# **Activity 4— Build a tiny DApp UI for your ERC-20 on DIDLab**

## **What you’ll have at the end**

A single-file web app that:

* Connects to MetaMask and **adds/switches** to your team’s DIDLab network
* Loads your **ERC-20 token** (name/symbol/decimals)
* Shows **your balance**
* Lets you **transfer** tokens
* Updates automatically when transfers happen

### **Prereqs**

* **Node 22.x** installed
* **MetaMask** in your browser
* Your team’s **RPC** & **Chain ID** and your **token address**
  + Team → RPC / ChainId (decimal):
    - 01 → <https://hh-01.didlab.org> / **31337**
    - 02 → <https://hh-02.didlab.org> / **31338**
    - …
    - 12 → <https://hh-12.didlab.org> / **31348**
* Import your **DIDLab faucet private key** into MetaMask (so the account has ETH and your token from deploy)

## **1) Make a folder & file**

mkdir didlab-dapp && cd didlab-dapp

Create index.html and paste everything below. **Edit just the CONFIG block** (team & token).

This uses ESM CDN imports (no build step). If your browser blocks <file://> module imports, serve it locally:

python3 -m http.server 8000 then open [http://localhost:8000](http://localhost:8000/)

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="utf-8" />  
 <title>DIDLab — ERC-20 DApp</title>  
 <meta name="viewport" content="width=device-width, initial-scale=1" />  
 <style>  
 html,body{font-family:system-ui,Segoe UI,Roboto,Helvetica,Arial,sans-serif;margin:0;padding:0;background:#0b1320;color:#e6eaf2}  
 .wrap{max-width:880px;margin:40px auto;padding:24px;border:1px solid #24314a;border-radius:14px;background:#0f1a2b}  
 h1{margin:0 0 16px;font-size:22px}  
 .row{display:flex;gap:12px;flex-wrap:wrap;margin:8px 0}  
 .row > \*{flex:1}  
 input,select,button{padding:10px 12px;border:1px solid #314164;background:#0b1527;color:#e6eaf2;border-radius:10px}  
 button{cursor:pointer}  
 button.primary{background:#335cff;border-color:#335cff}  
 .mono{font-family:ui-monospace,SFMono-Regular,Menlo,Consolas,monospace}  
 .card{padding:12px;border:1px solid #24314a;border-radius:10px;background:#0b1527}  
 .muted{opacity:.8}  
 .ok{color:#7be07b}.warn{color:#ffd166}.err{color:#ff6b6b}  
 a{color:#9bc1ff}  
 .kvs{display:grid;grid-template-columns:180px 1fr;gap:8px}  
 .kvs div{padding:6px 8px;border-bottom:1px dashed #22314e}  
 </style>  
</head>  
<body>  
<div class="wrap">  
 <h1>💠 DIDLab — ERC-20 DApp</h1>  
  
 <!-- CONFIG -->  
 <div class="card">  
 <div class="row">  
 <div>  
 <label>Team</label>  
 <select id="team">  
 <option>01</option><option>02</option><option>03</option><option>04</option>  
 <option>05</option><option>06</option><option>07</option><option>08</option>  
 <option>09</option><option>10</option><option>11</option><option>12</option>  
 </select>  
 </div>  
 <div>  
 <label>Token Address</label>  
 <input id="tokenAddr" class="mono" placeholder="0x…" />  
 </div>  
 </div>  
 <div class="row">  
 <button id="btnConnect" class="primary">1) Connect & Switch Network</button>  
 <button id="btnLoad">2) Load Token</button>  
 <button id="btnWatch">Add Token to MetaMask</button>  
 <button id="btnRefresh">Refresh Balance</button>  
 </div>  
 </div>  
  
 <!-- STATUS -->  
 <div class="row">  
 <div class="card">  
 <div class="kvs">  
 <div class="muted">Account</div><div id="acct" class="mono">—</div>  
 <div class="muted">Network</div><div id="net" class="mono">—</div>  
 <div class="muted">Token</div><div id="meta" class="mono">—</div>  
 <div class="muted">Balance</div><div id="bal" class="mono">—</div>  
 </div>  
 </div>  
 </div>  
  
 <!-- TRANSFER -->  
 <div class="card">  
 <h3>Transfer</h3>  
 <div class="row">  
 <input id="to" class="mono" placeholder="Recipient 0x…" />  
 <input id="amt" class="mono" placeholder="Amount (human units)" />  
 <button id="btnSend" class="primary">Send</button>  
 </div>  
 <div id="txlog" class="mono muted"></div>  
 </div>  
  
 <p class="muted">Tip: values persist in localStorage.</p>  
</div>  
  
<script type="module">  
 // ===== Imports (no build step)  
 import {  
 createPublicClient, createWalletClient, custom,  
 getAddress, formatUnits, parseUnits  
 } from "[https://esm.sh/viem@2.37.5](mailto:https://esm.sh/viem@2.37.5)";  
  
 // ===== Minimal ERC-20 ABI  
 const ERC20\_ABI = [  
 { type:"function", name:"name", stateMutability:"view", inputs:[], outputs:[{type:"string"}]},  
 { type:"function", name:"symbol", stateMutability:"view", inputs:[], outputs:[{type:"string"}]},  
 { type:"function", name:"decimals",stateMutability:"view", inputs:[], outputs:[{type:"uint8"}]},  
 { type:"function", name:"balanceOf", stateMutability:"view", inputs:[{name:"account",type:"address"}], outputs:[{type:"uint256"}]},  
 { type:"function", name:"transfer", stateMutability:"nonpayable", inputs:[{name:"to",type:"address"},{name:"amount",type:"uint256"}], outputs:[{type:"bool"}]},  
 { type:"event", name:"Transfer", inputs:[  
 {indexed:true, name:"from", type:"address"},  
 {indexed:true, name:"to", type:"address"},  
 {indexed:false,name:"value",type:"uint256"}  
 ]},  
 ];  
  
 // ===== Team → RPC/Chain config  
 const TEAM\_CHAINS = {  
 "01": { id: 31337, name:"DIDLab Team 01", rpc:"https://hh-01.didlab.org" },  
 "02": { id: 31338, name:"DIDLab Team 02", rpc:"https://hh-02.didlab.org" },  
 "03": { id: 31339, name:"DIDLab Team 03", rpc:"https://hh-03.didlab.org" },  
 "04": { id: 31340, name:"DIDLab Team 04", rpc:"https://hh-04.didlab.org" },  
 "05": { id: 31341, name:"DIDLab Team 05", rpc:"https://hh-05.didlab.org" },  
 "06": { id: 31342, name:"DIDLab Team 06", rpc:"https://hh-06.didlab.org" },  
 "07": { id: 31343, name:"DIDLab Team 07", rpc:"https://hh-07.didlab.org" },  
 "08": { id: 31344, name:"DIDLab Team 08", rpc:"https://hh-08.didlab.org" },  
 "09": { id: 31345, name:"DIDLab Team 09", rpc:"https://hh-09.didlab.org" },  
 "10": { id: 31346, name:"DIDLab Team 10", rpc:"https://hh-10.didlab.org" },  
 "11": { id: 31347, name:"DIDLab Team 11", rpc:"https://hh-11.didlab.org" },  
 "12": { id: 31348, name:"DIDLab Team 12", rpc:"https://hh-12.didlab.org" },  
 };  
  
 // ===== UI elements  
 const el = (id)=>document.getElementById(id);  
 const teamSel = el("team");  
 const tokenInput = el("tokenAddr");  
 const acctEl = el("acct"), netEl = el("net"), metaEl = el("meta"), balEl = el("bal");  
 const txlog = el("txlog");  
  
 // ===== State  
 let chain, token, decimals=18, symbol="TOKEN", name="Token";  
 let walletClient, publicClient, account;  
  
 // ===== Load/save simple prefs  
 const saved = JSON.parse(localStorage.getItem("didlab-ui")||"{}");  
 if (saved.team) teamSel.value = saved.team;  
 if (saved.token) tokenInput.value = saved.token;  
  
 function save() {  
 localStorage.setItem("didlab-ui", JSON.stringify({  
 team: teamSel.value, token: tokenInput.value  
 }));  
 }  
  
 // ===== Chain helpers  
 function currentChain() {  
 const t = TEAM\_CHAINS[teamSel.value];  
 return {  
 id: t.id,  
 name: t.name,  
 nativeCurrency: { name:"ETH", symbol:"ETH", decimals:18 },  
 rpcUrls: { default: { http: [t.rpc] } }  
 };  
 }  
 function hexChainId(id){ return "0x"+id.toString(16); }  
  
 async function ensureMetamask() {  
 if (!window.ethereum) throw new Error("MetaMask not found");  
 }  
  
 async function addOrSwitchNetwork() {  
 const t = TEAM\_CHAINS[teamSel.value];  
 const params = [{  
 chainId: hexChainId(t.id),  
 chainName: t.name,  
 rpcUrls: [t.rpc],  
 nativeCurrency: { name:"ETH", symbol:"ETH", decimals:18 }  
 }];  
 try {  
 await window.ethereum.request({ method: "wallet\_switchEthereumChain", params: [{ chainId: params[0].chainId }] });  
 } catch (e) {  
 // Not added? add it.  
 await window.ethereum.request({ method: "wallet\_addEthereumChain", params });  
 }  
 }  
  
 async function connect() {  
 await ensureMetamask();  
 chain = currentChain();  
  
 // bridge viem through MetaMask (no CORS issues)  
 walletClient = createWalletClient({ chain, transport: custom(window.ethereum) });  
 publicClient = createPublicClient({ chain, transport: custom(window.ethereum) });  
  
 await addOrSwitchNetwork();  
  
 const addrs = await window.ethereum.request({ method: "eth\_requestAccounts" });  
 account = getAddress(addrs[0]);  
  
 acctEl.textContent = account;  
 netEl.textContent = `${chain.name} (#${chain.id})`;  
 logOk("Connected. Network ready.");  
 }  
  
 async function loadToken() {  
 token = getAddress(tokenInput.value.trim());  
 save();  
  
 name = await publicClient.readContract({ address: token, abi: ERC20\_ABI, functionName: "name" });  
 symbol = await publicClient.readContract({ address: token, abi: ERC20\_ABI, functionName: "symbol" });  
 decimals = /\*\* @type {number} \*/(await publicClient.readContract({ address: token, abi: ERC20\_ABI, functionName: "decimals" }));  
 metaEl.textContent = `${name} (${symbol}), ${decimals}d, ${token}`;  
 logOk(`Loaded token ${symbol}`);  
  
 // Watch transfers involving this account and refresh balance  
 publicClient.watchContractEvent({  
 address: token, abi: ERC20\_ABI, eventName: "Transfer",  
 onLogs: (logs)=>{  
 for (const l of logs) {  
 if (l.args.from?.toLowerCase()===account.toLowerCase() ||  
 l.args.to?.toLowerCase()===account.toLowerCase()) {  
 refresh();  
 break;  
 }  
 }  
 }  
 });  
  
 await refresh();  
 }  
  
 async function refresh() {  
 if (!token || !account) return;  
 const bal = /\*\* @type {bigint} \*/(await publicClient.readContract({  
 address: token, abi: ERC20\_ABI, functionName: "balanceOf", args: [account]  
 }));  
 balEl.textContent = `${formatUnits(bal, Number(decimals))} ${symbol}`;  
 }  
  
 async function send() {  
 const to = getAddress(el("to").value.trim());  
 const amtStr = el("amt").value.trim();  
 if (!amtStr) throw new Error("Enter amount");  
 const amount = parseUnits(amtStr, Number(decimals));  
 const hash = await walletClient.writeContract({  
 address: token, abi: ERC20\_ABI, functionName: "transfer",  
 args: [to, amount],  
 // mild tips (EIP-1559)  
 maxPriorityFeePerGas: 2\_000\_000\_000n,  
 maxFeePerGas: 21\_000\_000\_000n,  
 account  
 });  
 logWarn(`Submitted: ${hash}`);  
 const rcpt = await publicClient.waitForTransactionReceipt({ hash });  
 logOk(`Mined in block ${rcpt.blockNumber} (gas ${rcpt.gasUsed})`);  
 }  
  
 async function watchAsset() {  
 if (!token || !symbol) return;  
 await window.ethereum.request({  
 method: "wallet\_watchAsset",  
 params: { type: "ERC20", options: { address: token, symbol, decimals: Number(decimals) } }  
 });  
 logOk("Token added to MetaMask (or already present).");  
 }  
  
 // ===== Logging  
 function logOk(m){ txlog.innerHTML = `<div class="ok">${escapeHtml(m)}</div>` + txlog.innerHTML; }  
 function logWarn(m){ txlog.innerHTML = `<div class="warn">${escapeHtml(m)}</div>` + txlog.innerHTML; }  
 function logErr(m){ txlog.innerHTML = `<div class="err">${escapeHtml(m)}</div>` + txlog.innerHTML; }  
 function escapeHtml(s){ return s.replace(/[&<>"]/g, c=>({ '&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;' }[c])); }  
  
 // ===== Wire up UI  
 el("btnConnect").onclick = async ()=>{ try{ await connect(); } catch(e){ logErr(e.message||e);} };  
 el("btnLoad").onclick = async ()=>{ try{ await loadToken(); } catch(e){ logErr(e.message||e);} };  
 el("btnRefresh").onclick = async ()=>{ try{ await refresh(); } catch(e){ logErr(e.message||e);} };  
 el("btnSend").onclick = async ()=>{ try{ await send(); } catch(e){ logErr(e.shortMessage||e.message||e);} };  
 el("btnWatch").onclick = async ()=>{ try{ await watchAsset(); } catch(e){ logErr(e.message||e);} };  
  
 // Restore from localStorage fast  
 (async ()=>{  
 // Pre-fill team/token if saved  
 if (tokenInput.value) metaEl.textContent = `Ready to load ${tokenInput.value}`;  
 })();  
</script>  
</body>  
</html>

### **2) Run it**

python3 -m http.server 8000  
# or: npx http-server -p 8000

Open [http://localhost:8000](http://localhost:8000/)

### **3) Use it**

1. Pick your **Team** and paste your **token address**.
2. Click **Connect & Switch Network** → approve in MetaMask.
3. Click **Load Token** → you’ll see name/symbol/decimals and your balance.
4. **Transfer**: enter recipient 0x… and amount (human units, e.g., 12.34) → **Send**.
5. Click **Add Token to MetaMask** to show balance inside the wallet.

#### **Troubleshooting**

* **Nothing happens on connect** → Make sure MetaMask is installed and you’re in a regular profile (not private window that blocks extensions).
* **Wrong network** → You must approve the network add/switch prompt.
* **Returned no data (0x)** → The **Token Address is wrong** or not a contract on your team’s chain.
* **Insufficient funds** → You don’t have token balance; transfer some to yourself from your deployer (or import the deployer key to MetaMask).