Rapier

Rapier is available as the @dimforge/rapier2d and @dimforge/rapier3d NPM packages. You may add the following to your package.json:

- Example 2DExample 3D

```
"dependencies": {
  "@dimforge/rapier3d": "*", // Replace the * by the latest version number.
}
```

Because Rapier is actually a WebAssembly module, it has to be loaded asynchronously. The following shows a basic example with a dynamic rigid-body falling on the ground.

- Example 2DExample 3D

```
import('@dimforge/rapier3d').then(RAPIER => {
    // Use the RAPIER module here.
    let gravity = { x: 0.0, y: -9.81, z: 0.0 };
    let world = new RAPIER.World(gravity);

    // Create the ground
    let groundColliderDesc = RAPIER.ColliderDesc.cuboid(10.0, 0.1, 10.0);
    world.createCollider(groundColliderDesc);

    // Create a dynamic rigid-body.
    let rigidBodyDesc = RAPIER.RigidBodyDesc.dynamic()
        .setTranslation(0.0, 1.0, 0.0);
    let rigidBody = world.createRigidBody(rigidBodyDesc);

    // Create a cuboid collider attached to the dynamic rigidBody.
    let colliderDesc = RAPIER.ColliderDesc.cuboid(0.5, 0.5, 0.5);
    let collider = world.createCollider(colliderDesc, rigidBody);

    // Game loop. Replace by your own game loop system.
    let gameLoop = () => {
        // Step the simulation forward.
        world.step();

        // Get and print the rigid-body's position.
        let position = rigidBody.translation();
        console.log("Rigid-body position: ", position.x, position.y);

        setTimeout(gameLoop, 16);
    };

    gameLoop();
```

```
})
```

See the testbed3d/src/demos and testbed2d/src/demos folders for examples on how to initialize a Rapier physics world using these bindings.

Using Rapier without a bundler

If you are attempting to use Rapier without a bundler, or if you are using a bundler that doesn't support WASM files properly, the previous solution will be difficult to get working. The alternative is to use our compatibility UMD packages @dimforge/rapier2d-compat or @dimforge/rapier3d-compat. These packages embed the WASM file (encoded in base64) into the main JS file. This results in a slightly different initialization process:

- Example 2DExample 3D

```
import RAPIER from 'https://cdn.skypack.dev/@dimforge/rapier3d-compat';

RAPIER.init().then(() => {
    // Run the simulation.
    _run_simulation(RAPIER);
});

// OR using the await syntax:
async function run_simulation() {
    await RAPIER.init();
    // Run the simulation.
    _run_simulation(RAPIER);
}
```

A complete example can be found on codepen.