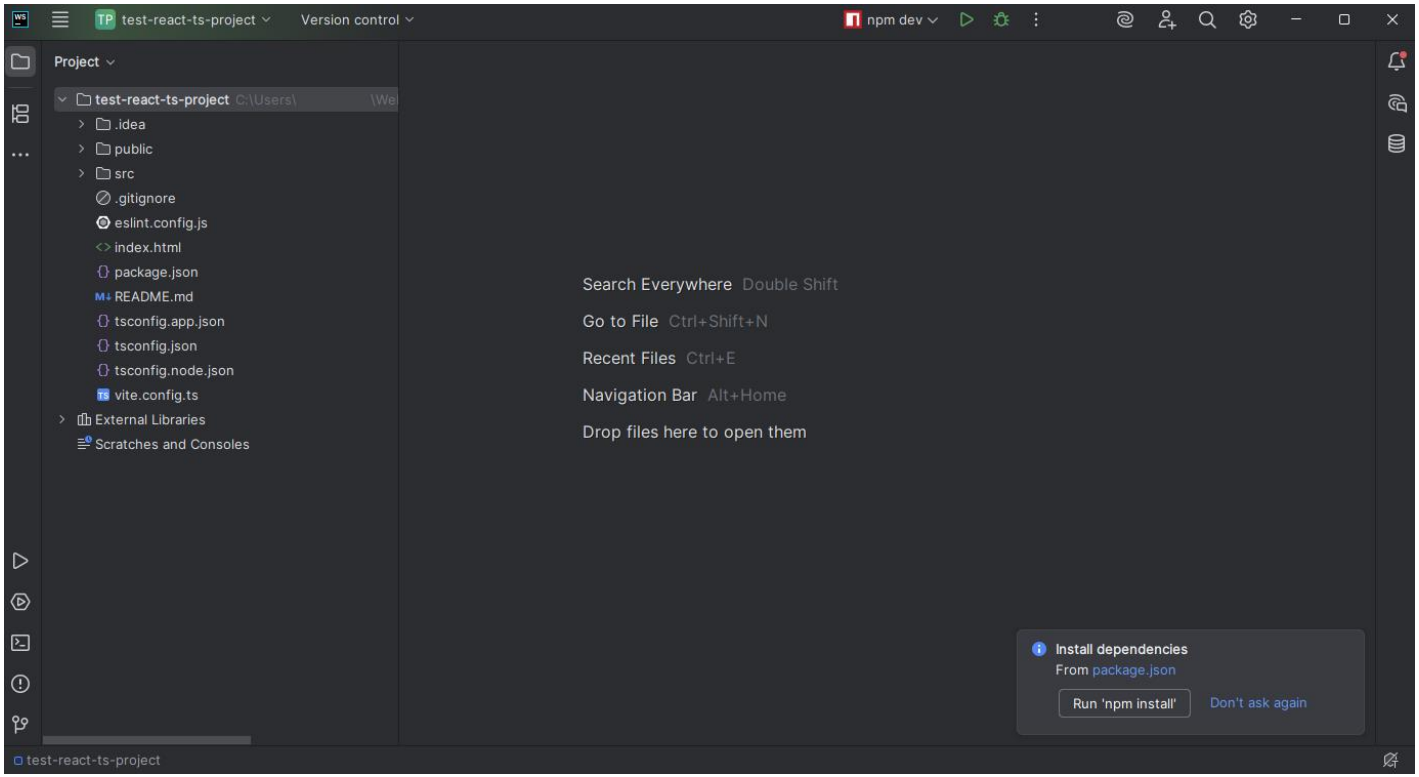
Έναρξη WebStorm (5)



- Στην επιλογή Node Interpreter διαλέγουμε την έκδοση Node.js που επιθυμούμε.
- Εάν δεν έχουμε εγκαταστημένη Node στο σύστημα μας τότε πατάμε "..." και επιλέγουμε το "Download Node.js" και μετά "OK"

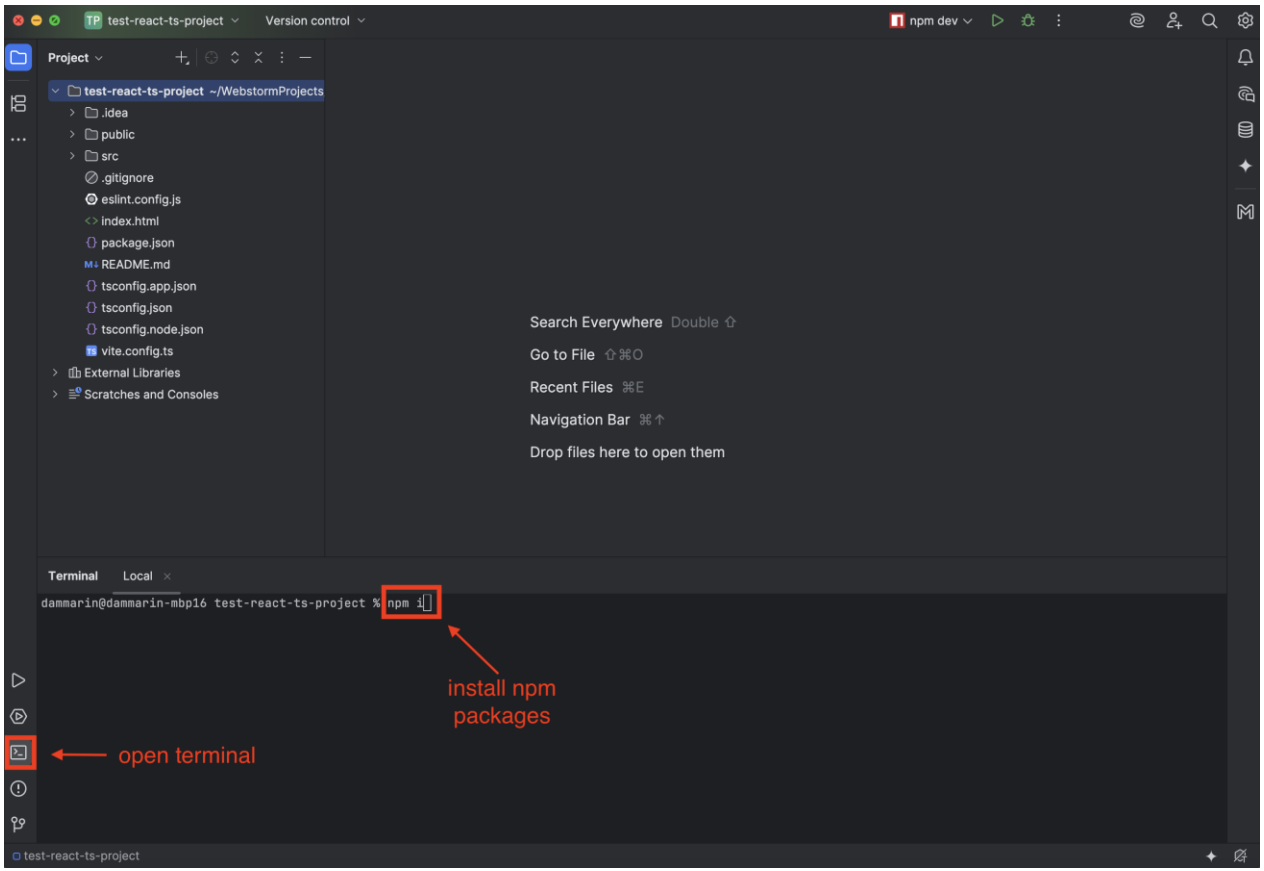
Έναρξη WebStorm (6)

- Μόλις δημιουργηθεί το project και ανοίξει στο WebStorm εμφανίζεται μήνυμα που μας ζητά να εγκαταστήσουμε τα npm packages πατώντας το κουμπί "Run 'npm install'"

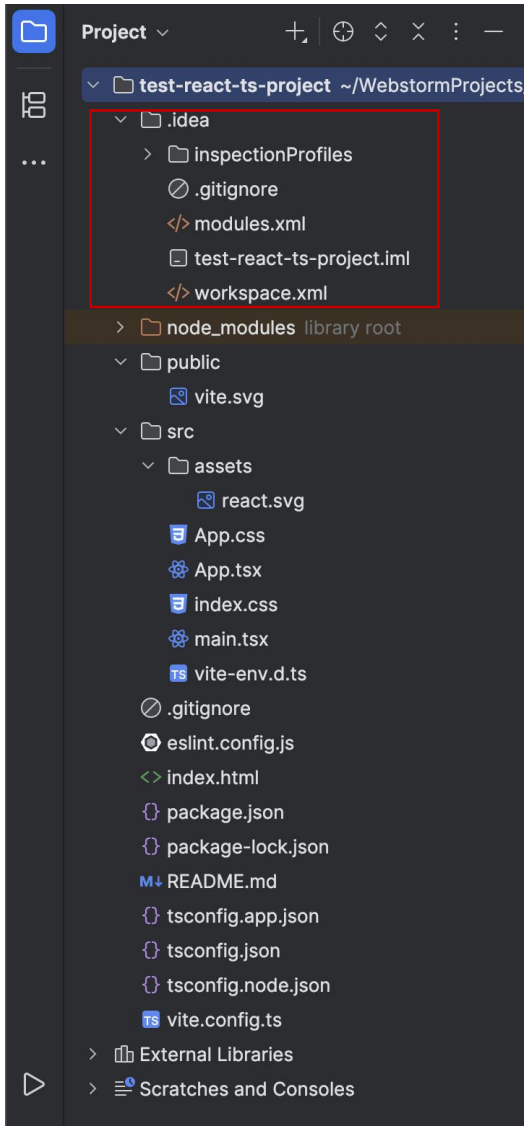


Έναρξη WebStorm (7)

- Εναλλακτικά, ανοίγουμε το terminal και πληκτρολογούμε χειροκίνητα το "npm install" ή "npm i".



Project Structure (1)

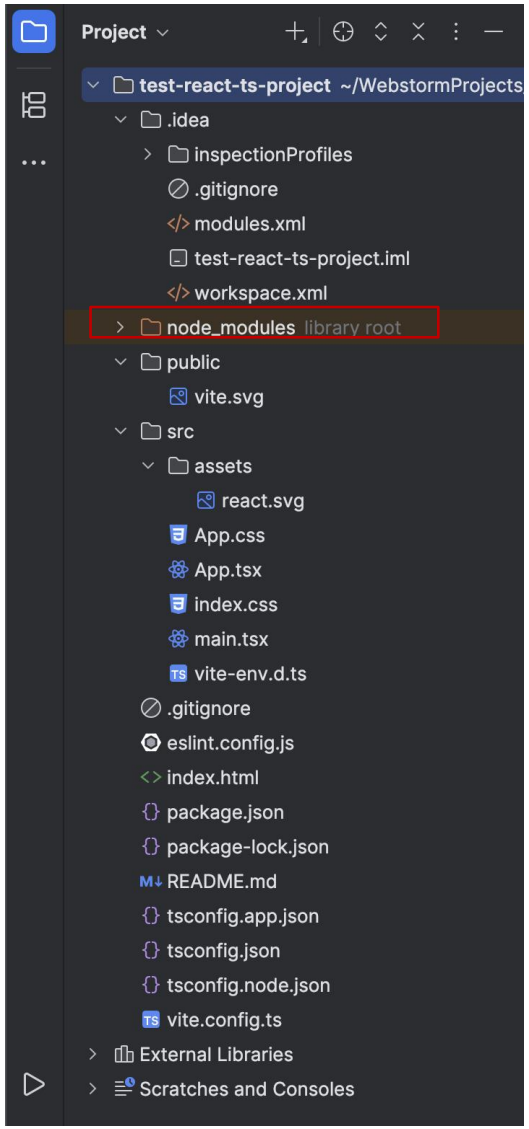


Η δομή περιλαμβάνει:

- Φάκελο ".idea" με τις ρυθμίσεις του WebStorm.

  Ο φάκελος αυτός δημιουργείται αυτόματα από το WebStorm (ή άλλα JetBrains IDEs όπως το IntelliJ) όταν ανοίγετε ένα project. Περιέχει μεταδεδομένα και ρυθμίσεις ειδικά για το project.

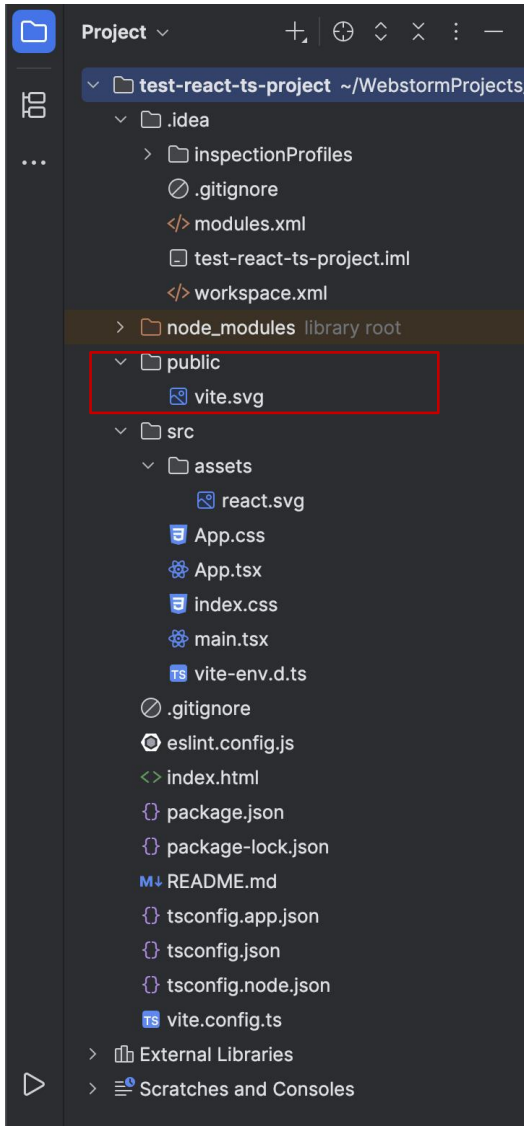
Project Structure (2)



- Φάκελο "node_modules" ο οποίος περιέχει όλες τις εξαρτήσεις (βιβλιοθήκες) που εγκαθίστανται μέσω του npm install ή yarn install. Χωρίς αυτόν, η εφαρμογή δεν θα μπορεί να μεταγλωττιστεί ή να τρέξει.

Προσοχή: Δεν το ανεβάζουμε ποτέ στο Git.

Project Structure (3)

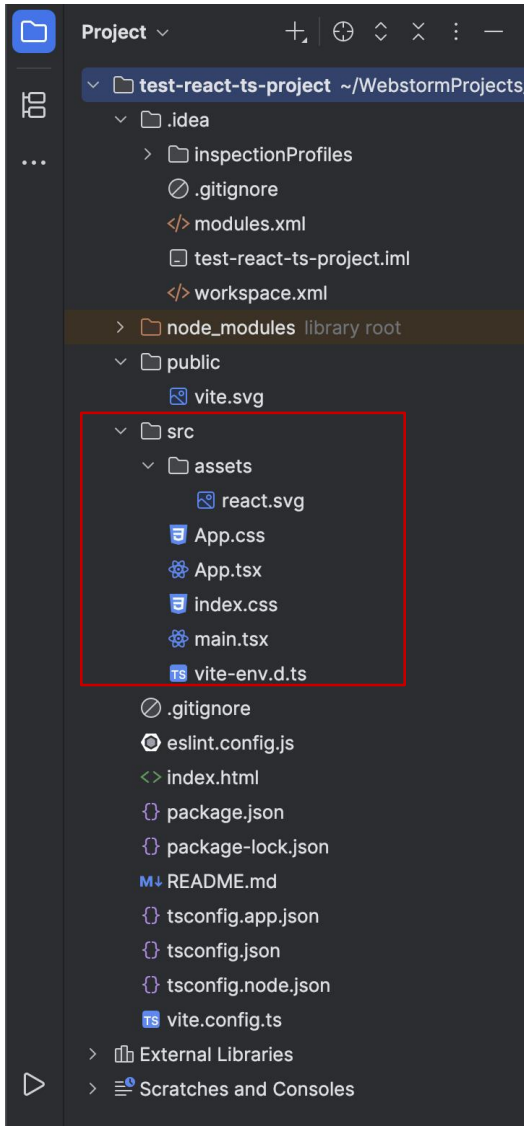


- Φάκελο "public" ο οποίος φιλοξενεί τα **static assets** που είναι προσβάσιμα απευθείας από το browser (π.χ. favicon.ico, vite.svg).

- Διαφορά από τον φάκελο src/assets:

  Τα αρχεία στο public δεν περνούν από το build process του Vite.

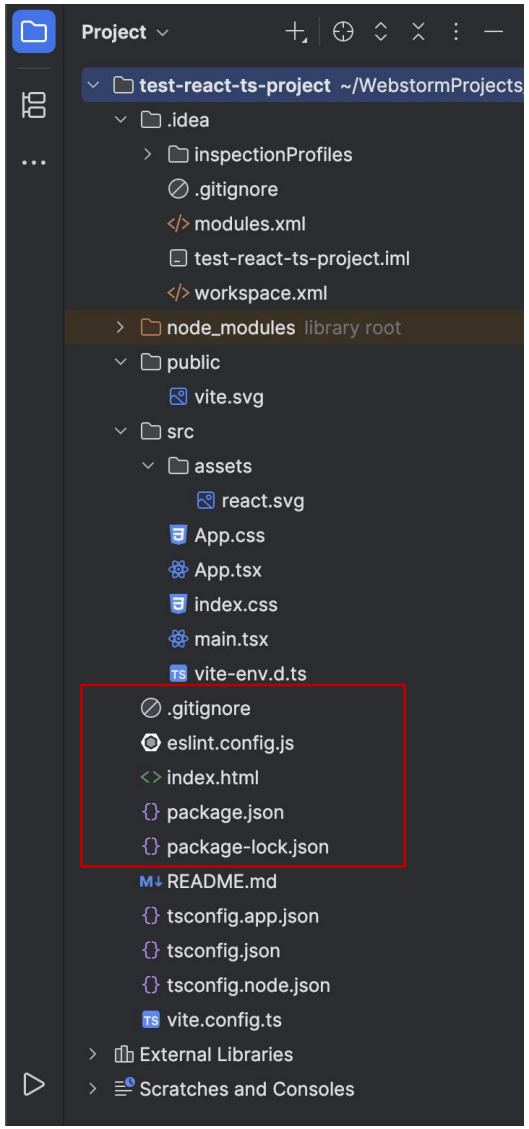
Project Structure (4)



- Ο φάκελος **src** (source) ο οποίος περιέχει όλα τα αρχεία της εφαρμογής μας.

App.tsx	Το βασικό component της εφαρμογής. Εδώ οργανώνουμε το layout ή το routing.
App.css	Styles που σχετίζονται με το App.tsx. Συνήθως για layout ή βασικά components.
index.css	Γενικά styles που εφαρμόζονται σε όλη την εφαρμογή (π.χ. reset, γραμματοσειρές).
main.tsx	Το "σημείο εκκίνησης" της εφαρμογής. Κάνει render το App.tsx στο DOM.
vite-env.d.ts	Types για την υποστήριξη ειδικών χαρακτηριστικών του Vite.
assets/	Φάκελος για εικόνες, icons, fonts κ.λπ.

Project Structure (5)



- .gitignore

Ορίζει ποια αρχεία/φάκελοι **δεν** θα παρακολουθούνται από το Git.

- eslint.config.js

Ρυθμίσεις για τον **ESLint**, το εργαλείο εντοπισμού και επιδιόρθωσης λαθών στον κώδικα.

- index.html

Το βασικό HTML αρχείο — το React app "μπαίνει" μέσα στο div #root.

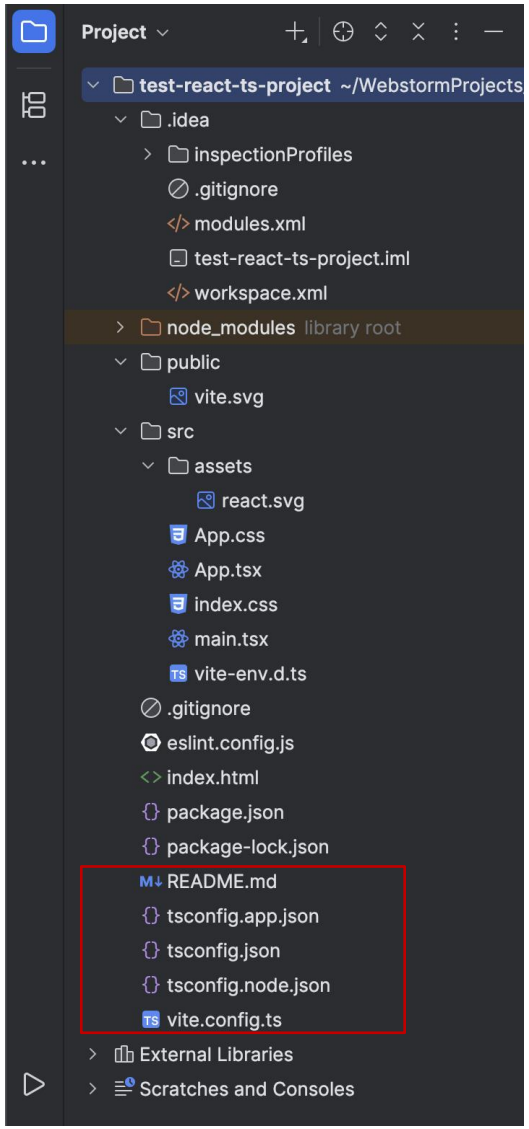
- package.json

Περιέχει πληροφορίες για το project, scripts, εξαρτήσεις (dependencies), κλπ.

- package-lock.json

Κλειδώνει τις ακριβείς εκδόσεις πακέτων για σταθερό builds (auto-generated).

Project Structure (6)



- README.md

Περιγραφικό αρχείο για το project.

- tsconfig.json

Κύριες ρυθμίσεις TypeScript (τι επιτρέπει/απαγορεύει στον κώδικα TS).

- tsconfig.app.json

Ρυθμίσεις TypeScript συγκεκριμένα για τον κώδικα (src/).

- tsconfig.node.json

Ρυθμίσεις TypeScript για αρχεία που τρέχουν στο Node.js (π.χ. config, scripts).

- vite.config.ts

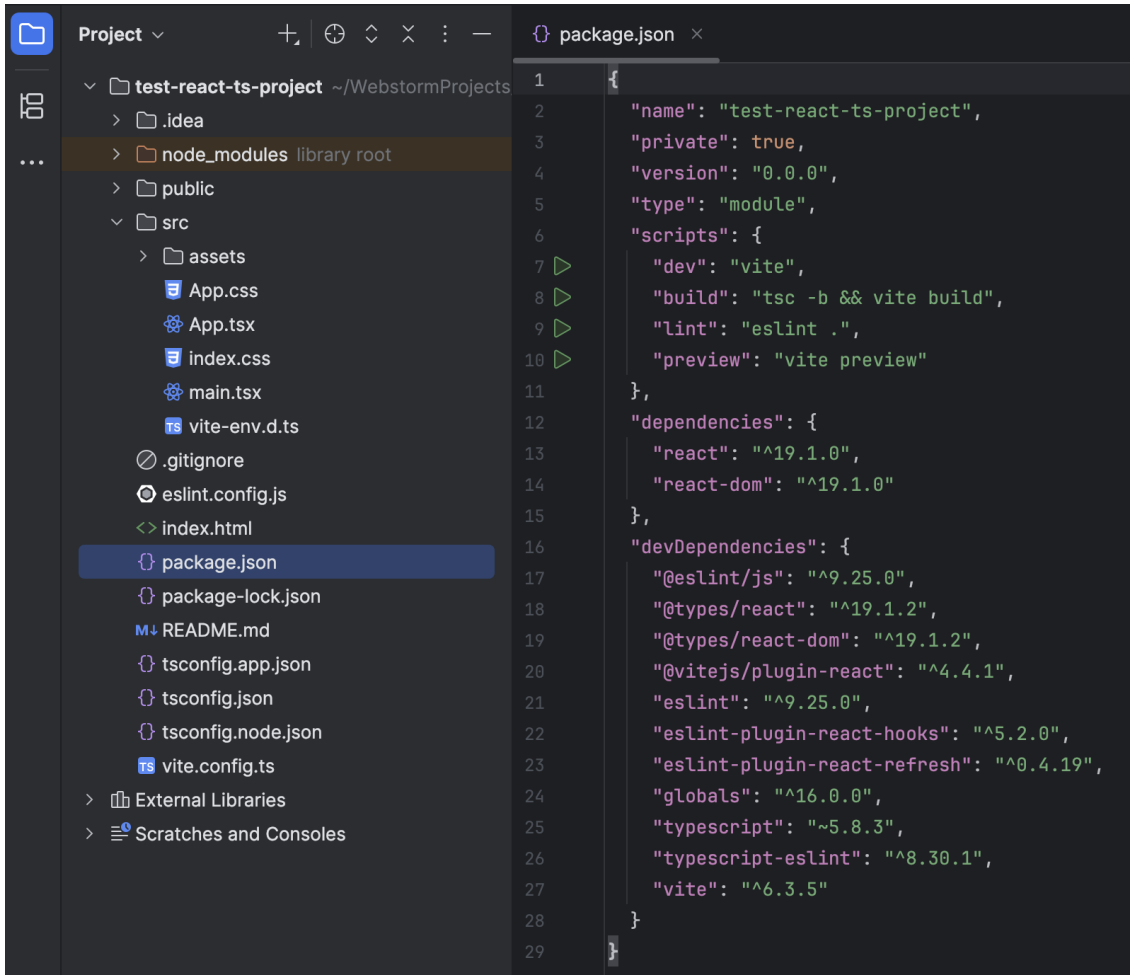
Ρυθμίσεις για το Vite (π.χ. plugins, aliases, περιβάλλοντα).

Project Structure (7)

- Ο φάκελος **src** (source) σε ένα πλήρες project μπορεί να περιλαμβάνει και παρακάτω:

components/	Επαναχρησιμοποιήσιμα React components (π.χ. buttons, forms, cards κ.λπ.).
pages/	Σελίδες της εφαρμογής (αν χρησιμοποιείται routing, π.χ. Home.tsx, About.tsx).
hooks/	Custom React hooks
utils/	Χρήσιμες συναρτήσεις/βοηθητικά (π.χ. format ημερομηνίας, number parser).
types/ ή @types/	Ορισμοί τύπων TypeScript που χρησιμοποιούνται σε όλη την εφαρμογή.
contexts/	React Context Providers (π.χ. AuthContext, ThemeContext).
store/	Αν χρησιμοποιείται state management (π.χ. Redux, Zustand).

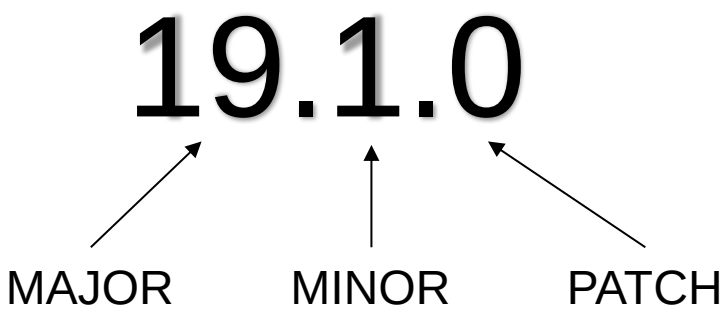
package.json



- Περιέχει πληροφορίες για το project, scripts, εξαρτήσεις (dependencies), κλπ.
- Τα dependencies χωρίζονται σε runtime (@runtime) τα dev (@compile time)
- Το caret (^) *επιτρέπει αλλαγές σε minor και patch versions* semantic versioning (SemVer).
- Το tilde (~) *επιτρέπει* αλλαγές μόνο σε patch versions

Semantic Versioning (SemVer)

- Το **SemVer** είναι μια μορφή έκδοσης που δείχνει τι αλλαγές έγιναν σε μια βιβλιοθήκη/εφαρμογή.



Μέρος	Πότε αλλάζει	Παράδειγμα αλλαγής
MAJOR	Αν γίνουν μη συμβατές αλλαγές	1.x.x → 2.0.0
MINOR	Νέα λειτουργία, αλλά συμβατή	1.4.2 → 1.5.0
PATCH	Διορθώσεις bug, συμβατές	1.4.2 → 1.4.3

package-lock.json

- Το package-lock περιέχει ότι έχει το package.json και επιπλέον τα transitive dependencies
- Περιέχει τα τελικά versions των dependencies που έχουν γίνει install (το package.json περιέχει version ranges)
- Την 1η φορά με βάση το package.json γίνεται το **npm i (ή npm install)** και δημιουργείται το package-lock.json

App.tsx (1)

App.tsx

```tsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() : Element { Show usages
  const [count, setCount] = useState( initialState: 0)
  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() : void => setCount((count : number ) : number => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.tsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
}

export default App Show usages
```

Τι είναι το App.tsx σε ένα νέο project;

- Είναι το κύριο αρχείο της εφαρμογής React — το αρχικό component από όπου ξεκινά όλη η εφαρμογή.
- Περιέχει την αρχική σελίδα που βλέπει ο χρήστης μόλις φορτώσει η εφαρμογή.

Τι κάνει αυτό το πρόγραμμα;

- Δημιουργεί ένα κουμπί που εμφανίζει έναν αριθμό (τον μετρητή).
- Όταν ο χρήστης πατάει το κουμπί, ο αριθμός αυξάνεται κατά 1.
- Εμφανίζει τα λογότυπα του Vite και του React και λειτουργούν ως σύνδεσμοι προς τις επίσημες σελίδες τους.

App.tsx (2)

App.tsx

```tsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() : Element { Show usages
  const [count, setCount] = useState( initialState: 0)
  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() : void => setCount((count : number ) : number => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.tsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
}

export default App Show usages
```

- **const [count, setCount] = useState(0)**
  Δημιουργεί μια μεταβλητή count με αρχική τιμή 0 και μια συνάρτηση setCount για να την αλλάζουμε.
- **onClick={...}**
  Δηλώνει τι θα γίνει όταν ο χρήστης κάνει κλικ στο button.
- **() => setCount((count) => count + 1)**
  Είναι μια συνάρτηση χωρίς όνομα (arrow function) που εκτελείται όταν πατηθεί το κουμπί. Καλεί τη setCount για να ενημερώσει την τιμή του count
- **setCount((count) => count + 1)**
  Είναι η σύσταση του React να χρησιμοποιείται η προηγούμενη τιμή (μέσω του callback) όταν αλλάζουμε το state, ειδικά αν η ενημέρωση εξαρτάται από την προηγούμενη τιμή.
- **count is {count}**
  Εμφανίζει το τρέχον state του count μέσα στο button (π.χ. count is 5).

Landing Page