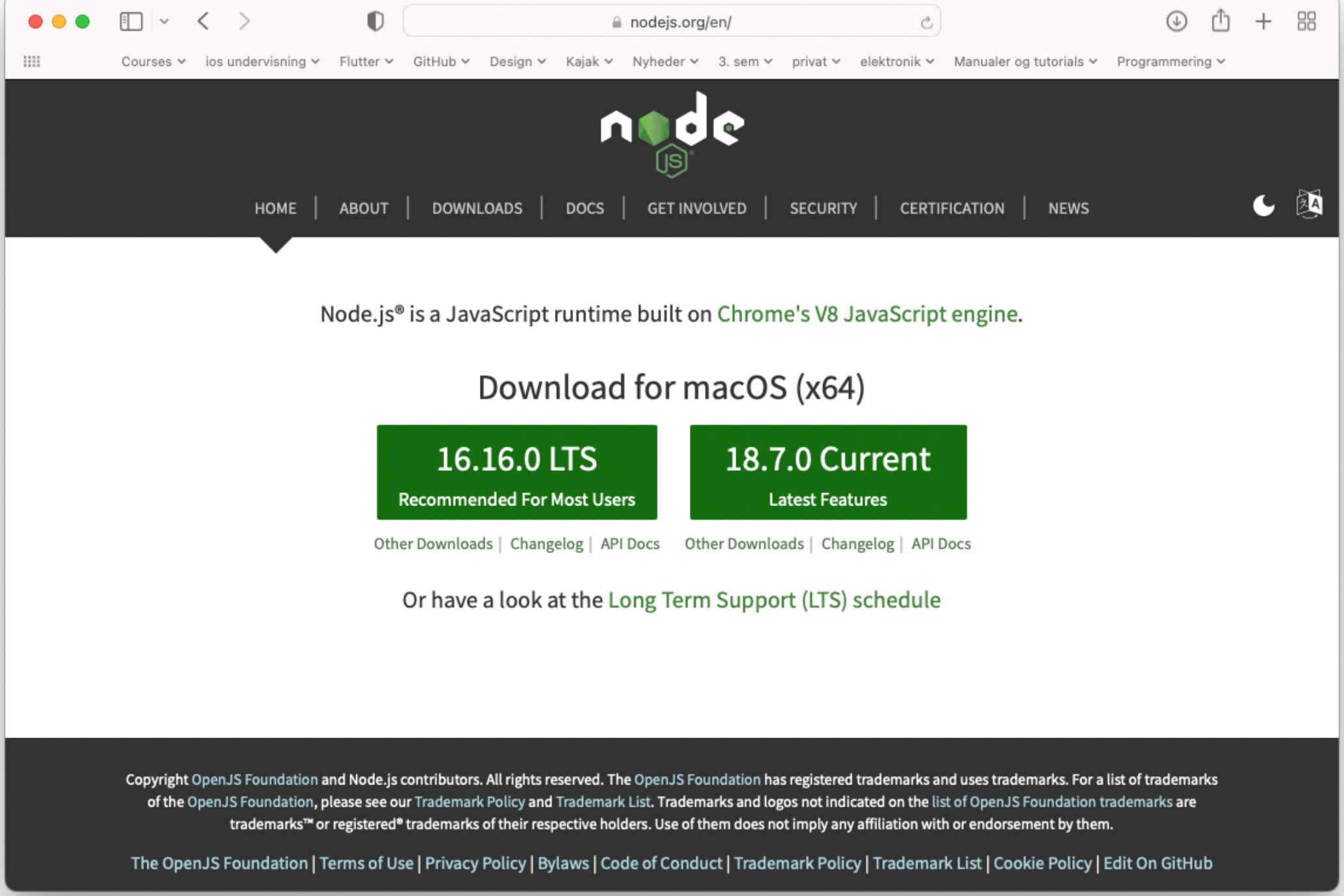
JavaScript introduktion

JavaScript

- Programmeringssproget der anvendes og afvikles i webbrowsere.
- Sammen med HTML5 og CSS3 udgør de tre teknologier internettes rygrad

javaScript

- Javascript kan også afvikles i en adskilt motor, V8 - udviklet her i århus af Google. Denne motor er pakket ind i et værktøj der gør det muligt at anvende javascript på serversiden så vi bruger det samme sprog overalt. Værktøjet er Node.js
- Node.js er en asynkron, eventstyret javascript runtime motor. Og den er yderst skalerbar.


nodejs.org/en/
Courses
ios undervisning
Flutter
GitHub
Design
Kajak
Nyheder
3. sem
privat
elektronik
Manualer og tutorials
Programmering
HOME
ABOUT
DOWNLOADS
DOCS
GET INVOLVED
SECURITY
CERTIFICATION
NEWS
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
Download for macOS (x64)
16.16.0 LTS
Recommended For Most Users
18.7.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs
Or have a look at the Long Term Support (LTS) schedule
Copyright OpenJS Foundation and Node.js contributors. All rights reserved. The OpenJS Foundation has registered trademarks and uses trademarks. For a list of trademarks of the OpenJS Foundation, please see our Trademark Policy and Trademark List. Trademarks and logos not indicated on the list of OpenJS Foundation trademarks are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.
The OpenJS Foundation | Terms of Use | Privacy Policy | Bylaws | Code of Conduct | Trademark Policy | Trademark List | Cookie Policy | Edit On GitHub

JavaScript dokumentation

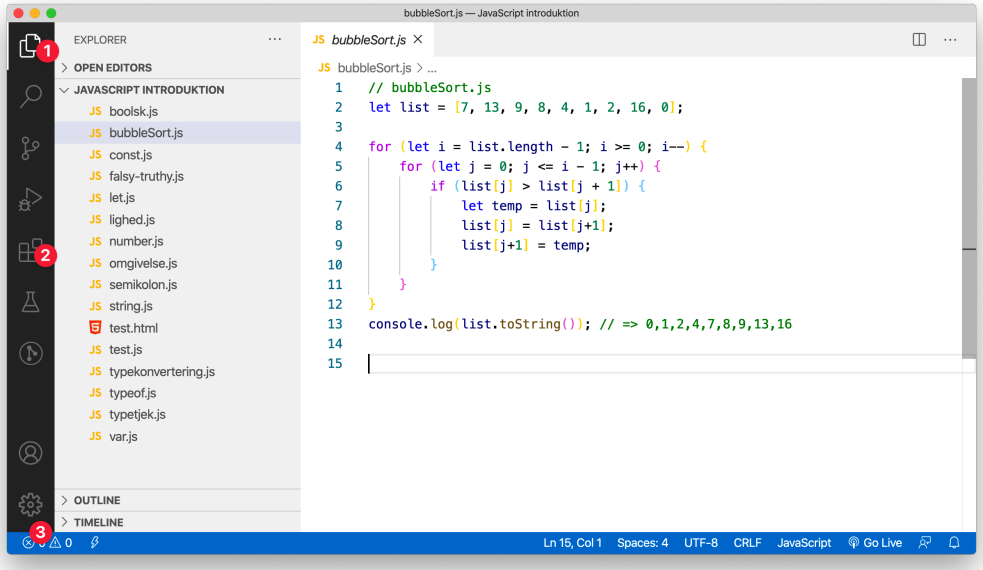
- **MDN**:
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
- **W3schools**:
  http://www.w3schools.com/jsref/default.asp

JavaScript eksempel

```
// bubbleSort.js
let list = [7, 13, 9, 8, 4, 1, 2, 16, 0];
for (let i = list.length - 1; i >= 0; i--) {
  for (let j = 0; j <= i - 1; j++) {
    if (list[j] > list[j + 1]) {
      let temp = list[j];
      list[j] = list[j+1];
        list[j+1] = temp;
    }
  }
}

console.log(list); // => 0,1,2,4,7,8,9,13,16
```

Visual Studio Code



1. Explorer. Altid åbn mapper og ikke workspaces
2. Plugins. Nogle bruger: Bracket Pair Colorizer 2, HTML Boilerplate, JavaScript Snippet Pack, Live Server, Npm, Npm intellisense, vscode icons
3. Terminal. Den bruges en del eller meget, afhænger...

VS Code: Keyboard Shortcuts

- Ctrl-K Ctrl-S Keyboard Shortcuts
- F1 Command palette
- F2 Rename
- Ctrl-K Ctrl-C Udkommenter linje
- Ctrl-K Ctrl-U Afkommenter linje
- Ctrl-Shift-K Slet linje
- Alt↑ Flyt op
- Alt↓ Flyt ned
- Alt-Shift↑ Dubler op
- Alt-Shift↓ Dubler ned
- Ctrl-K Ctrl-T Color Theme

Udleverede eksempler

Der anvendes eksempler i undervisningen. Disse er at finde i den `.zip` fil der vil ligge i Lektionen på Canvas. Opgaver til lektionen kan tage udgangspunkt i disse eksempler, så lær dem at kende!
Nogle lektioner uploades eksempler først efter præsentationen for at få demo-ændringer med

Kun 7 primitive typer

- number
- string
- boolean
- null
- undefined
- symbol
- bigInt

number

- Kun én tal type: number
- 64-bit floating point – svarende til Java's double
- Der er ingen særskilt integer type - udover bigint
- Specielle number værdier:
  - NaN (not-a-number)
  - Infinity

```
// number.js
console.log(1 == 1.0); // => true
console.log(7.0/2.0);// => 3.5
console.log(7/2); // => 3.5
console.log(Math.trunc(7/2)); // => 3
console.log(7/'two'); // => NaN
console.log(7/0); // => Infinity
```

string

- Kun én tekst type: string
- 16-bit Unicode
- strings er immutable
- string literals kan skrives med enten ', " eller `
- Der er ingen særskilt char type – så 'a' og "a" er samme tekststreng
- strings med
  - ' kan indeholde " uden escaping (")
  - " kan indeholde ' uden escaping (')
  - ' og " må ikke gå over flere linjer – brug i stedet \n
  - `** (kaldet template literals) kan indeholde linjeskift og indlejrede udtryk (${})

string - fortsat

```
// string.js
let s1 = 'En \' string kan indeholde "';
console.log(s1); // => En ' string kan indeholde "

let s2 = "En \" string kan indeholde '";
console.log(s2); // => En " string kan indeholde ‘

let s3 =
`En \` string kan indeholde ', " og linjeskift
samt udtryk: ${s1.length}`;
console.log(s3); // =>
// En ` string kan indeholde ', " og linjeskift
//      samt udtryk: 27
```

boolean

- boolean typen kan være `true` eller `false`

To specielle, primitive typer – med én værdi

- null – angiver
  no value eller no object
- undefined – angiver
  not initialized, does not exist eller no return value

Sætninger

- Sætningerne `if`, `switch`, `for`, `while`, `do`, `try`, `throw`, `return` og `break` fungerer som de tilsvarende sætninger i Java
- Derudover har JavaScript løkkerne `for…in` og `for…of`

let

- Variable (bindings) erklæres med `let` sætningen
- Variable har ingen type
- Hvis en variable ikke initialiseres, får den værdien `undefined`

```
// let.js
let x, y = 123;
console.log(x); // => undefined
console.log(y); // => 123
y = 'test';
console.log(y); // => test
y = null;
console.log(y); // => null
```

const

- Konstanter erklæres med const sætningen
- Konstanter skal initialiseres i erklæringen

```
// const.js
const x = 123;
console.log(x); // => 123
x = 'test'; // => TypeError: Assignment to constant variable
```

var

- Variable kan også erklæres med var sætningen, **men let sætningen bør anvendes i stedet**
- var sætningen har kun function scope og variable hoistes til starten af programmet (eller funktionen)
- let sætningen kom med ES6, har block scope og

```
// var.js
console.log(x); // => undefined
var x = 123;
console.log(x); // => 123
console.log(y); // => ReferenceError: y is not defined
let y = 456;
```

Se en fin beskrivelse (på US) HER:
https://scotch.io/tutorials/understanding-hoisting-in-javascript
vi kommer ikke mere ind på hoist, men let giver en semantik tættere på det man 'kender'

Automatisk typekonvertering

- JavaScript foretager automatisk en typekonvertering, hvis det er nødvendigt for at beregne et udtryk

```
// typekonvertering.js
console.log(true + 2); // => 3
console.log('3' + 4); // => 34
console.log('3' - 4); // => -1
console.log(5 * '6'); // => 30
console.log(7 * null); // => 0
console.log('x' - 8); // => NaN
console.log(!'ok' + 9); // => 9
```

falsy og truthy

- Om nødvendigt konverteres følgende værdier til false

  null`, `undefined`, ' ' (tom string), `0`og`NaN`

- Disse værdier kaldes falsy
- Alle andre værdier konverteres til true – og kaldes truthy

```
// falsy-truthy.js
let i = 2;
while (i--)
    console.log(i); // => 1 0
```

`===` og `!==`

- De fleste operatorer virker som i Java – men nogle er specielle
- `==` sammenligner om to værdier er lig med hinanden – om nødvendigt efter typekonvertering
- `===` sammenligner om to værdier er helt ens

```
// lighed.js
console.log(0 == false); // => true
console.log(0 === false); // => false
```

- Tilsvarende med `!=` og `!==`

`&&`, `||` og `!`

- `x && y` returnerer x, hvis x er falsy – ellers y
- `x || y` returnerer x, hvis x er truthy – ellers y

  y evalueres kun, hvis y skal returneres
- `!x` returnerer false, hvis x er truthy, ellers true

```
// boolsk.js
console.log('abc' && 123); // => 123
console.log('abc' || 123); // => abc
console.log(!'abc'); // => false
console.log(!0); // => true
```

`typeof`

- returnerer typen af en værdi – som en string

```
// 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
```

Typetjek

```
// typetjek.js
console.log(typeof 123 === 'number'); // => true
console.log(typeof '123' === 'number'); // => false
console.log(typeof ('123' - 0) === 'number'); // => true
console.log(typeof parseInt('123') === 'number'); // => true

console.log(typeof 'abc' === 'string'); // => true
console.log(typeof true === 'boolean'); // => true

console.log(typeof null === 'null'); // => false
console.log(null === null) // => true

console.log(typeof undefined === 'undefined'); // => true
console.log(undefined === undefined) // => true
console.log(undefined === null) // => false
console.log(undefined == null) // => true
```

Programmeringsomgivelse

- Der er defineret en række standard variable, der refererer til værdier, funktioner og objekter

```
// omgivelse.js
NaN;
parseInt;
console;
Number;
Math;

console.log(NaN); // => NaN
console.log(parseInt('111', 2)); // => 7
console.log(Number('0xFF')); // => 255
console.log(Number.isNaN('1o2')); // => false
console.log(Math.cos(Math.PI)); // => -1
```