Session 1 — Setting Up Your Trivia Game

Welcome to **React** Development! 🎉

You're about to launch your dev setup and start building your trivia game. 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

☁️ Creating Your Codespace

🎯 **Goal:** Set up your cloud development environment so you can start coding without any local installations.

Step 1: Sign in to GitHub

Go to github.com and log in with your account.

Step 2: Launch the template

Go to github.com/wizcamp/wizcamp-realms-demo and click **Use this template → Open in a codespace**.

Expected result: Your Codespace begins building (this takes 1-2 minutes).

Step 3: Wait for the environment to load

Once the build completes, VS Code will open in your browser.

Expected result: Project files appear in the file explorer on the left side.

Step 4: Customize your theme (optional)

Click the ⚙️ gear icon in the bottom left → **Themes** → **Color Theme** → pick your favorite (Dark+ is popular for coding).

💡 **Why Codespaces rocks:** Everyone gets the same setup — no more "it works on my machine" drama. Mess something up? Just delete and start fresh. It's like having a reset button for your entire dev environment. You'll use this Codespace for all 12 sessions.

🏆 **Bonus Challenge:** Go to github.com/codespaces to explore more about managing your Codespaces.

🗂️ Navigating the Project

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

```
wizcamp-realms/
├── 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.)
├── 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

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

🚀 Starting Your Development Server

🎯 **Goal:** Practice starting and stopping the local server you'll use to preview real-time changes as you build your app.

Step 1: Run the dev server

From the terminal at the bottom of your Codespace, run:

```
npm run dev
```

Step 2: Open the app in your browser

After running the command, you'll see output like:

```
VITE v7.1.7  ready in 2473 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Follow the link (ctrl + click), copy-paste it into a new browser tab, or click "Open in Browser" if a dialog appears.

Expected result: A web page displaying the starter app with placeholder content.

Step 3: Stop the server

Go back to your terminal and press `Ctrl + C`.

Expected result:

- Terminal returns to the command prompt
- Refreshing the browser shows a connection error (app no longer running)

💡 **Your dev server cheat sheet:** Run `npm run dev` to fire up your server and see your app live. Hit `Ctrl + C` to shut it down. You'll use these commands constantly — they're about to become muscle memory.

🧩 Editing Your First Component

🎯 **Goal:** Edit the App component to display your game's splash screen and experience React's live updates.

File: `src/App.jsx`

Step 1: Import SplashScreen

At the top of the file, add:

```
import SplashScreen from "./components/SplashScreen";
```

Step 2: Swap the placeholder for SplashScreen

In the App component's return statement, replace `<StartHere />` with `<SplashScreen />`:

```
export default function App() {
  return (
    <div className="app-container">
      <SplashScreen />  {/* Was: <StartHere /> */}
    </div>
  );
}
```

Expected result: The screen updates instantly — no save needed, no refresh required. That's the magic of Hot Module Replacement!

💡 **What just happened?** Components are React's building blocks — think digital LEGO pieces you snap together to build apps. That `.jsx` extension? It's JSX, a special syntax that looks like HTML but is actually JavaScript. And that instant update you just saw? That's Vite's Hot Module Replacement (HMR) doing its thing — your dev server is basically a live preview of your creation.

🏷️ Updating the Page Title

File: `index.html`

Open `index.html` and change the `<title>` tag to `Wizcamp Realms - Legends of Trivia`.

Expected result: The browser tab displays your new title.

💡 **Why this matters:** Even though React apps are built with components, they still need a standard HTML file as the entry point. A descriptive page title helps users identify your app when they have multiple tabs open — plus it's crucial for accessibility and SEO.

📚📋 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 `npm` commands. It's JavaScript unleashed.
📦 npm	Node's package manager — installs libraries and runs scripts (`npm run dev`).	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 `npm run dev`, 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 (`.jsx`).	You'll edit `.jsx` files (e.g., `src/App.jsx`) to swap components and change UI.
🧩 component	A reusable piece of UI that can include markup, styles, and logic (example: `<SplashScreen />`).	You'll replace a placeholder component with `SplashScreen` 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 — Setting Up Your Trivia Game

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?**