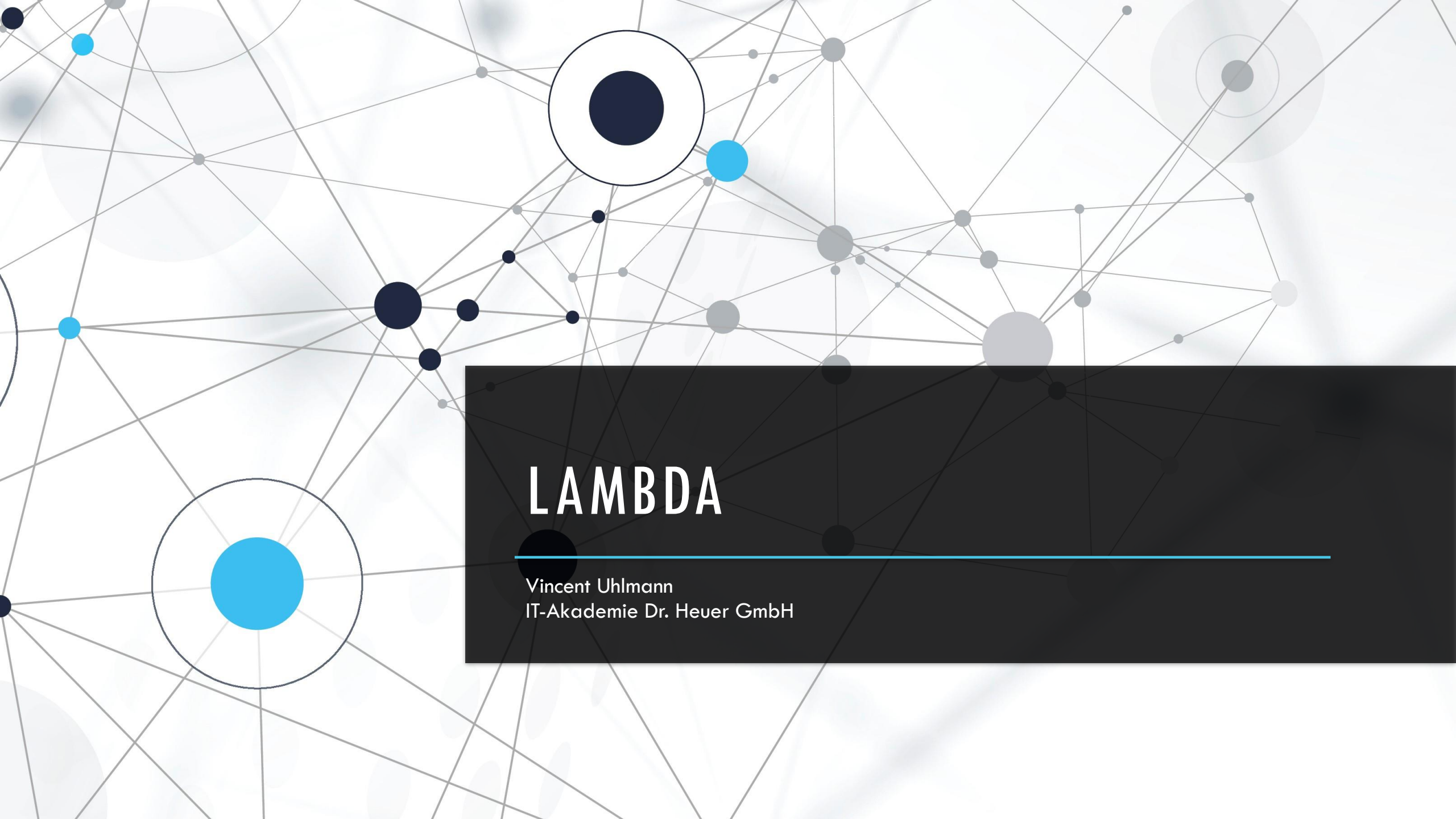

LAMBDA

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

LAMBDA

- Lambda-Ausdrücke sind anonyme Funktionen, die ohne Namen definiert werden können
- Sorgen für bessere Lesbarkeit und Reduzierung der Codezeilen, insbesondere für kurze und einfache Funktionen
- Können für die Erstellung von Delegaten verwendet werden (Func, Action, etc.)

VERGLEICH

- Anonymer Delegate

```
Delegate compare = delegate (int x, int y)
{
    return x == y;
};
```

- Lambda Ausdruck

```
Func<int, int, bool> compare2 = (x, y) => x == y;
```

SYNTAX

- Expression Lambdas: Körper ist ein Ausdruck
- (input-parameters) => expression

```
// Expression lambda
Func<int, int> func = (x) => x*x;
```

- Statement Lambdas: Körper ist ein Codeblock
- (input-parameters) => { <sequence-of-statements> }

```
// Statement lambda
Func<int, int> func = (x) => {
    Console.WriteLine(x);
    return x * x;
};
```

SYNTAX

- Liegt nur ein Parameter vor, können die Klammern weggelassen werden

```
Func<int, int> foo = x => x + x;
```

- Bei einer leeren Parameterliste müssen die runden Klammern angegeben werden

```
Action action = () => Console.WriteLine("Hallo");
```

- Standardwerte können in den Parametern angegeben werden, Parametertypen müssen explizit angegeben werden

```
Func<int, int, int> func = (int x, int y = 1) => {
    return x + y;
};
```

SYNTAX

- Parameter können ignoriert werden mit einem Unterstrich (_)

```
delegate int Test(int x, int z);

Test test = (_, _) => 25;
```