Тема 8

1.Работа с дебъгер

Описва предназначението и работата с дебъгер

Дебъгерът е инструмент за разработка на софтуер, който позволява на програмистите да анализират и отстраняват грешки в своя код.

Предназначението на дебъгера е да помогне на програмиста да намери и разбере проблемите, които пречат на програмата да функционира правилно. Това включва откриване на грешки в кода, проследяване на стойности на променливи и изпълнение на код ред по ред, за да се види къде точно в програмата се получава грешката.

Работата с дебъгера включва стъпки като поставяне на точки за прекъсване (breakpoints) в кода, за да се спре изпълнението на програмата в определени точки. При стартиране на програмата, дебъгерът ще спре изпълнението на кода на точката за прекъсване и ще позволи на програмиста да прегледа стойностите на променливите, изпълнението на кода, стека на извиканите функции и други полезни данни.

Дебъгерите могат да предоставят и други функционалности, като следене на стойностите на променливи в реално време, стъпка по стъпка изпълнение на програмата, преглед на стека на извиканите функции, използване на условни точки за прекъсване, проследяване на събития в приложенията и др.

2. Рефакториране на зле написан код.

Коректно рефакторира и модифицира зле написан код, който нарушава добрите практики и стилови ръководства

Когато рефакторирате и модифицирате зле написан код, който нарушава добрите практики и стилови ръководства, следвайте следните стъпки:

Прочетете и разберете кода: Прочетете кода и разберете какво прави, за да можете да идентифицирате проблемите и да ги поправите.

Идентифицирайте проблемите: Идентифицирайте всички проблеми в кода, като търсите нарушения на добрите практики и стилови ръководства, които следвате.

Поправете проблемите: Използвайте добрите практики и стилове, които следвате, за да поправите проблемите в кода. Това може да включва пренаписване на кода, премахване на ненужен код, добавяне на коментари и т.н.

Тествайте промените: Тествайте новия код, за да се уверите, че промените не са довели до нови проблеми или нарушения на добрите практики.

Документирайте промените: Документирайте промените, които сте направили, за да може бъдещи разработчици да разберат какво сте направили и защо.

Съветвайте се с колегите: Покажете новия код на колегите си и поискайте техните мнения. Те могат да ви дадат полезни съвети за това как да подобрите кода още повече.

Повторете процеса: Ако има още проблеми в кода, повторете процеса, за да ги поправите.

Следвайки тези стъпки, можете да рефакторирате и модифицирате зле написан код, който нарушава добрите практики и стилови ръководства, и да го направите по-четим, по-лесен за поддръжка и по-ефективен.

3. За да създадем компонентни тестове, трябва да изберем компонентите, които искаме да тестваме. Обикновено компонентите са функции или класове, които извършват определени операции и имат входни и изходни параметри.

Една от най-популярните библиотеки за създаване на тестове в JavaScript е Jest. Ще използваме Jest за нашите компонентни тестове.Създаване на компонентни тестове

За да създадем компонентни тестове, трябва да изберем компонентите, които искаме да тестваме. Обикновено компонентите са функции или класове, които извършват определени операции и имат входни и изходни параметри.

Първоначално използваме функцията require(), за да импортираме функцията add, която ще тестваме.

След това използваме функцията test(), за да опишем тестовете, които ще извършим. Първият аргумент е описание на теста, а вторият е функция, която съдържа кода на теста.

В кода на теста използваме функцията expect() за да проверим дали функцията add връща правилния резултат. Използваме метода toBe(), за да сравним резултата с очаквания резултат. Използваме метода toBeCloseTo() за да сравним резултата с точност до определен брой знаци след десетичната запетая.

Тези тестове ще се изпълнят, като използвате командата „jest“ в терминала в директорията на проекта. Ще видите, че всички тестове са минали успешно.

Това е един много прост пример за компонентни тестове. В реален проект може да има много по-сложни функции и класове.

4. IDE vs Text Editor

Дефинира и различава понятията интегрирана среда за разработка (IDE) и текстов редактор.

Интегрирана среда за разработка (IDE) и текстов редактор са два различни инструмента за разработка на софтуер, които имат свои характеристики и предназначения.

IDE е платформа за разработка на софтуер, която предоставя на разработчиците цялостна среда за разработка на приложения. Това включва функционалности като средства за управление на изходния код, инструменти за дебъгване, интегрирани компилатори и средства за тестване. IDE обикновено има графичен потребителски интерфейс (GUI), който позволява на разработчиците да работят с различни файлови формати и програмни езици.

От друга страна, текстовият редактор е инструмент за редактиране на текстови файлове. Текстовият редактор не предлага интегрирани средства за управление на изходния код или дебъгване, но предоставя прост и лесен начин за редактиране на кода. Текстовите редактори могат да бъдат по-леки и бързи от IDE, което ги прави предпочитани за някои програмисти, особено за по-малки проекти.

Основната разлика между IDE и текстов редактор е, че IDE е цялостна среда за разработка, която предоставя на програмистите интегрирани функционалности за разработка на приложения, докато текстовият редактор предоставя основни функционалности за редактиране на текст. IDE може да бъде по-мощен и по-функционален, но текстовият редактор е по-лек и по-бърз за малки проекти.

5.IDE Keyboard Shortcuts.

Описва и демонстрира инструменти и клавишни комбинации за работа с код

(бързо навигиране в кода, намиране на срещания и замяна, извличане – extract

На методи и класове и др.) в интегрирана среда за разработка и текстов редак-

Интегрираните среди за разработка (IDE) и текстови редактори предоставят множество инструменти и клавишни комбинации, които могат да ускорят работата върху кода. Някои от тези инструменти и клавишни комбинации варират в зависимост от избрания инструмент или редактор, но общите принципи и функционалности са подобни.

Ето някои от най-полезните инструменти и клавишни комбинации за работа с код:

Бързо навигиране в кода:

Ctrl + F (или Cmd + F за Mac): търсене на текст в текущия файл

Ctrl + Shift + F (или Cmd + Shift + F за Mac): търсене на текст във всички файлове в проекта

Ctrl + G (или Cmd + G за Mac): преминаване към определен ред в текущия файл

Ctrl + Shift + G (или Cmd + Shift + G за Mac): преминаване към определен ред в друг файл

Намиране на срещания и замяна:

Ctrl + D (или Cmd + D за Mac): избиране на следващото съвпадение на текущо маркираното изразение

Ctrl + Shift + D (или Cmd + Shift + D за Mac): избиране на всички съвпадения на текущо маркираното изразение

Ctrl + R (или Cmd + R за Mac): замяна на текущо маркираното изразение с друго

Ctrl + Shift + R (или Cmd + Shift + R за Mac): замяна на всички съвпадения на текущо маркираното изразение с друго

Извличане (Extract) на методи и класове:

Ctrl + Alt + M (или Cmd + Option + M за Mac): извличане на маркираното блокче код в нов метод

Ctrl + Alt + V (или Cmd + Option + V за Mac): извличане на маркираното блокче код в нова променлива

Ctrl + Alt + F (или Cmd + Option + F за Mac): извличане на маркираното блокче код в ново поле

Ctrl + Alt + C (или Cmd + Option + C за Mac): извличане на маркираното блокче код в нов клас

Други полезни клавишни комбинации:

Ctrl + Space (или Ctrl + Shift + Space за Mac): автоматично довършване на кода

Ctrl + Shift + T (или Cmd + Shift + T за Mac): отваряне на клас или метод по име.

6.Инструментите за работа със сорс-контрол система (Source Control Management tools) в интегрираните среди за разработка (IDE) и текстови редактори предоставят удобен начин за управление на кода и съхранението му във версионен контрол. Някои от най-популярните сорс-контрол системи включват Git, SVN и Mercurial.

Ето някои от инструментите и примерите за тяхната употреба в IDE и текстови редактори:

Git:

Интегриране на Git във Visual Studio Code: Въвеждане на Git във Visual Studio Code позволява на разработчиците да добавят, изтриват, модифицират и споделят код в реално време. В VS Code, инструментът за Git може да бъде намерен вляво от редактора. От там може да се използват всички Git команди като commit, push, pull, и много други.

Интегриране на Git в IntelliJ IDEA: IntelliJ IDEA предоставя мощен инструментариум за работа с Git, който включва удобна графична обвивка на Git-а. Разработчиците могат да работят с Git през контекстно меню или чрез специални клавишни комбинации, за да направят commit, push, pull, и много други.

SVN:

Интегриране на SVN в Eclipse: Eclipse предоставя интегриран плъгин за SVN, който позволява на разработчиците да работят със SVN през интерфейса на Eclipse. Разработчиците могат да направят checkout на проект от SVN, да го редактират и да направят commit на промените.

Интегриране на SVN в Visual Studio: Във Visual Studio може да се използва интегриран плъгин за SVN. Разработчиците могат да правят checkout на проект, да го редактират и да направят commit на промените.

Mercurial:

Интегриране на Mercurial в PyCharm: PyCharm предоставя интегриран плъгин за Mercurial, който позволява на разработчиците да работят с Mercurial през интерфейса на PyCharm. Разработчиците могат да правят checkout на проект, да го редактират и да направят commit на промените.

Интегриране на Mercurial в Visual Studio Code: Във Visual Studio Code може да се използва

Някои от другите инструменти за работа със сорс-контрол система в интегрираните среди за разработка и текстови редактори включват:

Sublime Text:

Интегриране на Git и SVN: Sublime Text предоставя интегриран пакет от плъгини, който позволява на разработчиците да работят с Git и SVN през интерфейса на редактора. Разработчиците могат да направят commit, push, pull и много други.

Atom:

Интегриране на Git: Atom предоставя интегриран плъгин за Git, който позволява на разработчиците да работят с Git през интерфейса на редактора. Разработчиците могат да направят commit, push, pull и много други.

Notepad++:

Интегриране на Git и SVN: Notepad++ предоставя интегриран пакет от плъгини, който позволява на разработчиците да работят с Git и SVN през интерфейса на редактора. Разработчиците могат да направят commit, push, pull и много други.

Всички тези инструменти предоставят визуални интерфейси и клавишни комбинации за работа със сорс-контрол системи. Разработчиците могат да следят промените в кода, да добавят, изтриват и модифицират файлове, да направят commit и push на промените си и да работят с различни клонове и тагове във версионен контрол. Това позволява на разработчиците да се концентрират върху разработката на кода си и да управляват ефективно процеса на разработка.