Viem Quick Reference

BigInt Tip: Use n (e.g., 5n) for all math. Regular Numbers will throw errors when used with Wei values.

1. Conversions & Formatting
parseEther('1.5') → 1500...0n (Wei)
formatEther(100n) → '0.000...1' (ETH)
parseGwei('20') → 20000000000n
formatGwei(20000000000n) → '20'
parseUnits('10', 6) → 10000000n
formatUnits(10n, 6) → '0.00001'
stringify(obj) → Returns a JSON string

2. Utilities & Validation
getAddress('0x...') → Returns checksummed address.
isAddress('0x...') → Returns true / false.
toHex(100) → '0x64'
fromHex('0x64', 'number') → 100
size('0x1234') → 2 (bytes)

3. Setup (Clients & Transports)
createPublicClient({ chain, transport: http() }) → Primary read interface.

```
import { createPublicClient, http } from 'viem';
import { mainnet } from 'viem/chains';

// 2. Set up your client with desired chain & transport.
const client = createPublicClient({
  chain: mainnet,
  transport: http(),
});
```

createWalletClient({ chain, transport: custom(window.ethereum) }) → Primary write interface (MetaMask). Can also be used with a local account (see code below).

```
import { createWalletClient, http, parseEther } from 'viem';
import { privateKeyToAccount, mnemonicToAccount } from 'viem/accounts';
import { mainnet } from 'viem/chains';

const client = createWalletClient({
  chain: mainnet,
  transport: http()
});

const account = privateKeyToAccount('0x...');
// const account = mnemonicToAccount('legal winner thank year wave sausage...');
```

```
const hash = await client.sendTransaction({
  account,
  to: '0xa5cc3c03994DB5b0d9A5eEdD10CabaB0813678AC',
  value: parseEther('0.001')
});
```

We can also register new chains, like Next Testnet.

```
import { defineChain } from 'viem';

export const nextChain = defineChain({
  id: 1337,
  name: 'Next',
  nativeCurrency: {
    name: 'Ether',
    symbol: 'ETH',
    decimals: 18
  },
  rpcUrls: {
    default: {
      http: ['https://eth.code-camp.org']
    }
  }
});
```

4. Reading Data (Public Client)
client.getBlockNumber() → Current height.
client.getBalance({ address }) → Native balance.
client.getTransaction({ hash }) → Details of a specific tx.
client.getTransactionReceipt({ hash }) → Status, gas used, and logs.
client.readContract({ abi, address, functionName, args }) → Call a view. (You can import erc20Abi from viem)

5. Writing & Signing (Wallet Client)
client.sendTransaction({ to, value, account }) → Transfer ETH.
client.writeContract({ abi, address, functionName, account, args }) → State-changing call.

6. Simulation & Observation
client.simulateContract({ address, abi, functionName, account }) → Dry-run a tx to prevent reverts.
client.estimateGas({ account, to, value }) → Get gas units needed.
client.estimateContractGas({ address, abi, functionName, account, args }) →Get gas needed for a contract write function.
client.waitForTransactionReceipt({ hash }) → Poll until tx is mined.