REACT (VITE) + REACT ROUTER → DEPLOY TO GITHUB PAGES (COMPLETE GUIDE)

INITIAL SETUP (ONLY ONCE PER PROJECT)

---

1. Configure Vite base path
   Edit vite.config.ts:
   base: "/REPO_NAME/"
   Why: GitHub Pages serves your site inside /REPO_NAME/
   Result: JS/CSS files load correctly instead of 404.
2. Configure React Router
   Edit App.tsx:

Why: The app runs inside a subfolder.
Result: /REPO_NAME/ works like / inside React.

3. Build the project
   Command:
   npm run build
   Why: GitHub Pages cannot run React source code.
   Result: Production website created inside dist/.
4. Fix refresh / direct link problem (VERY IMPORTANT)
   Command:
   cp dist/index.html dist/404.html
   Why: GitHub Pages does not understand SPA routing.
   Result: Refreshing the page (F5) no longer shows 404.
5. Deploy website
   Command:
   npm run deploy
   Why: dist folder must be uploaded to gh-pages branch.
   Result: Website becomes publicly accessible.
6. Enable GitHub Pages
   GitHub → Repository → Settings → Pages
   Set:
   Source: Deploy from a branch
   Branch: gh-pages
   Folder: /(root)
   Why: GitHub must know which branch hosts the site.
   Result: A live URL is generated.

WHAT TO DO EVERY TIME YOU CHANGE THE SITE

If you change UI / components / pages:

```
git add .
git commit -m "update"
git push origin main
npm run build
cp dist/index.html dist/404.html
npm run deploy
```

Why: main stores the code, but Pages reads the built output.
Result: Both the repository and the website update.

If you change router or base configuration:
Run the same commands above.
Why: Config changes only apply after rebuilding.
Result: Routing errors and blank page disappear.

If you only change README or documentation:

```
git add .
git commit -m "docs"
git push origin main
```

Why: Documentation does not affect the website.
Result: No deploy needed.

WHAT THESE FOLDERS AND FILES ARE

---

src/
Your real application code (components, pages, routing).
Result: This is what you develop.

public/
Static files copied directly (favicon, static images).
Result: Available as-is on the site.

dist/
Production build output (optimized HTML/CSS/JS).
Result: This is the real website GitHub Pages hosts.

node_modules/
Installed npm libraries.
Result: Very large — never committed to git.

package.json
Project dependencies and scripts (dev, build, deploy).
Result: npm commands come from here.

vite.config.ts
Vite build configuration (especially base path).
Result: Controls asset paths and build behavior.

main branch
Your source code branch.
Result: Development history lives here.

gh-pages branch
Contains only the built site (dist content).
Result: GitHub Pages serves the website from this branch.

HOW GITHUB PAGES ACTUALLY READS YOUR SITE

You actually have TWO separate worlds inside one repository:

main branch:
Contains React source code (development files).

gh-pages branch:
Contains built HTML/CSS/JS (the real website).

The command:
npm run deploy

uses the "gh-pages" npm package and does this automatically:

1. Takes the dist folder
2. Creates a commit
3. Pushes it to the gh-pages branch

GitHub Pages is configured to read ONLY the gh-pages branch.

Important:
GitHub never runs React.
GitHub only serves ready HTML files.

Mental model:

main = kitchen (raw ingredients)
npm run build = cooking
dist = prepared food
gh-pages = restaurant table
GitHub Pages = waiter serving users

MOST COMMON WHITE PAGE CAUSES

---

1. Wrong base path in vite.config.ts
2. Missing basename in BrowserRouter
3. Forgot to create 404.html
4. Forgot to run deploy after build