

Session 1 — Setting Up Your Trivia Game

Welcome to **React** Development! 🎉

You're about to launch your dev setup and start building like a pro. This guide breaks everything down into bite-sized tasks so you can move fast, learn by doing, and see results right away. Ready to dive in? Let's go!

Table of Contents

- [Launch Your Codespace](#)
- [Project Navigation](#)
- [Start the Development Server](#)
- [Replace the Placeholder Component](#)
- [Update the Page Title](#)
- [Essential Terms](#)
- [Ask the AI](#)

Launch Your Codespace

Let's get your cloud coding environment up and running so you can start building without messing with local installs. This is your dev playground in the cloud — no setup headaches, no installs, just code.

1. Head over to github.com and log in using the account you set up during pre-camp.
2. Go to github.com/wizcamp/wizcamp-realms-demo and click **Use this template** → **Open in codespace**.
3. Wait a bit while your Codespace builds. You'll see a VS Code editor pop up in your browser.
4. Once it loads, make sure you can see the project files in the file explorer on the left.
5. **Customize your theme:** Click the gear icon in the bottom left, go to **Themes -> Color Theme**, and pick your favorite (Dark+ is popular for coding).

Why This Matters

Codespaces give everyone the same setup — no more “it works on my machine” drama. If you mess something up, just delete and start fresh. It's like having a reset button for your entire

dev environment. You will use this Codespace for all sessions.

Bonus Challenge

Visit github.com/codespaces to explore more about managing your Codespaces.

Project Navigation

Quick orientation to help you find files during today's tasks:

```
wizcamp-realms-demo/
├── src/                    # Your React code lives here
│   ├── components/        # React components (SplashScreen, etc.)
│   └── App.jsx            # Main app component (you'll edit this!)
├── public/                # Static files (images, etc.)
├── session-guides/        # Step-by-step guides (like this one)
├── index.html             # HTML entry point (you'll edit this too!)
└── package.json           # Project configuration (npm scripts, dependencies)
```

For today's tasks, you'll only work with:

- `src/App.jsx` — to swap components
- `index.html` — to update the page title
- `session-guides/` — for instructions (that's where you are now!)

Don't worry about the other folders yet — we'll explore them in future sessions.

Start the Development Server

Preview the app in your browser by running the dev server to confirm everything is wired up correctly.

1. Launch the dev server from the terminal: `npm run dev`.
2. Click "Open in Browser" or visit the provided localhost URL (e.g., `http://localhost:5173/`).
3. The starter app should load, showing a placeholder component.

Why This Matters

Your dev server is like having a live preview of your creation. Every change you make shows up instantly — you're watching your code come to life in real-time.

Bonus Challenge

Try stopping and restarting the dev server:

- Stop the server with `ctrl + c` in the terminal.
- Check localhost again to see the app is gone.
- Restart it with `npm run dev` and refresh the browser to see the app again.

Replace the Placeholder Component

With the development server still running, replace the placeholder component with the game's splash screen component to experience live updates in action.

1. **Open** `src/App.jsx`
2. **Add the import:** `import SplashScreen from "../components/SplashScreen";`
3. **Replace** `<StartHere />` **with** `<SplashScreen />`
4. **Watch the screen update instantly** — no save needed!

Why This Matters

Components are the building blocks of **React** web apps — kind of like digital LEGO pieces. You build apps by snapping them together.

You probably noticed the `.jsx` file extension. That's because these **components** are written in a special syntax called **JSX**. It looks a lot like HTML, but is actually JavaScript under the hood. JSX lets you describe what the UI should look — using a syntax that's readable like HTML but powered by JavaScript.

The live update “magic” you experienced is actually powered by a build tool we are using called **Vite**, which uses a process known as **Hot Module Replacement (HMR)** to apply “smart updates” to your app instantly as you code.

Update the Page Title

Even though React apps are built with components, they still use a standard HTML file as the entry point. Let's update the page title to reflect our project name.


1. **Open** `index.html`
2. **Update the** `<title>` **tag** to: `Wizcamp Realms - Legends of Trivia`
3. **Confirm the browser tab** displays the new title.





💡 Why This Matters

A descriptive page title is important for usability, accessibility, and SEO. It helps users identify your app when they have multiple tabs open and improves discoverability in search engines.

📚 Essential Terms

Quick reference for all the tools and concepts you just experienced:

Term	Definition	Why it matters
 Codespace	A cloud dev environment from GitHub — a ready-made VS Code workspace that runs in your browser.	You'll launch this first; it gives everyone the same setup so you can jump straight to coding.
 VS Code	Your coding headquarters — think Photoshop but for building apps instead of editing photos.	This is where the magic happens. File explorer, code editor, terminal — all in one place.
 Node.js	JavaScript that runs on your computer (not just in browsers) — like having a JavaScript engine everywhere.	Powers your dev tools and lets you run <code>npm</code> commands. It's JavaScript unleashed.
 npm	Node's package manager — installs libraries and runs scripts (<code>npm run dev</code>).	Use it to install dependencies and start the dev server.
 Vite	The Ferrari of dev servers — crazy fast and makes your app load instantly during development.	When you run <code>npm run dev</code> , Vite serves your app at lightning speed. You'll see why it's so popular.

 Hot Module Replacement (HMR)	Updates only the changed code in the browser without a full reload, often keeping app state.	Lets you see edits instantly (CSS/JS) while you work — you'll notice changes apply without losing progress.
 React	A library for building UIs out of components; it updates the UI when data changes.	The project is a React app — you'll edit components to change what users see.
🌟 JSX	JavaScript syntax that looks like HTML — used to describe UI in React components (<code>.jsx</code>).	You'll edit <code>.jsx</code> files (e.g., <code>src/App.jsx</code>) to swap components and change UI.
 component	A reusable piece of UI that can include markup, styles, and logic (example: <code><SplashScreen /></code>).	You'll replace a placeholder component with <code>SplashScreen</code> to practice editing and imports.
 Document Object Model (DOM)	The browser's object model of the page — JS code (including React) reads and updates the DOM to change what users see.	React updates the DOM when you change components or state (e.g., button clicks, title updates).

Ask the AI — What Just Happened?

You just launched your Codespace, ran your dev server, swapped a component, and updated your page title — nice work!

Now let's make sure you understand what you did and why it matters. Here are the most impactful questions to ask your AI assistant about today's session:

- **Why is cloud development better for beginners?**
- **What is a development server and why do we need it?**
- **What does the localhost URL mean?**
- **What's the difference between `npm run dev` and `npm start`?**
- **How and why do I import a component in React? Where am I importing all that from?**

- What does `import SplashScreen from './components/SplashScreen'` mean?
- In a React app, what does the `index.html` file do?