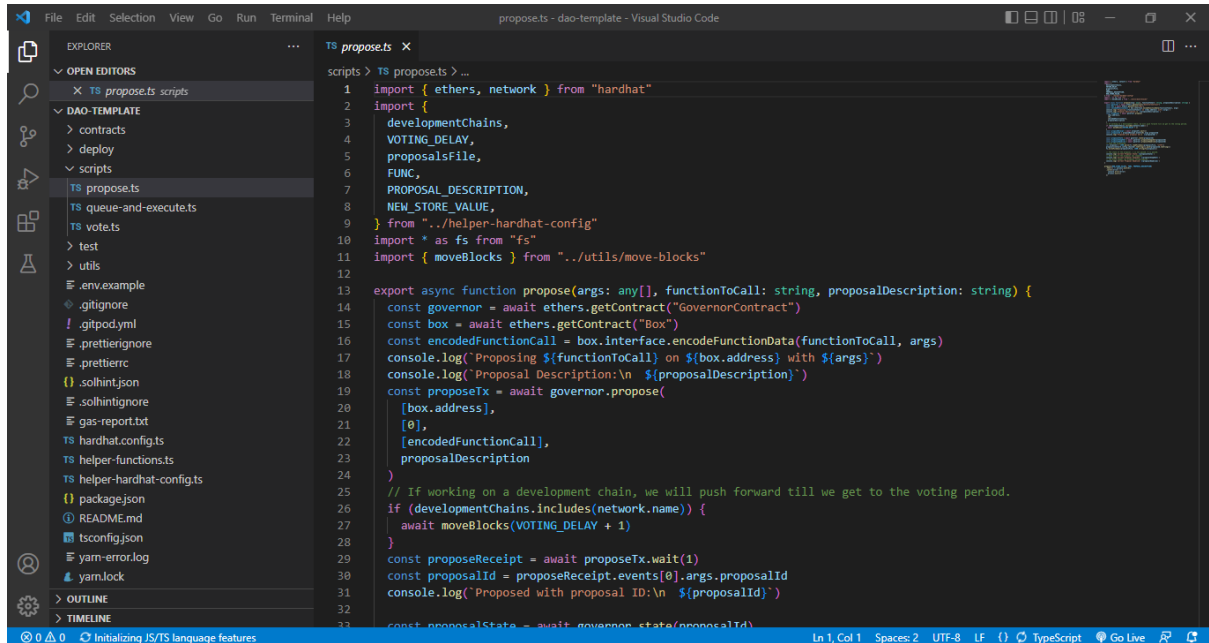


# Lesson 17

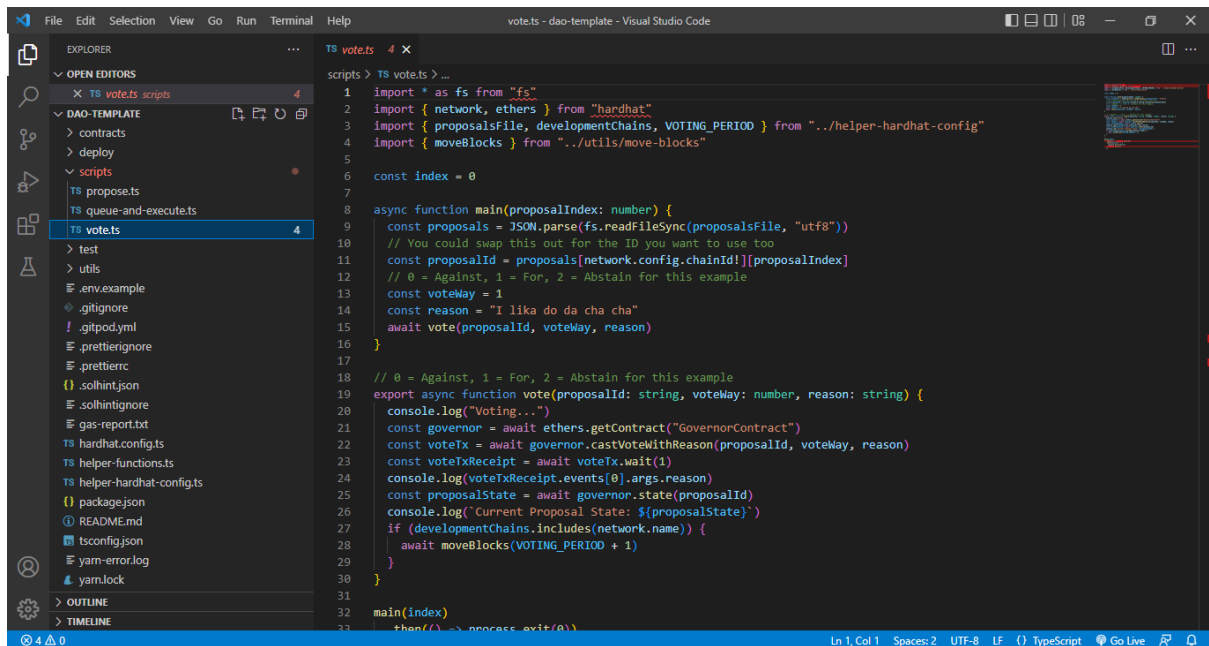
## Propose.ts



The screenshot shows the Visual Studio Code editor with a project named 'dao-template'. The Explorer sidebar on the left shows the file structure, with 'scripts' > 'propose.ts' selected. The main editor displays the content of 'propose.ts'.

```
1 import { ethers, network } from "hardhat"
2 import {
3   developmentChains,
4   VOTING_DELAY,
5   proposalsFile,
6   FUNC,
7   PROPOSAL_DESCRIPTION,
8   NEW_STORE_VALUE,
9 } from "../helper-hardhat-config"
10 import * as fs from "fs"
11 import { moveBlocks } from "../utils/move-blocks"
12
13 export async function propose(args: any[], functionToCall: string, proposalDescription: string) {
14   const governor = await ethers.getContract("GovernorContract")
15   const box = await ethers.getContract("Box")
16   const encodedFunctionCall = box.interface.encodeFunctionData(functionToCall, args)
17   console.log("Proposing ${functionToCall} on ${box.address} with ${args}")
18   console.log("Proposal Description:\n ${proposalDescription}")
19   const proposeTx = await governor.propose(
20     [box.address],
21     [0],
22     [encodedFunctionCall],
23     proposalDescription
24   )
25   // If working on a development chain, we will push forward till we get to the voting period.
26   if (developmentChains.includes(network.name)) {
27     await moveBlocks(VOTING_DELAY + 1)
28   }
29   const proposeReceipt = await proposeTx.wait(1)
30   const proposalId = proposeReceipt.events[0].args.proposalId
31   console.log("Proposed with proposal ID:\n ${proposalId}")
32
33   const proposalState = await governor.state(proposalId)
```

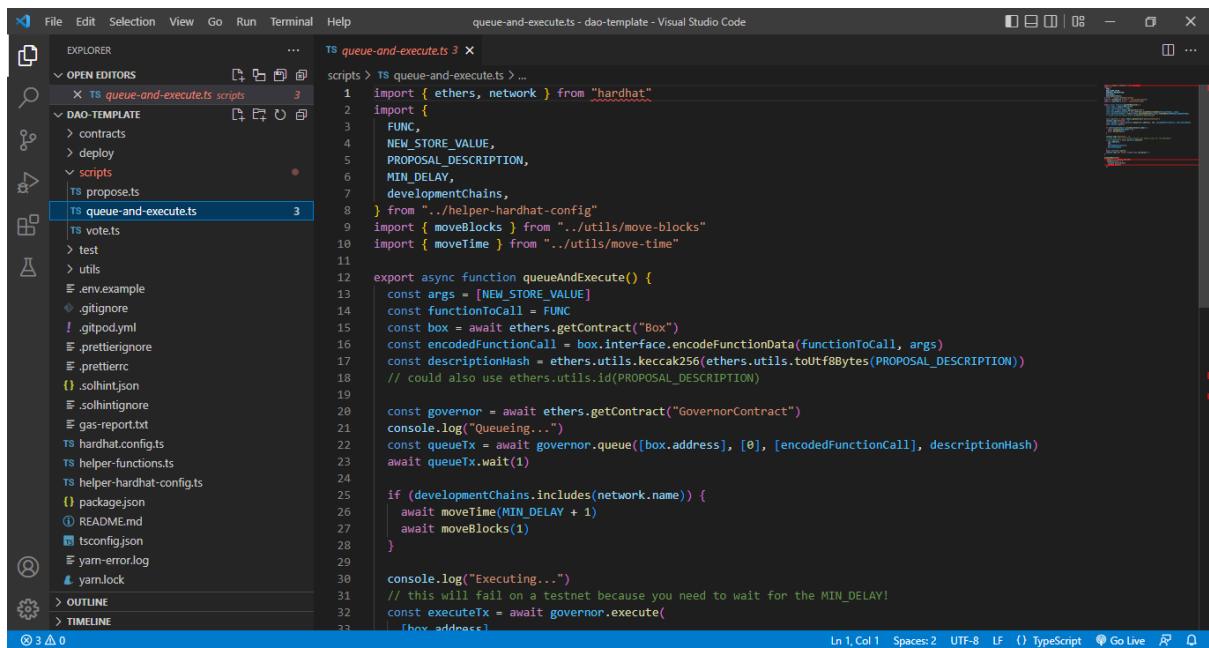
## Vote.ts



The screenshot shows the Visual Studio Code editor with a project named 'dao-template'. The Explorer sidebar on the left shows the file structure, with 'scripts' > 'vote.ts' selected. The main editor displays the content of 'vote.ts'.

```
1 import * as fs from "fs"
2 import { network, ethers } from "hardhat"
3 import { proposalsFile, developmentChains, VOTING_PERIOD } from "../helper-hardhat-config"
4 import { moveBlocks } from "../utils/move-blocks"
5
6 const index = 0
7
8 async function main(proposalIndex: number) {
9   const proposals = JSON.parse(fs.readFileSync(proposalsFile, "utf8"))
10   // You could swap this out for the ID you want to use too
11   const proposalId = proposals[network.config.chainId!][proposalIndex]
12   // 0 = Against, 1 = For, 2 = Abstain for this example
13   const voteWay = 1
14   const reason = "I lika do da cha cha"
15   await vote(proposalId, voteWay, reason)
16 }
17
18 // 0 = Against, 1 = For, 2 = Abstain for this example
19 export async function vote(proposalId: string, voteWay: number, reason: string) {
20   console.log("Voting...")
21   const governor = await ethers.getContract("GovernorContract")
22   const voteTx = await governor.castVoteWithReason(proposalId, voteWay, reason)
23   const voteTxReceipt = await voteTx.wait(1)
24   console.log(voteTxReceipt.events[0].args.reason)
25   const proposalState = await governor.state(proposalId)
26   console.log("Current Proposal State: ${proposalState}")
27   if (developmentChains.includes(network.name)) {
28     await moveBlocks(VOTING_PERIOD + 1)
29   }
30 }
31
32 main(index)
33   .then(() => process.exit(0))
```

## Queue-and-execute.ts



```
1 import { ethers, network } from "hardhat"
2 import {
3   FUNC,
4   NEW_STORE_VALUE,
5   PROPOSAL_DESCRIPTION,
6   MIN_DELAY,
7   developmentChains,
8 } from "../helper-hardhat-config"
9 import { moveBlocks } from "../utils/move-blocks"
10 import { moveTime } from "../utils/move-time"
11
12 export async function queueAndExecute() {
13   const args = [NEW_STORE_VALUE]
14   const functionToCall = FUNC
15   const box = await ethers.getContract("Box")
16   const encodedFunctionCall = box.interface.encodeFunctionData(functionToCall, args)
17   const descriptionHash = ethers.utils.keccak256(ethers.utils.toUtf8Bytes(PROPOSAL_DESCRIPTION))
18   // could also use ethers.utils.id(PROPOSAL_DESCRIPTION)
19
20   const governor = await ethers.getContract("GovernorContract")
21   console.log("Queueing...")
22   const queueTx = await governor.queue([box.address], [0], [encodedFunctionCall], descriptionHash)
23   await queueTx.wait(1)
24
25   if (developmentChains.includes(network.name)) {
26     await moveTime(MIN_DELAY + 1)
27     await moveBlocks(1)
28   }
29
30   console.log("Executing...")
31   // this will fail on a testnet because you need to wait for the MIN_DELAY!
32   const executeTx = await governor.execute(
33     [box.address]
```