Programmiergrundlagen

Sascha Püttner (BIG) | Sven Seyfert (BIG)

Agenda

1. Gängige Programmiersprachen
2. Programmierparadigmen
3. IDEs & Entwicklungsumgebungen
4. Hello World
5. Variablen & Datentypen
6. Operatoren
7. Bedingungen
8. Schleifen
9. Algorithmen
10. Datenbanken – kurzer Exkurs in SQL und SQL Management Studio
11. Frontend und Backend – Zusammenhang
12. Fragerunde, Feedback

1.

Gängige Programmiersprachen
Direkt für Dich.

Programmiersprachen

Sprache	Anwendungsbereiche	Besonderheiten
Python	Datenanalyse, KI, Webentwicklung, Automatisierung	Einsteigerfreundlich, große Community
JavaScript	Webentwicklung (Frontend & Backend mit Node.js)	Läuft direkt im Browser
Java	Enterprise-Anwendungen, Android-Apps	Plattformunabhängig, stark typisiert
C#	Enterprise-Anwendungen, Windows-Apps, Spieleentwicklung mit Unity	Gute Integration in Microsoft-Ökosystem
SQL	Datenbankabfragen	Deklarativ, für strukturierte Daten
PowerShell	Systemadministration, Automatisierung unter Windows	Cmdlet-basiert, stark in Windows integriert
TypeScript	Webentwicklung mit Typisierung	Superset von JavaScript
PHP	Webentwicklung (Server-seitig)	Häufig in CMS wie WordPress
Go	Cloud-native Anwendungen, Microservices	Schnell, kompiliert, einfach
Rust	Systemnahe Programmierung, sichere Speicherverwaltung	Sehr performant, speichersicher

2.

Programmierparadigmen
Direkt für Dich.

Programmierparadigmen

Paradigma	Beschreibung	Vorteile	Nachteile	Beispiele
Skriptsprachen	Werden meist interpretiert, für Automatisierung und kleinere Aufgaben genutzt	Schnell einsatzbereit, einfache Syntax, ideal für kleine Aufgaben	Weniger performant, schwerer zu strukturieren bei großen Projekten	PowerShell, Python, SQL, JavaScript
Objektorientierte Sprachen	Strukturieren Code in Klassen und Objekte mit Eigenschaften und Methoden	Gute Wiederverwendbarkeit, klare Struktur, geeignet für große Projekte	Kann komplex und überladen wirken, höhere Einstiegshürde	Java, C#, Python, C++, Rust
Funktionale Sprachen	Arbeiten mit Funktionen als zentrale Bausteine, vermeiden Zustände & Nebenwirkungen	Sehr gut für parallele Verarbeitung, mathematisch elegant	Ungewohnt für Einsteiger, weniger verbreitet in klassischen Anwendungen	F#, Racket , Haskell

3.

IDEs & Entwicklungsumgebungen

Direkt für Dich.

IDEs & Entwicklungsumgebungen

IDE / Editor	Geeignet für Sprachen	Vorteile	Nachteile
Visual Studio Code	JS, TS, Python, PowerShell, Go, etc.	Kostenlos, leichtgewichtig, viele Erweiterungen	Nicht alle Features wie große IDEs (z. B. Debugging in Java)
Visual Studio	C#, .NET, PowerShell	Vollwertige IDE, gute Integration in Windows	Ressourcenintensiv, nur Windows/macOS
IntelliJ IDEA	Java, Kotlin	Sehr leistungsfähig, gute Codeanalyse	Lizenzkosten für volle Version
Eclipse	Java, C/C++	Open Source, viele Plugins	Etwas altbackene UI, langsamer als moderne IDEs
PowerShell ISE / VS Code mit PowerShell Extension	PowerShell	Speziell für PowerShell-Skripte, gute Autovervollständigung	ISE ist veraltet, VS Code braucht Einrichtung
WebStorm	JavaScript, TypeScript	Beste JS/TS-Unterstützung, Debugging & Testing integriert	Kostenpflichtig
Rider	C#, .NET	Cross-Plattform, moderne UI	Kostenpflichtig
Sublime Text	Viele Sprachen	Schnell, minimalistisch	Eingeschränkte Funktionen ohne Plugins
Notepad ++	Viele Sprachen	Kostenlos	Eingeschränkte Funktionen ohne Plugins

4.

Hello World in verschiedenen Programmiersprachen Direkt für Dich.

PowerShell

```
# ------------------------------------
# 2. Hello World
# ------------------------------------
Write-Host "Hello World“
```

GO

```
// ------------------------------------
// 2. Hello World
// ------------------------------------
fmt.Println("Hello World")
```

Shell

```
# ------------------------------------
# 2. Hello World
# ------------------------------------
echo "Hello World"
```

5.

Variablen & Datentypen
Direkt für Dich.

Variablen & Datentypen

- Variablen speichern Daten
- Datentypen geben an, welche Art von Daten gespeichert werden

```powershell
# String
$name = "Sascha"

# Integer
$age = 35

# Boolean
$isAdmin = $true

# Array
$colors = @("Rot", "Grün", "Blau")
```

```powershell
# Hashtable
$user = @{
    Name = "Sascha"
    Role = "Admin"
}

# Explizite Typdeklaration
[int]$number = 42
[string]$text = "Hallo"
```

6.

Operatoren
Direkt für Dich.

Operatoren

```powershell
# Arithmetische Operatoren
$a = 5 + 3 # Addition
Write-Host "a = $a"

$b = 10 - 2 # Subtraktion
Write-Host "b = $b"

$c = 4 * 2 # Multiplikation
Write-Host "c = $c"

$d = 8 / 2 # Division
Write-Host "d = $d"

$e = 10 % 3 # Modulo (Rest der Division)
Write-Host "e = $e"
```

Vergleichsoperatoren

```powershell
# Vergleichsoperatoren
Write-Host "Gleich:       $($a -eq $b)" # ==
Write-Host "Ungleich:     $($a -ne $b)" # != oder <>
Write-Host "Größer als:   $($a -gt $b)" # >=
Write-Host "Kleiner als:  $($a -lt $b)" # <=
```

Logische Operatoren

```
# Logische Operatoren
$a = 5
$b = 8

(($a -gt 3) -and ($b -lt 7)) # false
(($a -eq 7) -or  ($b -eq 8)) # true
-not ($a -eq 5) # !=
```

Zuweisungsoperatoren

```powershell
# Zuweisungsoperatoren
$x = 10
$x += 5 # x = x + 5
$x -= 3 # x = x - 3
$x *= 2 # x = x * 2
$x /= 4 # x = x / 4
$x %= 3 # x = x % 3
Write-Host "x = $x" # Rest ist 0
```

7.

Bedingungen
Direkt für Dich.

If/Else

```powershell
# If/Else
$age = 18


if ($age -ge 18) {
    Write-Host "Volljährig"
}
else {
    Write-Host "Minderjährig"
}
```

If/ElseIf/Else

```
$score = 85


if ($score -ge 90) { # >=
    Write-Host "Ausgezeichnet"
}
elseif ($score -ge 80) {
    Write-Host "Gut"
}
elseif ($score -ge 70) {
    Write-Host "Befriedigend"
}
elseif ($score -ge 60) {
    Write-Host "Ausreichend"
}
else {
    Write-Host "Ungenügend"
}
```

Switch

```powershell
$number = 2


switch ($number) {
    1 { Write-Host "Eins" }
    2 { Write-Host "Zwei" }
    3 { Write-Host "Drei" }
    { $_ -gt 10 } { Write-Host "Große Zahl" }
    default { Write-Host "Andere Zahl: $_" }
}
```

8.

Schleifen
Direkt für Dich.

For-Schleife

```powershell
for ($i = 0; $i -lt 5; $i++) {
    Write-Host "Zähler: $i"
}
```

While-Schleife Kopfgesteuert

```powershell
# While-Schleife (Kopfgesteuert (Abbruchbedingung am Anfang))
$i = 0
while ($i -lt 5) {
    Write-Host "While: $i"
    $i++
}
```

Do-While-Schleife Fußgesteuert

```powershell
# Do-While-Schleife (Fußgesteuert (Abbruchbedingung am Ende))
$i = 0
do {
    Write-Host "Do-While: $i"
    $i++
} while ($i -lt 5)
```

Foreach-Schleife

```powershell
$colors = @("Rot", "Grün", "Blau")
foreach ($color in $colors) {
    Write-Host "Farbe: $color"
}
```

9.

Algorithmen
Direkt für Dich.

Algorithmen

- ist eine Schritt-für-Schritt-Anleitung
- um ein Problem zu lösen

Algorithmen: Übung

- Schreibe ein Programm, das die Zahlen von **1** bis **21** ausgibt.
- Folgenden Regeln:
  - Wenn die Zahl durch **3** teilbar ist → gib „**Fizz**“ aus.
  - Wenn die Zahl durch **5** teilbar ist → gib „**Buzz**“ aus.
  - Wenn die Zahl durch **3** und **5** teilbar ist → gib „**FizzBuzz**“ aus.
  - Ansonsten → gib die Zahl selbst aus.

Algorithmen: Erwartetes Ergebnis

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
```

Algorithmen: Lösungsbeispiel

```powershell
for ($i = 1; $i -le 20; $i++) {
    # Zuerst auf Teilbarkeit durch 3 und 5 prüfen,
    # da diese Bedingung spezieller ist als die einzelnen Teilbarkeiten.
    if (($i % 3 -eq 0) -and ($i % 5 -eq 0))
    {
        Write-Host "FizzBuzz"
    }
    elseif ($i % 3 -eq 0)
    {
        Write-Host "Fizz"
    }
    elseif ($i % 5 -eq 0)
    {
        Write-Host "Buzz"
    }
    else {
        Write-Host $i
    }
}
```

Datenbanken

- Kurzer Exkurs in SQL und SQL Management Studio
  - Live Demo

Frontend und Backend

- Zusammenhang
  - Live Demo

Fragen? Feedback!

- Danke -

Vielen Dank für Ihre Aufmerksamkeit.

Sascha Püttner
Digital Factory
Sascha.Puettner@big-direkt.de
+49 231 5557 1038

Sven Seyfert
Digital Factory
sven.seyfert@big-direkt.de
+49 231 5557 1277