Personal Project Alpha Release

Name: Aydin Kaan Cinar

Name of my library: kaan.js (named it after myself)

Description:

This is a 2-D physics simulation library. Its main purpose is for users to create any type of matter they would like by playing with a matter's specifications. For example, they can create a solid, liquid, or gas matter, and they can change their mass, volume, color. If it is solid, they can give it a shape such as a box, circle, triangle, or some random amorph shape. The library will also let users chose some other specifications such as the boiling, and evaporating point, etc. Also, users will be able to drag these matters they created into space, where they can adjust the temperature and gravity of that space and see how their matter reacts to that space. The library will also let the user be able to add some interaction between the matters they create, they can make two matters react-able between each other, and they can add some other specifications to the new matter that will be created on that interaction. Other than reactions like this, liquid and solid objects can collide as well. I will also let the user drag the solid objects by clicking. I can also add some keyboard functions to move solid objects the user choose, such as if they press space, this object might move with some velocity and user can observe how the object will move with the specified gravity, or they can observe how this object will move in a liquid (if it can). I will try to add more functions to this library to be able to simulate an almost real physical environment. It should be useful for developers because instead of writing their own functions for these objects, they can just use these library's functions, such as they don't need to implement gravity, this attribute can just be added to space. And they don't need to write code to specify that there will be an interaction between two types of matters, they can just use libraries functions to create that functionality. This app can be helpful for end-users as well. Teachers can create the environment, and the matters by specifying their properties, and then demonstrate it to a class. For example, they can create a liquid object and play with its properties to make it similar to water (H2O), and they can create several solid objects with different densities to demonstrate buoyancy. Another demonstration could be, changing the gravity of the space, and then showing how an object reacts to the gravity change. Other than educational purposes, this library also could be used for games. As I noted above, the library will let the user chose some keyboard functionality to move the objects, this could be used in a game. The developer can create the environment using the library functions, create platforms on other obstacles, and assign keyboard functions to a matter he creates and make it the player of the game.

Features implemented:

Users can create an environment, assign gravity to this environment, start the environment functions whenever they want, apply gravity to objects, induce reactions. Users can also create matters in solid, liquid, and gas form. Users can add the drag and drop functionality to solid objects (while dragging the objects mouse should be on the object all the time), or they can be set immovable. If the solid objects collide with another solid object they stop moving. User can

assign and change position. User can assign color, mass, height and width for these objects. Matters respond to the gravity of the environment. A matter can be set react-able with another type of matter, and when these two objects touch, they create a new product specified by the developer automatically, or if they chose to do it developers can use API functions to induce reactions. Objects can be removed from the environment and added back again. Developers also can choose to add an environment controller UI to the environment, where an end-user can add new matters to the environment through this UI. They can set the height, width, position, color, mass, and form (solid, liquid, gas) of the object before creating the object.

Link to my library: https://thawing-forest-48724.herokuapp.com/examples.html

Objects that represent something in DOM:

- Matter: Matter object stores the div corresponding to itself, and it stores some style properties that are directly correlated with the matter's specifications: width, height, x position, y position, color, div id. The library manipulates these variables to simulate how a matter would be.
- Environment: This also has the div corresponding to itself, width, height, color, and id. This also has event listeners attached to it to keep track of mouse movements, so that the objects can be moved.
- Environment Controller: This creates many DOM elements since it is essentially a built in UI for the environment. It has a corresponding div as well, and it currently only has Matter creation functionality. It has a screen div, which later might be changeable as well, for instance I might add a functionality where user can switch between properties of different items on the screen and display that properties on UI. But right now, it only displays a Matter Creator.
- Matter Creator: This creates many buttons, sliders, and input fields from user to create a new matter. As color assignment changes, it shows the user what the new color is by changing color property of a div.

Some API functions:

- Enviroment(name, height, width, color, temparature, gravity):
    - name: name of the environment, takes the id of a div that is in the html document, for example if the environment is going to be named env1, there should be a div defined such that .
    - height: height of the environment
    - width: width of the environment
    - color: color of the environment
    - temperature: temperature of the environment
    - gravity: gravity of the environment
    - createEnviroment(): Initializes the environment using the properties assigned by the user. Adds event listeners to the div corresponding to this object.
    - applyGravity(): This function is used inside the library, but can be used by the developers as well, it applies gravity to all the objects in environment.

    - applyGravityOnList(lst): Same as above, but applies gravity on a given lst of objects, some codes might be repetitive here, might clean some code in applyGravity() for the next phase.
    - startEnviroment(): This function starts interactions in environment, this has to be called by developer manually.
    - initiateReaction(reactant1, reactant2, product): This is called automatically by the library functions if to react-able objects collide, but can also be called manually by developer to induce a reaction.
    - removeItem(item): Removes the passed in matter from the environment

- EnviromentController(enviroment):
    - enviroment: this is the environment that the controller controls
    - create(): Creates the EnviromentController

- Matter(enviroment, width, height, x, y, color="White", mass=0, movable=true, displayItem=true, name = ""):
    - environment: environment that the matter is in
    - width: width of the matter
    - height: height of the matter
    - x: x position
    - y: y position
    - color: color of the matter, white by default
    - mass: mass of the matter, 0 by default, might change it to a very small float later, as it doesn't really make sense to have a matter with mass 0.
    - movable: assigns if the matter is movable by dragging it with mouse, true by default.
    - displayItem: assigns if the matter should be displayed right after the initialization, true by default
    - name: the name of the matter, this is the main identifier for a type of matter.
    - createSolid(): creates a solid matter with above specifications
    - createLiquid(): creates a liquid matter with above specifications
    - createGas(): creates a gas matter with above specifications
    - isColliding(): used a lot by the library for object interactions, buy might be useful for the developers too, returns a list of other matters this matter is touching.
    - changeColor(r, g, b): changes the color of the matter to the color specified with the passed red, green, blue values
    - changePosition(x, y): changes the position of the matter to the x and y
    - logProperties(): sends a message which has the properties of this matter
    - showItem(): sets displayItem true and adds the matter to the DOM
    - setReactable(other, product): makes this matter react-able with the other matter, if a reaction occurs later, the product is going to be the result of the reaction.
    - getProduct(other): returns the product that is formed by reacting with other, returns null if there is no reaction with other.

- duplicateMatter(x, y): creates another matter with same specifications as this, at x and y position that is passed in.
- isSame(): returns if the matters share the same name (are they same type of matter)

Features that will be implemented:

I am thinking of simulating the movement of objects in liquids and gases as well. For example, properties that will simulate buoyancy. The objects will move in gases and liquids depending on their density. Also, I am thinking to add more features to the UI functionality of the library, maybe in addition to adding new objects, remove functionality as well, and I will add a reaction creation functionality to the UI. I will also add a temperature setting to environments and matters. I might also add more shapes for solid matters for developer/user to choose for, at the moment all the solid matters are only can have a rectangle shape.