Overloading von Funktionssignaturen

```
export class Test{

    public add( a: Number, b:Number):void;
    public add( a: String, b:String):void;
    public add( a: boolean, b:boolean):void;
    public add( a, b):void{
        console.log(a+b);
    }
    public static run():any{
        let t:Test = new Test();
        t.add('a','b');
        t.add(3,5);
        t.add(true,true);
    }
}
Test.run();
Erzeugt die Ausgaben:
ab
8
2
```

https://www.bennadel.com/blog/3339-using-method-and-function-overloading-in-typescript.htm

Typescript Bindungsarten

[xxx]	@Input	Input Parameter
(xxx)	@Output	Callback Parameter

Conditional Types

```
declare type StringOrNull<T> = T extends string
  ? string
  : T extends null
  ? null
  : never;
export function upperCase<T extends string | null>
(text: T): StringOrNull<T>;
export function upperCase(text: string | null):
string | null {
  if (typeof text === 'string') {
    return text.toUpperCase();
  }
  return null;
}
// String
upperCase('').toLocaleLowerCase();
// Null
upperCase(null);
// Union
let maybe: string | null = null as any;
upperCase(maybe);
```

https://speakerdeck.com/gregonnet/conditional-types

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
stackoverflow.com/users/story/373498

Published 2nd July, 2019.
Last updated 7th July, 2019.
Page 1 of 1.