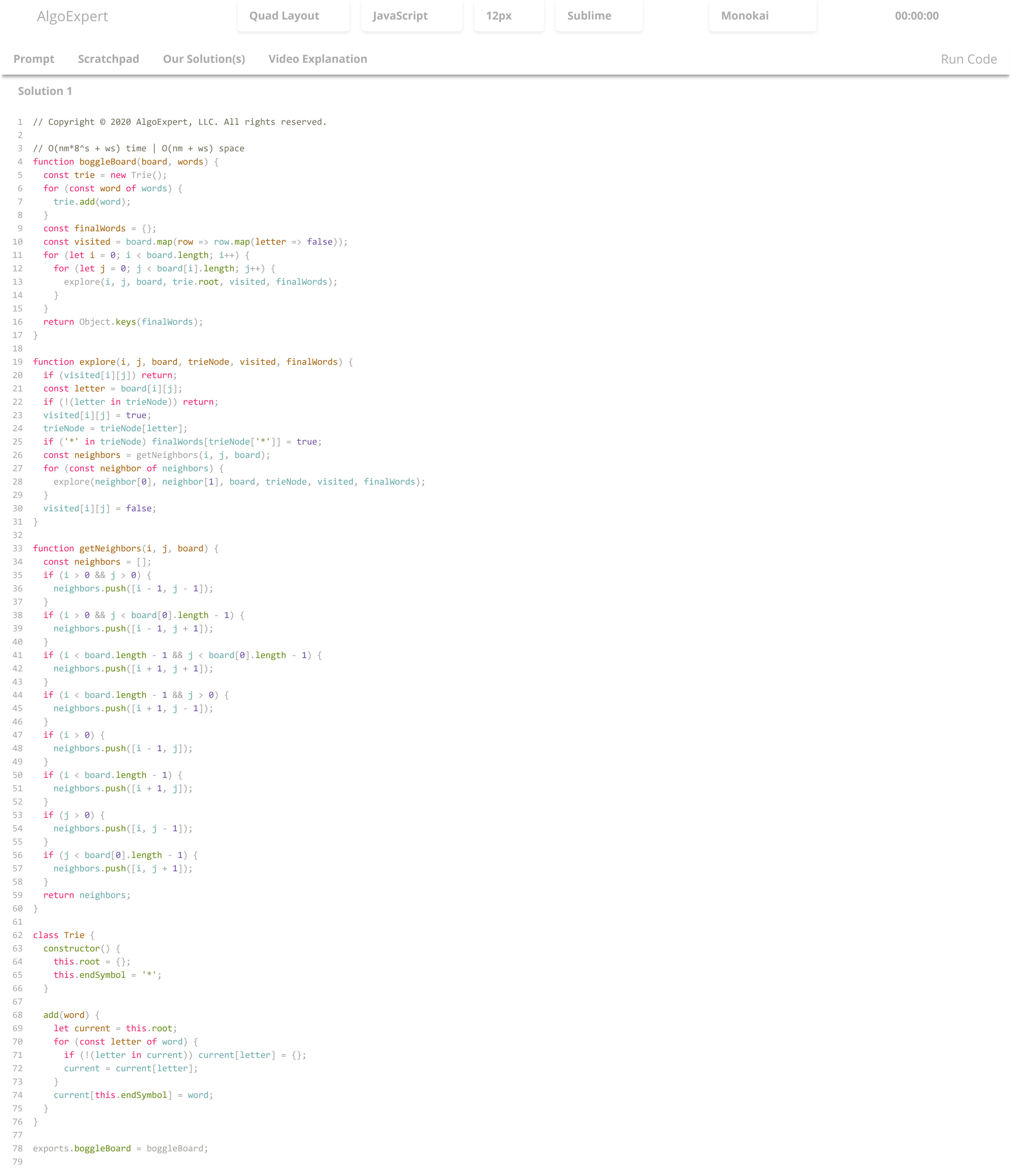

Solution 1

```javascript
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

// O(nm*8^s + ws) time | O(nm + ws) space
function boggleBoard(board, words) {
  const trie = new Trie();
  for (const word of words) {
    trie.add(word);
  }
  const finalWords = {};
  const visited = board.map(row => row.map(letter => false));
  for (let i = 0; i < board.length; i++) {
    for (let j = 0; j < board[i].length; j++) {
      explore(i, j, board, trie.root, visited, finalWords);
    }
  }
  return Object.keys(finalWords);
}

function explore(i, j, board, trieNode, visited, finalWords) {
  if (visited[i][j]) return;
  const letter = board[i][j];
  if (!(letter in trieNode)) return;
  visited[i][j] = true;
  trieNode = trieNode[letter];
  if ('*' in trieNode) finalWords[trieNode['*']] = true;
  const neighbors = getNeighbors(i, j, board);
  for (const neighbor of neighbors) {
    explore(neighbor[0], neighbor[1], board, trieNode, visited, finalWords);
  }
  visited[i][j] = false;
}

function getNeighbors(i, j, board) {
  const neighbors = [];
  if (i > 0 && j > 0) {
    neighbors.push([i - 1, j - 1]);
  }
  if (i > 0 && j < board[0].length - 1) {
    neighbors.push([i - 1, j + 1]);
  }
  if (i < board.length - 1 && j < board[0].length - 1) {
    neighbors.push([i + 1, j + 1]);
  }
  if (i < board.length - 1 && j > 0) {
    neighbors.push([i + 1, j - 1]);
  }
  if (i > 0) {
    neighbors.push([i - 1, j]);
  }
  if (i < board.length - 1) {
    neighbors.push([i + 1, j]);
  }
  if (j > 0) {
    neighbors.push([i, j - 1]);
  }
  if (j < board[0].length - 1) {
    neighbors.push([i, j + 1]);
  }
  return neighbors;
}

class Trie {
  constructor() {
    this.root = {};
    this.endSymbol = '*';
  }

  add(word) {
    let current = this.root;
    for (const letter of word) {
      if (!(letter in current)) current[letter] = {};
      current = current[letter];
    }
    current[this.endSymbol] = word;
  }
}

exports.boggleBoard = boggleBoard;
```