Index.html

BEGIN HTML DOCUMENT

// Document type declaration for HTML5

DECLARE document type as HTML5

// Start of the HTML document

START HTML with language set to English

// Head section of the document

START HEAD section

// Specify the character set for the document meta charset=””

SET character encoding to UTF-8

// Define the viewport settings for responsive design. Meta name=”” content=””

SET viewport to width=device-width, initial-scale=1.0

// Link external CSS stylesheets link rel=”stylesheet”

LINK to external stylesheet "styles.css"

// Set the title of the webpage

SET title to "RPG - Dragon Repeller"

END HEAD section

// Body section of the document

START BODY section

// Main game container

CREATE a DIV with id "game"

// Player stats display area

CREATE a DIV with id "stats"

// Display player's experience points (XP)

CREATE a SPAN with class "stat" containing text "XP:"

CREATE a STRONG element containing a SPAN with id "xpText" and initial value "0"

// Display player's health points -repeat the above

CREATE a SPAN with class "stat" containing text "Health:"

CREATE a STRONG element containing a SPAN with id "healthText" and initial value "100"

// Display player's gold amount

CREATE a SPAN with class "stat" containing text "Gold:"

CREATE a STRONG element. Within that element also create a SPAN with id "goldText" and initial value "50"

END DIV with id "stats"

// Control buttons for player actions

CREATE a DIV with id "controls"

// Button to navigate to the store

CREATE a BUTTON with id "button1" and text "Go to store"

// Button to navigate to the cave

CREATE a BUTTON with id "button2" and text "Go to cave"

// Button to initiate a fight with the dragon

CREATE a BUTTON with id "button3" and text "Fight dragon"

END DIV with id "controls"

// Monster stats display area

CREATE a DIV with id "monsterStats"

// Display the monster's name

CREATE a SPAN with class "stat" containing text "Monster Name:"

CREATE a STRONG element containing a SPAN with id "monsterName"

// Display the monster's health points

CREATE a SPAN with class "stat" containing text "Health:"

CREATE a STRONG element containing a SPAN with id "monsterHealth"

END DIV with id "monsterStats"

// Text area for game narrative and instructions

CREATE a DIV with id "text"

SET initial text to "Welcome to Dragon Repeller. You must defeat the dragon that is preventing people from leaving the town. You are in the town square. Where do you want to go? Use the buttons above."

END DIV with id "game"

// Include JavaScript files in the correct order

LINK to JavaScript file "../model/gameData.js"

LINK to JavaScript file "../controller/gameController.js"

END BODY section

END HTML document

END HTML DOCUMENT

//stylesheet.css

A screenshot of a computer

Description automatically generated

BEGIN STYLESHEET

// Set the background color for the entire page. background-color:

SET background color of BODY to dark blue (#0a0a23)

// Styling for the text area that displays game instructions and messages

SET background color of TEXT area (#text) to dark blue (#0a0a23)

SET text color of TEXT area to white (#ffffff)

SET padding of TEXT area to 10 pixels

// Styling for the main game container

SET maximum width of GAME container (#game) to 500 pixels

SET maximum height of GAME container to 400 pixels

SET background color of GAME container to white (#ffffff)

SET text color of GAME container to white (#ffffff)

SET top margin of GAME container to 30 pixels and auto for left and right to center it

SET padding of GAME container to 10 pixels

// Styling for controls and player stats areas

SET border of CONTROLS and STATS containers (#controls, #stats) to 1 pixel solid dark blue (#0a0a23)

SET padding of CONTROLS and STATS containers to 5 pixels

SET text color of CONTROLS and STATS containers to dark blue (#0a0a23)

// Styling for monster stats area

SET display of MONSTER STATS area (#monsterStats) to none (hidden)

SET border of MONSTER STATS area to 1 pixel solid dark blue (#0a0a23)

SET padding of MONSTER STATS area to 5 pixels

SET text color of MONSTER STATS area to white (#ffffff)

SET background color of MONSTER STATS area to red (#c70d0d)

// Styling for stat text elements

SET right padding of STAT elements (.stat) to 10 pixels

// Styling for buttons

SET cursor of BUTTONS to pointer (indicates clickable)

SET text color of BUTTONS to dark blue (#0a0a23)

SET background color of BUTTONS to orange (#feac32)

SET background image of BUTTONS to a linear gradient from light yellow (#fecc4c) to orange (#ffac33)

SET border of BUTTONS to 3 pixels solid orange (#feac32)

END STYLESHEET

// gameData.js

The elements are defined as let including inventory (probably because at some point the entire inventory will be replaced). Weapons monsters and locations as cont. Copying " in word could give a problem as it gives “ “ and not ".

|  |  |  |  |
| --- | --- | --- | --- |
| xp | 0 |  |  |
| health | 100 |  |  |
| gold | 50 |  |  |
| currentWeapon | 0 |  |  |
| fighting |  |  |  |
| monsterHealth |  |  |  |
| inventory | array | "stick" |  |
| weapons | array | Name:  Power: | Stick  5 |
|  |  | Name:  Power: | Dagger  30 |
|  |  | Name:  Power: | Claw hammer  50 |
|  |  | Name:  Power: | Sword  100 |

{ name: 'stick', power: 5 },

|  |  |  |  |
| --- | --- | --- | --- |
| monsters | array | Name:  Level:  Health: | Slime  2  15 |
|  |  | Name:  Level:  Health: | fanged beast  8  60 |
|  |  | Name:  Level:  Health: | dragon  20  300 |
| locations | array | Name:  "button text"  "button functions"  text: | "town square"  ["Go to store", "Go to cave", "Fight dragon"]  [],  // Leave empty or use placeholder values  You are in the town square. You see a sign that says \"Store\"." |
|  |  | Name:  "button text"  "button functions"  text: | "store",  ["Buy 10 health (10 gold)", "Buy weapon (30 gold)", "Go to town square"],  [],  // Leave empty or use placeholder values  "You enter the store." |
|  |  | Name:  "button text"  "button functions"  text: | "cave",  "Fight slime", "Fight fanged beast", "Go to town square"],  [],  // Leave empty or use placeholder values  "You enter the cave. You see some monsters." |
|  |  | Name:  "button text"  "button functions"  text: | "fight"  ["Attack", "Dodge", "Run"]  [],  // Leave empty or use placeholder values  "You are fighting a monster." |
|  |  | Name:  "button text"  "button functions"  text: | "kill monster",  ["Go to town square", "Go to town square", "Go to town square"],  [],  // Leave empty or use placeholder values  'The monster screams "Arg!" as it dies. You gain experience points and find gold.' |
|  |  | Name:  "button text"  "button functions"  text: | "lose",  ["REPLAY?", "REPLAY?", "REPLAY?"]  [],  // Leave empty or use placeholder values  "You die. &#x2620;" |
|  |  | Name:  "button text"  "button functions"  text: | "win",  ["REPLAY?", "REPLAY?", "REPLAY?"]  [],  // Leave empty or use placeholder values  "You defeat the dragon! YOU WIN THE GAME! &#x1F389;" |
|  |  | Name:  "button text"  "button functions"  text: | "easter egg",  ["2", "8", "Go to town square?"],  [],  // Leave empty or use placeholder values  "You find a secret game. Pick a number above. Ten numbers will be randomly chosen between 0 and 10. If the number you choose matches one of the random numbers, you win!" |

// gameController.js

A screenshot of a medical test

Description automatically generated

**select an HTML element** from the webpage and assign it to a JavaScript variable. **Thus, for example**: Find the HTML element with the id of button1 on the webpage. Store a reference to this element in the variable button1

|  |  |
| --- | --- |
| const button1 | const button1 = document.querySelector('#button1'); |
| const button2 | const button2 = document.querySelector("#button2"); |
| const button3 | const button3 = document.querySelector("#button3"); |
| const text | const text = document.querySelector("#text"); |
| const xpText | const xpText = document.querySelector("#xpText"); |
| const healthText | const healthText = document.querySelector("#healthText"); |
| const goldText | const goldText = document.querySelector("#goldText"); |
| const monsterStats | const monsterStats = document.querySelector("#monsterStats"); |
| const monsterName | const monsterName = document.querySelector("#monsterName"); |
| const monsterHealthText | const monsterHealthText = document.querySelector("#monsterHealth"); |

 monsterStats.style.display = "none"; //is used to hide an HTML element from the webpage.

text.innerHTML = location.text; is used to set the HTML content of an HTML element (text) to a specific string value stored in the location object.

button1.innerText

button1.onclick

**Description of the update Function in Simple Terms**

When a player moves to a new location in the game, the function update is called to change the game's interface to match that new location.

**Hide the Monster Information:**

First, any information about monsters (like their stats) is hidden. This might be because the player is moving to a place where no monsters are currently visible or relevant.

**Change the Text on the Buttons:** button1.innerText

The game has three buttons that players can use to choose actions. The function updates the text on these buttons to show what actions are available at the new location. For example, if the player moves to a store, the buttons might change to say "Buy Health," "Buy Weapon," and "Return to Town."

**Update What Happens When Buttons are Clicked:** button1.onclick

Not only does the text on the buttons change, but what happens when the player clicks them also changes. The function assigns new actions to each button based on what the player can do in the new location. For instance, clicking a button might now lead to buying an item or starting a fight, depending on where the player is.

**Display New Location Description:**

The function then updates a section of text on the screen to describe the new location. This could be a message like "You are in the town square. You see a sign that says 'Store'." This text gives the player information about where they are and what they might do next.

In summary, this function dynamically changes the game's display and controls whenever the player moves to a different location, making the game interactive and responsive to the player's actions. It ensures that the information shown and the available actions are always appropriate for where the player currently is in the game.

**Pseudocode for : FUNCTION update(location)**

**Note: we use location instead of locations[0] because we want to dynamically load any collation in the update function. We dont know yet which function and that will be called in a different function. In Eamonn’s mind you may call this setting up the skeleton. When placing the code in gameController.js we start with this but that is confusing. We can only start with this if we know there are 8 locations and the update function is the skeleton that has the placeholder parameter location. It might have been less confusing if we use “location\_placeholder” as a parameter. I will also add the explanation text that location\_placeholder will be used in later functions. So when we call update() in a later function we use locations[0]. For example:**

**function goTown() {**

**update(locations[0]);**

**}**

**The real magic is that the update function know that when it is called, whatever is within brackets is what replaces location\_placeholder. In Eamonn’s head there is an “error” . He does not know that function update(location\_placeholder) {} is a placeholder, especially since that placeholder can further be connected (or concatenated) with [“button text”] or [“ button functions”] . Note since the [“button text”] variable is associated with an array you also have to indicate with array. That means [“button text][0] is needed. So:**

**button1.innertext=location\_placeholder[“button text”][0];**

// Hide the monster statistics element

SET display property of monsterStats to "none"

// Update button1 text to the first value in the "button text" array from the location object

SET button1 text to location's "button text" at index 0

// Update button2 text to the second value in the "button text" array from the location object

SET button2 text to location's "button text" at index 1

// Update button3 text to the third value in the "button text" array from the location object

SET button3 text to location's "button text" at index 2

// Set button1's click action to the first function in the "button functions" array from the location object

SET button1 click event to location's "button functions" at index 0

// Set button2's click action to the second function in the "button functions" array from the location object

SET button2 click event to location's "button functions" at index 1

// Set button3's click action to the third function in the "button functions" array from the location object

SET button3 click event to location's "button functions" at index 2

// Update the text element to show the current description from the location object

SET text content to location's "text" property

END FUNCTION

**Pseudo Code for goTown, goStore, and goCave Functions**

FUNCTION goTown

// Call the update function to change the game to the "town square" location

CALL update with the first location in the locations array (locations[0])

END FUNCTION

FUNCTION goStore

// Call the update function to change the game to the "store" location

CALL update with the second location in the locations array (locations[1])

END FUNCTION

FUNCTION goCave

// Call the update function to change the game to the "cave" location

CALL update with the third location in the locations array (locations[2])

END FUNCTION

**Pseudo Code for buyHealth Function**

FUNCTION buyHealth

// Check if the player has at least 10 gold

IF gold is greater than or equal to 10 THEN

// Deduct 10 gold from the player's total

SUBTRACT 10 from gold

// Add 10 health points to the player's health

ADD 10 to health

// Update the displayed gold amount on the screen

SET goldText to the current value of gold

// Update the displayed health amount on the screen

SET healthText to the current value of health

ELSE

// If the player does not have enough gold, show a message

SET text to "You do not have enough gold to buy health."

END FUNCTION

**Pseudo Code for buyWeapon Function**

FUNCTION buyWeapon

// Check if the player can buy a new weapon (current weapon is not the most powerful)

IF currentWeapon is less than the last index in the weapons array THEN

// Check if the player has enough gold to buy a weapon

IF gold is greater than or equal to 30 THEN

// Deduct 30 gold from the player's total

SUBTRACT 30 from gold

// Move to the next weapon in the weapons array

INCREMENT currentWeapon by 1

// Update the displayed gold amount on the screen

SET goldText to the current value of gold

// Get the name of the new weapon

SET newWeapon to the name of the weapon at the index currentWeapon in the weapons array

// Display a message to the player about the new weapon

SET text to "You now have a " + newWeapon

// Add the new weapon to the player's inventory

ADD newWeapon to inventory

// Display the updated inventory to the player

APPEND " In your inventory you have: " + inventory to text

ELSE

// If the player does not have enough gold, show a message

SET text to "You do not have enough gold to buy a weapon."

ELSE

// If the player already has the most powerful weapon

SET text to "You already have the most powerful weapon!"

// Offer the player an option to sell the weapon

SET button2 text to "Sell weapon for 15 gold"

SET button2's click action to the sellWeapon function

END FUNCTION

**Pseudo Code for sellWeapon Function**

FUNCTION sellWeapon

// Check if the player has more than one weapon in their inventory

IF the number of items in inventory is greater than 1 THEN

// Add 15 gold to the player's total

ADD 15 to gold

// Update the displayed gold amount on the screen

SET goldText to the current value of gold

// Remove the first weapon from the inventory and store it in currentWeapon

REMOVE the first item from inventory and STORE it in currentWeapon

// Display a message to the player about the weapon sold

SET text to "You sold a " + currentWeapon + "."

// Display the updated inventory to the player

APPEND " In your inventory you have: " + inventory to text

ELSE

// If the player has only one weapon, show a message

SET text to "Don't sell your only weapon!"

END FUNCTION

**Pseudo Code for fightSlime, fightBeast, and fightDragon Functions**

FUNCTION fightSlime

// Set the "fighting" state to represent fighting a slime

SET fighting to 0

// Call the function to initiate a fight

CALL goFight

END FUNCTION

FUNCTION fightBeast

// Set the "fighting" state to represent fighting a beast

SET fighting to 1

// Call the function to initiate a fight

CALL goFight

END FUNCTION

FUNCTION fightDragon

// Set the "fighting" state to represent fighting a dragon

SET fighting to 2

// Call the function to initiate a fight

CALL goFight

END FUNCTION

**Pseudo Code for goFight Function**

FUNCTION goFight

// Update the game to show the "fight" location

CALL update with the fourth location in the locations array (locations[3])

// Set the monster's health based on the type of monster being fought

SET monsterHealth to the health value of the monster at index "fighting" in the monsters array

// Make the monster's statistics visible on the screen

SET display property of monsterStats to "block"

// Display the name of the monster being fought

SET monsterName text to the name of the monster at index "fighting" in the monsters array

// Display the monster's current health

SET monsterHealthText to the current value of monsterHealth

END FUNCTION

**Pseudo Code for attack Function**

FUNCTION attack

// Describe the monster's attack on the player

SET text to "The " + name of the monster being fought + " attacks."

// Describe the player's counterattack

APPEND " You attack it with your " + name of the player's current weapon + "." to text

// Decrease the player's health by the monster's attack value

DECREASE health by the value returned from getMonsterAttackValue with the monster's level as the argument

// Check if the player hits the monster

IF isMonsterHit returns true THEN

// Decrease the monster's health based on the player's weapon power and experience points

DECREASE monsterHealth by the sum of the player's weapon power, a random value based on experience points, and 1

ELSE

// Describe a missed attack

APPEND " You miss." to text

END IF

// Update the player's health on the screen

SET healthText to the current value of health

// Update the monster's health on the screen

SET monsterHealthText to the current value of monsterHealth

// Check if the player has lost the game

IF health is less than or equal to 0 THEN

CALL lose function

// Check if the monster has been defeated

ELSE IF monsterHealth is less than or equal to 0 THEN

// If fighting the dragon (fighting index 2), win the game

IF fighting equals 2 THEN

CALL winGame function

// Otherwise, defeat the monster

ELSE

CALL defeatMonster function

END IF

END IF

// Randomly check if the player's weapon breaks

IF a random number is less than or equal to 0.1 AND inventory length is not 1 THEN

// Describe the weapon breaking

APPEND " Your " + name of the last item in inventory + " breaks." to text

// Remove the last weapon from the inventory

REMOVE the last item from inventory

// Downgrade the player's weapon

DECREASE currentWeapon by 1

END IF

END FUNCTION

**Pseudo Code for getMonsterAttackValue Function**

FUNCTION getMonsterAttackValue(level)

// Calculate the monster's attack value

SET hit to (level \* 5) minus a random number between 0 and (xp - 1)

// Print the calculated hit value to the console (for debugging)

PRINT hit to the console

// Return the hit value if it's greater than 0; otherwise, return 0

RETURN hit if hit is greater than 0, otherwise RETURN 0

END FUNCTION

**Pseudo Code for isMonsterHit Function**

**FUNCTION isMonsterHit**

// Determine if the player's attack hits the monster

RETURN true if a random number is greater than 0.2 OR the player's health is less than 20

OTHERWISE, RETURN false

**END FUNCTION**

**Pseudo Code for dodge Function**

FUNCTION dodge

// Display a message indicating the player dodged the monster's attack

SET text to "You dodge the attack from the " + name of the monster being fought

END FUNCTION

**Pseudo Code for defeatMonster Function**

FUNCTION defeatMonster

// Increase the player's gold based on the defeated monster's level

ADD the result of (monster's level \* 6.7) rounded down to the nearest integer to gold

// Increase the player's experience points based on the defeated monster's level

ADD the monster's level to xp

// Update the displayed gold amount on the screen

SET goldText to the current value of gold

// Update the displayed experience points on the screen

SET xpText to the current value of xp

// Update the game state to show the location after the monster's defeat

CALL update with the location at index 4 in the locations array

END FUNCTION

**Pseudo Code for lose Function**

FUNCTION lose

// Update the game state to the "lose" location

CALL update with the location at index 5 in the locations array

END FUNCTION

**Pseudo Code for winGame Function**

FUNCTION winGame

// Update the game state to the "win" location

CALL update with the location at index 6 in the locations array

END FUNCTION

**Pseudo Code for restart Function**

FUNCTION restart

// Reset player stats

SET xp to 0

SET health to 100

SET gold to 50

SET currentWeapon to 0

SET inventory to ["stick"]

// Update displayed stats

SET goldText to the current value of gold

SET healthText to the current value of health

SET xpText to the current value of xp

// Return to the town location to start over

CALL goTown

END FUNCTION

**Pseudo Code for easterEgg Function**

FUNCTION easterEgg

// Update the game state to the "easter egg" location

CALL update with the location at index 7 in the locations array

END FUNCTION

FUNCTION pickTwo

// Call the pick function with the number 2

CALL pick with argument 2

END FUNCTION

FUNCTION pickEight

// Call the pick function with the number 8

CALL pick with argument 8

END FUNCTION

**Pseudo Code for pick Function**

FUNCTION pick(guess)

// Create an empty array to store random numbers

CREATE an empty array called numbers

// Generate 10 random numbers between 0 and 10

WHILE numbers contains less than 10 items DO

ADD a random integer between 0 and 10 to numbers

END WHILE

// Display the player's guess and the random numbers

SET text to "You picked " + guess + ". Here are the random numbers:\n"

FOR each number in numbers DO

APPEND the number to text with a newline character

END FOR

// Check if the player's guess is among the random numbers

IF guess is in numbers THEN

// Player guessed correctly

APPEND "Right! You win 20 gold!" to text

ADD 20 to gold

SET goldText to the current value of gold

ELSE

// Player guessed incorrectly

APPEND "Wrong! You lose 10 health!" to text

SUBTRACT 10 from health

SET healthText to the current value of health

// Check if player's health is zero or less

IF health is less than or equal to 0 THEN

CALL lose function

END IF

END IF

END FUNCTION