Distribueret programmering:

Lektion 02: Arrays

Arrays

- Et array er en dynamisk liste af indekserede elementer
- Arrays er objekter, hvor elementerne er værdier til properties med navne '0', '1', '2', ... – med typekonvertering 0, 1, 2, ...
- Elementerne kan være af vilkårlig type

```
// elementer.js
let a = [0, 'en', true, null, undefined, [7, 9, 13]];
console.log(a); // => [ 0, 'en', true, null, undefined, [ 7, 9, 13 ] ]
console.log(a['1']); // => en
console.log(a[5]); // => [7, 9 13]
console.log(a[6]); // => undefined
```

Array metoder

Arrays har en length property og en række metoder:

push(), pop(), unshift(), shift(), includes(), slice(), concat(), indexOf(), ...

```
// metoder.js
let a = [0,1,2,3];
console.log(a.length); // => 4
a.push(4);
console.log(a); // => [ 0, 1, 2, 3, 4 ]
console.log(a.shift()); // => 0
console.log(a); // => [ 1, 2, 3, 4 ]
```

Tæt eller hullet array

- Arrays kan være tæt eller med huller
- Implementationen af et tæt array er mest effektiv

```
// hullet.js
let a = [0, 1];
a[4] = 4;
console.log(a); // => [ 0, 1, <2 empty items>, 4 ]
console.log(a.toString()); // => 0,1,,,4
console.log(a.length); // => 5
```

length

- Hvis et array er tæt, er length lig antallet af elementer
- Hvis et array er hullet, er length > max index
- Hvis length tildeles en lavere værdi, slettes elementer med index ≥ length

```
// length.js
let a = [0, 1];
console.log(a.length); // => 2
a.length = 1;
console.log(a); // => [ 0 ]
a[2] = 2;
console.log(a); // => [ 0, <1 empty item>, 2 ]
a.length = 4;
console.log(a); // => [ 0, <1 empty item>, 2, <1 empty item> ]
```

CRUD på arrays

```
// crud.js
let a = [0, 1, 2, 3];
a[4] = 4;
console.log(a[1]); // => 1
a[2] = 'two';
delete a[3];
console.log(a); // => [ 0, 1, 'two', <1 empty item>, 4 ]
```

Iteration over elementer

```
// iteration.js
let a = [0, 1, 2, 3, 4];
delete a[2];

let s = '';
for (let i = 0; i < a.length; i++)
    s += a[i] + ', ';
console.log(s); // => 0, 1, undefined, 3, 4,

s = '';
for (let i in a)
    s += a[i] + ', ';
console.log(s); // => 0, 1, 3, 4,

s = '';
for (let e of a)
    s += e + ', ';
console.log(s); // => 0, 1, undefined, 3, 4,
```

Mange metoder returnerer et array

- Object.keys(), Object.values() og Object.entries()
- String metoden split()

```
// arrays.js
let o = {id: 123, '3 tal': [7, 9, 13]};
console.log(Object.keys(o)); // => [ 'id', '3 tal' ]
console.log(Object.values(o)); // => [ 123, [ 7, 9, 13 ] ]
console.log(Object.entries(o));
          // => [ [ 'id', 123 ], [ '3 tal', [ 7, 9, 13 ] ] ]

let s = 'Dette er en prøve';
console.log(s.split(' ')); // => [ 'Dette', 'er', 'en', 'prøve' ]
```

Rest og spread

```
// rest-spread.js
function sum(a, b, ...rest) {
    let sum = a + b;
    for (let e of rest)
        sum += e;
    return sum;
}
console.log(sum(1)); // => NaN
console.log(sum(1, 2)); // => 3
console.log(sum(1, 2, 3, 4)); // => 10

let spread = [1, 2, 3];
console.log(spread); // => [ 1, 2, 3 ]
console.log([0, ...spread, 4]); // => [ 0, 1, 2, 3, 4 ]
```

Destructuring

```
// destructuring.js
let [a, b, ...rest] = [10, 20, 30, 40, 50];
console.log(a); // => 10
console.log(b); // => 20
console.log(rest); // => [30, 40, 50]

[a, b] = [b, a];
console.log(a); // => 20
console.log(b); // => 10

let { x, z } = { x: 1, y: 2, z: 3 };
console.log(x); // => 1
console.log(z); // => 3
```

Arrays er objekter

```
// object.js
let a = [0, 1];
a[-1] = '-1';
a.size = function(){return this.length;};
console.log(a); // => [ 0, 1, '-1': '-1', size: [Function] ]
console.log(a.size()); // => 2
for (let n in a)
    console.log(n + ': ' + a[n]);
// => 0: 0
// => 1: 1
// => -1: -1
// => size: function(){return this.length;}
```

Array-like objekter

- Objekter med en length property og indekserede properties kaldes array-like objekter
- Array-like objekter kan bruge som arrays – dog uden array'ets metoder
- string er fx array-like
- Array-like objekter kan konverteres til arrays med Array.from() metoden

```
// array-like.js
let s = 'abcd', a = [];
console.log(s.length); // => 4
for (let i in s)
    a.unshift(s[i]);
console.log(a); // => [ 'd', 'c', 'b', 'a' ]
console.log(Array.from(s)); // => [ 'a', 'b', 'c', 'd' ]
s.push('e'); // TypeError: s.push is not a function
```

typeof

```
// typeof.js
console.log(typeof 123); // => number
console.log(typeof "abc"); // => string
console.log(typeof true); // => boolean
console.log(typeof null); // => object
console.log(typeof undefined); // => undefined
console.log(typeof {}); // => object
console.log(typeof function(){}); // => function
console.log(typeof []); // => object

let x = null;
console.log(x === null); // => true
console.log(Array.isArray([])); // => true
```