Statischen Typisierung (statically typed)

Bei der statischen Typisierung, wie sie Sprachen wie Java und C# nutzen, steht der Typ zur Übersetzungszeit bereits immer fest und kann nicht mehr geändert werden. Dies wird in der Regel dadurch erreicht, dass der Programmierer bereits zur Entwicklungszeit im Source-Code den Typ für Variablen, Parameter, Rückgabewerte etc. angibt

```
int number1 = 4;
float number2 = 4.25;
boolean check = 3 > 2;
```

Der große Vorteil von statisch typisierten Sprachen ist, dass der falsche Umgang mit Typen zur Laufzeit nicht möglich ist, da der Typ bereits zur Übersetzungszeit bekannt ist wird der Compiler, falls er einen Verstoß findet, diesen melden und kein fertig kompiliertes Programm bereitstellen. Dadurch haben Programme, welche mit statischen Programmiersprachen entwickelt werden, eine höhere Robustheit, da viele mögliche Fehlerquellen bereits durch den Compiler ausgeschlossen werden.

Dynamische Typisierung (dynamically typed)

Im Gegensatz zu statischen wird bei der dynamischen Typisierung der Typ erst zur Laufzeit ermittelt und ist auch veränderbar. Das sorgt dafür, dass dynamische Sprachen oft flexibler sind und weniger Code geschrieben werden muss. Zum Beispiel kann in Javascript einfach ein Objekt erstellt werden, das eine dynamische Anzahl von Variablen mit variablen Typen hat.

```
function printName ( p ) {
console.log ( p.name )
}

let obj = { name: "Test", age  21 }
printName (obj)
//Output der Funktion => Test
obj = "Random String"
printName(obj)
//Output der Funktion => undefined (Bug)
```

Starke Typisierung (strongly typed)

Eine stark typisierte Programmiersprache ist eine Sprache, in der bestimmte Operationen nur mit bestimmten Datentypen zulässig sind.

Der Begriff stark typisierte Programmiersprache bezieht sich auf eine Idee, die die Durchsetzung fester Beschränkungen für die Vermischung verschiedener Datentypen und Werte beschreibt. Bei einem Verstoß treten Fehler auf, die auch als Ausnahmen bezeichnet werden. Stark typisierte Programmiersprachen verwenden einen Sprachcompiler, um die Einhaltung der Datentypisierung zu erzwingen.

```
//Java ist stark typisiert
//es ist nicht möglich einen String mit einer Nummer zu verbinden

int number1 = 5;
String number2 = "10";
```

```
System.out.println(number1 + number2); //Error
```

Schwache Typisierung (weakly typed)

Bei Laufzeit wird herausgefunden, welche Datentypen die Variablen haben, und nimmt die notwendigen Anpassungen vor. Mit den Anpassungen bei Laufzeit muss man die verschiedenen Datentypen nicht neu definieren. Schwach typisierte Sprachen sind anfälliger für Laufzeitfehler.

```
//Javascript ist schwach typisiert
value = 21;
value = value + "dot";
console.log(value); //eine Nummer und ein String wird zu einem String verbunden => "21dot"
```