**PROJECT DOCUMENTATION**

SUBJECT**:** **Bernoulli’s Experiment Simulator Using JavaScript.**

PREPARED BY**: Aditya Rathi**

**1. INTRODUCTION**

The Bernoulli’s Experiment Simulator is the JavaScript version of the pre-existing simulator programmed in Java3D. The simulator runs on all the modern web-browsers like Chrome, Firefox, Edge etc., and does not need any requirement of any third party application to run, unlike the previous version.

**The following is the information related to the Tech-Stack used in the project:**

* Programming Language Used: **JavaScript**
* Markup Language Used: **HTML5**
* Style-Sheet Language Used: **CSS3**
* External Libraries/Frameworks Used: **SVG.js, Canvas.js, jQuery and Bootstrap**

**FILES:**

1. Codes/index.html
2. Codes/style.css
3. Codes/script.js
4. Libraries/svg.js
5. Libraries/jquery.js

*Canvas.js and Bootstrap are imported through a link in the index.html file*

**2. OVERALL DESCRIPTION**

**2.1 BUTTONS:**

**2.1.1 START BUTTON:**

FUNCTIONALITY: Starts the animation of the simulation.

IMPLEMENTATION: The “START” button has been given ID “start” in the index.html file. A variable “startButton” has been declared in the script.js file to which this ID has been assigned by using DOM function document.getElementById(). Function toggleAnimation() executes on click of this button.

**2.1.2 RESET BUTTON:**

FUNCTIONALITY: Refreshes the webpage index.html

IMPLEMENTATION: The “RESET” button has been given ID “reset” in the index.html file. A variable “resetButton” has been declared in the script.js file to which this ID has been assigned by using DOM. location.reload() function executes on click of this button.

**2.1.3 “CALCULATIONS” BUTTON:**

FUNCTIONALITY: Hyperlink to the Calculations section of the webpage.

IMPLEMENTATION: The “CALCULATIONS” button has been given ID “calc-button” in the index.html file. A variable “calcButton” has been declared in the script.js file to which this ID has been assigned by using DOM. On click of this button, page scrolls into the section of ID “myForm” by using DOM and scrollIntoView() function.

**2.1.4 “CALCULATE!” BUTTON**:

FUNCTIONALITY: Performs calculations related to the experiment. Answer being displayed in a JavaScript alert box and plots a graph displayed below the button.

IMPLEMENTATION: The “CALCULATE!” button has been given ID “result” in the index.html file. A variable “resultButton” has been declared in the script.js file to which this ID has been assigned using DOM. On click of this button, function calculate() executes in which answer displays in an alert box. In calculate() function, a plotGraph() function is called which plots the graph related to the experiment.

**2.1.5 “CLEAR” BUTTON:**

FUNCTIONALITY: Clears the values of the form and the graph.

IMPLEMENTATION: The “CLEAR” button has been given ID “clear-btn” and its button type has been set to “reset” in the index.html file. A variable “clearButton” has been declared in the script.js file to which this ID has been assigned using DOM. On click of this button, form values get cleared and function plotGraph() function has been called without any arguments, thereby removing the previous values (if entered).

**2.2 THE SIMULATION**

*The simulation mainly SVG.js library to draw the outlines and the perform horizontal or downward water animations of the experiment. Upward water animations have been programmed using jQuery.*

**2.2.1 HTML STRUCTURE**

The simulation has been programmed for mainly two <div> elements in the index.html file. The first one has ID “bernoulli-setup” and class “bernoulli-class”, which contains JavaScript code for drawing the outlines of the experiment and for the horizontal water animations. The second <div> has ID “upward-animations”. The second <div> has 7 sub divisions tags to program the vertical water animations of the experiment which will be explained further below.

**2.2.2 OUTLINES OF THE EXPERIMENT (SVG.js)**

IMPLEMENTED BY: Function **experimentSetup()** in script.js file.

An SVG drawing variable “draw” has been declared at the top of script.js file as a global variable which creates an SVG area of size 1300x620 px.

*Variables in experimentSetup():*

1. tankLeft - Denotes the left beaker of water
2. incomingPipe - Denotes the pipe from where water initiates
3. ductLower - Denotes the middle tube of the setup
4. tankRight - Denotes the right beaker of water
5. verticalPipe1 - Denotes first vertical pipe above the middle tube
6. verticalPipe2 - Denotes second vertical pipe above the middle tube
7. verticalPipe3 - Denotes third vertical pipe above the middle tube
8. verticalPipe4 - Denotes fourth vertical pipe above the middle tube
9. measureTank - Denotes the measuring tank at the bottom

*polyline() / polygon() functions of SVG.js have been used to draw these.*

**2.2.3 WATER ANIMATION**

IMPLEMENTED BY: Function **toggleAnimation()**. Function toggleAnimation() is called when the onclick event of “START” button occurs.

**Part 1: USING SVG.js**

*For horizontal and downward water animations*

Variables Used:

1. incomingPipeHorizontal - Denotes water animation of incoming pipe (1)
2. incomingPipeVertical - Denotes water animation of incoming pipe (2)
3. ductRect - Denotes water animation in rectangular part of middle tube
4. ductTriangle1 - Denotes animation in first triangular part of middle tube
5. ductTriangle2 - Denotes animation in exit triangular part of middle tube
6. outgoingPipeHorizontal - Denotes animation in outgoing pipe (1)
7. outgoingPipeVertical - Denotes animation in outgoing pipe (2)

***Concept used:*** *Water of a particular shape is drawn for a negligible size (not visible) and its dimensions are increased in a direction using animate().size() function of SVG.js*

**Part 2: USING jQuery**

*For upward water animations*

7 <div> elements inside the <div> with ID “upward-animations”. These <div> elements are as follows:

**IDs CLASSESTime taken:**

1. tankLeft tank-left
2. verticalPipe1 vertical-pipe1
3. verticalPipe2 vertical-pipe2
4. verticalPipe3 vertical-pipe3
5. verticalPipe4 vertical-pipe4
6. tankRight tank-right
7. measureTank measure-tank

***Concept used:*** *These <div> elements have been styled in style.css file initially with negligible height. After respective waiting time for each element, height changes accordingly using animate() function of jQuery, in the toggleAnimation() function inside script.js*

**2.3 CALCULATIONS**

**2.3.1 PERFORMING CALCULATIONS**

IMPLEMENTED BY: Function **calculate()**.

A form for filling the values of experiment has been created by using HTML and CSS. The values of the placeholders are used in the calculate() function by using DOM property.

Variables Used for Reading 1:

1. Volume 6. discharge
2. Time 7. velocityHead
3. Area 8. pressureHead
4. Pressure 9. totalHead
5. Velocity

Variables Used for Reading 2:

1. volume2 6. discharge2
2. time2 7. velocityHead2
3. area2 8. pressureHead2
4. pressure2 9. totalHead2
5. velocity2

The answer avgTotalHead is displayed using JavaScript alert.

*Exceptions are thrown if values of time/area are non-positive and that of pressure/volume are negative or if all values are not filled in the placeholders of form. Exception messages are displayed in JavaScript alert.*

**2.3.2 GRAPH (USING Canvas.js)**

HTML: Placed inside the <div> element with ID “chartContainer”

IMPLEMENTED BY: Function **plotGraph()**.

plotGraph() function takes six variables as arguments -

1. velocityHead
2. velocityHead2
3. pressureHead
4. pressureHead2
5. totalHead
6. totalHead2

There are two Y-Axes in the graph, one for Velocity Head and the other for Pressure Head. X-Axis is for Total Head. All the length units are centimeters.

When plotGraph() function is called without any arguments, an empty graph is displayed. This empty graph has been displayed on the refresh of the page and after the onclick event of the “CLEAR” button.