Übung 01

Variablen Definitionen

```
var a = 1; // number
var b = 1.1; // number
var c = true; // boolean
var d = "hello world"; // string
var e = new Object(); // object
var f = [1,2,3,4]; // object
var g = ["hello", 1, true, e]; // object
var h = function getA() {return a}; // function

console.log(type of a); // etc...
```

Array Definitionen

```
var colors = ["red", "blue", "yellow"];
console.log(colors[0]); // etc...
colors.push("orange");
var colorString = colors[0] + " " + colors[1] + " " + colors[2] + " " + colors[3];
colors.sort();
```

Arrays

```
const numbers = [4,6,1,27,5,4];
const sumSquared = numbers.reduce((total, amount) => (total + amount*amount),0);
```

Konstruktoren

```
Circle.prototype.get = function() { return [this.radius, this.center] };
```

```
var object1 = new Object();
var date1 = new Date();
```

```
function Intervall(x, y) {
this.start = x;
this.end = y;
}
```

```
Intervall.prototype.isInside = function(value) {
    var isIn = (value >= this.start) && (value <= this.end);
    return isIn;
};
```

Funktionen

Funktion Mult definieren:

```
function mult(a,b) { return a*b; }
```

```
var mult = function(a,b) { return a*b; }
```

Rekursive Funktion:

```
var recursion = function(value) {
    return recursion(value + 1);
};
```

Memoization Pattern:

```
var recursion = (function() {
    var memo = {};

    function recursion(value) {
        var result;

        if(value in memo) { // existiert Wert bereits?
            result = memo[value];
        } else {
            result = recursion(value + 1);
        }
        return result;
    }

    return recursion;
})();
```