Hanoi Tower Activity

#### **Learning Outcome Addressed**

 4. Write recursive functions

Now you can practice solving the Hanoi Tower problem in Next Tech. Click the link below to launch the activity in a new window and follow the instructions.

**Recursion To Solve The Hanoi Tower Problem**

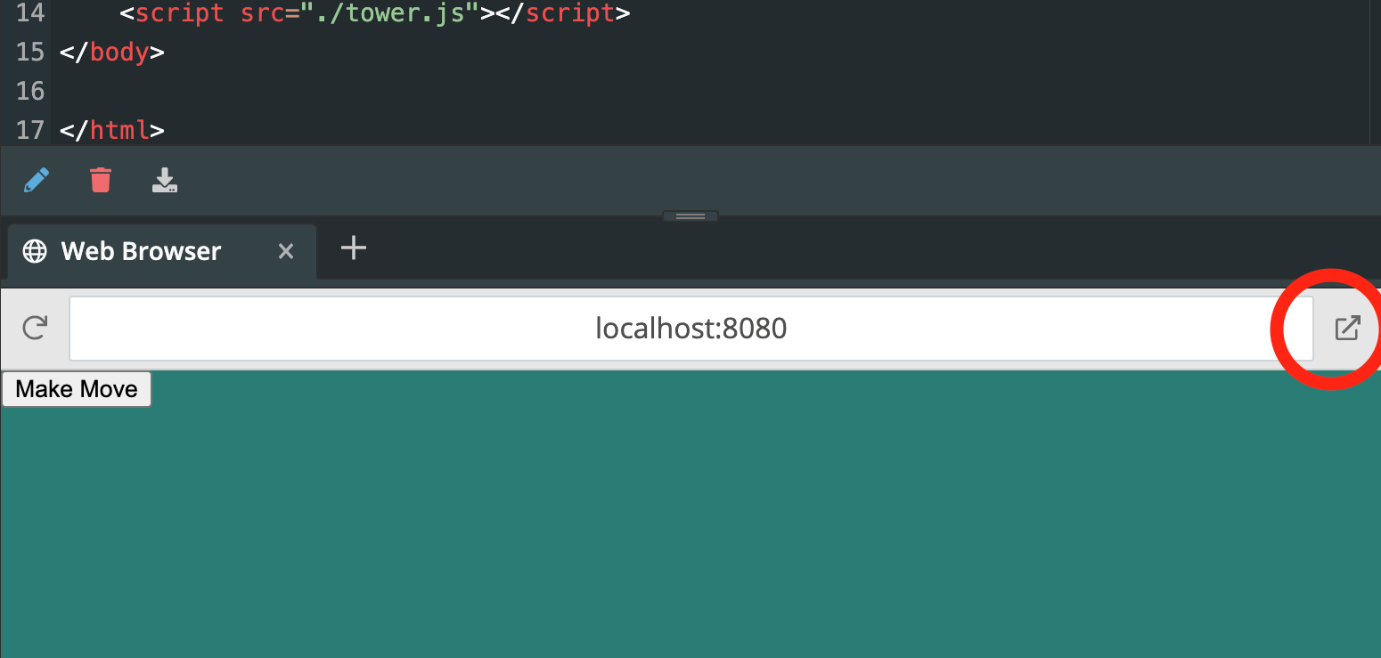
### Hanoi Tower

Now that you're familiar with the Hanoi Tower problem, you can explore how to customize its algorithm to add more disks.

The starter code for this activity includes the solution to the Hanoi Tower problem. Currently, this solution works for five disks only.

You can use the application by pressing the Make Move button. The first press will load the Hanoi Tower. Every button press after that iterates through solving the Hanoi Tower. When the Hanoi Tower is solved, on the last button press, you will receive an alert stating Tower is Finished.

Note: The integrated web browser is too small on most screen sizes to display the entire Hanoi Tower application. Use the "Open in new browser tab" button. See the screenshot below.

[](https://cdn.filestackcontent.com/QjOq0hp6RNmqGTFhlcOq)

**Your task in this activity is to edit this solution to add a custom number of disks to the tower.**

You can accomplish this task by doing the following:

* Add an input box with the ID: numDisk under the make move button in the index.html file. This input box will contain the custom number of disks.
* Use the input from that box to adjust the Hanoi Tower algorithm to account for all the disks.

Hint: In the *tower.js* file, find the *makeMove* function. Then think about how to retrieve the value that the user typed in the *<input>*. Finally, consider where in the function to set the *nDisks* variable to the user's input value.

Task

Edit the hanoi tower algorithm to add a custom number of disks to the tower

const writeAll = (pegMap) => {

  let s = JSON.stringify(pegMap);

  // write output

  console.log(s);

  pegHist.push(JSON.parse(s));

};

const pegHist = [];

const pegMap = {

  A: [],

  B: [],

  C: [],

};

const pegArray = ['A', 'B', 'C'];

const updateMapAndPlot = (pegMap, from, to) => {

  try {

    let theDisk = pegMap[from].pop(); // this actually moves the disk

    pegMap[to].push(theDisk);

    writeAll(pegMap);

  } catch (e) {

    console.log('no disk in Map');

  }

};

// Pegs  A   B   C

// disks are labeled 0 to N

let moves = 0;

const moveDisks = function (n, from, to, spare) {

  if (n === 1) {

    updateMapAndPlot(pegMap, from, to);

    moves++;

  } else {

    moveDisks(n - 1, from, spare, to);

    updateMapAndPlot(pegMap, from, to);

    moves++;

    moveDisks(n - 1, spare, to, from);

  }

};

// given from and to figure out the other peg

const missing = (from, to) => {

  const all = {

    A: 1,

    B: 2,

    C: 3,

  };

  let total = all[from] + all[to];

  let miss = 5 - total; // 0 based so not 6 but one less

  let keys = Object.keys(all); // array of keys

  return keys[miss];

};

const initialize = (nDisks, pegMap) => {

  for (let i = nDisks; i >= 1; i--) {

    pegMap['A'].push(i); // this stacks disks with smallest on top

  }

  writeAll(pegMap);

};

<html>

  <head>

    <link rel="stylesheet" href="./styles.css" />

    <script src="./hanoi.js"></script>

  </head>

  <body>

    <button onclick="makeMove()" id ="makemove" width="40">Make Move</button>

    <div class="game" id="game"></div>

    <script src="./tower.js"></script>

  </body>

</html>

body {

  margin: 0;

  padding: 0;

  background: hsla(175, 87%, 24%, 0.92);

}

.game {

  position: absolute;

  width: 100%;

  height: 100%;

  background: rgba(175, 85%, 10%, 0.92);

  margin: 0 25%;

}

.peg {

  width: 30px;

  height: 300px;

  background: rgba(236, 231, 231, 0.509);

  position: absolute;

  margin: 40px;

  border-radius: 10%;

}

.disk {

  width: 80px;

  height: 200px;

  background: rgb(211, 35, 35);

  margin: 10px;

  border-radius: 5%;

  position: absolute;

}

let counter = 0;

let pegs = [];

let diskArray = []; // this stores the disks

const game = document.getElementById('game');

function plotAll(pegMap) {

  pegArray.map((peg, pegindex) => {

    // loop over all pegs

    let disks = pegMap[peg]; // the array of disks on pegMap.A for example

    if (disks.length > 0) {

      disks.map((disk, index) => {

        pickDisk = diskArray.filter((item) => {

          // pick out correct disk from diskArray

          return item.id == disk;

        });

        positionDisk(pickDisk[0].newdiv, disk, index, pegindex); // disk is the disk number where larger is wider disk

      });

    }

  });

}

function positionDisk(domdiv, diskNumber, indexOnPeg, pegindex) {

  // set its position

  let pegCenter = 300 \* pegindex + 100;

  let diskWidth = diskNumber \* 40 + 20;

  let base = 400;

  let diskHeight = 20;

  domdiv.style.left = pegCenter - diskWidth / 2;

  domdiv.style.width = diskWidth;

  domdiv.style.top = 400 - diskHeight \* indexOnPeg;

  console.log('Disk:' + diskNumber + ' at top: ' + domdiv.style.top);

  domdiv.style.height = diskHeight;

  domdiv.innerHTML = '  ' + diskNumber;

}

const sleep = (milliseconds) => {

  return new Promise((resolve) => setTimeout(resolve, milliseconds));

};

function diskFactory(id, indexOnPeg, pegNumber, type) {

  // this determines start positions

  let newdiv = document.createElement('div');

  positionDisk(newdiv, id, indexOnPeg, pegNumber);

  newdiv.setAttribute('class', type); // these are the pegs

  newdiv.setAttribute('id', id);

  game.appendChild(newdiv);

  return {

    id,

    newdiv,

  };

}

function pegFactory(id, indexOnPeg, pegNumber, type) {

  // this determines start positions

  let newdiv = document.createElement('div');

  newdiv.setAttribute('class', 'peg'); // these are the pegs

  newdiv.setAttribute('id', id);

  let pegCenter = 300 \* pegNumber + 60;

  let base = 400;

  let diskHeight = 200;

  newdiv.style.left = pegCenter;

  newdiv.style.width = 20;

  newdiv.style.top = 200;

  newdiv.style.height = 200;

  game.appendChild(newdiv);

}

function initializeDisks(pegMap) {

  let disks = pegMap[pegArray[0]]; // all disks start on peg 0

  diskArray = disks.map((diskid, indexOnPeg) => {

    // this is a kind of shadow DOM

    return diskFactory(diskid, indexOnPeg, 0, 'disk');

  });

  pegArray.map((item, pegId) => {

    pegFactory(pegId, 0, pegId, 'peg');

  });

}

function makeMove() {

  if (counter == 0) {

    const nDisks = 5;

    initialize(nDisks, pegMap);

    moveDisks(nDisks, 'A', 'C', 'B');

    initializeDisks(pegHist[0]);

  }

  if (counter < pegHist.length) {

    plotAll(pegHist[counter]);

  } else {

    alert('Tower is Finished');

  }

  counter++;

}