# Exercises

Just to give you some context, sometimes we make some not very smart mistakes (we are only human…). Tools like eslint are a valuable help in these situations. Eslint statically analyzes code to quickly find issues.

For all the exercises proposed you should take some time to analyze the code WITHOUT using the tool. As an exceptionally talented programmer, you should be able to catch some of these issues without using it.

Then you can run eslint and the issues will be outlined! As easy as that!

You should take into account that, in some cases, fixing one issue reveals another set of brand new issues for you to enjoy! Fun right?

For additional information on the issues go here.

## Exercise 1

The main goal of this exercise is for you to understand how the tool works and the different issues it identifies. In order to do that, four simple functions are presented to you. You should start fixing them in order.

Did you notice anything strange? Eslint is only reporting one error...weird...lets see what happens when you try to fix it!

### Exercise 1.a.

In this function, it is expected that you fix 4 different issues.

```
function exA(a, b, op, op) {
  let res;
  switch (op) {
    case 'SUM':
      res = a + b;
      break;
    case 'SUB':
      res = a - b;
      break;
    case 'PROD':
      res = a * b;
    case 'SUM':
      res = a + b;
      break;
    case 'DIV':
      res = a / b;
      break;
  }
  return res;
}
```

### Exercise 1.b.

In this function, it is expected that you fix 5 different issues.

```
function exB(a, b, op) {
  const value = exC(a, b, op);
  if (vallue == 'POSITIVE') {
    const c = true;
  } else {
    const b = false;
  }
  return c;
}
```

### Exercise 1.c.

In this function, it is expected that you fix 5 different issues.

```
function exC(a, b, op) {
  const value = exA(a, b);
  if (value < 0) {
    return 'NEGATIVE';
  } else if (value == 0) {
    return 'ZERO';
  } else if (value_ > 0) {
    return 'POSITIVE';
  } else if (value < 0) {
    return 'NEGATIVE';
  }
}
```

### Exercise 1.d.

In this function, it is expected that you fix 2 different issues.

```
function exD(a, b, op) {
  value = exB(a, b, op) ? 10 : 20;
  let value;
  let count = 0;
  let res;
  while (value < 30) {
    count += 1;
  }
  return count;
}
```

## Exercise 2

This exercise presents a more “real” application of this tool. It is a very simple program that implements the tic tac toe game. As in the previous exercise, we advise you to fix the issues in order.

We are expecting you to encounter around 14 issues (it will depend on your resolution of previous issues) and 2 warnings.

Again, there is only one error...but you already know what that means...

```
const statusDisplay = document.querySelector('.game--status');

const gameActive = true;
let currentPlayer = 'X';
let gameState = ['', '', '', '', '', '', '', '', ''];

const winningMessage = () => `Player ${currentPlayer} has won!`;
const drawMessage = () => 'Game ended in a draw!';
const currentPlayerTurn = () => `It's ${currentPlayer}'s turn`;

statusDisplay.innerHTML = currentPlayerTurn();

const winningConditions = [
  [0, 1, 2],
  [3, 4, 5],
  [6, 7, 8],
  [0, 3, 6],
  [1, 4, 7],
  [2, 5, 8],
  [0, 4, 8],
  [2, 4, 6],
];

function handleCellPlayed(clickedCell, clickedCellIndex) {
  gameState[clickedCellIndex] = currentPlayer;
  clickedCell.innerHTML = currentPlayer;
}

function handlePlayerChange() {
  currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
  nextPlayer = currentPlayer;
  statusDisplay.innerHTML = currentPlayerTurn();
}

function handleResultValidation() {
  let roundWon = false;
  console.log(roundWon);

  // i += 1
  for (let i = 0; i <= 7; i++) {
    const winCondition = winningConditions[i];
    const a = gameState[winCondition[0]];
    let b = gameState[winCondition[0]];
    let b = gameState[winCondition[1]];
```

```
    const c = gameState[winCondition[2]];

    if (a == '' || b == '' || c == '') {
      continue;
    }
    if (a == b && b == c) {
      roundWon = true;
      break;
      console.log(roundWon);
    }

    let val = false;
    while (val) {
      roundWon = true;
    }
  }

  if (roundWon) {
    statusDisplay.innerHTML = winningMessage();
    gameActive = false;
    return;
  }

  const roundDraw = !gameState.includes('');
  if (roundDraw) {
    statusDisplay.innerHTML = drawMessage();
    gameActive = false;
    return;
  }

  handlePlayerChange();
}

function handleCellClick(clickedCellEvent) {
  const clickedCell = clickedCellEvent.target;
  const clickedCellIndex = parseInt(clickedCell.getAttribute('data-cell-
index'));

  // !== instead of !=
  if (gameState[clickedCellIndex] != '' || !gameActive) {
    return;
  }

  handleCellPlayed(clickedCell, clickedCellIndex);
  handleResultValidation();
}

function handleRestartGame() {
  gameActive = true;
  currentPlayer = 'X';
  gameState = ['', '', '', '', '', '', '', '', ''];
  statusDisplay.innerHTML = currentPlayerTurn();
  document.querySelectorAll('.cell').forEach((cell) => cell.innerHTML =
'');
```

```
}

// function declared but never called
function setGameActive(value, oldValue) {
  gameActive = value;
}

document.querySelectorAll('.cell').forEach((cell) =>
cell.addEventListener('click', handleCellClick));
document.querySelector('.game--restart').addEventListener('click',
handleRestartGame);
```

Need some additional help?

1. Start by fixing the issue ate line 44;
2. go to line 31;
3. line 45;
4. line 48;
5. line 62;
6. line 81;
7. lines 97-99;
8. lines 54-57;
9. line 51;
10. line 46;
11. line 36.
12. done, ezpz.