1. **Builds & Tooling:**
2. Grunt: Erklären Sie die Bestandteile folgender Gruntconfig.js Datei

'use strict'

module.exports = function(grunt) {

const config = {

app: 'app',

dist: 'dist'

}

grunt.initConfig({

pkg: grunt.file.readJSON('package.json'),

config: config,

browserify: {

dist: {

files: {

'<%= config.dist %>/scripts/app.js': ['<%= config.app %>/scripts/\*\*/\*.js']

},

options: {

browserifyOptions: {

debug: true

},

transform: [

'hbsfy',

'babelify',

'uglifyify'

]

}

}

},

clean: {

dist: ['<%= config.dist %>']

},

eslint: {

target: ['<%= config.app %>/scripts/\*.js']

},

grunt.registerTask('default', [

'eslint',

'clean',

'browserify'

]

)

}

1. Projekt: Bauen Sie eine Projektstruktur auf und erklären Sie Zweck und Aufgabe der einzelnen Komponenten
2. **GIT:**
3. Nennen Sie einige Vorteile von GIT als Versionskontrollsystem.
4. Was verstehen Sie unter nicht linearer Entwicklung?
5. Welche GIT Kommandos kennen Sie und was kann man damit machen (min. 3 Kommandos, max. 5)?
6. Was ist ein Merge Konflikt und wie geht man damit um?
7. Erklären Sie den 3-Wege-Merge anhand einer Darstellung.
8. **Refactoring:**
9. Welchen Autor würden Sie nachschlagen wenn Sie mehr über Refactoring lernen möchten?
10. Was ist die ultimative Voraussetzung um ein Refactoring durchzuführen?
11. Suchen Sie sich ein Refactoring aus, erklären Sie dieses anhand der Struktur die Fowler in seinem „Catalog of Refactorings“ gewählt hat.
12. Geben Sie ein Codebeispiel für das von Ihnen gewählte Refactoring. Schreiben Sie auf einem Blatt Papier - links den ursprünglichen Code und rechts das Refactoring (Programmiersprache nach Wahl)