Dokumentation: Statische Codeanalyse mit ESLint

1) Einleitung

Diese Dokumentation beschreibt die Schritte zur Durchführung einer statischen Codeanalyse eines JavaScript-Projekts mithilfe von ESLint. Es wird ein einfaches "Schere-Stein-Papier"-Spiel analysiert, Fehler werden behoben, und das korrigierte Projekt wird dokumentiert.

2) Projektbeschreibung

Das Projekt ist ein JavaScript-Spiel, das "Schere-Stein-Papier" simuliert. Der Benutzer kann eine der drei Optionen wählen, und das Spiel vergleicht die Wahl des Benutzers mit einer zufälligen Wahl des Computers. Das Ergebnis des Spiels (Gewinn, Verlust oder Unentschieden) wird angezeigt.

3) Einrichtung des Projekts

1) Verzeichnisstruktur:

```
static-code-analysis-js/
├── js/
│   ├── Schere-Stein-Papier.js
├── node_modules/
├── .eslintrc.json
├── package.json
├── package-lock.json
```

2) Installation von ESLint:

```bash
npm init -y
npm install eslint --save-dev
```

3. **Initialisierung von ESLint:

```bash
npx eslint --init
```

Auswahloptionen bei der Initialisierung:

- How would you like to use ESLint? **To check syntax, find problems, and enforce code style**

- What type of modules does your project use? **JavaScript modules (import/export)**
- Which framework does your project use? **None of these**
- Does your project use TypeScript? **No**
- Where does your code run? **Node**
- How would you like to define a style for your project? **Use a popular style guide**
- Which style guide do you want to follow? **Airbnb**
- What format do you want your config file to be in? **JSON**

4) **Codeanalyse mit ESLint**

1. Ursprünglicher Code:

```javascript
// Schere-Stein-Papier.js
const choices = ["rock", "paper", "scissors"];

let playerScore = 0;
let computerScore = 0;
let drawScore = 0;

function playGame(playerChoice) {
  const computerChoice = choices[Math.floor(Math.random() * 3)];
  let result = "";
  if (playerChoice === computerChoice) {
    result = "It's a draw!";
    drawScore++;
  } else if (
    (playerChoice === "rock" && computerChoice === "scissors") ||
    (playerChoice === "paper" && computerChoice === "rock") ||
    (playerChoice === "scissors" && computerChoice === "paper")
  ) {
    result = "You win!";
    playerScore++;
  } else {
    result = "You lose!";
    computerScore++;
  }

  console.log(
```

```
    `You chose ${playerChoice}, computer chose ${computerChoice}. ${result}`
  );
  console.log(
    `Player: ${playerScore} | Computer: ${computerScore} | Draws: ${drawScore}`
  );
}

function unusedFunction() {
  console.log("This function is never used");
}

let unusedVar = 42;

if ((playerScore = 10)) {
  console.log("Player reached 10 points");
}

let computerChoice = "rock";

console.log("This is a debugging statement");

function sayHello(name) {
  console.log("Hello, " + name);
  return;
  name = name + "!";
}

sayHello("World");

const obj = {
  prop1: "value1",
  prop2: "value2",
  prop3: "value3",
};

const { prop1, prop2 } = obj;

playGame("rock");
```

```
playGame("paper");
playGame("scissors");

playerChoice = 'rock';
var exampleVar = "This should be let or const";
console.log("This is a missing semicolon");

function anotherFunction() {
  console.log("This is another function");
}

const anotherVar = "Another unused variable";

if ((playerScore = 20)) {
  console.log("Player reached 20 points");
}
```

2) Ausführung von ESLint:

```bash
npx eslint Schere-Stein-Papier.js
```

3) Fehlerbehebung:

Fehler und Lösungen:

- Nicht verwendete Funktion `unusedFunction` und `anotherFunction` entfernt.
- Nicht verwendete Variablen `unusedVar`, `exampleVar`, `anotherVar`, und `computerChoice` entfernt.
- Ungültige Gleichheit in `if`-Bedingungen `if (playerScore = 10)` und `if (playerScore = 20)` korrigiert zu `===`.
- Unreachable code im `sayHello` entfernt.

4) Korrigierter Code:

```javascript
// Schere-Stein-Papier.js
const choices = ["rock", "paper", "scissors"];
```

```
let playerScore = 0;
let computerScore = 0;
let drawScore = 0;

function playGame(playerChoice) {
 const computerChoice = choices[Math.floor(Math.random() * 3)];
 let result = "";
 if (playerChoice === computerChoice) {
  result = "It's a draw!";
  drawScore++;
 } else if (
  (playerChoice === "rock" && computerChoice === "scissors") ||
  (playerChoice === "paper" && computerChoice === "rock") ||
  (playerChoice === "scissors" && computerChoice === "paper")
 ) {
  result = "You win!";
  playerScore++;
 } else {
  result = "You lose!";
  computerScore++;
 }

 console.log(
  `You chose ${playerChoice}, computer chose ${computerChoice}. ${result}`
 );
 console.log(
  `Player: ${playerScore} | Computer: ${computerScore} | Draws: ${drawScore}`
 );
}

function sayHello(name) {
 console.log("Hello, " + name);
 return;
}

sayHello("World");
```

```
  console.log("This is a debugging statement");

  if (playerScore === 10) {
   console.log("Player reached 10 points");
  }

  if (playerScore === 20) {
   console.log("Player reached 20 points");
  }

  // Funktion zum Testen des Spiels in der Konsole
  function testGame() {
   const choices = ["rock", "paper", "scissors"];
   for (let i = 0; i < 10; i++) {
    const playerChoice = choices[Math.floor(Math.random() * 3)];
    playGame(playerChoice);
   }
  }

  testGame();

  console.log("This is a missing semicolon");
```

5) Zusätzliche Testfunktion:
Die `testGame`-Funktion simuliert zehn Spielrunden mit zufälligen Spielerauswahlen.

6) Ausführung von Code:
```bash
node Schere-Stein-Papier.js
```

Ausgabe:
Hello, Anthony
This is a debugging statement
You chose rock, computer chose rock. It's a draw!
Player: 0 | Computer: 0 | Draws: 1
You chose paper, computer chose paper. It's a draw!
Player: 0 | Computer: 0 | Draws: 2

You chose scissors, computer chose rock. You lose!

Player: 0 | Computer: 1 | Draws: 2

You chose scissors, computer chose scissors. It's a draw!

Player: 0 | Computer: 1 | Draws: 3

You chose paper, computer chose scissors. You lose!

Player: 0 | Computer: 2 | Draws: 3

You chose scissors, computer chose scissors. It's a draw!

Player: 0 | Computer: 2 | Draws: 4

You chose scissors, computer chose scissors. It's a draw!

Player: 0 | Computer: 2 | Draws: 5

You chose rock, computer chose paper. You lose!

Player: 0 | Computer: 3 | Draws: 5

You chose rock, computer chose scissors. You win!

Player: 1 | Computer: 3 | Draws: 5

You chose rock, computer chose rock. It's a draw!

Player: 1 | Computer: 3 | Draws: 6

7) Abschluss:

Diese Dokumentation zeigt die Durchführung der statischen Codeanalyse mit ESLint, die Behebung der gefundenen Fehler und die Erstellung eines Testspiels zur Verifizierung der Korrekturen. Die Schritte sind klar beschrieben, und die entsprechenden Codeänderungen und Ergebnisse sind dokumentiert.