**1. index.html Documentation**

**Overview:**

The index.html file serves as the entry point for the 3D model viewer web application. It defines the basic structure of the webpage, links to necessary stylesheets and scripts, and provides a canvas element where the 3D model will be rendered.

**Breakdown:**

* **DOCTYPE and HTML Structure**:
  + **<!DOCTYPE html>**: Declares that this is an HTML5 document.
  + **<html lang="en">**: The root element, setting the language of the content to English.
  + **<head>**: Contains metadata, links to stylesheets, and the title of the document.
  + **<body>**: Contains the main content of the page.
* **Head Section**:
  + **<meta charset="UTF-8">**: Sets the character encoding for the document.
  + **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Ensures the webpage is responsive and scales appropriately on different devices.
  + **<title>**: Sets the title displayed in the browser tab.
  + **<link rel="stylesheet" href="style.css">**: Links to the style.css file, which contains the styles for the webpage.
* **Body Section**:
  + **<div id="info">3D Model Viewer</div>**: A div element that displays the title or any other information at the top of the webpage.
  + **<canvas id="modelCanvas"></canvas>**: The canvas element where the 3D scene is rendered.
* **Scripts**:
  + **<script src="main.js"></script>**: Links to the main.js file, which contains the JavaScript code that powers the 3D model viewer.

**2. style.css Documentation**

**Overview:**

The style.css file is responsible for the styling of the webpage. It defines the appearance of the HTML elements, including the layout, colors, fonts, and overall design.

**Breakdown:**

* **Global Styles**:
  + **body, html { margin: 0; padding: 0; overflow: hidden; font-family: Arial, sans-serif; }**:
    - **Purpose**: Removes default margin and padding, hides overflow to prevent scrollbars, and sets a default font for the entire page.
* **Canvas Styling**:
  + **canvas { display: block; }**:
    - **Purpose**: Ensures the canvas element is treated as a block-level element, taking up the full width of the parent element (usually the viewport).
* **Info Box Styling**:
  + **#info { position: absolute; top: 10px; left: 10px; color: white; background-color: rgba(0, 0, 0, 0.5); padding: 10px; border-radius: 5px; }**:
    - **Purpose**: Styles the information box that displays the title "3D Model Viewer" at the top-left of the screen. It has a semi-transparent background and rounded corners.

**3. main.js Documentation**

**Overview:**

The main.js file contains the core logic for the 3D model viewer. It sets up the 3D scene using Three.js, handles user interactions, and renders the 3D model in the browser.

**Breakdown:**

* **Initialization**:
  + **const scene = new THREE.Scene();**: Creates a new Three.js scene where all objects will be added.
  + **const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);**:
    - **Purpose**: Creates a perspective camera, simulating the way human eyes perceive the world, with a field of view of 75 degrees.
  + **const renderer = new THREE.WebGLRenderer({ canvas: document.getElementById('modelCanvas') });**:
    - **Purpose**: Sets up the WebGL renderer, which will render the 3D scene to the canvas element.
* **Lighting Setup**:
  + **const light = new THREE.DirectionalLight(0xffffff, 1);**:
    - **Purpose**: Adds a directional light to the scene, simulating light from a distant source like the sun.
  + **light.position.set(1, 1, 1).normalize();**: Positions and normalizes the light source.
* **3D Model Setup**:
  + **const geometry = new THREE.BoxGeometry();**:
    - **Purpose**: Defines the shape of a 3D object—in this case, a simple cube.
  + **const material = new THREE.MeshStandardMaterial({ color: 0x0077ff });**:
    - **Purpose**: Defines the material and appearance of the object, using a standard material with a blue color.
  + **const cube = new THREE.Mesh(geometry, material);**:
    - **Purpose**: Combines the geometry and material into a mesh, which is then added to the scene.
* **Camera and Controls**:
  + **camera.position.z = 5;**: Positions the camera so that it is 5 units away from the center of the scene.
  + **const controls = new THREE.OrbitControls(camera, renderer.domElement);**:
    - **Purpose**: Adds orbit controls, allowing the user to interact with the scene using the mouse (e.g., rotating the camera).
* **Animation Loop**:
  + **function animate() {**
    - **Purpose**: Defines a loop that continuously updates the scene.
  + **requestAnimationFrame(animate);**: Ensures the animate function is called repeatedly to create smooth animations.
  + **controls.update();**: Updates the camera controls on each frame.
  + **renderer.render(scene, camera);**: Renders the scene from the perspective of the camera.
* **Window Resizing**:
  + **window.addEventListener('resize', () => {**
    - **Purpose**: Adjusts the camera's aspect ratio and the renderer's size whenever the window is resized to maintain the correct display proportions.