Ethers.js Cheatsheet

```
import { ethers } from 'ethers';


const connectWallet = async ()=>{


  if(window.ethereum){
     const provider = new ethers.providers.Web3Provider(window.ethereum)


     // MetaMask requires requesting permission to connect users accounts
     await provider.send("eth_requestAccounts", []);
  }

}
```

Connect RPC javascript

```
import { ethers } from 'ethers';


// JSON RPC provider for the network you want to connect
const rpc = 'https://rinkeby.infura.io/v3/9aa3d95b3bc440fa88ea12eaa4456161';


const provider = new ethers.providers.JsonRpcProvider(rpc);
```

Query blockchain javascript

```
import { ethers } from 'ethers';


const provider = new ethers.providers.JsonRpcProvider(rpc);
// Look up the current block number
await provider.getBlockNumber()
// 14681280


// Get the balance of an account (by address or ENS name, if supported by network)
balance = await provider.getBalance("ethers.eth")
// { BigNumber: "182826475815887608" }


// Often you need to format the output to something more user-friendly,
// such as in ether (instead of wei)
```

```
ethers.utils.formatEther(balance)
// '0.182826475815887608'


// If a user enters a string in an input field, you may need
// to convert it from ether (as a string) to wei (as a BigNumber)
ethers.utils.parseEther("1.0")
// { BigNumber: "1000000000000000000" }
```

Historic Events javascript

```
// Get the address of the Signer
myAddress = await signer.getAddress()
// '0x8ba1f109551bD432803012645Ac136ddd64DBA72'


// Filter for all token transfers from me
filterFrom = daiContract.filters.Transfer(myAddress, null);


// Filter for all token transfers to me
filterTo = daiContract.filters.Transfer(null, myAddress);


// List all transfers sent from me in a specific block range
await daiContract.queryFilter(filterFrom, 9843470, 9843480)


// List all transfers sent in the last 10,000 blocks
await daiContract.queryFilter(filterFrom, -10000)


// List all transfers ever sent to me
await daiContract.queryFilter(filterTo)
```

Send Transaction javascript

```
// Send 1 ether to an ens name.
const tx = signer.sendTransaction({
  to: "ricmoo.firefly.eth",
  value: ethers.utils.parseEther("1.0")
});
```

Sign Message javascript

```
// To sign a simple string, which are used for
// logging into a service, such as CryptoKitties,
// pass the string in.
signature = await signer.signMessage("Hello World");
```

```
// A common case is also signing a hash, which is 32
// bytes. It is important to note, that to sign binary
// data it MUST be an Array (or TypedArray)


// This string is 66 characters long
message = "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef"


// This array representation is 32 bytes long
messageBytes = ethers.utils.arrayify(message);
// Uint8Array [ 221, 242, 82, 173, 27, 226, 200, 155, 105, 194, 176, 104, 252, 55, 141, 170, 149,
43, 167, 241, 99, 196, 161, 22, 40, 245, 90, 77, 245, 35, 179, 239 ]


// To sign a hash, you most often want to sign the bytes
signature = await signer.signMessage(messageBytes)
```

Contract Actions

Signed ContractWrite to ContractCall payableListen to eventsProviderContractQuery Contract Configure Contract javascript

```
// You can also use an ENS name for the contract address
const daiAddress = "dai.tokens.ethers.eth";


// The ERC-20 Contract ABI, which is a common contract interface
// for tokens (this is the Human-Readable ABI format)
const daiAbi = [
 // Some details about the token
 "function name() view returns (string)",
 "function symbol() view returns (string)",


 // Get the account balance
 "function balanceOf(address) view returns (uint)",


 // Send some of your tokens to someone else
 "function transfer(address to, uint amount)",


 // An event triggered whenever anyone transfers to someone else
```

```
 "event Transfer(address indexed from, address indexed to, uint amount)"
];


// The Contract object
const daiContract = new ethers.Contract(daiAddress, daiAbi, provider);
Query Contract javascript
// Get the ERC-20 token name
await daiContract.name()
// 'Dai Stablecoin'


// Get the ERC-20 token symbol (for tickers and UIs)
await daiContract.symbol()
// 'DAI'


// Get the balance of an address
balance = await daiContract.balanceOf("ricmoo.firefly.eth")
// { BigNumber: "3519207045588426820163 1" }


// Format the DAI for displaying to the user
ethers.utils.formatUnits(balance, 18)
// '35192.070455884268201631'
Listning to events javascript
// Receive an event when ANY transfer occurs
daiContract.on("Transfer", (from, to, amount, event) => {
  console.log(`${ from } sent ${ formatEther(amount) } to ${ to}`);
  // The event object contains the verbatim log data, the
  // EventFragment and functions to fetch the block,
  // transaction and receipt and event functions
});


// A filter for when a specific address receives tokens
myAddress = "0x8ba1f109551bD432803012645Ac136ddd64DBA72";
filter = daiContract.filters.Transfer(null, myAddress)


// Receive an event when that filter occurs
daiContract.on(filter, (from, to, amount, event) => {
  // The to will always be "address"
  console.log(`I got ${ formatEther(amount) } from ${ from }.`);
```

```
});
Write to contract javascript
const signedContract = contract.connect(signer);


const tx = await signedContract.addComment(postId, commentString);
// wait for the transaction to be finished
await tx.wait();
Call payable method javascript
const signedContract = contract.connect(signer);


const options = { value: ethers.utils.parseEther('0.002') };
const tx = await signedContract.addPaidComment(postId, commentString, options);
await tx.wait();
```