Praktikum

- Review Aufgabe 8

Rückmeldung

Vorlesung 11

- Funktionen (Methoden)

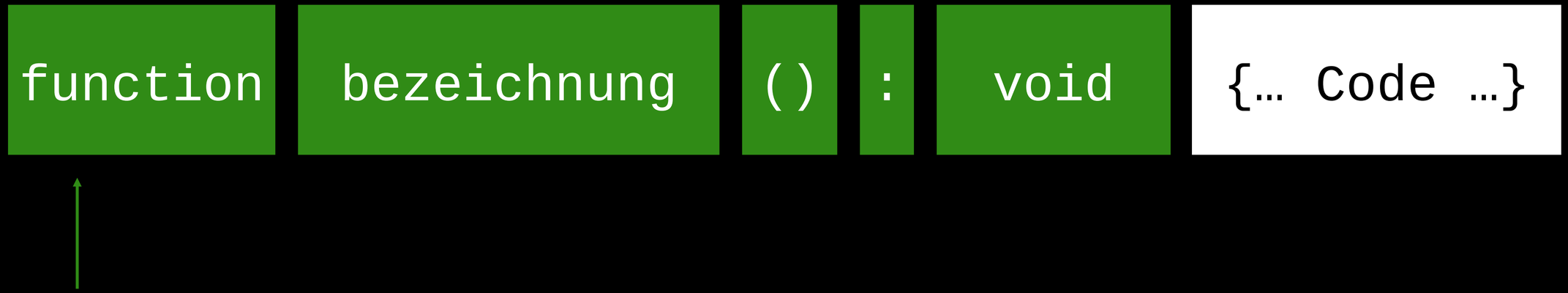
Einfache Funktionen

- Zentralisierung / Bündelung von Code

Einfache Funktionen

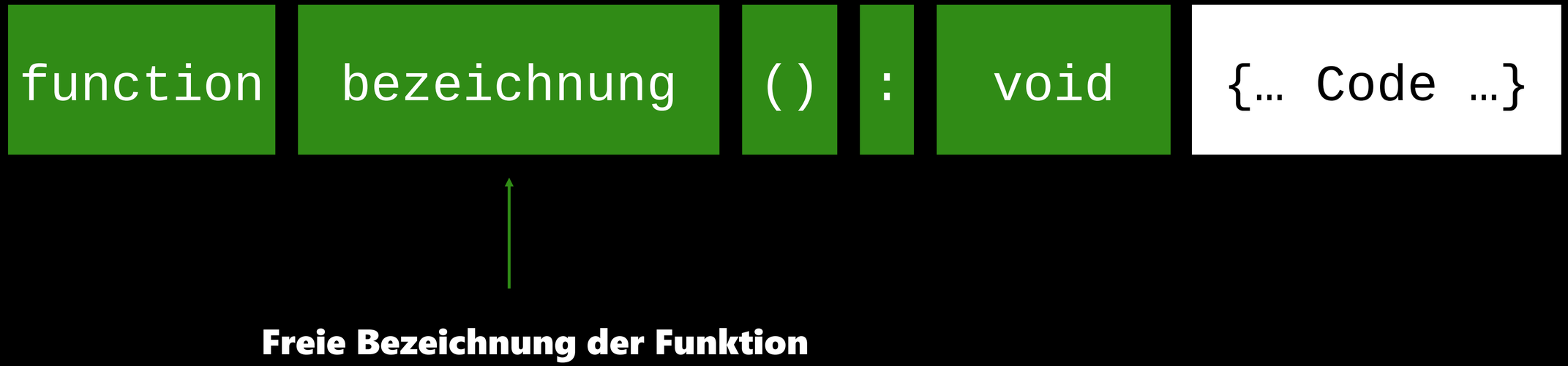
```
function greeter (): void {

    var message : string = "Hallo, mein Name ist Pingu";
    console.log (message);

}

// Funktionsaufruf
greeter ();
```

Einfache Funktionen Deklaration – in TypeScript

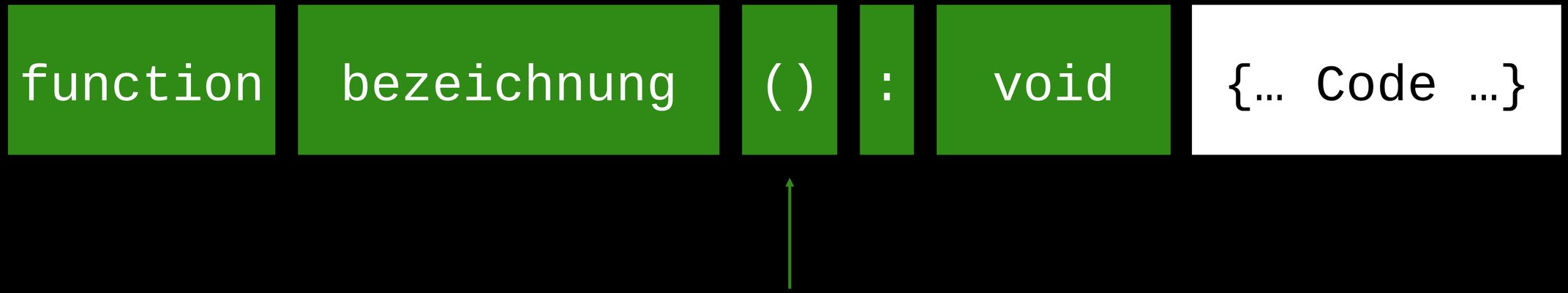


Schlüsselwort zur Deklaration einer Funktion

Einfache Funktionen Deklaration – in TypeScript

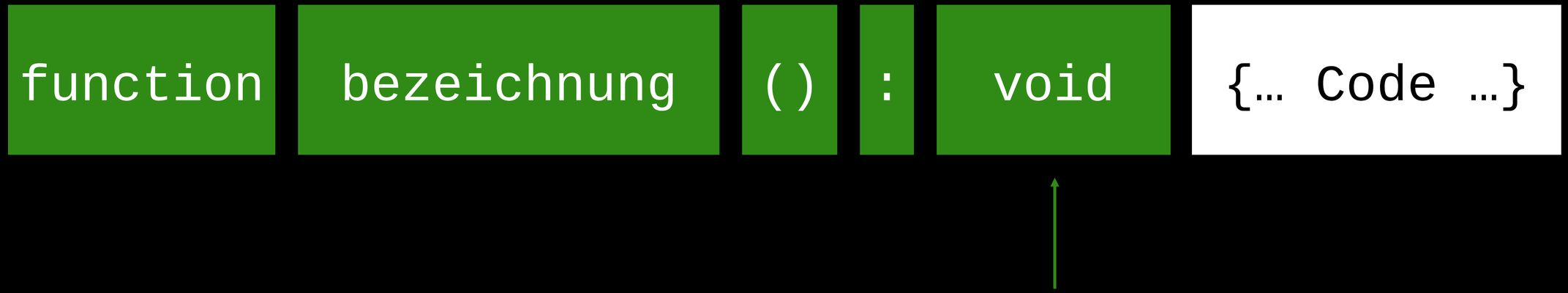


Einfache Funktionen Deklaration – in TypeScript



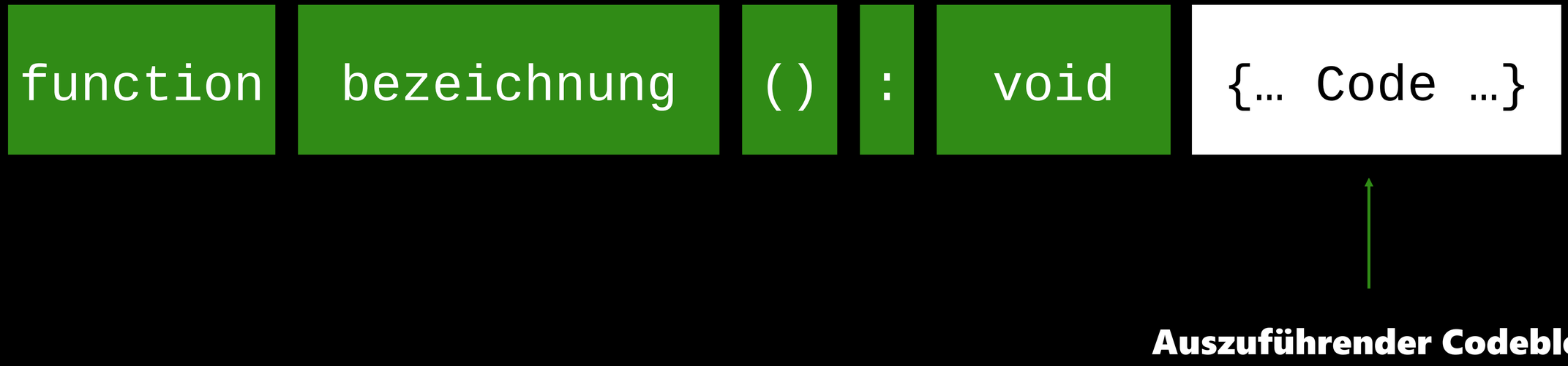
Optionale Übergabeparameter – hier: keine Parameter

Einfache Funktionen Deklaration – in TypeScript



Optional: Typ des Rückgabewertes – hier: void, EN für „nichts“

Einfache Funktionen Deklaration – in TypeScript



Einfache Funktionen Deklaration – in TypeScript

```
function bezeichnung () : void {… Code …}
```

Beispiel

```
function greeter () : void { … }
```

Aufruf

```
greeter ();
```

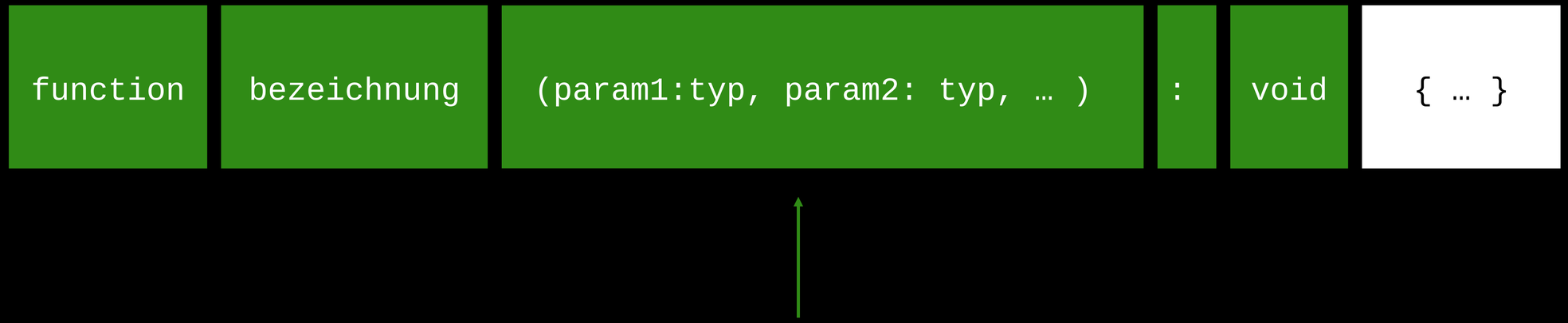
Parametrisierte Funktionen

- Funktionen mit Übergabe Parameter

Parametrisierte Funktionen

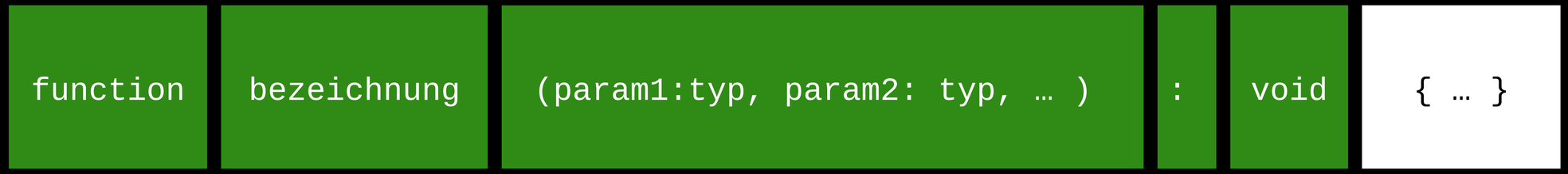
```
function greeter (surname: string, name: string) : void {

       var message = "Hallo, mein Name ist ";
       message += surname;
       message += " ";
       message += name;
       message += ". ";

       console.log (message);
 }

 // Funktionsaufruf
 greeter ("Pingu","Penguin");
 greeter ("Leon","Löwe");
```

Parametrisierte Funktionen



Freie Anzahl an Parametern
Getrennt durch Kommas

Parametrisierte Funktionen



Beispiel

```
function greeter (surname: string, name: string) : void { … }
```

Aufruf

```
greeter ("Pingu","Penguin");
```

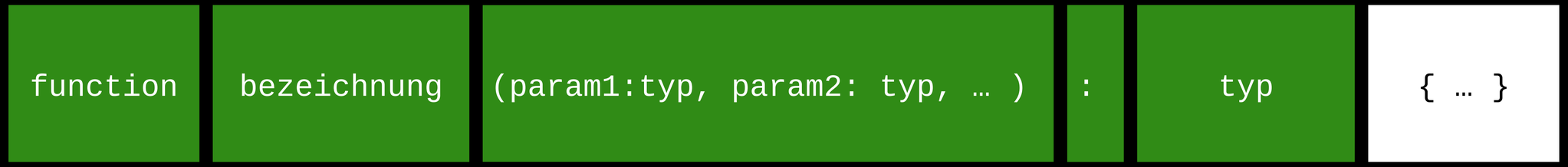
Funktionen mit Rückgabewert

- Funktionen zur Ergebnisberechnung

Funktionen mit Rückgabewert

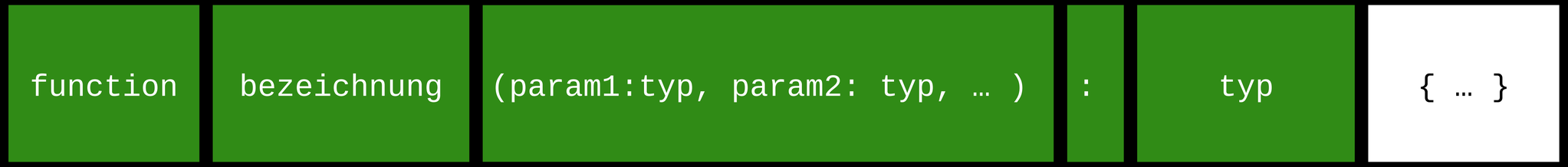
```
function calculateSum (age1 : number, age2 : number) : number {

            var sum : number = age1 + age2;
            return sum;

}

// Funktionsaufruf
var ageInTotal : number = calculateSum (2,14);
console.log ("Zusammen sind wir " + ageInTotal + " Jahre alt");
```

Funktionen mit Rückgabewert



Typ des Rückgabewerts,
z.B.
string, number, boolean,
string [], number[],...

Funktionen mit Rückgabewert



Beispiel

```
function calculateSum (age1 : number, age2 : number) : number
```

Aufruf

```
calculateSum (2,14)
```

Gültigkeit von Variablen

- sog. „Scope“


const welcomeFirstTime : string = "Willkommen!";
const welcomeBack : string = "Willkommen zurück"
var counter : number = 0;
function handleDOMContentLoaded() : void {
var i : number = 3929;
document.getElementById("helloWorldButton").addEventListener ("click", function(){
if (counter == 0){
greet (welcomeFirstTime, "Max", "Mustermann");
} else {
greet (welcomeBack, "Max", "Mustermann");
}
counter++;
});
}
function greet (welcomeMsg: string, vorname: string, nachname: string){
var greeting : string;
var pipes : string = "";
greeting = welcomeMsg + ", " + vorname + " " + nachname;
for (var i : number = 0; i < counter; i++){
pipes += "|";
}
}
document.addEventListener('DOMContentLoaded', function () { // anonyme Funktion
var i : number = 23456;
handleDOMContentLoaded();
console.log (i);
});
Sichtbarkeit: global
Sichtbarkeit: lokal, innerhalb des Funktionsblocks
Sichtbarkeit: lokal innerhalb der Schleife
Sichtbarkeit: lokal innerhalb des anonymen Funktionsblocks

Funktionen

- Zusammenfassung

Zusammenfassung Funktionen

Modularisierter, wiederverwendbarer Code
Funktionen können in verkürzter Form immer wieder aufgerufen werden

Variabler Einsatz
Funktionen können variable Übergabeparameter für die weitere Verarbeitung erhalten

Berechnung
Funktionen können einen Rückgabewert liefern

Schnittstellen
Funktionen bilden einen abgeschirmten Sichtbarkeitsbereich – theoretisch wird nur die Schnittstellendefinition in Form der Signatur benötigt.

Vielen Dank.