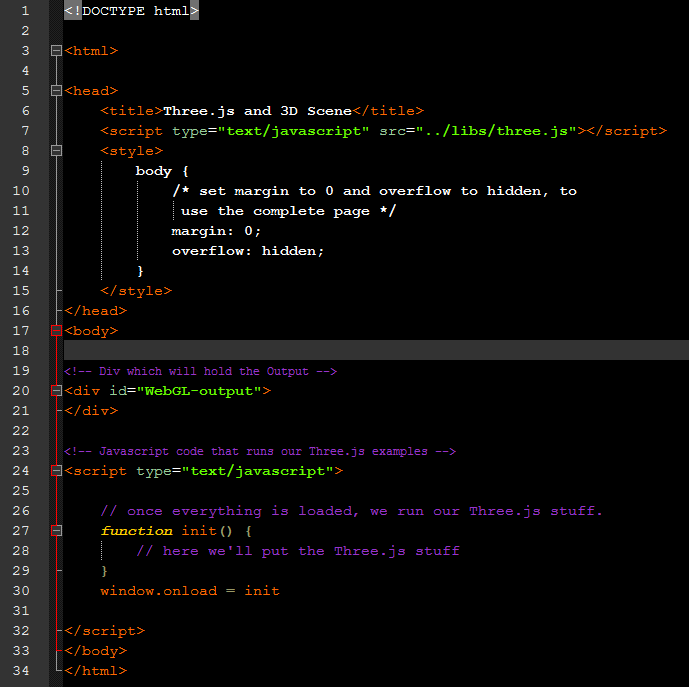
**Basic Steps in Three.js in Creating 3D Scene**

**Code-Template**

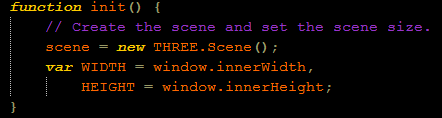
จากการบ้านก่อนหน้านี้ เราจะเริ่มต้นจาก code template ที่เป็นไฟล์ HTML ต่อไปนี้



**สร้างฉาก (Scene)**

Three.js ใช้แนวคิดของฉาก (scene) ในการกำหนดพื้นที่ที่คุณสามารถวางสิ่งต่างๆ เช่น รูปทรงเรขาคณิต, ไฟ, กล้องและอื่น ๆ

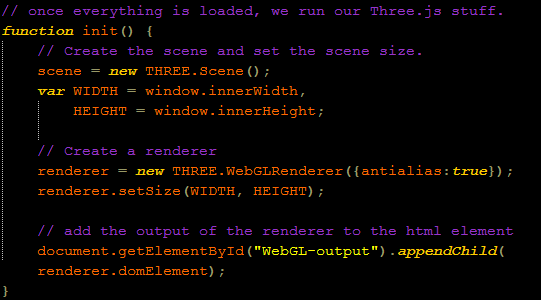
จาก code-template ก่อนหน้านี้ ฉากจะถูกสร้างในฟังก์ชัน init() นอกจากนี้ ในตัวอย่างนี้ เราจะทำการสร้างฉากที่มีขนาดเท่ากับความกว้างและความสูงของหน้าต่างของเบราว์เซอร์ โดยเราจะเก็บความกว้างและความสูงของหน้าต่างเบราว์เซอร์ในตัวแปร WIDTH และ HEIGHT ส่วนของโค้ดที่เพิ่มแสดงดังรูปต่อไปนี้



หลังจากแก้ไข หากนักศึกษาลองเปิดไฟล์ HTML ที่แก้ไขแล้ว จะพบว่าไม่มีอะไรแสดงบนหน้าเบราว์เซอร์เลย

**ทำการ Render ภาพ**

ต่อไปเราจะตั้งค่า renderer ของ Three.js เพื่อทำการ render ภาพ เราสามารถใช้ SVG หรือ canvas ในการ render ได้ แต่เราต้องการที่จะใช้ WebGL ในการ render เพราะมันสามารถใช้ประโยชน์ของ GPU ได้ ซึ่งทำให้การทำงานมีประสิทธิภาพ มากขึ้น หลังจากที่สร้าง renderer เราทำการเพิ่มส่วน renderer เข้าไปใน <div> element ID ว่า “WebGL-output” โดยใช้คำสั่งตามมาตรฐาน DOM ของ JavaScript DOM ซึ่งจะทำให้ Three.js สร้าง canvas ภายใน <div> element ดังกล่าว



**สร้างกล้อง (Camera)**

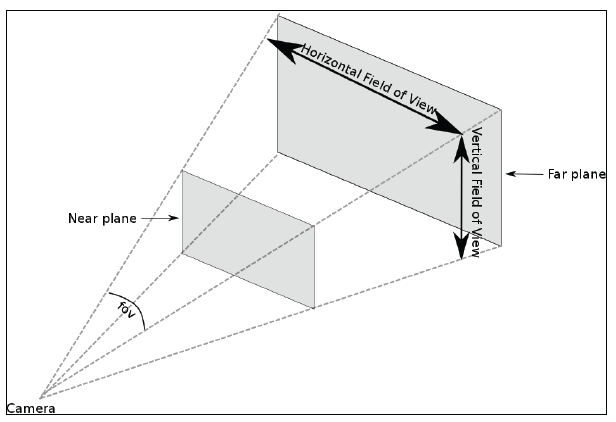
เมื่อเราทำการสร้าง “ฉาก (Scene)” และ “Renderer” แล้ว เราจะต้องการสร้าง “กล้อง (Camera)” โดยในตัวอย่างนี้ เราจะใช้กล้องแบบมุมมอง Perspective ซึ่งสามารถสร้างได้โดยการสร้าง object ของ Three.js ที่ชื่อ PerspectiveCamera โดยทำการ set ค่าพารามิเตอร์ที่เกี่ยวข้อง ดังนี้

FOV – ให้เป็น 45 องศา

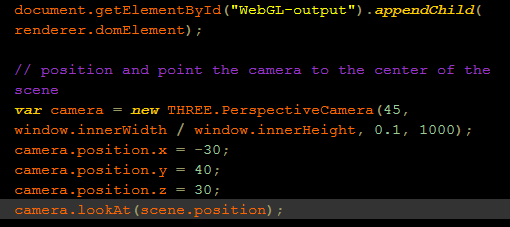
Aspect – เพื่อให้ง่าย เราทำการกำหนดค่านี้ โดยนำค่า WIDTH และ HEIGHT มาหารกัน

Near – ระยะทางจากกล้องที่เริ่มทำการ render ฉากและวัตถุ

Far – อะไนก็ตามที่อยู่เกินกว่าระยะนี้ จะไม่ถูกแสดงในฉาก



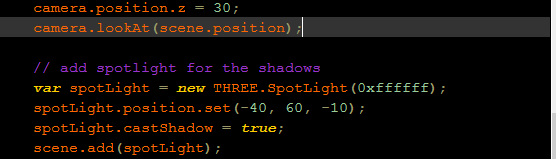
หลังจากสร้างกล้องแล้ว เราก็ทำการกำหนดตำแหน่งที่วางกล้อง โดยในเบื้องต้น สมมติว่า กล้องอยู่ที่ตำแหน่ง (x, y, z) = (-30, 40, 30)



**เพิ่มแสง (Light)**

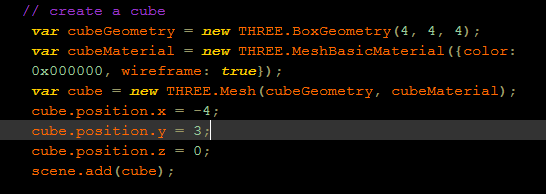
เราจะเริ่มทำการตกแต่งฉากของเรา โดยการเรียกฟังก์ชัน “setClearColorHex” ของ object “WebGLRendere” เราจะสามารถที่จะทำการกำหนดสีของ background ได้ สมมติว่าเราต้องการกำหนดสี background เป็นสี 0xEEEEE (สีเทา)

หลังจากนั้น เราจะเพิ่มแสงเข้าไปในฉาก เพื่อให้เห็นวัตถุเป็น 3D โดยเราจะเพิ่ม “แหล่งกำเนิดแสง” เข้าไปในฉากพร้อมทั้งระบุตำแหน่ง ในตัวอย่างนี้ เราจะใช้แหล่งกำเนิดแสงแบบ SpotLight ซึ่งในความเป็นแล้ว Three.js มีแสงหลายประเภทมากที่เราสามารถใช้งานได้



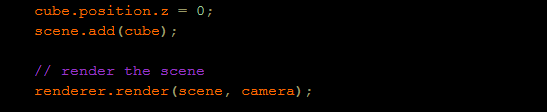
**เพิ่มวัตถุ (Object)**

ทั้งหมดที่ผ่าน เป็นการเตรียมความพร้อมของฉากที่จะใช้ในการวางวัตถุ ซึ่งเมื่อทุกอย่างพร้อมแล้ว เราก็ทำการเพิ่มวัตถุที่เราต้องการเข้าไปในฉาก พร้อมทั้งระบุตำแหน่งของมัน สมมติว่า เราจะทำการเพิ่มลูกบาศก์ (Cube) เข้าไปในฉากที่ตำแหน่ง (-4, 3, 0) รวมทั้ง เราต้องทำการระบุพื้นผิวของวัตถุว่าเป็นแบบไหน ในตัวอย่างนี้ เราจะใช้พื้นผิวพื้นฐานที่ Three.js ให้มา โดยเรียกฟังก์ชัน MeshBasicMaterial

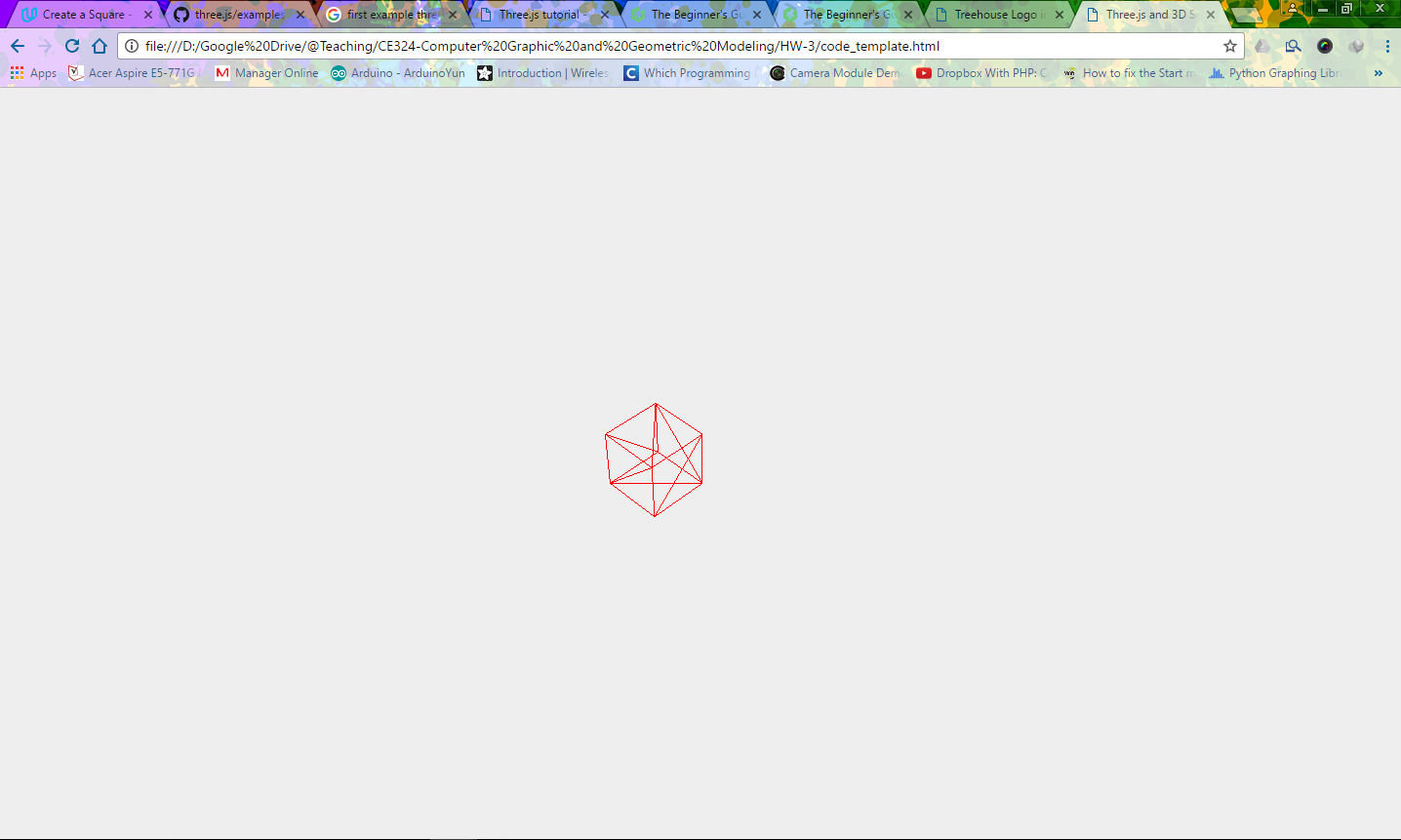


**Render ฉากทั้งหมด**

หลังจากที่เราเสร็จสิ้นการเตรียมการทั้งหมด เราจะทำการ render ฉาก 3 มิติที่เราสร้างไว้ โดยการเรียกฟังก์ชัน render ของวัตถุ renderer ที่เราได้สร้างก่อนหน้านี้แล้ว ดังตัวอย่าง



**ผลลัพธ์ที่ได้จากการรัน**



Note:

1. กระบวนการทั้งหมดที่อธิบายมา เป็นขั้นตอนพื้นฐานเท่านั้นในการสร้างฉาก 3D ขึ้นมา
2. การสร้างฉาก 3D ที่อธิบายมา หากทำการ resize หน้าจอของเวบ Browser การแสดงผลอาจจะผิดพลาด จะต้องมีการเพิ่มขั้นตอนอีกนิดนึงในส่วนของการแก้ปัญหาการ resize ของขนาดหน้าจอ Browser

**ให้นักศึกษาทำ** (ทำเป็นกลุ่มๆละ ไม่เกิน 3 คน ถ้าเป็นไปได้ ให้ใช้กลุ่มเดิม)

1. เพิ่มทรงกลมลงในฉาก ที่พิกัดใดๆก็ได้ในฉาก (แต่ต้องปรากฏในฉาก)
2. เปลี่ยนพื้นผิวของลูกบาศ์กและทรงกลมให้เป็นแบบ MeshLambertMaterial ตามที่นักศึกษาต้องการ
3. เปลี่ยนสีฉากหลังให้เป็นสีอื่น
4. ห้ามใช้สีซ้ำกันในฉาก วัตถุ พื้นหลัง สำหรับนักศึกษาแต่ละกลุ่ม