Functional Programming

Was ist functional Programming?

- Programmierparadigma (wie objektorientierte Programmierung)
- Stellt Funktionen in den Mittelpunkt
- Wesentliche Eigenschaften:
  - Funktionen, Funktionen, Funktionen!
  - Keine Side-Effects
  - Higher order functions
  - Immutable data

Das ist nicht functional

```
const name = „Frederic“;
const greeting = „Hi, ich bin „;

console.log(greeting + name);
=> „Hi, ich bin Frederic“
```

Das ist functional

```
function greet(name) {
   Return „Hi, ich bin „ + name;
}

greet(„Frederic“)
=> „Hi, ich bin Frederic“
```

Side Effects vermeiden/ Pure Functions

Not pure

```
const name = „Frederic“;

function greet() {
  console.log(„Hi, ich bin „ + name);
}
```

Pure

```
function greet(name) {
   return „Hi, ich bin „ + name;
}

console.log(greet(„Frederic“));
```

Arrow Functions

Arrow-Functions

```
function (name) {
   return „Hallo, ich bin „ + name;
}
```

```
(name) => {
   return „Hallo, ich bin „ + name;
}
```

Arrow-Functions

```
(name) => {
   return „Hallo, ich bin „ + name;
}
```

```
name => {
   return „Hallo, ich bin „ + name;
}
```

Arrow-Functions

```
name => {
   return „Hallo, ich bin „ + name;
}
```

```
name => „Hallo, ich bin „ + name;
```

Higher Order Functions

Funktionen als Return-Value

```
function makeGreeter(greeting) {
    return (name) => {
        return greeting + „ „ + name;
    }
}

Const sayHi = makeGreeter(„Hallo“);

console.log(sayHi(„Frederic“));
=> „Hallo Frederic“
```

Funktionen als Parameter

```
function printSomething( getPrintString ) {
    console.log( getPrintString() );
}
```

```
const filterGreaterFive = filter((el) => {
  if( el >= 5 ) {
    return true;
  }

  return false;
});

const myList = [1,5,3,7,8,2];

console.log(filterGreaterFive(myList));
=> [5,7,8]
```

Funktionen als Parameter

```
function filter( filterCondition ) {
    Return (arr) => {
        const outArray = [];

        for(let i=0; i < arr.length; i++) {
            if( filterCondition(arr[i]) == true) {
                outArray.push(arr[i]);
            }
        }

        return outArray;
    }
}
```

```
const filterGreaterFive = filter((el) => {
  if( el >= 5 ) {
    return true;
  }

  return false;
});

const myList = [1,5,3,7,8,2];

console.log(filterGreaterFive(myList));
=> [5,7,8]
```

Function Composing

Immutable Data

Veränderung bestehender Daten

```
let color = „red“;
color = „green“;

console.log( color );
=> „green“


const colorRed = „red“;
const colorGreen = „green“;

console.log( colorGreen );
=> „green“
```

Veränderung bestehender Daten

```
const names = [„Guido“,“Wolf-Dieter“,“Frederic“];
names[2] = „Thomas“

=> [„Guido“,“Wolf-Dieter“,“Thomas“]
```

```
const names = [„Guido“,“Wolf-Dieter“,“Frederic“];
const newNames = names.map(el => {
  if( el == „Frederic“ ) {
    return „Thomas“;
  }

  return el;
}
=> [„Guido“,“Wolf-Dieter“,“Thomas“]
```