



DevOps Build Tools



Building Competence. Crossing Borders.

Build

Einführung in Build-Tools

Dependency Management

Beispiel: Gradle – ein Java-Build Tool

Beispiel: Node/NPM – Node Package Manager für JavaScript

Was ist ein Build Tool?

Ein Build-Tool ist eine Software die die Erstellung von ausführbaren Applikationen aus dem Quellcode automatisiert (z.B. für eine Android App, für eine Web-Anwendung auf dem Server).

Aufgaben

- Quellcode kompilieren (sofern notwendig)
- Dateien für das Ziel-Betriebssystem erstellen
- Externe Dateien (Libraries, Bilder, ...) hinzufügen
- kann weitere Schritte übernehmen: Tests, Qualitätskontrolle, ...

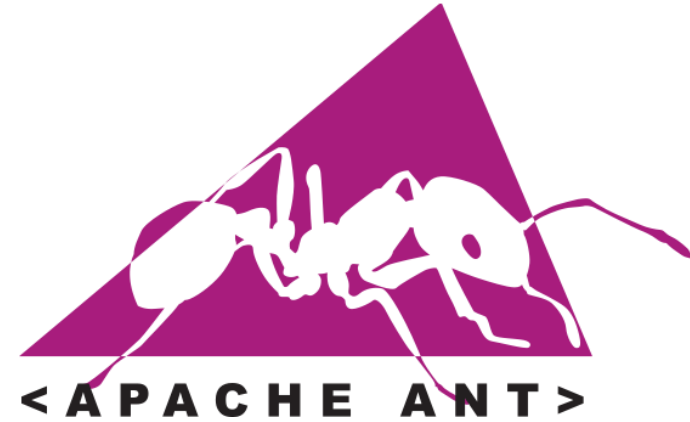
Automatisierung

Grundsätzlich könnte alles von Hand gemacht werden. Mit DevOps muss dieser Prozess aber zwingend **automatisiert** werden, so dass er effizient und beliebig oft wiederholt werden kann.

Build Tools

Maven™

PyB



sbt

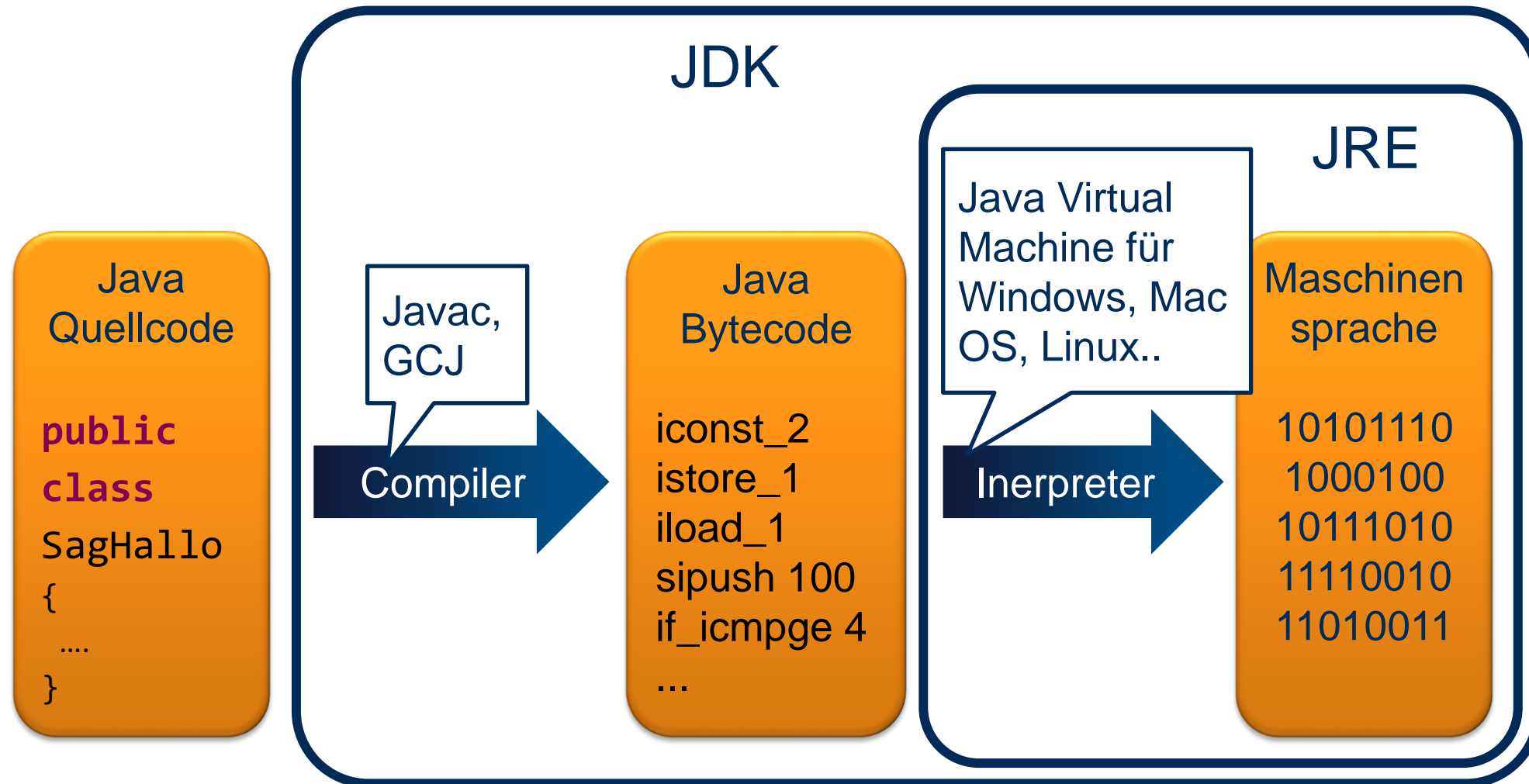


make (C, Linux, Unix)



Warum braucht es ein Build-Tool?

Java: write once, run anywhere.



- Java Compiler
- Erstellt aus .java Dateien .class Dateien

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
Verzeichnis: C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> java Main.java
Hello World
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> javac Main.java
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> dir

Verzeichnis: C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava

Mode                LastWriteTime         Length Name
----                -
d-----          06.01.2021    16:14             .gradle
d-----          06.01.2021    16:22             .settings
d-----          06.01.2021    14:18             .vscode
-a----          06.01.2021    16:22           464 .project
-a----          06.01.2021    14:18           42 build.gradle
-a----          07.01.2021    09:37          413 Main.class
-a----          06.01.2021    14:18          127 Main.java
-a----          06.01.2021    14:18          738 Test.jar

PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava>
```

Java Archiver

- Erstellt aus (mehreren) .class Dateien ein Archiv, welches ausgeführt werden kann
- Das JAR kann danach gestartet werden, ohne dass der Quellcode weitergegeben werden muss bzw. verändert werden kann



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  1: powershell  +  [ ]  [X]  ^  X

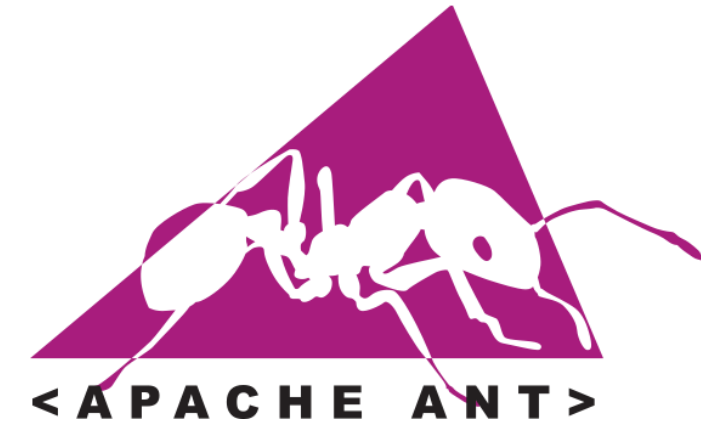
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> jar cf Test.jar *.class
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> java -classpath Test.jar Main
Hello World
PS C:\mosa\dev\vscode\projects\DevOps\02 DevOpsJava> 
```

Unser erster Build 😊

- Wird kaum «von Hand» gemacht, sondern automatisiert
- Auch für Sprachen mit Interpreter (z.B. JavaScript, Code-Minimierung bzw. Code-Obfuscation)

Apache ANT

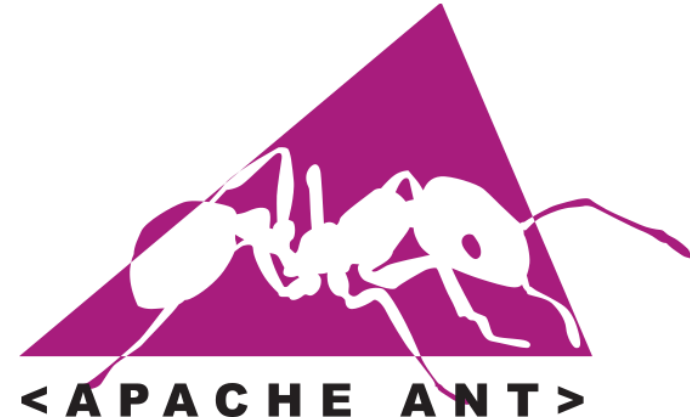
- Apache ANT war eines der ersten Build Tools



```
<target name="jar">
  <mkdir dir="build/jar"/>
  <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
    <manifest>
      <attribute name="Main-Class" value="oata.HelloWorld"/>
    </manifest>
  </jar>
</target>
```

Apache ANT Nachteile

- Zu flexibel
- Jeder kann nach seiner eigenen Struktur arbeiten
- Schwierige Einarbeitung in gewachsene Projekte
- Viele Zeilen Code (XML) notwendig



Neue Ideen

- Einfacher Setup
- Best Practices
- Grundsatz: **Convention over Configuration**
- Dependency Management



Dependency Management

- In der Praxis ist jede Software auf **Libraries** angewiesen
- Maven kann diese automatisch in der richtigen Version aus dem Internet laden
- Zudem kann Maven **transitive Dependencies** (Libraries, welche eine verwendete Library benötigt) automatisch ebenfalls laden

Apache Maven

POM

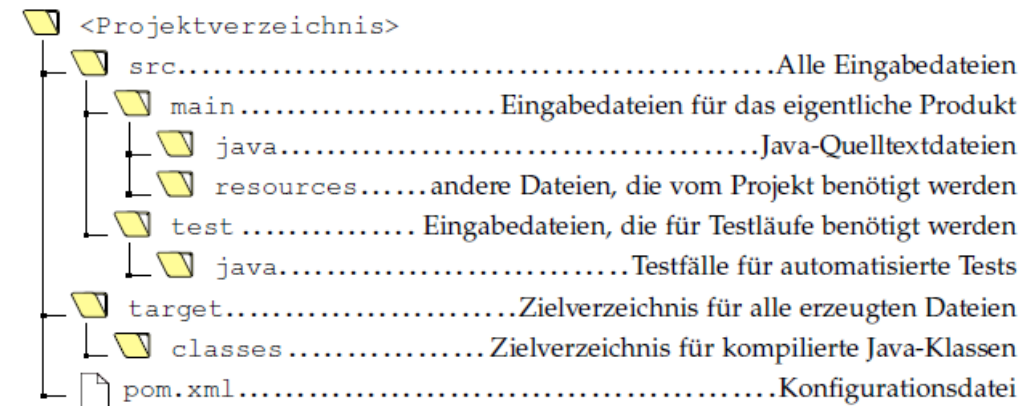
- Project Object Model
- Hierarchisch

Konventionen

- Standard-Projektstruktur

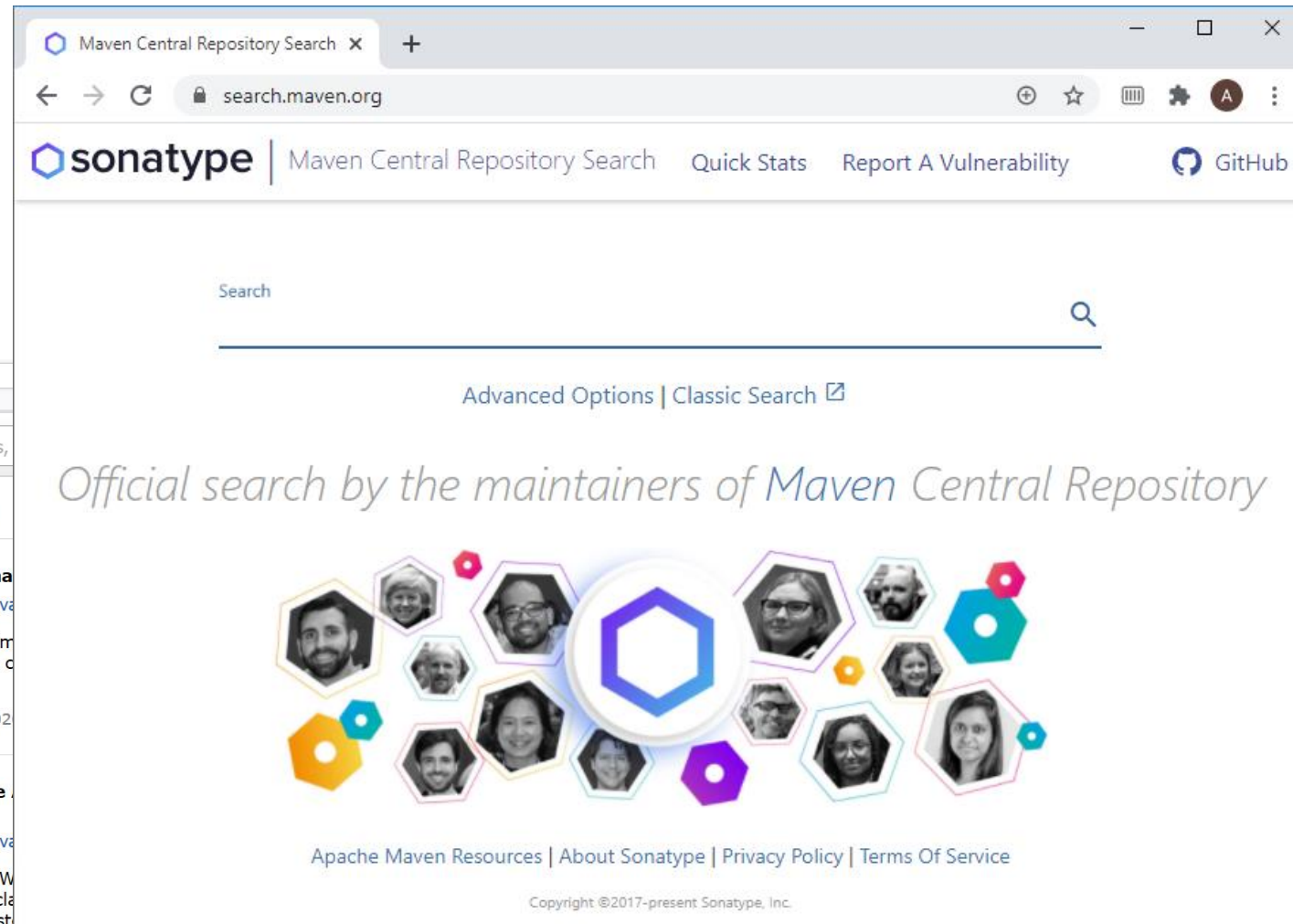
```
<project>
<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany.app</groupId>
<artifactId>my-module</artifactId>
<version>1</version>
</project>
```



Maven Repository / Maven Search

<https://search.maven.org/>



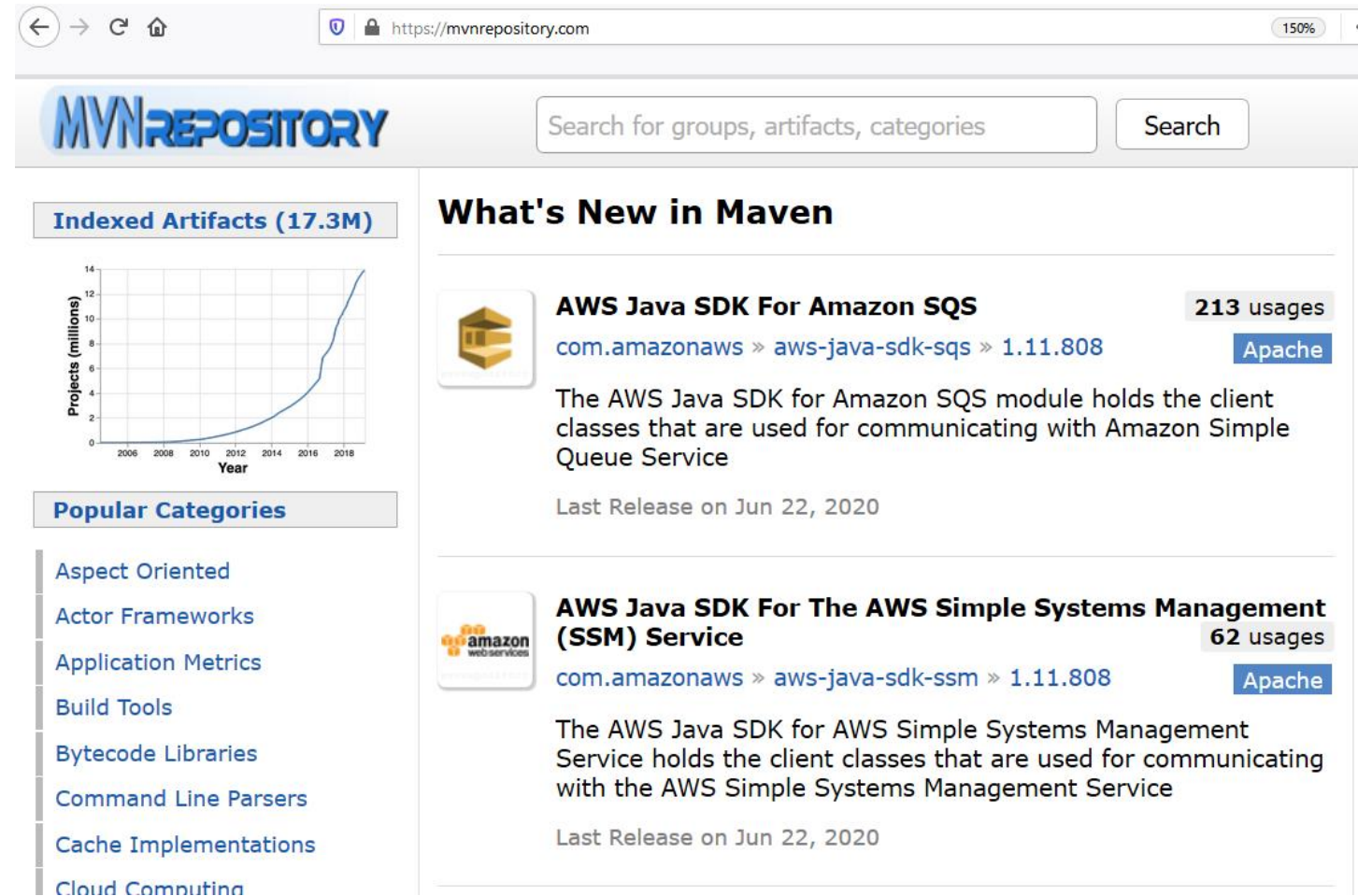
Maven Repository

Repositories

- Maven Central Repository
- praktisch alle Java-Libraries in allen Versionen verfügbar
- Für Closed-Source kann firmenintern ein Repository mit privatem Libraries betrieben werden

	Version	Repository	Usages	Date
4.13.x	4.13	Central	3,671	Jan, 2020
	4.13-rc-2	Central	55	Dec, 2019
	4.13-rc-1	Central	53	Oct, 2019
	4.13-beta-3	Central	270	May, 2019
	4.13-beta-2	Central	118	Feb, 2019
	4.13-beta-1	Central	67	Nov, 2018
4.12.x	4.12	Central	51,505	Dec, 2014
	4.12-beta-3	Central	30	Nov, 2014
	4.12-beta-2	Central	31	Sep, 2014
	4.12-beta-1	Central	31	Jul, 2014
4.11.x	4.11	Central	25,212	Nov, 2012
	4.11-beta-1	Central	23	Oct, 2012
4.10.x	4.10	Central	9,169	Sep, 2011
4.9.x	4.9	Central	1,243	Aug, 2011
4.8.x	4.8.2	Central	6,020	Oct, 2010
	4.8.1	Central	5,430	Feb, 2010

Maven Repository



The screenshot shows the Maven Repository website. The header includes the Maven Repository logo, a search bar with the placeholder text "Search for groups, artifacts, categories", and a search button. The left sidebar contains two sections: "Indexed Artifacts (17.3M)" with a line graph showing growth from 2006 to 2018, and "Popular Categories" with a list of categories. The main content area is titled "What's New in Maven" and lists two new releases: "AWS Java SDK For Amazon SQS" and "AWS Java SDK For The AWS Simple Systems Management (SSM) Service".

Indexed Artifacts (17.3M)

Projects (millions)

Year

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing

What's New in Maven

AWS Java SDK For Amazon SQS 213 usages
com.amazonaws » aws-java-sdk-sqs » 1.11.808 Apache

The AWS Java SDK for Amazon SQS module holds the client classes that are used for communicating with Amazon Simple Queue Service

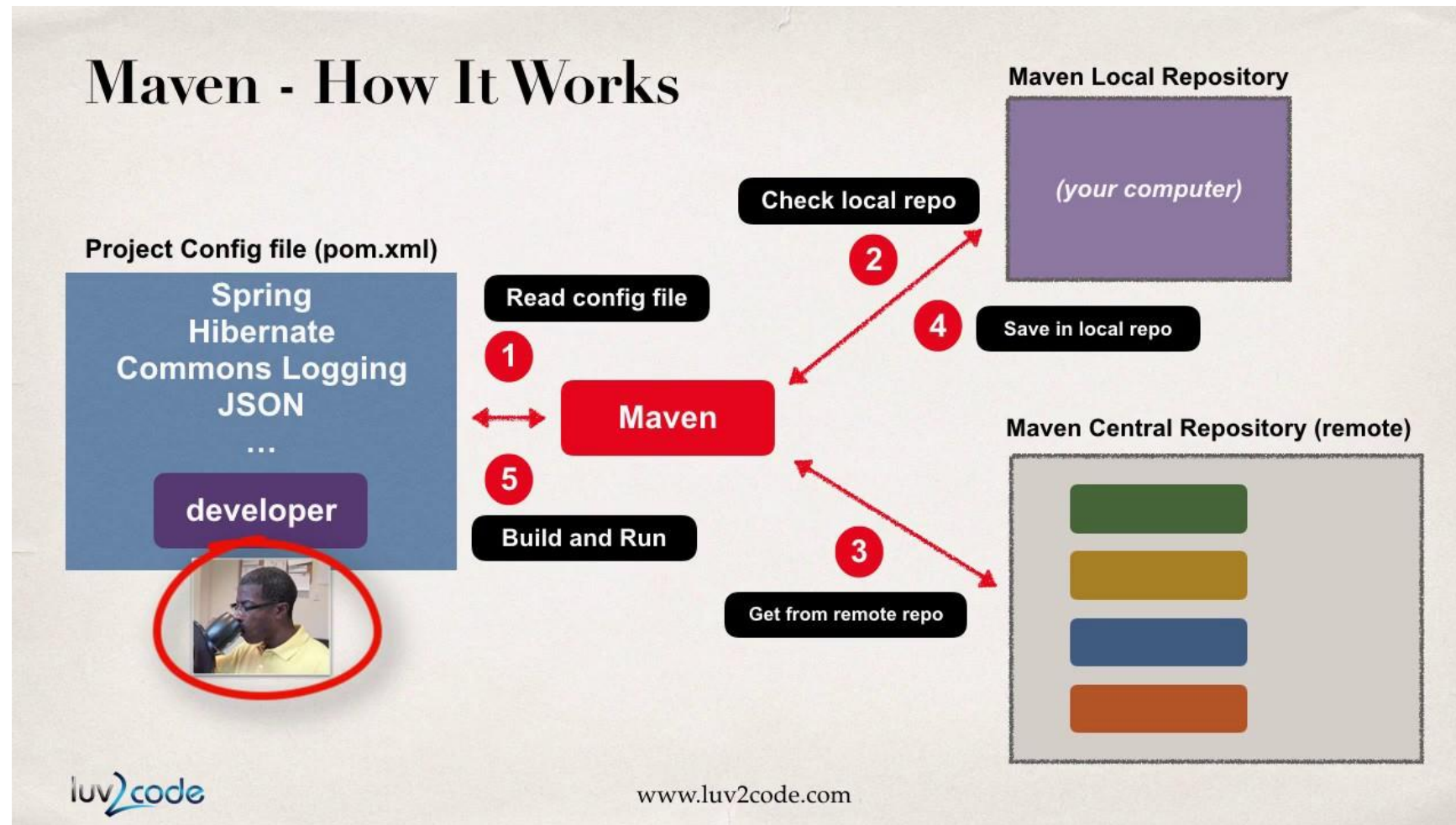
Last Release on Jun 22, 2020

AWS Java SDK For The AWS Simple Systems Management (SSM) Service 62 usages
com.amazonaws » aws-java-sdk-ssm » 1.11.808 Apache

The AWS Java SDK for AWS Simple Systems Management Service holds the client classes that are used for communicating with the AWS Simple Systems Management Service

Last Release on Jun 22, 2020

Maven Dependency Management



https://www.youtube.com/watch?v=m_eWc9pjyg4

Gradle



Gradle ist ein Build-Tool

Viele Ideen von Maven übernommen

Basiert statt auf XML auf der Programmiersprache «Groovy»

Apache Maven / Gradle

- Die XML von Maven werden sehr lang
- Maven löst 90% der Fälle gut, den Rest aber mit viel Aufwand
- Gradle übernimmt die Ideen von Maven, ersetzt aber XML durch Groovy

The image displays a side-by-side comparison of build configuration files for Maven and Gradle. On the left, a Maven `pom.xml` file is shown, characterized by its verbose XML structure with numerous tags for project information, dependencies, and build phases. On the right, a Gradle `build.gradle` file is shown, which is significantly more concise and uses a Groovy-based DSL. Below the Maven XML, the 'maven' logo is visible. Below the Gradle file, the 'Gradle' logo is present. In the bottom right corner, the text 'Polyglot maven' is displayed. At the bottom center, the phrase 'Convention Over Configuration' is written in a large, stylized font. In the bottom right corner, there is a reference to 'Tomek's Blog' with the URL <http://kaczanowscy.pl/tomek>. The Apache Ant logo is also visible at the bottom left of the Maven XML section.

Gradle Konzepte

Bedienung

Über Console / Terminal oder über Visual Studio Code Plugins (User Interface).

Projekt

Ein Projekt ist z.B. eine Java-Applikation, aus der ein JAR-File gebaut werden muss und das auf einen Server deployed wird

Task

Diese Projekt-Schritte sind Tasks. Ein Projekt besteht aus mehreren Tasks.

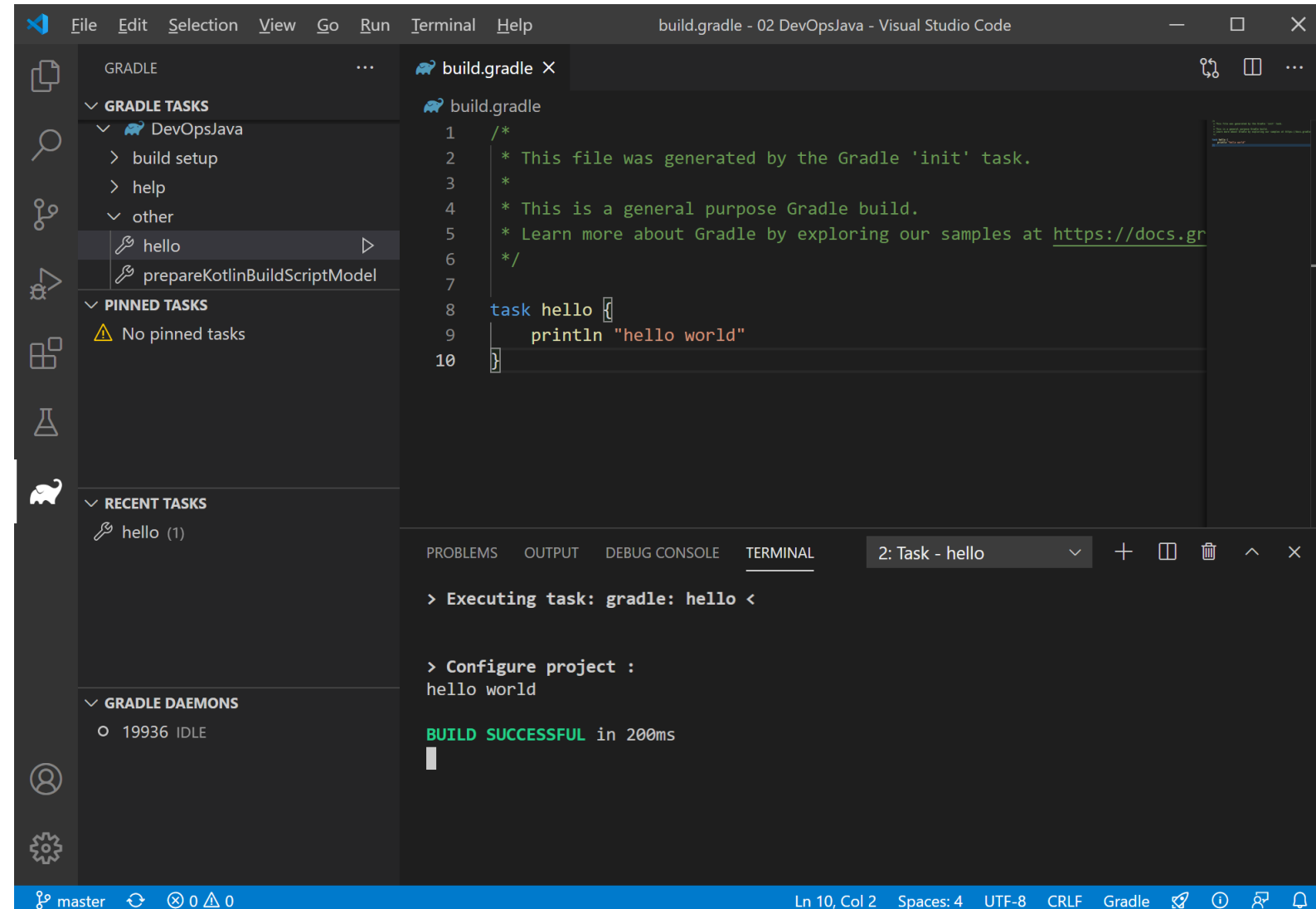
Gradle Daemon

Der Daemon ist ein Hintergrund-Prozess, der den Build jeweils ausführt. So muss nicht immer ein neuer Prozess gestartet werden.

Gradle Hello World

«Hello World» auf
der Konsole / im
Terminal
ausgeben

```
task hello {  
    println "hello world"  
}
```



Neues Gradle Projekt anlegen

Neues Projekt anlegen

- Neuen Ordner erstellen, in Visual Studio Code öffnen
- Terminal öffnen
- **gradle init** ausführen

Gradle initialisieren

- Application (Typ)
- Java (Implementation Language)
- one Application Project
- Groovy (Build Script DSL)
- JUnit Jupiter (Test Framework)

Test mit «**gradle run**». Damit VS Code Gradle erkennt, Ordner neu öffnen, anschliessend wird Java-Projekt importiert.

gradle init: Beispiel

```
PS C:\...test> gradle init
```

```
Select type of project to generate:
```

- 1: basic
- 2: application
- 3: library
- 4: Gradle plugin

```
Enter selection (default: basic) [1..4] 2
```

```
Select implementation language:
```

- 1: C++
- 2: Groovy
- 3: Java
- 4: Kotlin
- 5: Scala
- 6: Swift

```
Enter selection (default: Java) [1..6] 3
```

```
Generate multiple subprojects for application? (default:  
no) [yes,no]
```

```
Select build script DSL:
```

- 1: Kotlin
- 2: Groovy

```
Enter selection (default: Kotlin) [1..2] 2
```

```
Select test framework:
```

- 1: JUnit 4
- 2: TestNG
- 3: Spock
- 4: JUnit Jupiter

```
Enter selection (default: JUnit Jupiter) [1..4] 4
```

```
Project name (default: test):
```

```
Source package (default: test):
```

```
Enter target version of Java (min. 7) (default: 21):
```

```
Generate build using new APIs and behavior (some features  
may change in the next minor  
release)? (default: no) [yes, no]
```

Um den Default-Wert zu wählen, kann «Enter» eingegeben werden

Resultat: Neues Gradle Projekt

Projektstruktur: init erstellt build.gradle im Ordner «app»

Gradle Plugins

- Java Application

Repositories

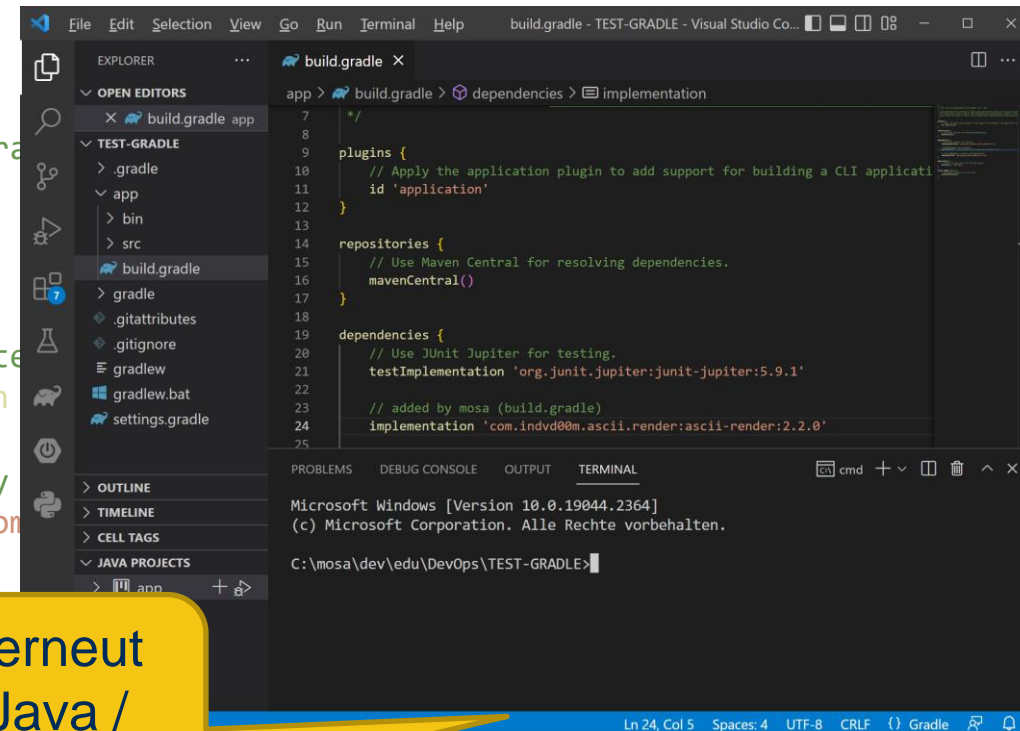
- mavenCentral

Dependencies

- Guava
- JUnit

Main-Klasse

```
plugins {  
    // Apply the application plugin to add support for building a CLI app in  
    // Java.  
    id 'application'  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    // Use JUnit Jupiter for testing.  
    testImplementation 'org.junit.jupiter:junit-jupiter:5.9.1'  
  
    // This dependency is added by the application plugin.  
    implementation 'com.indvd0m.ascii.render:ascii-render:2.2.0'  
}
```



In VS Code erneut
öffnen und Java /
Gradle abwarten.

Verwendung einer Library

Ziel

Es wird eine ASCII-Ausgabe gewünscht.

Eine passende Library wurde gefunden: <https://github.com/indvd00m/java-ascii-render>

PseudoText

Antialiasing option is customizable, enabled by default.



```
IRender render = new Render();
IContextBuilder builder = render.newBuilder();
builder.width(120).height(20);
builder.element(new PseudoText("PseudoText"));
ICanvas canvas = render.render(builder.build());
String s = canvas.getText();
System.out.println(s);
```


Gradle: Neue Dependency hinzufügen

Dependency-Bezeichnung suchen und finden

auffindbar über <https://search.maven.org>

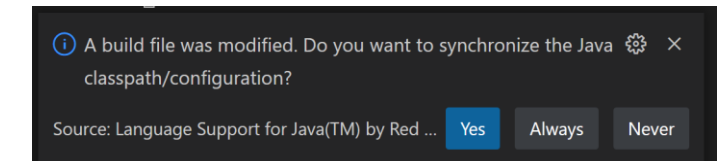
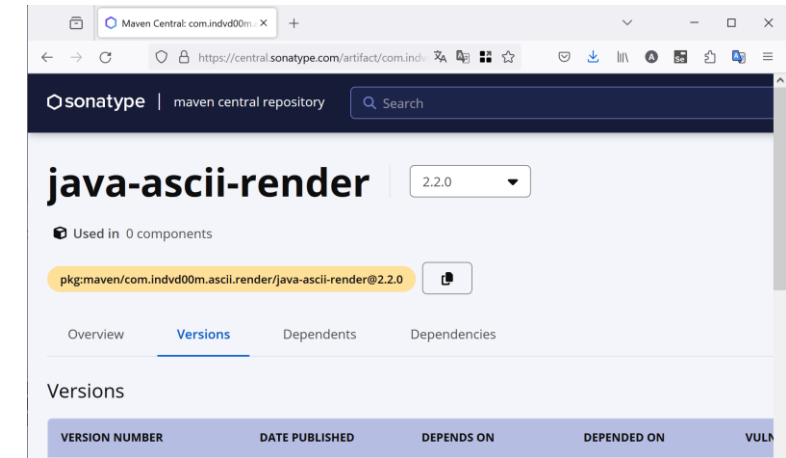
Gruppe:ID:Version

gradle.build anpassen

```
// added by mosa (build.gradle)
implementation 'com.indvd00m.ascii.render:ascii-render:2.2.0'
```

Visual Studio Code

Die neue Dependency sollte automatisch geladen werden. **Wenn nicht:** Ctrl-Shift-P: Java: Clean Language Server Workspace und/oder gradle build ausführen.



Achtung: Copy/Paste von Code aus PDF kann zu Fehlern führen!

Java: Code erweitern

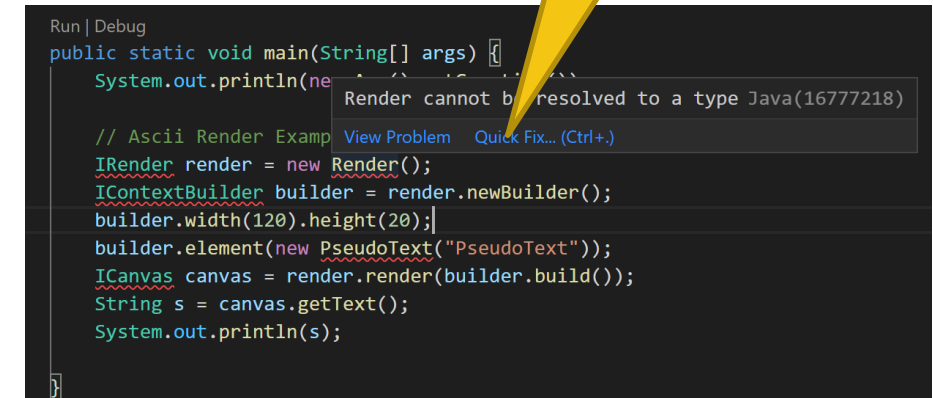
java-ascii-render: Beispiele

Beispiel-Code von <https://github.com/indvd00m/java-ascii-render#pseudotext>

```
// Ascii Render Example
IRender render = new Render();
IContextBuilder builder = render.newBuilder();
builder.width(width:120).height(height:20);
builder.element(new PseudoText(text:"DevOps"));
ICanvas canvas = render.render(builder.build());
String s = canvas.getText();
System.out.println(s);
```

Nach App.java kopieren,
Imports bereinigen (Quick Fix),
Ausführen (siehe auch nächste Folie)

QuickFix

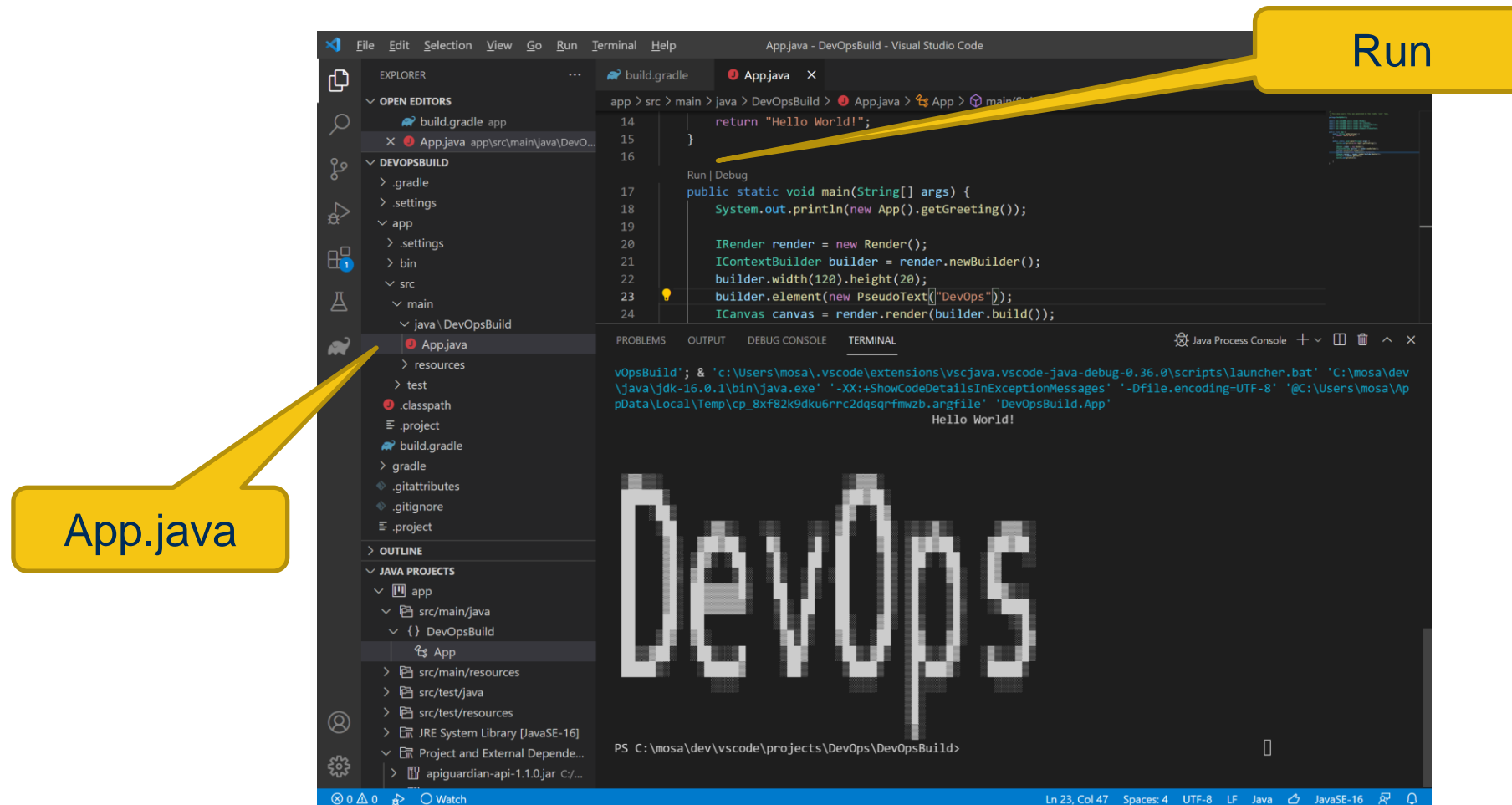


The screenshot shows a code editor with the following Java code:

```
public static void main(String[] args) {
    System.out.println(ne
// Ascii Render Examp
IRender render = new Render();
IContextBuilder builder = render.newBuilder();
builder.width(120).height(20);
builder.element(new PseudoText("PseudoText"));
ICanvas canvas = render.render(builder.build());
String s = canvas.getText();
System.out.println(s);
```

A compiler error is visible: "Render cannot be resolved to a type Java(16777218)". A yellow callout bubble labeled "QuickFix" points to the error.

Java: Code erweitern



Weiteres Beispiel: PDF mit pdfbox

The image shows two browser windows. The left window displays the Apache PDFBox Library page on <https://pdfbox.apache.org>. The right window shows the Maven Central repository page for **pdfbox** at <https://central.sonatype.com/org.apache.pdfbox>. The version 3.0.1 is selected. A snippet of the Gradle dependency is highlighted and copied to the clipboard:

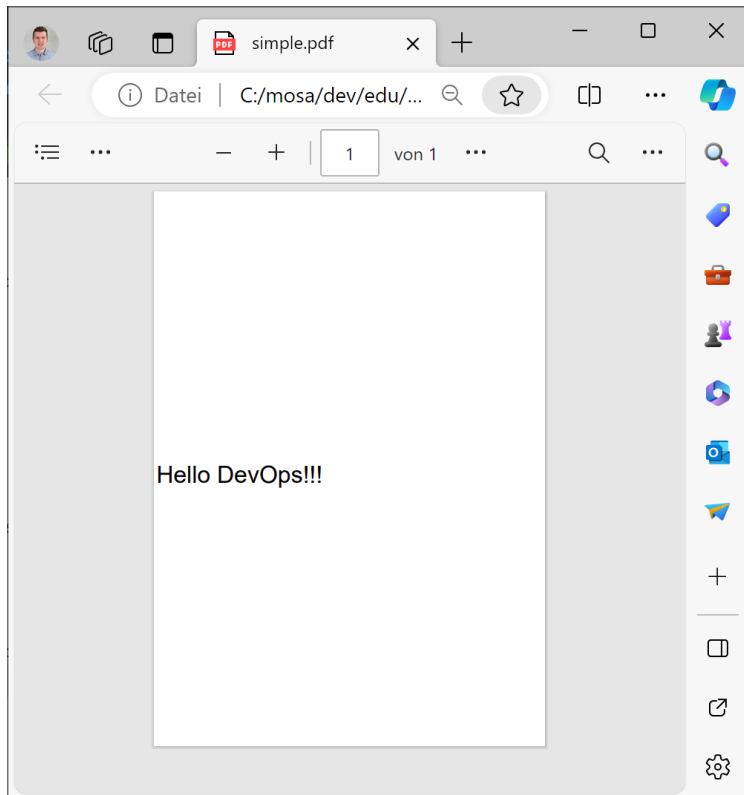
```
implementation 'org.apache.pdfbox:pdfbox:3.0.1'
```

Library suchen, Gradle-Zeile kopieren, in build.gradle einfügen

Beispiel-Code suchen: https://carbonrider.github.io/pdfbox_tutorial/introduction.html

pdfbox verwenden

Achtung: Beispiele können sich auf alte Version beziehen, fehlerhaft sein, ...
Imports ergänzen, Exceptions hinzufügen, ...



```
// PDF Box
PDDocument helloPdf = new PDDocument();
PDPage page = new PDPage(PDRectangle.A4);
helloPdf.addPage(page);

PDPageContentStream contentStream = new PDPageContentStream(helloPdf, page);
contentStream.beginText();
contentStream.setFont(new PDType1Font(Standard14Fonts.FontName.HELVETICA), 36);
contentStream.newLineAtOffset(5, 400);
contentStream.showText("Hello DevOps!!!");
contentStream.endText();
contentStream.close();

helloPdf.save(new File("simple.pdf"));
helloPdf.close();
```

NPM

Grundlagen: HTML und JavaScript

NPM – ein Build und Dependency Manager für JavaScript

Node.js – Laufzeitumgebung für JavaScript

Aufbau eines HTML-Dokuments

Um den Inhalt einer Webseite zu gestalten, wird HTML verwendet.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Ueli</title>
</head>
<body>
  <h2>Homepage von Ueli</h2>
  <p>Hallo, das ist meine Homepage.</p>
</body>
</html>
```

Zwischen <body> und
</body> kommt der
Inhalt der Webseite.



Bin jetzt im
Internet?

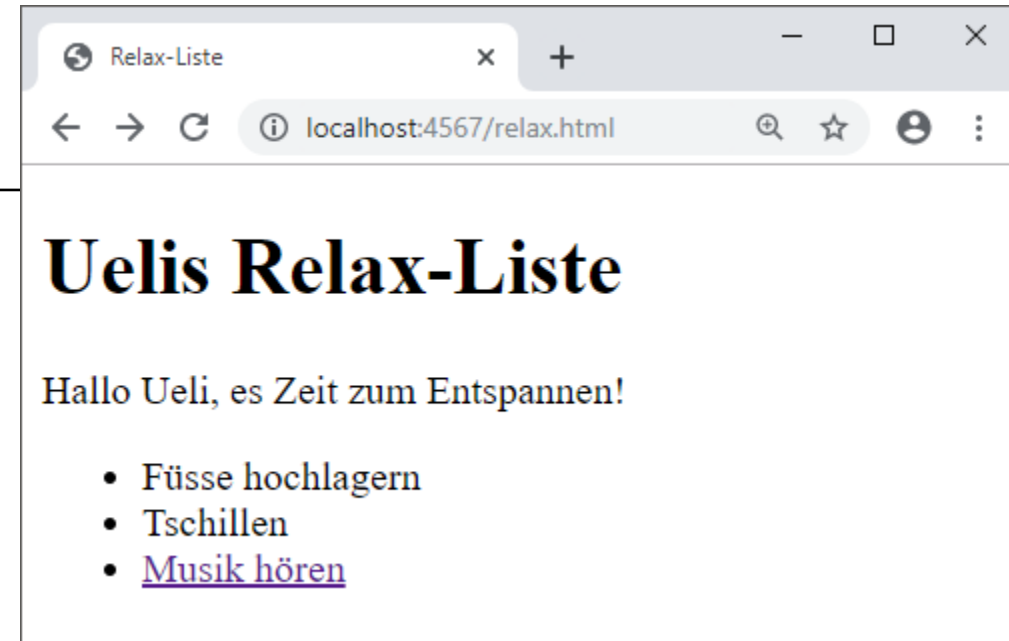


HTML Elemente

HTML	Beschreibung
<code><h1>Titel</h1></code>	Erste Überschrift
<code><h2>Untertitel</h2></code>	Zweite Überschrift
<code><p>Text Text..</p></code>	Paragraph
<code>
</code>	Zeilenumbruch
<code>Mein Link</code>	Link
<code> Warte Luege </code>	Aufzählung
<code><table> <tr><th>Tag</th><th>Likes</th></tr> <tr><td>Montag</td><td>23</td></tr> <tr><td>Dienstag</td><td>1024</td></tr> </table></code>	Tabelle

HTML Beispieldokument

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Relax-Liste</title>
</head>
<body>
  <h1>Uelis Relax-Liste</h1>
  <p>Hallo Ueli, es Zeit zum Entspannen!</p>
  <ul>
    <li>Füsse hochlagern</li>
    <li>Tschillen</li>
    <li><a href="http://www.radioswisspop.ch/de/webplayer">Musik hören</a></li>
  </ul>
</body>
</html>
```



Geschichte

- Entstanden 1995 mit den ersten Webbrowsern (Netscape)
- Kein Zusammenhang mit Java

Wichtigste Eigenschaften

- Nicht fest typisiert
- Objektorientiert (nach Bedarf)
- Dateiendung *.js
- Scriptsprache, kein Compiler notwendig



Node.js

- Laufzeitumgebung für JavaScript
- Ausführen von JavaScript ohne Webbrowser
- Basiert auf JavaScript Engine V8 von Google

Analogie

- Entspricht Java Virtual Machine (JVM)



NPM

NPM

Node Package Manager

Paketmanager für JavaScript

seit 2010

Analogie

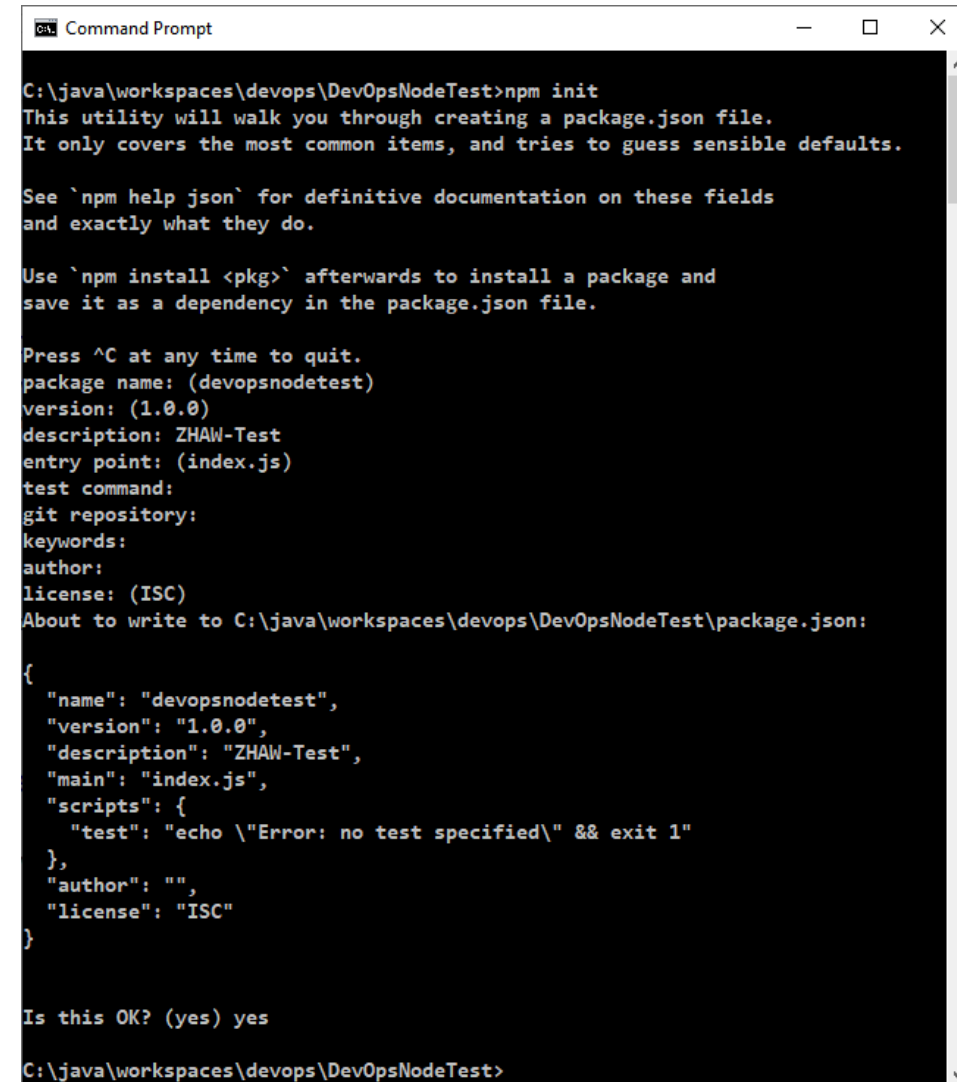
entspricht Gradle & Repository für Java



NPM Hello World

Das File **package.json** definiert das NPM Projekt. Es kann mit **npm init** erstellt werden.

```
{
  "name": "devopsnodetest",
  "version": "1.0.0",
  "description": "ZHAW-Test",
  "main": "index.js",
  "scripts": {
    "test": "echo ..."
  },
  "author": "",
  "license": "ISC"
}
```



```
Command Prompt
C:\java\workspaces\devops\DevOpsNodeTest>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (devopsnodetest)
version: (1.0.0)
description: ZHAW-Test
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\java\workspaces\devops\DevOpsNodeTest\package.json:
{
  "name": "devopsnodetest",
  "version": "1.0.0",
  "description": "ZHAW-Test",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
Is this OK? (yes) yes
C:\java\workspaces\devops\DevOpsNodeTest>
```

Tutorial: NPM Beispiel mit chart.js

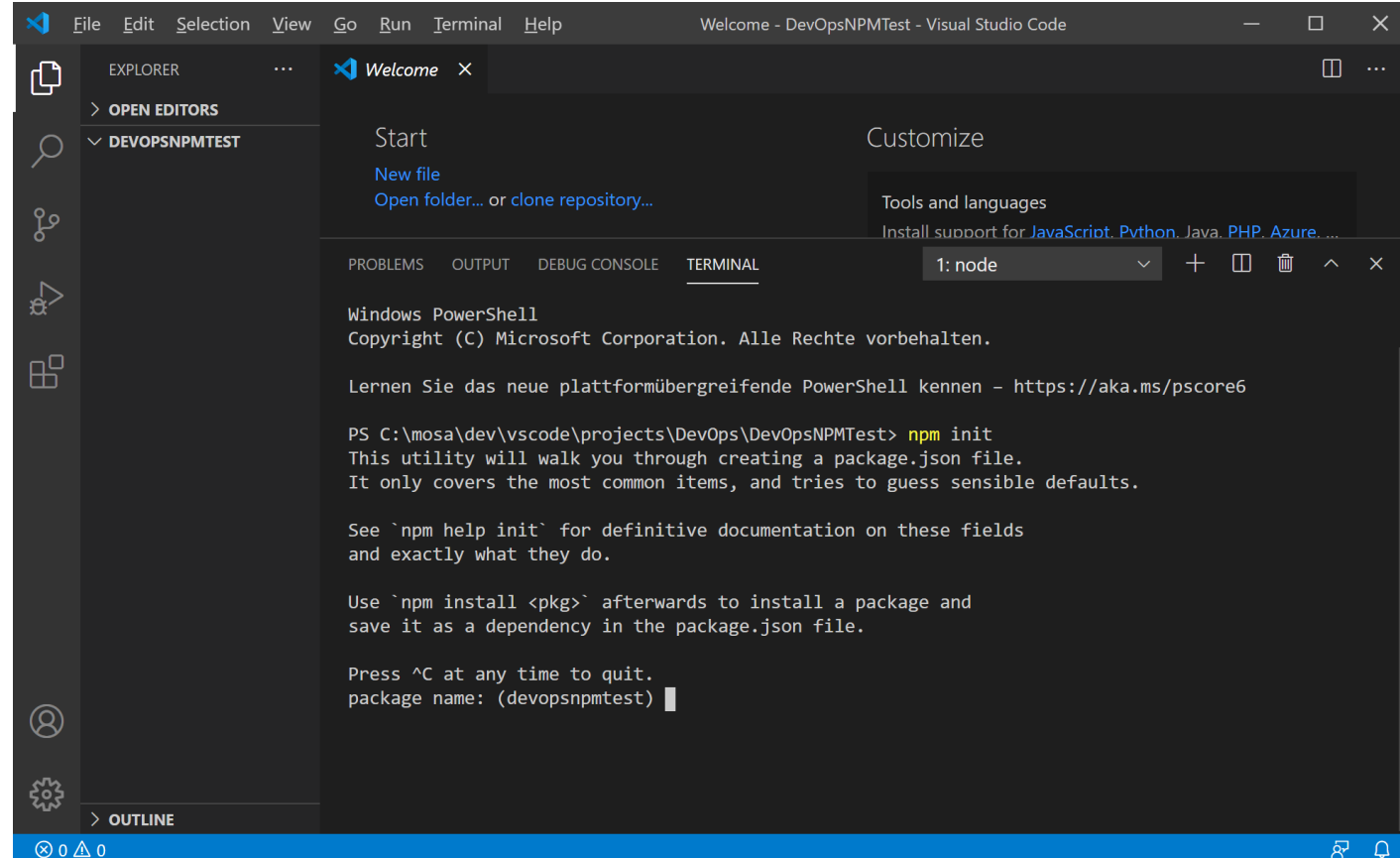
Tutorial

Neues Projekt

- neuen Ordner erstellen
- In Visual Studio Code öffnen

NPM-Projekt erstellen

- Terminal öffnen
- npm init starten, Name angeben und mit “Enter” durchgehen (Standard-Werte)



The screenshot shows the Visual Studio Code interface with the 'Terminal' panel open. The terminal window is titled '1: node' and shows the output of the 'npm init' command. The output text is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - https://aka.ms/pscore6

PS C:\mosa\dev\vscode\projects\DevOps\DevOpsNPMTTest> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

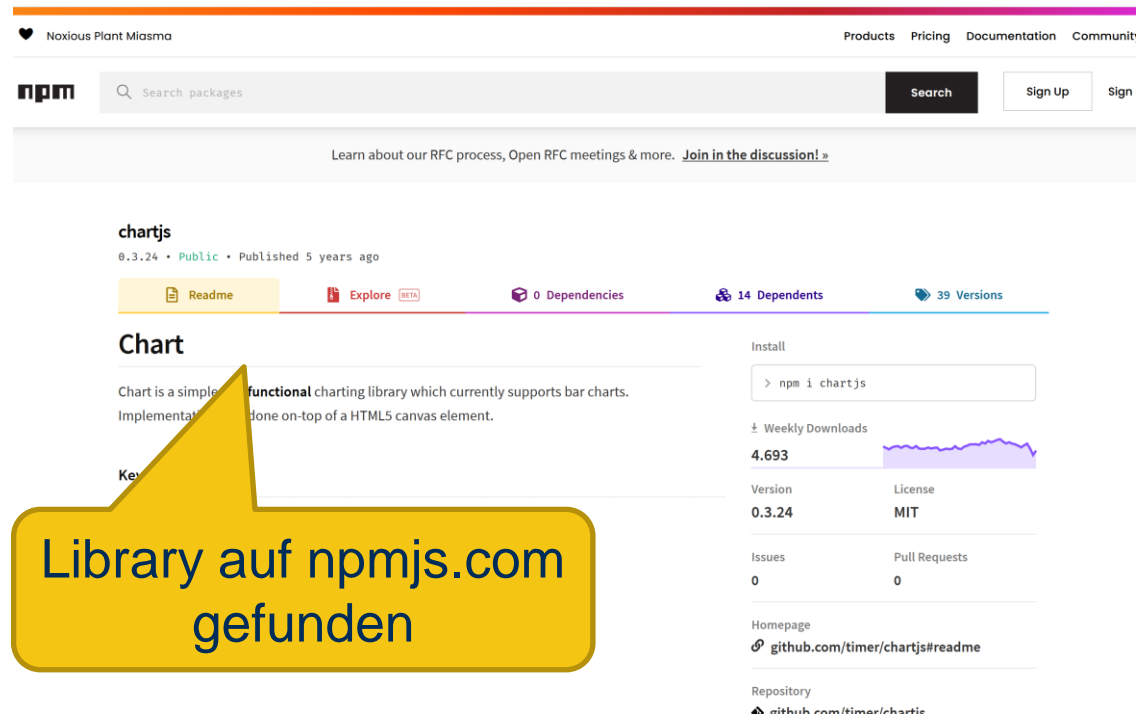
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (devopsnptest) 
```

NPM Dependencies einfügen

Tutorial

Wir fügen die Bibliothek zu unserem Projekt hinzu: **npm install chart.js@4.4.1 -save**



```
{
  "name": "devopsnodetest",
  "version": "1.0.0",
  "description": "ZHAW-Test",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: ...\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "chart.js": 4.4.1
  }
}
```

Tutorial: NPM Beispiel mit chart.js

Tutorial

HTML-Datei erstellen

chart.html Datei erstellen

Beispielcode kopieren

chart.html in Browser öffnen



```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <script src="node_modules/chart.js/dist/chart.umd.js"></script>
  <title>DevOps Chart</title>
</head>

<body>
  <canvas id="myChart" width="400" height="400"></canvas>
  <script>
    var ctx = document.getElementById('myChart').getContext('2d');
    var myChart = new Chart(ctx, {
      type: 'bar',
      data: {
        labels: ['Dev', 'Ops', 'DevOps', 'IT', 'Hardware', 'Software'],
        datasets: [{
          label: '# of Votes',
          data: [12, 19, 3, 5, 2, 3],
          backgroundColor: [
            'rgba(255, 99, 132, 0.2)',
            'rgba(54, 162, 235, 0.2)',
            'rgba(255, 206, 86, 0.2)',
            'rgba(75, 192, 192, 0.2)',
            'rgba(153, 102, 255, 0.2)',
            'rgba(255, 159, 64, 0.2)'
          ],
          borderColor: [
            'rgba(255, 99, 132, 1)',
            'rgba(54, 162, 235, 1)',
            'rgba(255, 206, 86, 1)',
            'rgba(75, 192, 192, 1)',
            'rgba(153, 102, 255, 1)',
            'rgba(255, 159, 64, 1)'
          ],
          borderWidth: 1
        }]
      },
      options: {
        scales: {
          y: {
            ticks: {
              beginAtZero: true
            }
          }
        }
      }
    });
  </script>
</body>
</html>
```


npm install (Funktionsweise und Terminal)

Tutorial

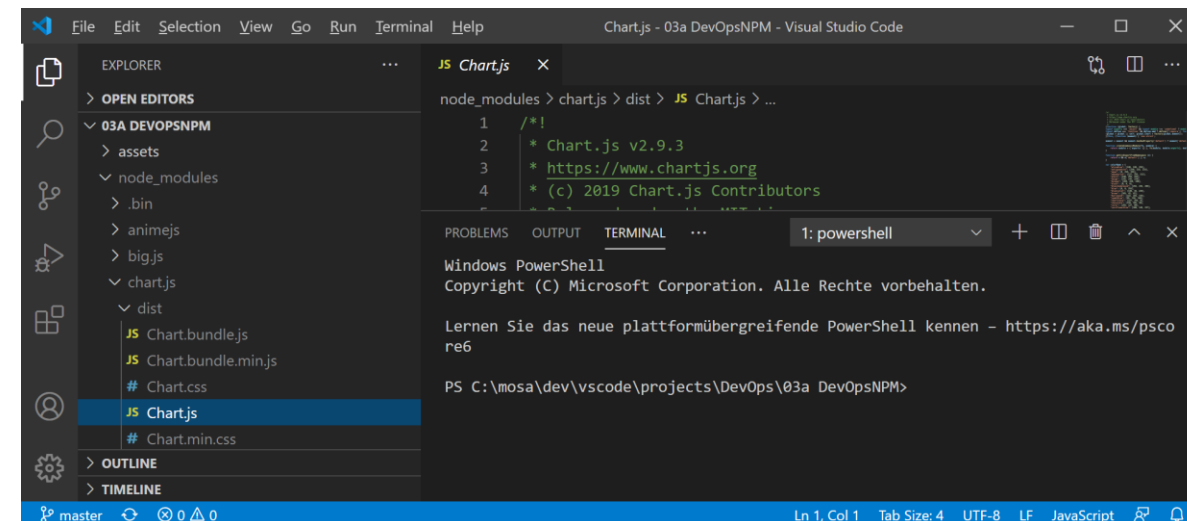
Wie wird die Chart-Bibliothek nun auf unseren Computer geladen?

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <script src="node_modules/chart.js/dist/chart.umd.js"></script>
  <title>My Game</title>
</head>
```

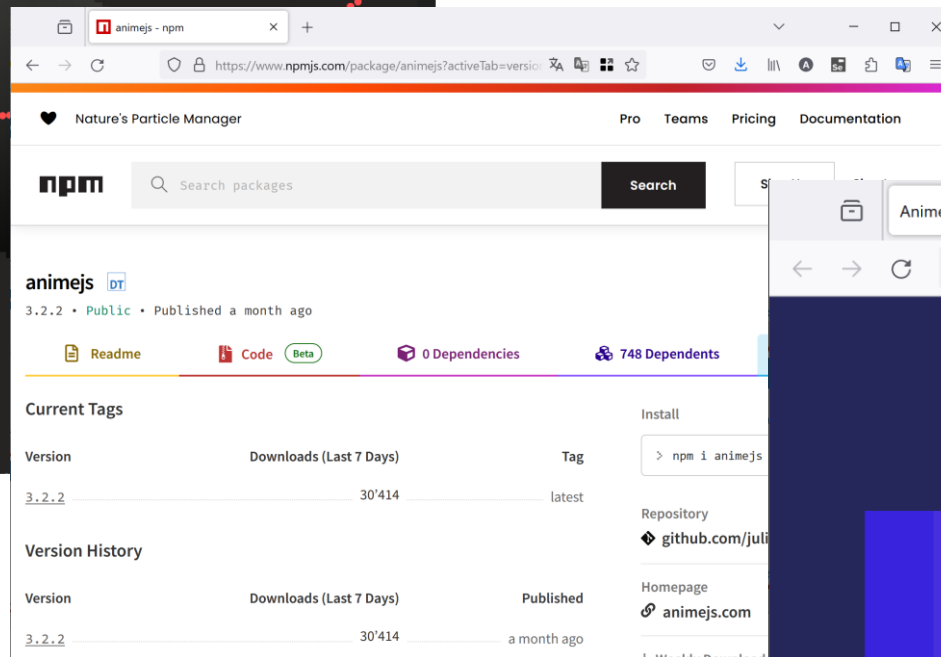
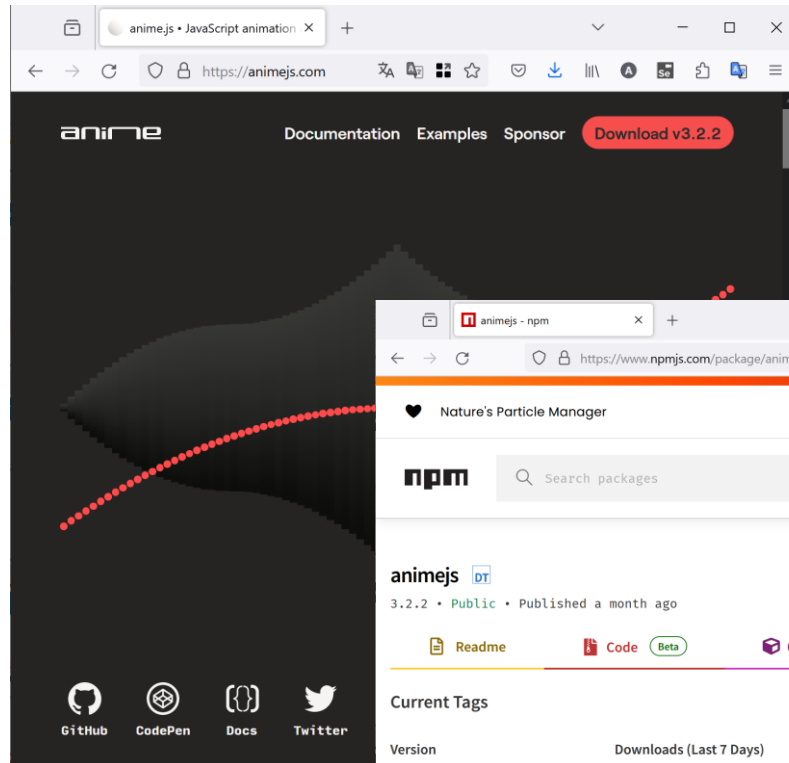
Chart wird im Ordner
node_modules gesucht

Dazu müssen zuerst alle
Abhängigkeiten mit **npm install**
geladen werden:

- In Terminal npm install eingeben
und die Bibliothek wird aus dem
Internet geladen
- Kontrolle im Filesystem



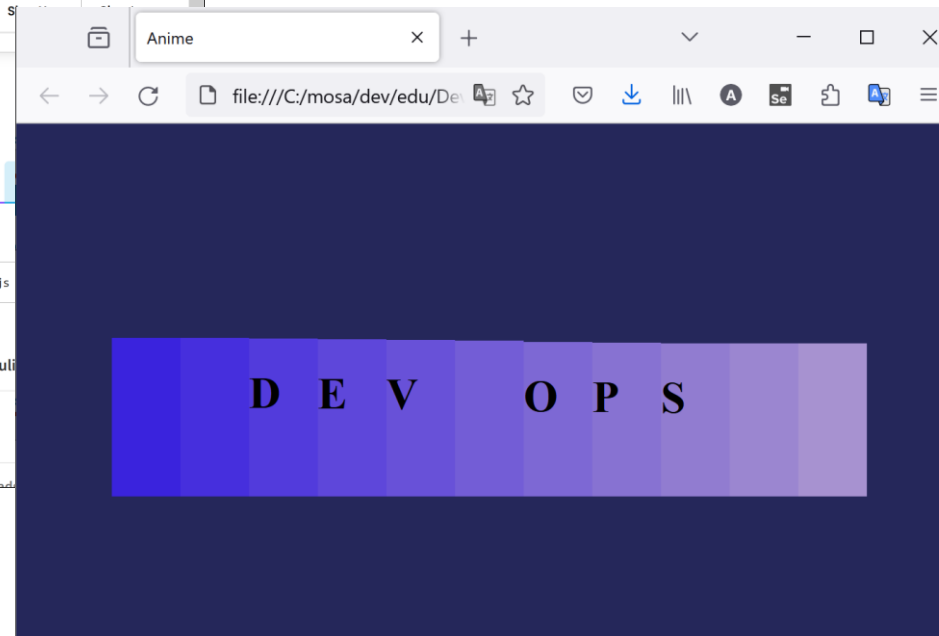
Weiteres Beispiel: anime.js



<https://animejs.com/>

<https://www.npmjs.com/search?q=animejs>

npm install animejs@3.2.2 --save
anime.html erstellen



Lernjournal

Lernziele

- Den Zweck von Build-Tools am Beispiel Gradle verstehen
- Mit Gradle als Build-Tool arbeiten können

Checkliste

- ✓ Neues Gradle-Projekt erstellen
- ✓ Suche nach Dependency dokumentieren
- ✓ Neue Dependencies (Libraries) hinzugefügt
- ✓ Dependency-Beispiele aus Vorlesung (Minimal) oder eigene Beispiele finden und erstellen (Advanced)
- ✓ Neue Dependency im Code verwenden (z.B. Hello World)
- ✓ Projekt auf GitHub gepushed
- ✓ gradle run ausführen, Beispiele funktionieren

Lernziele

- Den Zweck von Build-Tools am Beispiel NPM verstehen
- Mit NPM als weiterem Build-Tool arbeiten können

Checkliste

- ✓ Neues NPM-Projekt erstellen
- ✓ Frontend-Library mit npm install einbinden (Minimal chart.js und Anime.js aus Vorlesung) oder eigene Beispiele finden und erstellen (Advanced)
- ✓ Beispiel starten und ausprobieren
- ✓ Resultat auf GitHub pushen