

# 钱包登录DAPP

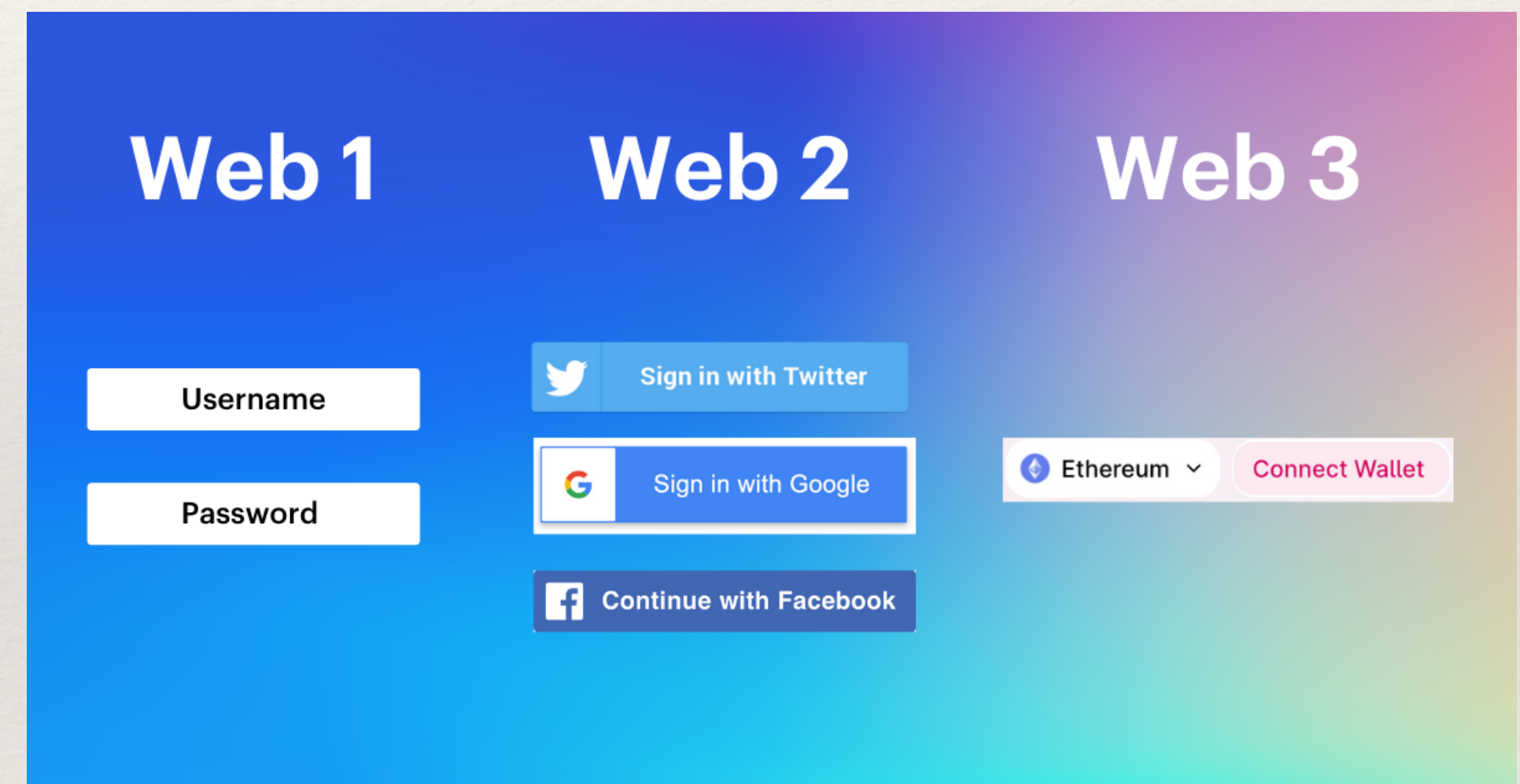
七哥

<https://x.com/0xqige>



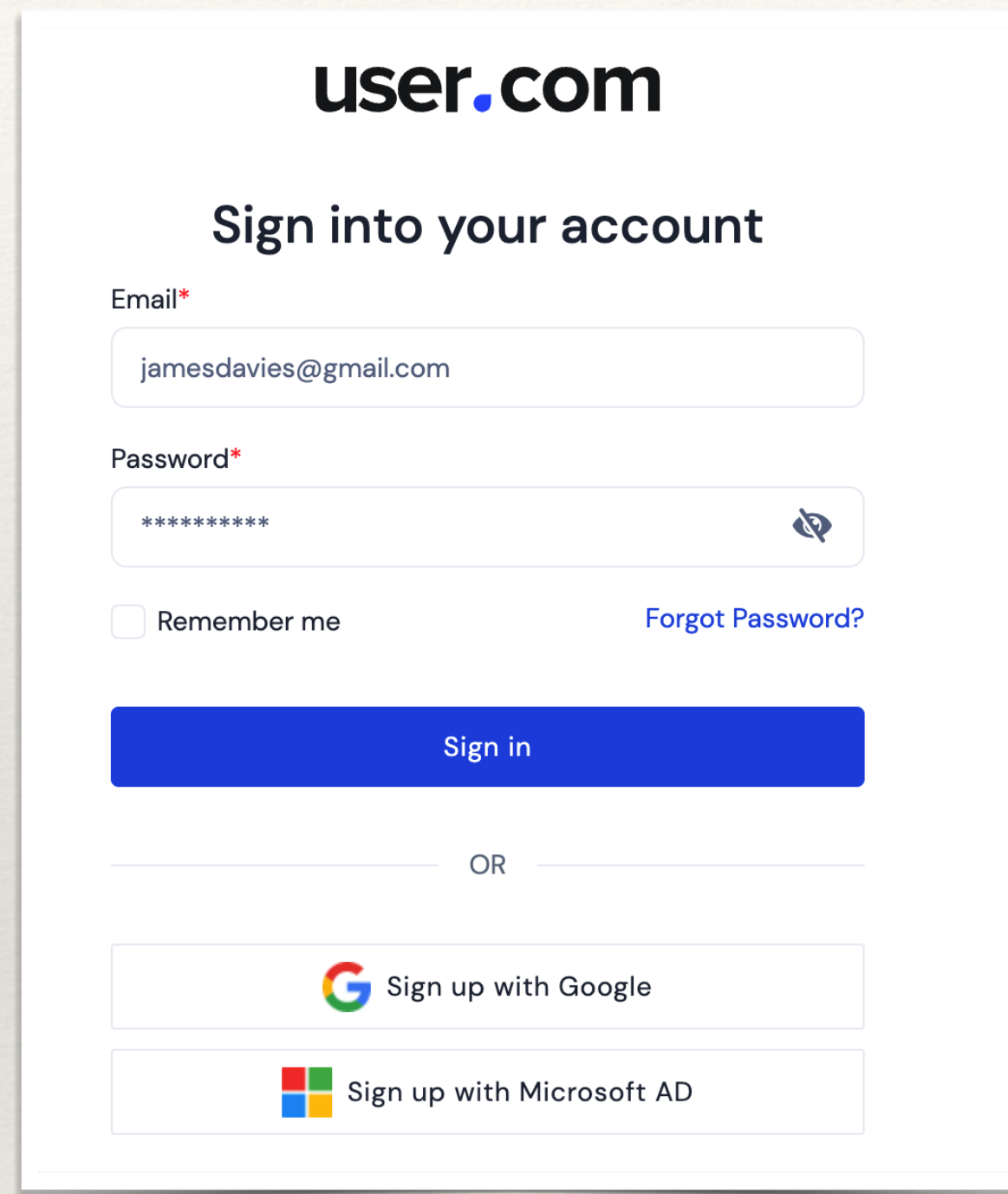
# 目录

- ❖ Web2 vs Web3 登录方式
- ❖ DAPP 登录的多种实现
- ❖ Wallet Connect 原理
- ❖ DAPP UI 集成 Web3 Modal



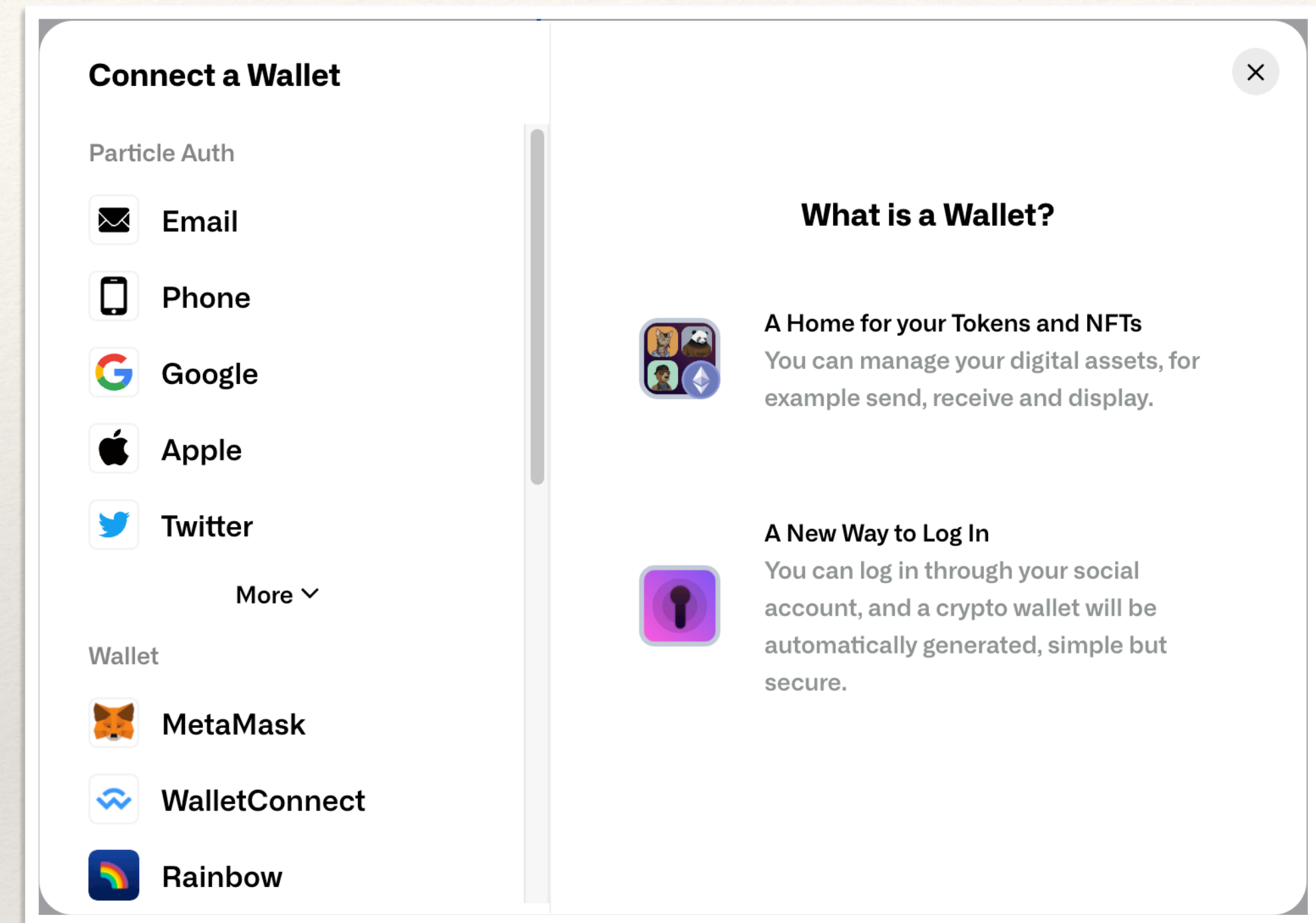


# Web2 vs Web3 登录方式



The image shows a typical Web2 login page for 'user.com'. It features a central form with fields for 'Email\*' and 'Password\*'. The email field contains 'jamesdaves@gmail.com' and the password field contains '\*\*\*\*\*'. Below the password field, there is a 'Remember me' checkbox and a 'Forgot Password?' link. A large blue 'Sign in' button is positioned below the form. At the bottom, there are two buttons for social login: 'Sign up with Google' and 'Sign up with Microsoft AD', separated by an 'OR' label.

- 在多个应用中重复注册不同的帐号，管理多个用户名和密码
- 弱密码被破解
- 容易被盗号



The image shows a Web3 login interface titled 'Connect a Wallet'. It is divided into two panels. The left panel lists various authentication methods under 'Particle Auth': Email, Phone, Google, Apple, and Twitter, followed by a 'More' dropdown. Below this is a 'Wallet' section with options for MetaMask, WalletConnect, and Rainbow. The right panel, titled 'What is a Wallet?', explains that a wallet is a 'Home for your Tokens and NFTs' and provides a 'New Way to Log In' by linking a social account to a crypto wallet.

- 无需中央服务器
- 单一身份
- 高度安全性



# 实现 Web3 注册与登录的技术

智能合约

address => data

加密钱包

Private Key

去中心化存储(IPFS)

url id => content



nick.eth



# DAPP 实现加密钱包登录



1. 原始： 直接利用 Metamask 浏览器插件JS
2. 进阶：
  - 使用 Viem 连接 Metamask
  - 使用 WalletConnect 连接移动钱包
3. 简易： 使用 AppKit(Web3modal) 按钮



# 利用 Metamask 浏览器插件JS

MetaMask 提供易于使用的 API，允许开发者与其钱包进行交互

```

// 检查 MetaMask 是否安装
if (typeof window.ethereum !== 'undefined') {
  console.log('MetaMask is installed!');
}

// 请求账户访问权限
async function connectMetaMask() {
  try {
    const accounts = await ethereum.request({ method: 'eth_requestAccounts' });
    console.log('Connected', accounts[0]);
    // 在此处添加用户登录的逻辑，例如将账户信息发送到服务器进行验证
  } catch (error) {
    console.error(error);
  }
}

// 触发 MetaMask 连接
document.getElementById('connectButton').addEventListener('click', connectMetaMask);
```



# 使用 Viem 连接 Metamask



```
// 1. check if the browser extension wallet is installed
const ethereum = window.ethereum;
if (!ethereum) {
  alert("Please install MetaMask first.");
  return;
}
try {
  // 2. create a wallet client
  const ethereumWalletClient = createWalletClient({
    transport: custom(ethereum),
  });
  // 3. request get addresses: this will trigger the browser extension wallet to show a
  popup to the user to allow access
  const allowAccounts = await ethereumWalletClient.requestAddresses();
  console.log('Connected', allowAccounts[0]);
} catch (error) {
  alert(error);
  return;
}
```



# 使用 WalletConnect 连接

WalletConnect 通过使用二维码 (QR Code) 技术来实现扫描登录



```
import { EthereumProvider } from "@walletconnect/ethereum-provider";

const handle = async () => {
  const provider = await EthereumProvider.init({
    projectId: "e4c87e9143dd57a65fc4749cbcd1c88",
    metadata: {
      name: "My Website",
      description: "My Website Description",
      url: "https://mywebsite.com", // origin must match your domain & subdomain
      icons: ["https://avatars.githubusercontent.com/u/37784886"],
    },
    showQrModal: true,
    optionalChains: [1, 10],
  });

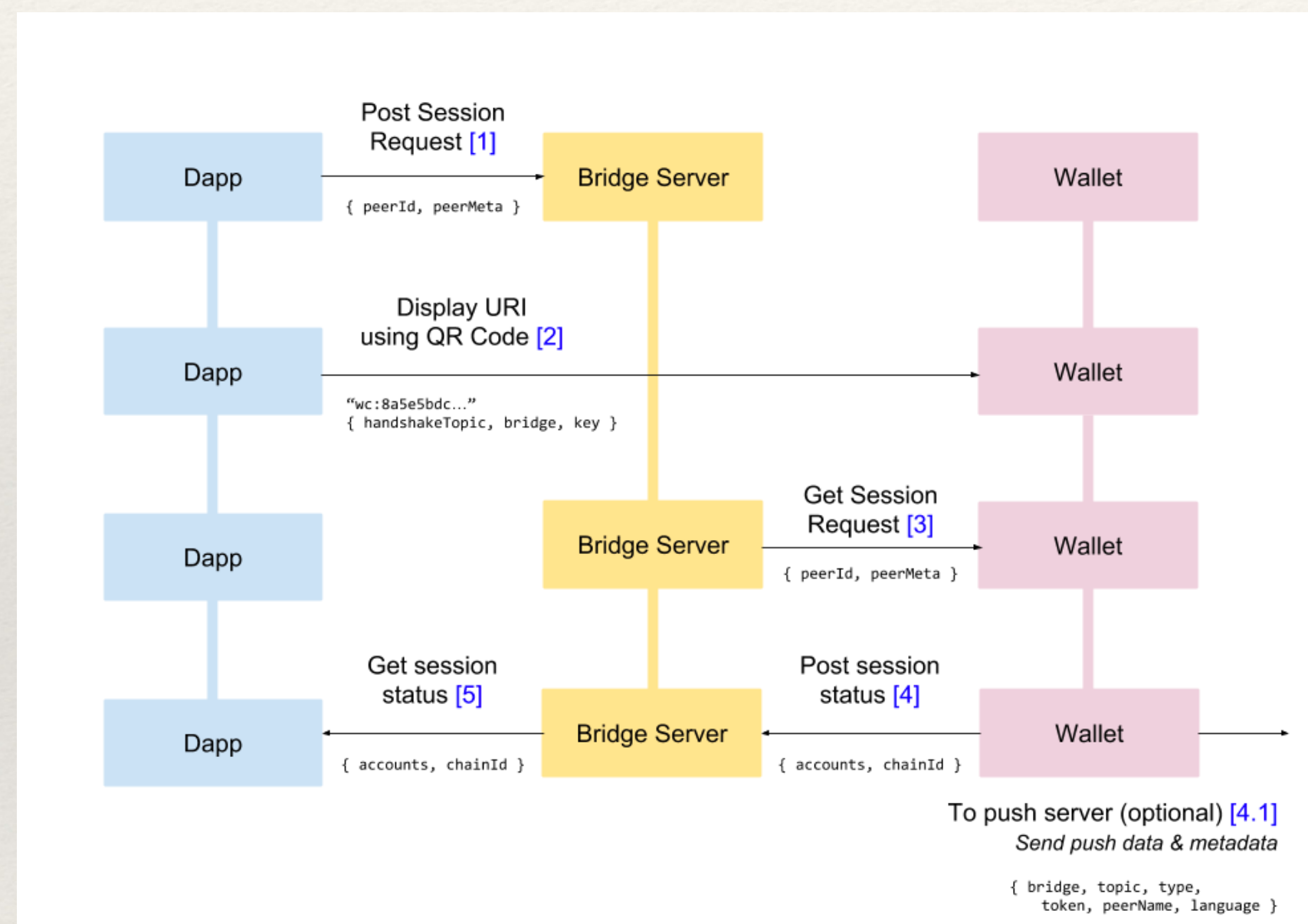
  await provider.connect();
  const result = await provider.request<string[]>({
    method: "eth_requestAccounts",
  });
  console.log("Connected:", result);
  setAccounts(result);
};
```

<https://walletconnect.com/>



# WalletConnect 原理

利用 Pairing API 实现通信： WalletConnect 通过使用二维码（QR Code）技术来实现扫描登录



1. DApp 生成 WalletConnect 会话
2. DApp 显示二维码
3. 用户扫描二维码
4. 钱包连接到会话
5. DApp 处理会话确认

<https://specs.walletconnect.com/2.0/specs/clients/core/pairing#successful-request-handling-if-b-registers-protocol-p>



# WalletConnect 原理

二维码信息

WalletConnect 协议的版本号



会话标识符

wc:0b5096dbb3be5ee0721ac2bc0c5adf5b4c767e260813c20ed0a093e7a25cce56@2?  
expiryTimestamp=1719841255&relay-  
protocol=irn&symKey=59c3a0b32ecde687fac8258c0c23595dc934c159ce299d72009f0f8969781640

会话过期时间戳

消息中继协议

用于配对加密的对称密钥



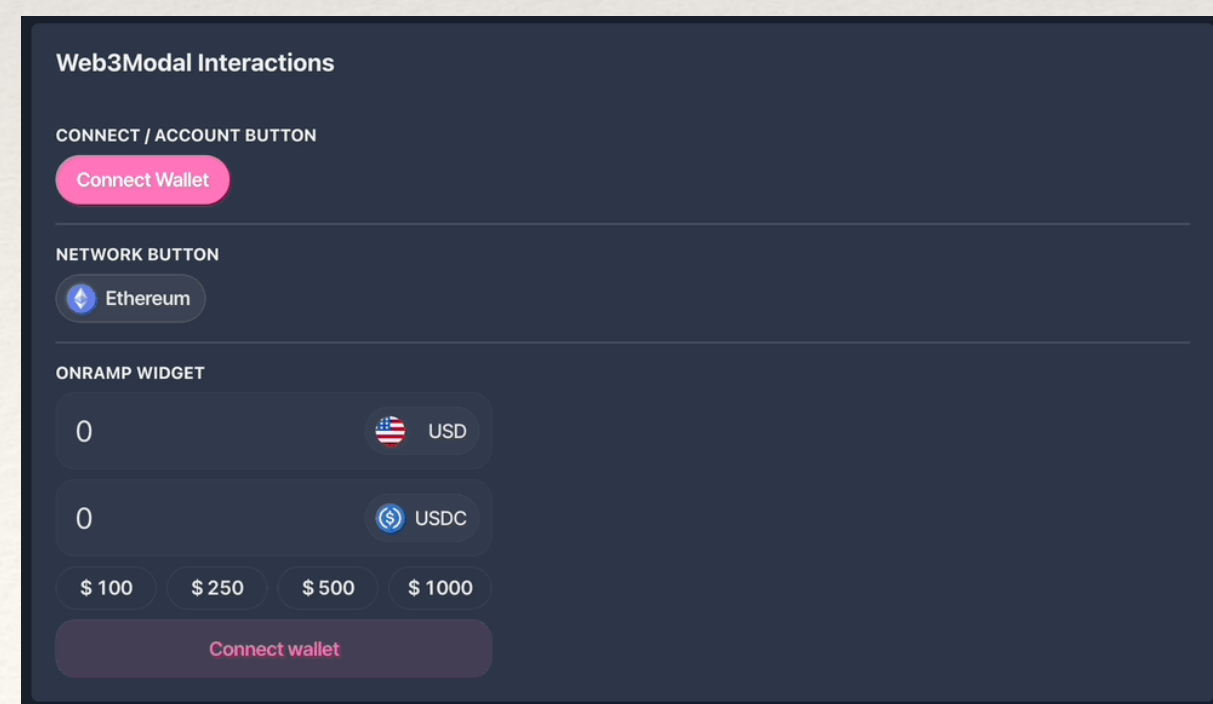
# 使用 AppKit(Web3modal) 按钮

## 登录加密钱包痛点

- 多钱包支持的复杂性
- 用户体验的一致性
- 开发和维护成本
- 钱包检测和管理

## Web3Modal 优势

- 统一的接口
- 增强的用户体验
- 简化集成
- 开箱即用的功能



Demo : <https://lab.web3modal.com/library/wagmi-email/>



# WebModal Connect

```
import { createWeb3Modal } from "@web3modal/wagmi/react";
import { defaultWagmiConfig } from "@web3modal/wagmi/react/config";
import { WagmiProvider } from "wagmi";
import { arbitrum, mainnet } from "wagmi/chains";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { ReactNode } from "react";

// 0. Setup queryClient
const queryClient = new QueryClient();

// 1. Get projectId from https://cloud.walletconnect.com
const projectId = import.meta.env.VITE_PROJECT_ID;
if (!projectId) {
  throw new Error("VITE_PROJECT_ID is not set");
}

// 2. Create wagmiConfig
const metadata = {
  name: "Web3Modal",
  description: "Web3Modal Example",
  url: "https://web3modal.com", // origin must match your domain & subdomain
  icons: ["https://avatars.githubusercontent.com/u/37784886"],
};

const chains = [mainnet, arbitrum] as const;
const config = defaultWagmiConfig({
  chains,
  projectId,
  metadata,
});
```

```
// 3. Create modal
createWeb3Modal({
  wagmiConfig: config,
  projectId,
});

function Web3ModalProvider({ children }: { children: ReactNode }) {
  return (
    <WagmiProvider config={config}>
      <QueryClientProvider client={queryClient}>{children}</QueryClientProvider>
    </WagmiProvider>
  );
}

export default function LoginWithWeb3Modal() {
  return (
    <>
      <Web3ModalProvider>
        <w3m-button />
      </Web3ModalProvider>
    </>
  );
}
```



# 作业说明

- ❖ 代码在自己的 github 提交
- ❖ 在 decert.me 提交领取证书
- ❖ **不可抄袭作业**，一经发现将不再检查抄袭者作业！



# 作业

► 使用 <https://vercel.com/> 部署项目，实现 Web3Modal连接到MetaMask，提交部署的网站链接和 Github 分支链接。

<https://decert.me/quests/aebe24be-0bec-4c6c-bef1-22eb08817621>

<https://docs.walletconnect.com/appkit/javascript/core/installation>



谢谢