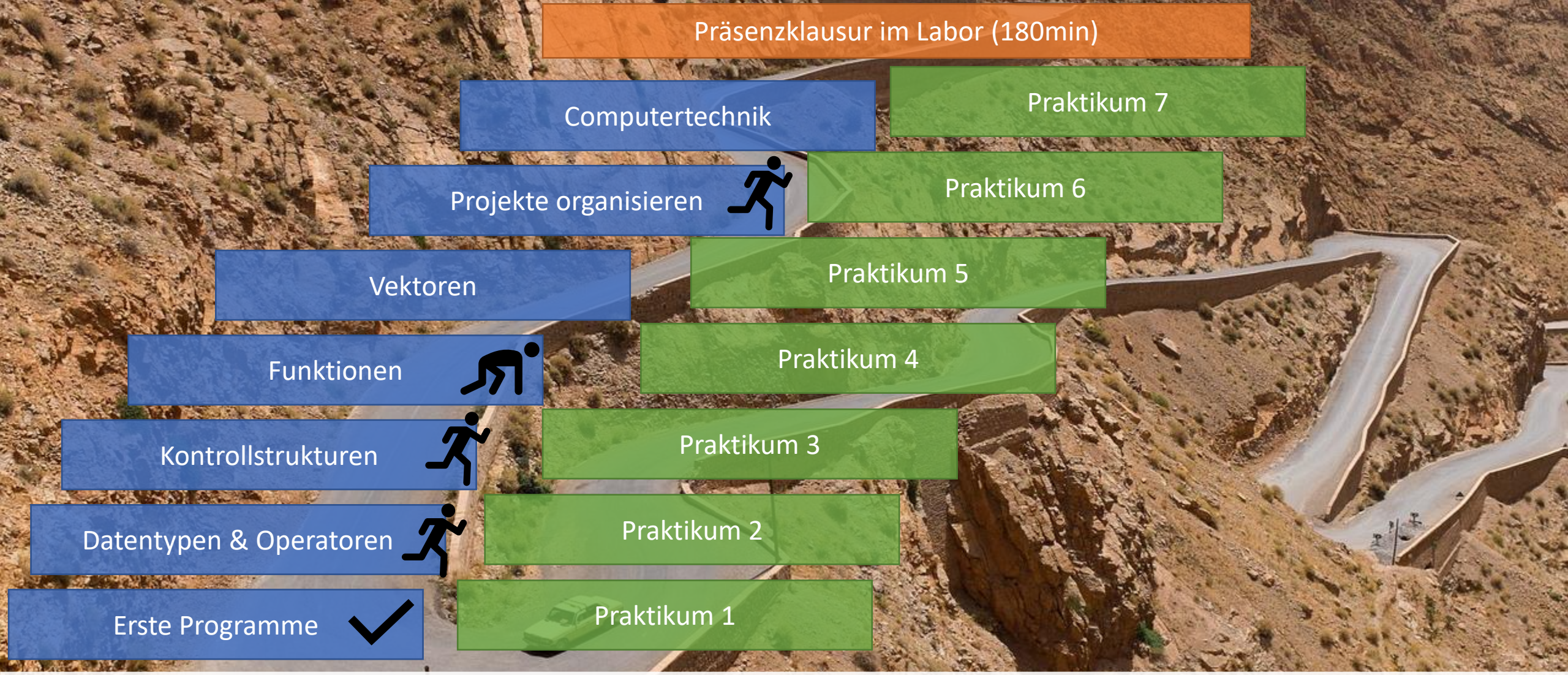
PROGRAMMIEREN I

WS 2022

Prof. Dr.-Ing. Kolja Eger
Hochschule für Angewandte Wissenschaften Hamburg

HAW HAMBURG

Check-In



Unser Weg durch das Semester

FUNKTIONEN

HAW
HAMBURG

Funktionen (Unterprogramme)

- Teile des Codes können in Funktionen ausgelagert werden
- Funktionen werden in Bibliotheken bereitgestellt oder können selbst geschrieben werden
- Funktionen aus der Bibliothek *stdio* sind z.B. `printf()` und `scanf()`
- Andere Bibliotheken umfassen mathematische Funktionen (z.B. *math.h*) oder Hilfestellung zur Verarbeitung von Zeichenketten (z.B. *string.h*)

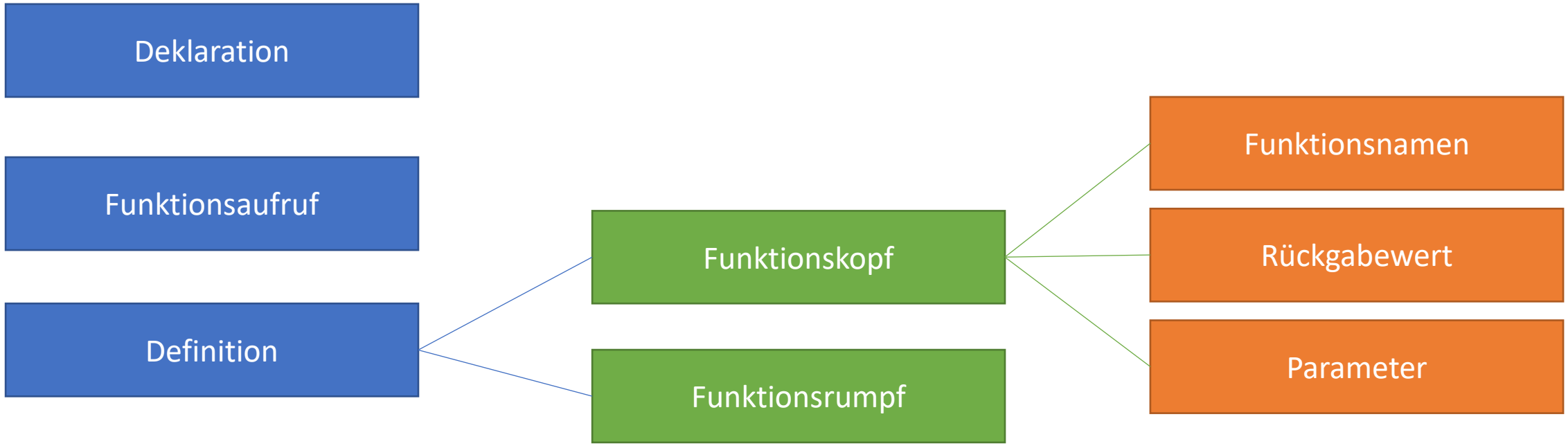
- Eine Übersicht zu den Funktionen in den Standard-Bibliotheken finden Sie in Sprachreferenzen im Internet, z.B.
  - https://www.cplusplus.com/reference/clibrary/
  - https://docs.microsoft.com/de-de/cpp/c-language/c-language-reference?view=msvc-170

Vorteile von Funktionen

- Modularität – Strukturierung des Programms und Auslagerung von Teilaufgaben („Kapselung“)
- Wiederverwendbarkeit – Funktionen lassen sich in unterschiedlichen Programmen verwenden
- Entwicklung – Aufgaben lassen sich einfacher im Entwicklerteam verteilen
- Wartbarkeit – einzelne Teile können einfacher von einander aktualisiert werden
- Lesbarkeit – durch die Strukturierung in Funktionen ergeben sich einzelne Abschnitte, die kürzer und einfacher verständlich sind

Eigene Funktionen schreiben

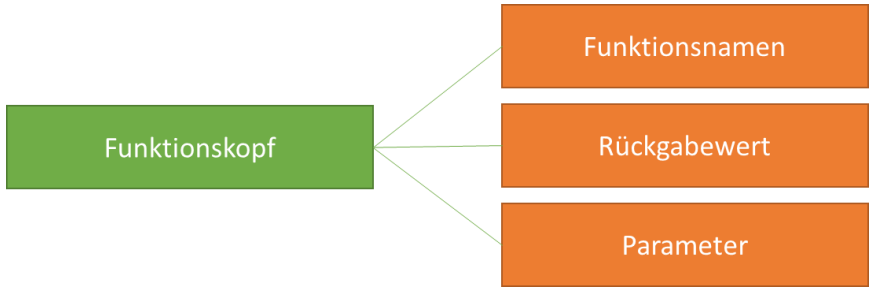
- Folgende Punkte werden wir im Folgenden kennenlernen:



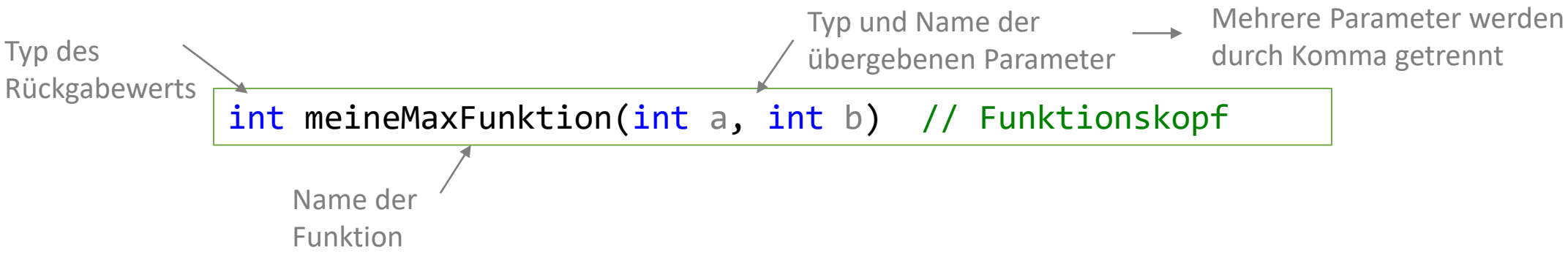
→ Beispiel in Visual Studio

```c
#include <stdio.h>

int meineMaxFunktion(int a, int b);     // Funktionsdeklaration

int main()
{
    int a, b, c;

    a = 100;
    b = 1;

    c = meineMaxFunktion(a, b);    // Funktionsaufruf

    printf("max(%d,%d)=%d", a,b, meineMaxFunktion(a, b));

    return 0;
}

// Funktionsdefinition
int meineMaxFunktion(int a, int b)  // Funktionskopf
{
    // Funktionsrumpf
    if (a >= b)
        return a;
    else
        return b;
}
```

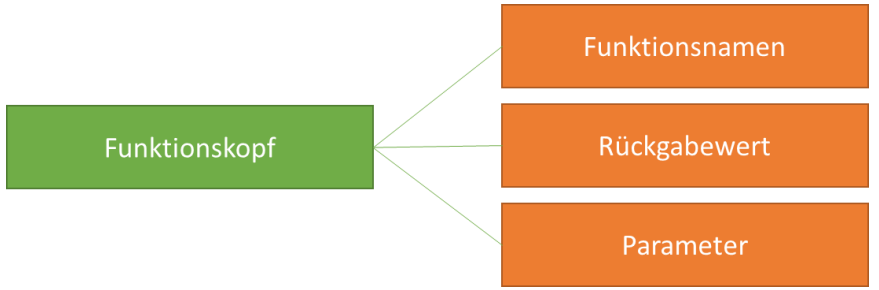
Funktionskopf



- Im Funktionskopf ist angegeben
  - wie die Funktion heißt,
  - was für einen Typ der Rückgabewerte der Funktion und
  - welche Parameter der Funktion übergeben werden können



Funktionskopf (mit *void*)



- Mit dem Schlüsselwort `void` geben Sie an, wenn keine Werte übergeben werden sollen (sowohl bei Rückgabe als auch bei Parametern)

Beispiel:

```
// Funktion ohne Rückgabewert
void func1(int a) {  /* ... */ }
// Funktion ohne Parameter
int func2(void) {  /* ... */ }
```

- Der C-Standard sieht die Angabe von `void` vor (auch wenn ein Compiler keinen Fehler oder keine Warnung ausgibt, wenn keine Datentypen angegeben werden)
- Dies gilt auch für das Hauptprogramm *main*

```
int main(void)
{
    /* ... */
    return 0;
}
```

`void` wird häufig weggelassen, aber Standard empfiehlt die Angabe von `void`

Funktionsrumpf

```
// Funktionsdefinition
int meineMaxFunktion(int a, int b)  // Funktionskopf
{
    // Funktionsrumpf
    if (a >= b)
        return a;
    else
        return b;
}
```

Der Funktionsrumpf enthält den Code der Funktion bestehend aus lokalen Variablen und Anweisungen

Beim Schlüsselwort `return` wird eine Funktion sofort verlassen. Der Rückgabewert wird in der `return`–Anweisung angegeben.

Eine Funktion kann mehrere `return`–Anweisung enthalten.

Funktionsaufruf

- Eine Funktion wird mit ihrem Namen aufgerufen
- Die zu übergebenen Parameter werden in runden Klammern angegeben (Reihenfolge beachten!)
- Der Rückgabewert der Funktion kann einer Variablen zugewiesen werden (Datentyp beachten!)

```
c = meineMaxFunktion(a, b);     // Funktionsaufruf
```

- Ein Funktionsaufruf kann auch in verschachtelten Anweisungen erfolgen

```
printf("max(%d,%d)=%d\n", a,b, meineMaxFunktion(a, b));
```

Deklaration

- Bevor eine Funktion aufgerufen werden kann, muss die Funktion (und sein Aufbau) dem Compiler bekannt sein
- Ein (unüblicher) Weg wäre es die Definition der Funktion im Quellcode z.B. vor der *main* einzufügen
- Ein besserer Weg ist die Funktion separat dem Compiler bekannt zu machen → **Deklaration**
- Die Deklaration sieht (in der Regel) wie der Funktionskopf in der Definition aus und wird mit einem Semikolon abgeschlossen (die Variablennamen können auch weggelassen werden)

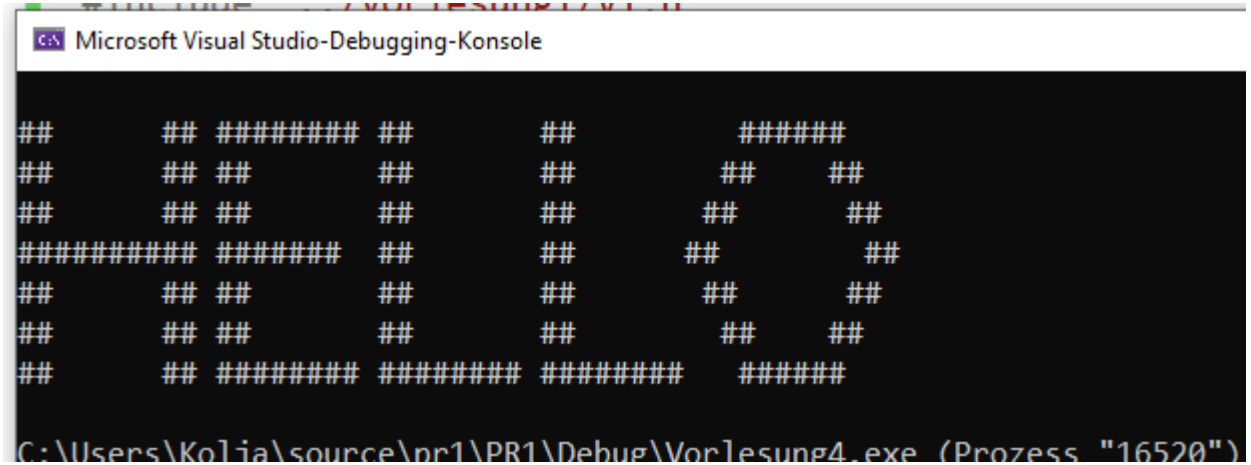
Beispiel:

```
int meineMaxFunktion(int a, int b);     // Funktionsdeklaration
```

Übung

- Aufgabe 1a):
  - Schreiben Sie ein Programm mit einer Funktion, die aus 3 int-Werten den größten Wert zurückgibt
  - Rufen Sie die Funktion im Hauptprogramm auf und erstellen Sie eine sinnvolle Ausgabe

- Aufgabe 1b):
  - Ergänzen Sie eine Funktion, die einen Willkommensgruß ausgibt
  - Mit ASCII lassen sich auch „Bilder malen“



VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!

HAW HAMBURG