Styling A Dynamic Grid Using JavaScript Activity (60:00)

#### **Learning Outcome Addressed**

 5. Use JavaScript to set web page styles

Follow the instructions in NextTech to complete the activity.

**Styling A Dynamic Grid Using JavaScript**

### Dynamically Applying Styles

As you've seen so far, JavaScript enables you to dynamically add HTML elements and style them using CSS classes. This is important as you start building more dynamic web pages.

In this activity, you're going to create a dynamic web page containing a grid and two buttons:

* CREATE GRID: When clicked, a grid of 100 squares is created dynamically on the page. This is accomplished by calling the function createGrid()
* MOVE: When clicked, the background color of the grid cells need to change from one cell to the next every 100ms. This is accomplished by calling the function move()

The starter code for this activity contains the following files:

* index.html: This file contains the HTML code. Note the two buttons referenced above and the main div with the id: target. This div will hold the dynamically created grid
* styles.css: This file contains the CSS style rules. Note the "on" class here. This class sets the background color to orange
* grid.js: This file contains the JavaScript code needed to complete this task. Within this file, note the following functions:
  + createGrid(): This function will create a 100 square grid, which will be dynamically inserted into the HTML element div id="target" using appendChild. You should also place a label on each grid cell that has the cell's number, as well as give each cell a unique ID.
  + move(): This function will increment the position variable by one every 100ms, taking into consideration the gridSize variable value. It will also make use of the toggle function to make CSS changes (turning the cell orange) to its specific div element.
  + toggle(): Takes a position parameter referencing a grid cell and sets the class name of that cell to "on".

In grid.png, you can find an example of what your completed grid should look like.

Make sure you read all the code comments within these files to get a better understanding of how each function should perform.

**Your task in this activity is to implement the createGrid(), move() and toggle() functions.**

Note Due to the timing of the exercise, you need to wait a minute between checking the separate tasks.

Hints:

* Reference Dr. William's video on creating a dynamic grid for a live demo of what this grid's functionality is.
* Each cell is its own div element.
* You can use setTimeout() to delay the execution of the code by a set number of milliseconds.
* This exercise makes use of loops, dynamic HTML, and comparison operators. Make sure you are using each of these programming elements to help complete the exercise.
* You can use appendChild() to append new HTML elements to the existing ones.

Tasks

Follow the hints on grid.js to create the "Create Grid" function

**1**

Follow the hints in grid.js to create the move and toggle functions. Both of them work together to dynamically create the grid cells turning orange.

<!DOCTYPE html>

<html>

<head>

    <title>Grid</title>

    <link rel="stylesheet" href="./styles.css">

</head>

<body>

    <button onclick="createGrid()" id="btn\_createGrid">CREATE GRID</button>

    <button onclick="move()" style="margin-bottom: 10px;" id="btn\_move">MOVE</button>

    <main class="content" id="target">

    </main>

    <script>

    </script>

</body>

<script src="./grid.js"></script>

</html>

let position = 1; // The starting position of the grid

const gridSize = 100; // The total number of the grid cells

function createGrid() {

  // Should create a new Div based on the gridSize variable value and append these divs to the element with id "target"

}

function move() {

  // Should increment the position variable by 1 each 100ms taking into consideration the gridSize variable value

  // This function should make use of the toggle function below to change the CSS class on a specific div element

}

function toggle(position) {

  // Takes a position parameter referencing a grid cell and sets the class name of that cell (or div) to the class 'on'

  // The CSS class "on" is defined in the styles.css file

}

//don't change this line

if (typeof module !== 'undefined') {

  module.exports = { createGrid, move, toggle };

}

.on {

  background: orange;

}

.content {

  display: grid;

  grid-template-columns: repeat(10, 1fr);

  grid-template-rows: repeat(10, 1fr);

  gap: 5px;

}

div {

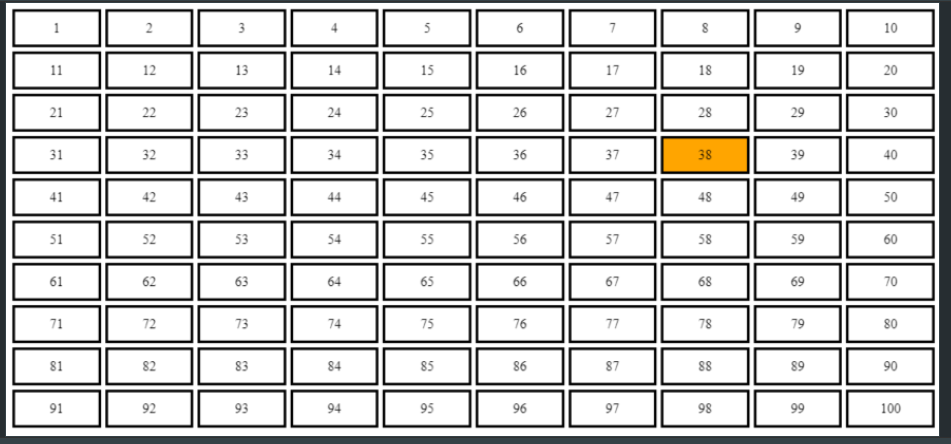
  padding: 10px;

  text-align: center;

  border-style: solid;

  font-size: 1em;

}



Solution

let position = 1; // The starting position of the grid

const gridSize = 100; // The total number of the grid cells

function createGrid() {

  // Should create a new Div based on the gridSize variable value and append these divs to the element with id "target"

  for (let i=1; i<=gridSize; i++){

                var div = document.createElement("div");

                div.id = 'n' + i;

                div.innerText = i;

                document.getElementById("target").appendChild(div);

            }

}

function move() {

  // Should increment the position variable by 1 each 100ms taking into consideration the gridSize variable value

  // This function should make use of the toggle function below to change the CSS class on a specific div element

  setTimeout(() => {

                if (position > 100) return;

                toggle(position);

                toggle(position-1);

                position += 1;

                move();

            }, 100);

}

function toggle(position) {

  // Takes a position parameter referencing a grid cell and sets the class name of that cell (or div) to the class 'on'

  // The CSS class "on" is defined in the styles.css file

  if (position < 1) return;

            const name = 'n' + position;

            const element = document.getElementById(name);

            element.classList.toggle('on');

}

//don't change this line

if (typeof module !== 'undefined') {

  module.exports = { createGrid, move, toggle };

}