

# Internetwork og Web-programmering

## Introduktion til Web Teknologier

### Klient-side og server-side teknologi

Forelæsning 1

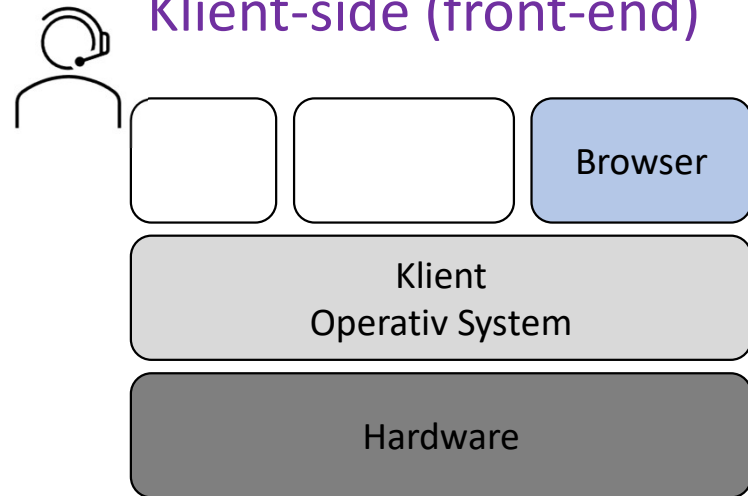
Brian Nielsen

Distributed, Embedded, Intelligent Systems



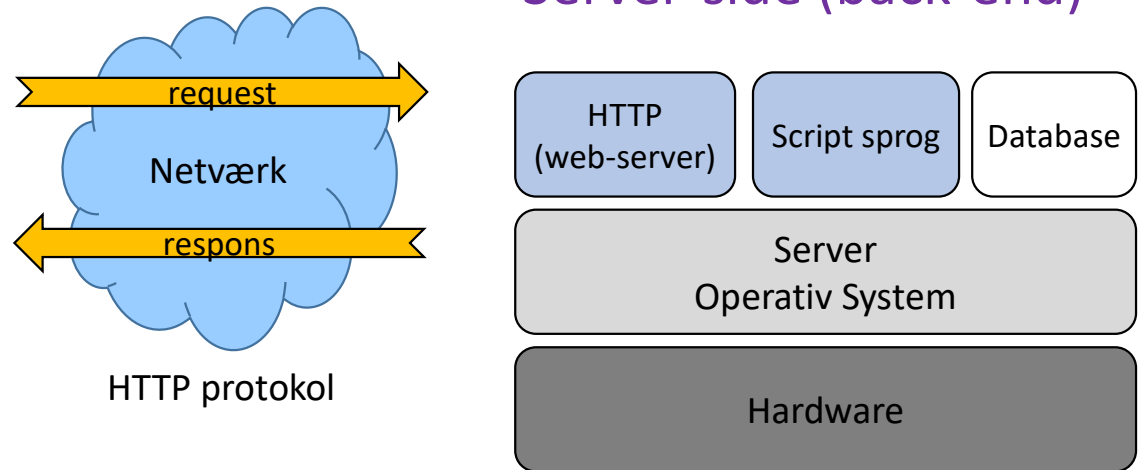
# Simpel web klient-server arkitektur

## Klient-side (front-end)



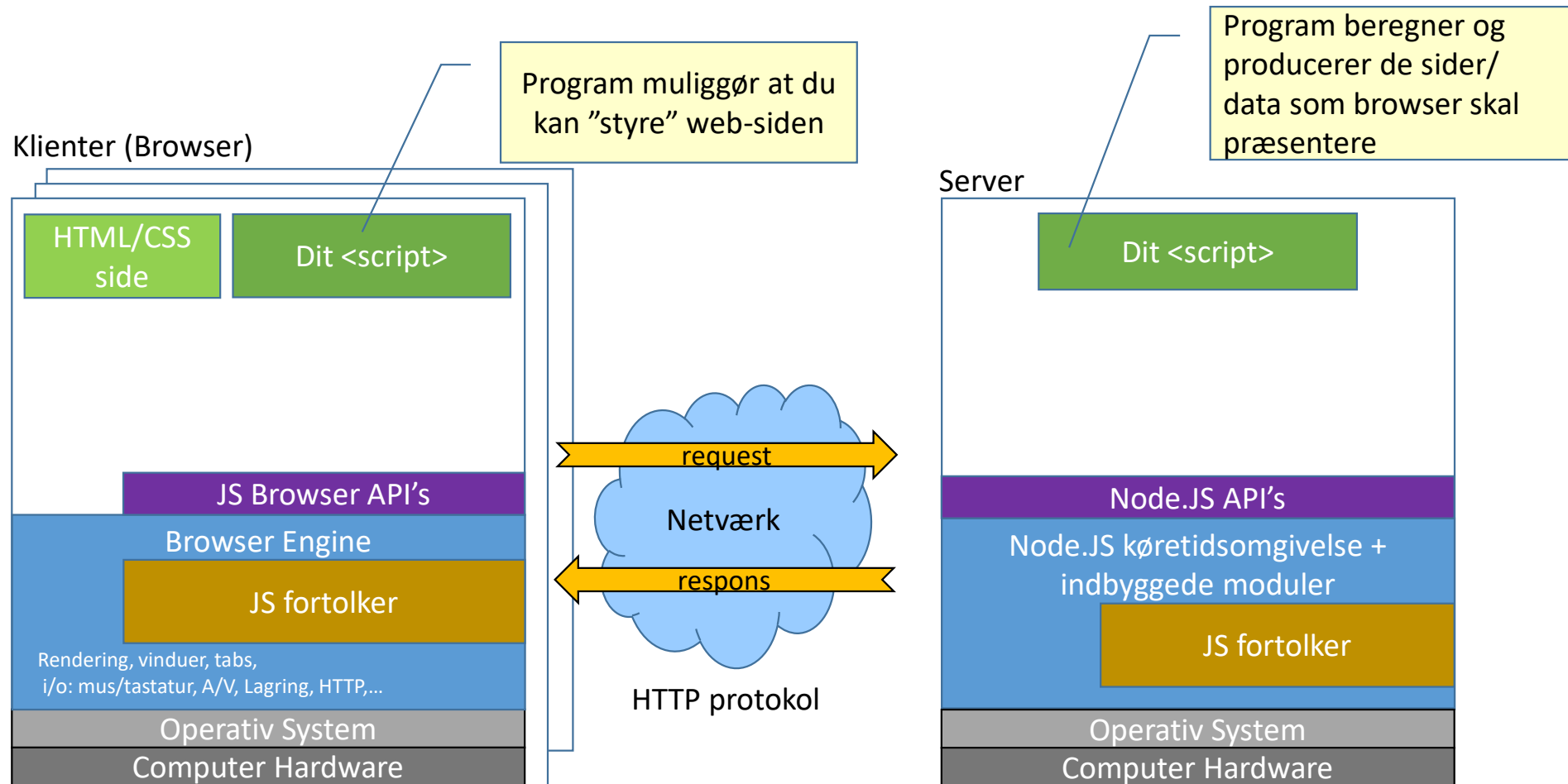
- Hyppigst tilgås en web-side via en browser
- (Men kan være andre programmer, som "snakker" HTTP)
- "UI" og nogle funktioner

## Server-side (back-end)



- Server, særlig maskine
  - Always-on, offentlig IP, klar til at modtage requests, kraftig
- Implementerer server-siden af en applikation
  - Gemmer tilstand, foretager optagelser og beregninger
- Server leverer information til klienten
  - HTML dokumenter: statiske filer, eller dynamisk genereret
  - Data-objekter (JSON)
- (Men kan også give adgang til data fra "smart devices", fx lys- og varme-styring)

# Overordnet arkitektur i IWP



# Front-end tre-enigheden: HTML5, CSS, JavaScript



**HTML** beskriver  
indhold og dets struktur, opbygning  
i bestanddele

"semantic markup"

HTML5



3 forskellige separate  
sprog, men  
dokumenterne er delvist  
indbyrdes afhængige



**Javascript** giver  
Dynamisk optegning af side,  
Dynamik, Interaktion,  
Animation, UI,  
input validering,  
funktioner,  
Dynamisk data-indlæsning fra server ...

JS



CSS

**Cascading Style Sheets**  
Definerer dokumentets  
præsentation: Udseende, layout,  
farver, fonts,...

Samme html dokument kan have flere udseender,  
fx afhængigt af

- Skærmstørrelser
- Brugerens/udviklerens præferencer

# Front-end tre-enighed: Code/Demo1/toggleDemo.html

```
<!DOCTYPE html>
<html lang="da">
  <head>
    <title>IWP DEMO</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <script src="demo.js"></script>
    <h1 onclick="doToggle(event)"> IWP: click mig! </h1>
  </body>
</html>
```

HTML5



3 forskellige separate  
sprog, men  
dokumenterne er delvist  
indbyrdes afhængige

```
let toggleState=true;
let oldText="";
function doToggle(event){
  let elem=event.target;
  if(toggleState){
    oldText=elem.textContent;
    elem.textContent="Hejsa Dejsa";
    toggleState=false;
  }
  else {
    elem.textContent=oldText;
    toggleState=true;
  }
}
```

JS

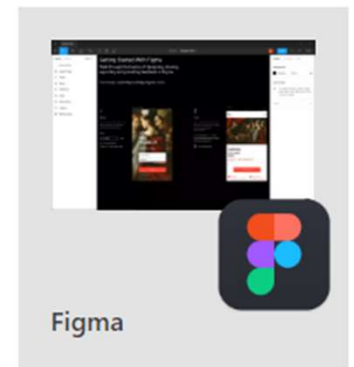
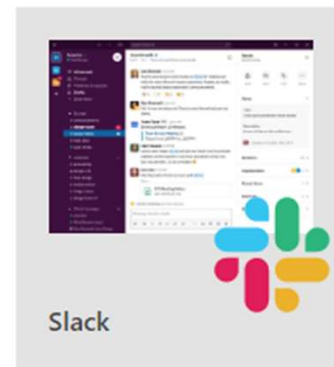
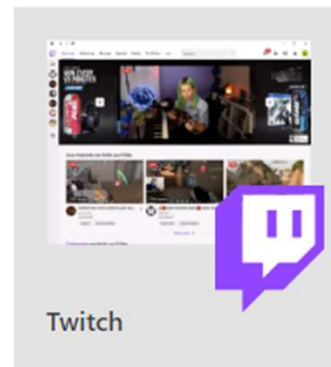
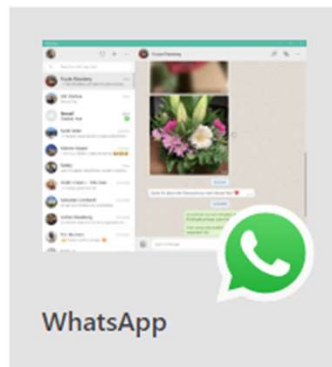
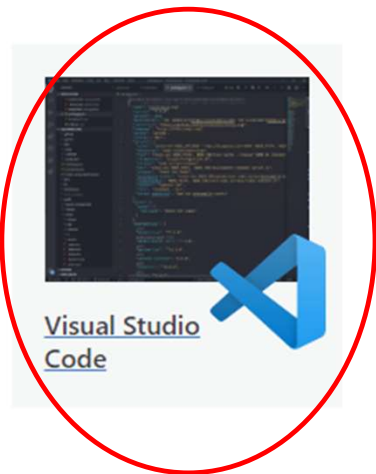
CSS

```
h1 {
  text-align: left;
  width: fit-content;
  height: auto;
  padding: 5px;
  margin-top: 10px;
  border: none;
  border-radius: 10px;
  background-color: lightskyblue;
}
```

# (JS til desktop Applikationer)



- Build cross-platform desktop apps with JavaScript, HTML, and CSS
  - <https://www.electronjs.org/>



# Back-end (server-side) teknologier

- "Development stack": Samling af sprog, biblioteker, værktøjer, databaser, designet til at arbejde sammen.
- Mange muligheder
  - LAMP (Linux-Apache-MySQL-PHP)
    - Den klassiske, velkendt, arbejdshesten bag mange web-sites.
  - WISA (Windows-IIS-Sql-ASP.net)
    - C# (.net) programmer på windows server
  - Django Stack: Python-Django-Apache-MySQL
  - Ruby-on-rails
  - Java-servlets: Java-Tomcat
  - JavaScript/node.js
    - MEAN (MongoDB-Express-Angular-Node) \*)
      - MERN: "React" framework i stedet for "Angular".
      - MEVN: "Vue" framework i stedet for "Angular".
  - ...

\*) Angular, React, Vue er front-end frameworks til udvikling af avancerede bruger grænseflader/apps.

# Internetwork og Web-programmering

## Introduktion til Web Teknologier

### JavaScript

Forelæsning 1

Brian Nielsen

Distributed, Embedded, Intelligent Systems





# Javascript

- Oprindeligt tiltænkt mindre opgaver i web-sider, hvor Java var for tungt og klodset.
  - Introduceret i Netscape browser, 1996
- JavaScript, Mocha, LiveScript, JScript, ECMAScript,
- ECMA: standardiseringskommitté
  - [6th Edition - ECMAScript 2015](#) henviser til en bestemt standardiseret version
- **JavaScript ≠ Java:** Meget forskellige sprog
  - Fuldstændigt “rigtigt” programmeringssprog (*fortolket, dynamisk, svagt typet*)
  - The good, The Bad, The Ugly
  - Understøtter imperativ, funktionsorienteret, og tildels objekt-orienteret programmering
  - Vi kan langt hen ad vejen klare os med en “imperativ” stil
  - Under stadig forbedring og udvikling:
    - Især mange forbedringer fra ES6, 2016: “Modern JavaScript”
    - Nyeste “Ecmascript 2022” [https://en.wikipedia.org/wiki/ECMAScript\\_version\\_history](https://en.wikipedia.org/wiki/ECMAScript_version_history)

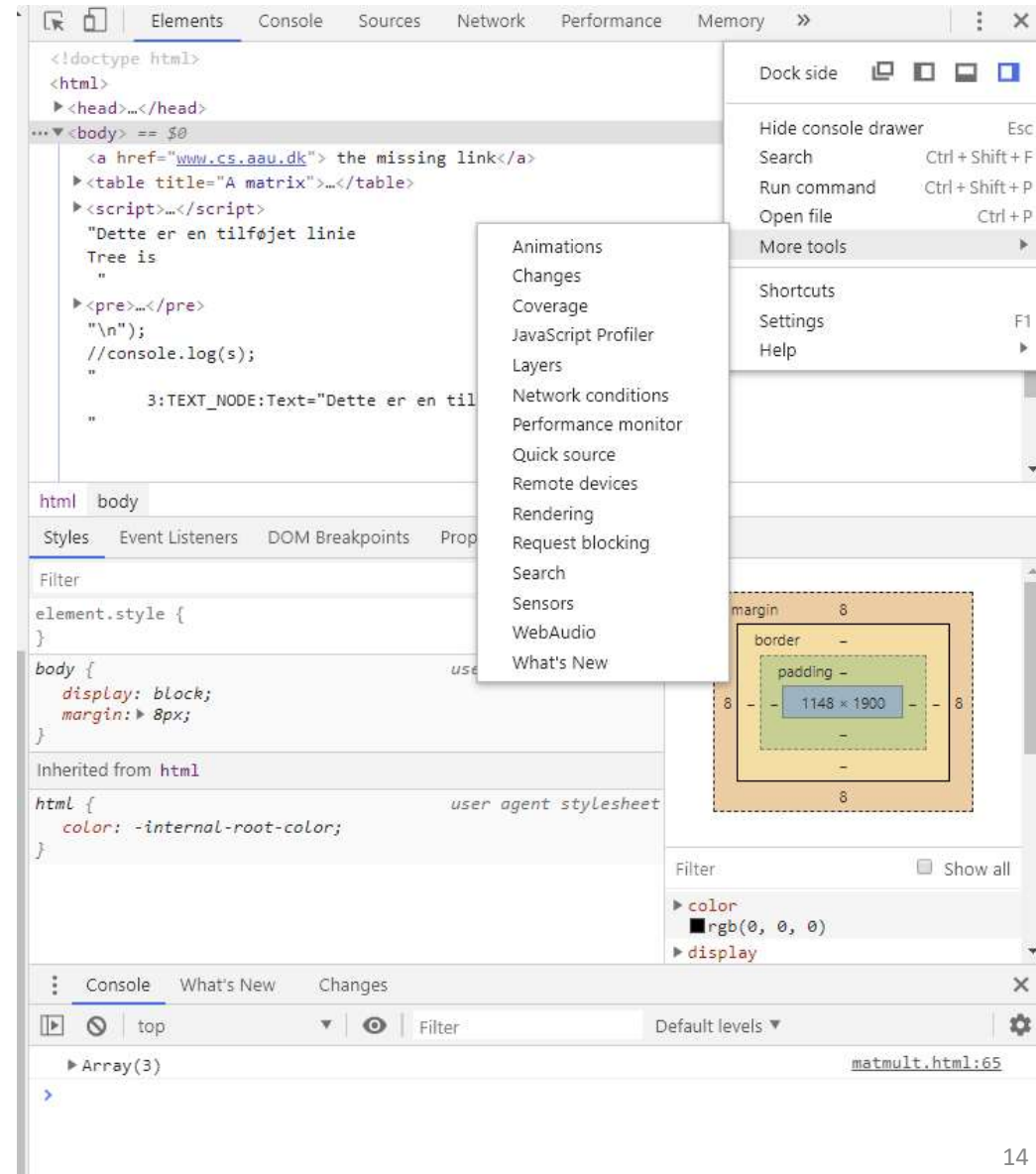
# JS i browsere

Findes som "Ctrl+Shift+i" i chrome

- Værktøjer → Developer Tools
- Inspektion og ændring af HTML og CSS
- Konsol
- Write-eval-print loop
  - Skrive og afvikle code-snips
- Debugger
- Optimering
  - Hastighed
  - Hukommelse
  - Brug af netværk
  - Profilers (flaskehalse)
- Dækningsmåling (Test)

<https://developers.google.com/web/tools/chrome-devtools>

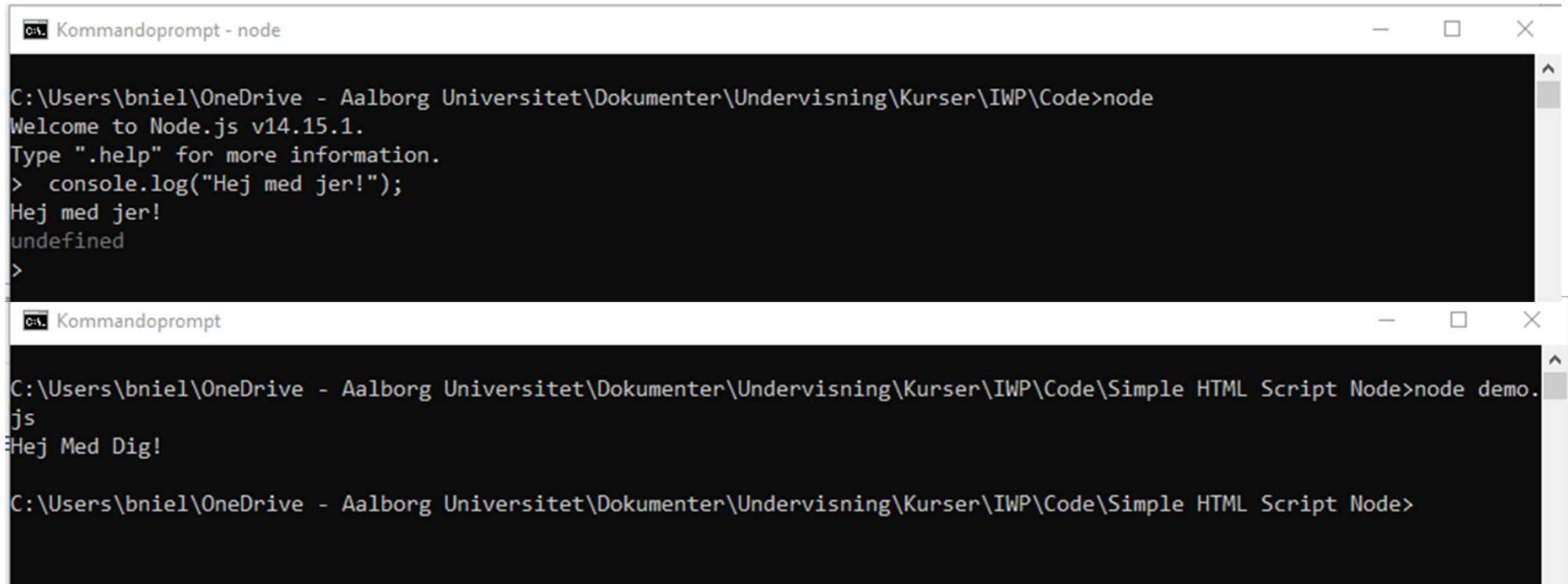
<https://javascript.info/debugging-chrome>



# Hvad bruges JS til på front-end (clients)

- Interaktivitet
  - Validering af bruger input og meningsfyldte fejlmeddelelser
  - Dynamisk fremstilling af web-side, afhængigt af brugers valg
  - Styring af GUI-elementer: sliders, menuer, pop-ups,...
  - Applikations funktionalitet
- Server-kommunikation
  - Dynamisk indlæsning af data fra server, filtreret visning
  - Opdatering af web-side uden explicit "reload"
  - Minimere server kommunikation: en del data behandling kan ske lokalt uden at bruge netværk og server.
- Program funktioner
  - Lettere beregninger og program dele, som klienten "bekvem" kan foretage lokalt
  - Funktioner den skal være tage, hvis server er "nede"

# Server side: Node.js



The image shows two screenshots of a Windows Command Prompt window. The top screenshot shows the Node.js REPL (Read-Eval-Print Loop) being started with the command 'node'. It displays the welcome message 'Welcome to Node.js v14.15.1.' and the prompt 'Type ".help" for more information.'. The user enters the command 'console.log("Hej med jer!");', and the output 'Hej med jer!' is displayed. The bottom screenshot shows the Node.js command-line interface being used to run a script. The user enters the command 'node demo.js', and the output 'Hej Med Dig!' is displayed. The command prompt window title is 'Kommandoprompt - node'.

```
C:\Users\bniel\OneDrive - Aalborg Universitet\Dokumenter\Undervisning\Kurser\IWP\Code>node
Welcome to Node.js v14.15.1.
Type ".help" for more information.
> console.log("Hej med jer!");
Hej med jer!
undefined
>

C:\Users\bniel\OneDrive - Aalborg Universitet\Dokumenter\Undervisning\Kurser\IWP\Code\Simple HTML Script Node>node demo.js
Hej Med Dig!

C:\Users\bniel\OneDrive - Aalborg Universitet\Dokumenter\Undervisning\Kurser\IWP\Code\Simple HTML Script Node>
```

- Afvikling af JS programmer
- Kan sættes op til at fungere som web-server, og håndtere requests i JS

# Hvad bruges JS til på back-end (server)

- Det samme som andre server scripting sprog
  - Modtagelse af HTTP requests fra klienter
  - Dekodning af parametre og formularer
  - Validering af modtagne data
  - Applikations funktioner og beregning (som indgår i svaret til klienter)
  - Opslag i og opdatering af databaser eller filer
  - Generering af dynamiske HTML sider
- Implementation af WEB-APIer
- Servering af filer
- Håndtering af data og funktioner, som skal være delte mellem forskellige klienter

# Visual Studio Code - Intellisense

Avanceret editor med

- Syntax highlighting
- Code-completion
- Assistanse og hints
- En stor hjælp til sprog som JS

Muligheder for fuldendelse af "rou"

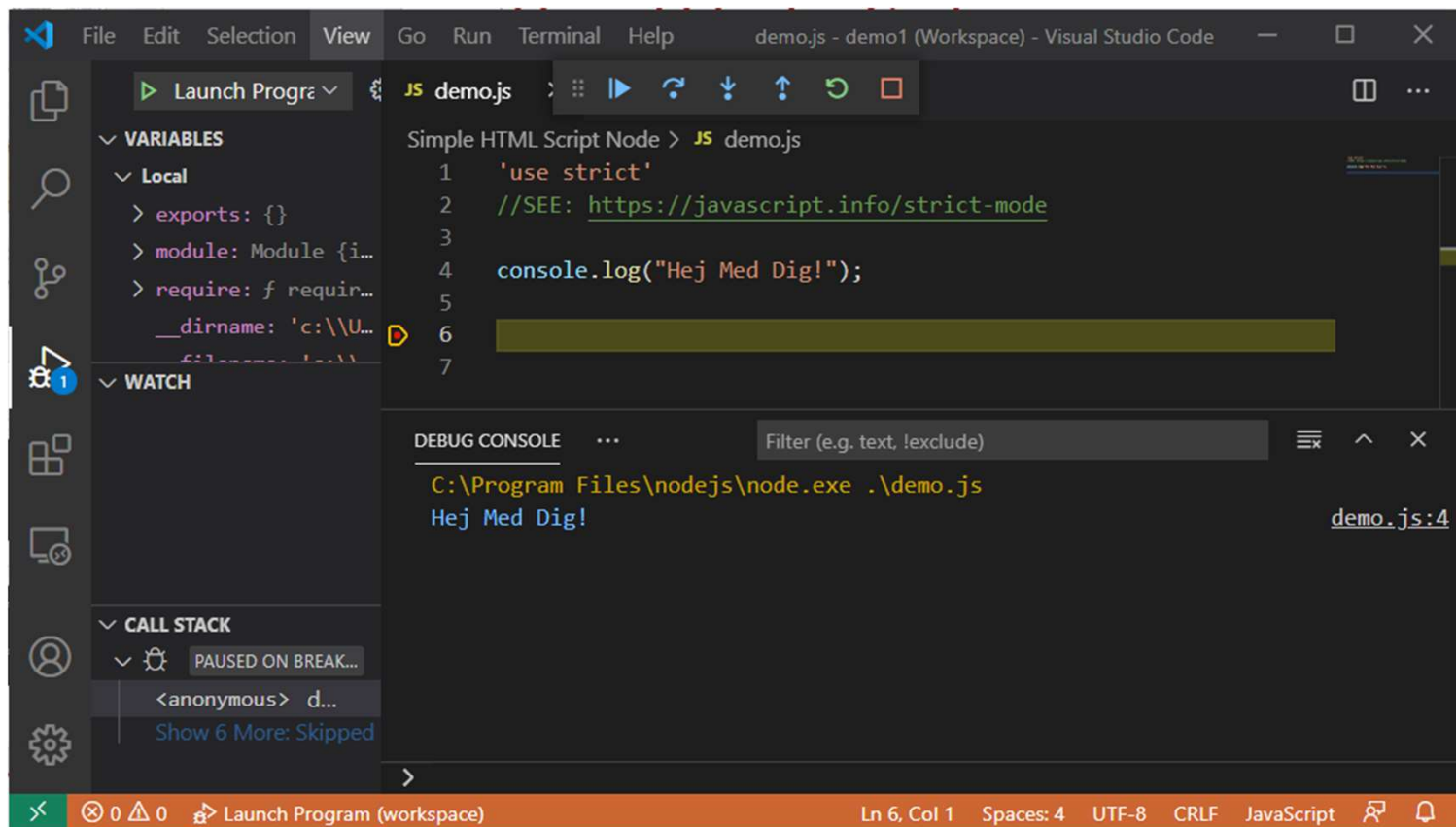
Viser hints om erklæringen af den valgte fuldendelse

```
17 const rounds={ //as C-enums doesn't directly exist in JS, we emulate it using an object
18   ones: 0,
19   twos: 1,
20   threes: 2,
21   fours: 3,
22   fives: 4,
23   sixes: 5,
24   sum: 6,
25   bonus: 7,
26   onePair: 8,
27   twoPairs: 9,
28   threeId: 10,
29   fourId: 11,
30   littleS: 12,
31   bigS: 13,
32   house: 14,
33   chance: 15,
34   yatzy: 16,
35   total: 17
36 };
37
38 function isSpecialRound(round){
39   return ((round===rounds.sum || round===rounds.bonus || ro
40 }
41
42 rou
43 con[Ⓜ] rounds
44 "Ⓜ roundsText
45 ]; abc round
46 abc roundID
47 abc roundName
48 abc roundNo
49 [Ⓜ] noRounds
50 [Ⓜ] isSpecialRound
51 [Ⓜ] playRound
52 [Ⓜ] runInContext
53 [Ⓜ] resourceUsage
54 [Ⓜ] runInNewContext
```

```
const rounds: {
  ones: number;
  twos: number;
  threes: number;
  fours: number;
  fives: number;
  sixes: number;
  sum: number;
  bonus: number;
  onePair: number;
  twoPairs: number;
  threeId: number;
  fourId: number;
  littleS: number;
  ... 4 more ...;
  total: number;
}
```

# Visual Studio Code - debugger

- Indsættelse af break-points, inspektion/ændring af variable, ...



# Demo Applikation





Multi-Yatzy



# Multi-Yatzy V1

- V1: Konsol applikation, evt. med udskrift af scoreboard til HTML fil.
- Programmeret i enkel C-lign. stil

1s	3261454135	2
2s	6563651165	0
3s	6632621344	6
4s	6243612632	4
5s	2432264366	0
6s	4213143313	0
Sum		12
Bonus		0
1 Pair	2352213551	10
2 Pairs	3261144244	12
Three Identical	5636444654	18
Four Identical	5253132314	0
Little Straight	4533646221	15
Big Straight	3635631554	0
House	5231455551	17
Chance	3552544441	23
Yatzy	2154131454	0
Total Score		107

Yatzy Scores		
Player		
Round Name	Dice	Score
1s		2
2s		0
3s		6
4s		4
5s		0
6s		0
Sum		12
Bonus		0
1 Pair		10
2 Pairs		12
Three Identical		18
Four Identical		0
Little Straight		15
Big Straight		0
House		17
Chance		23
Yatzy		0
Total Score		107

# Multi-Yatzy V2

- V2: Som dynamisk HTML baseret "Web-site"
  - Spillet foretages på server-siden, som genererer HTML sider
  - Bruger input data konfigureres gennem "HTML Formularer"
  - Spillet styres af en formular
  - Spil status overføres som HTML sider

← → ↻ 🏠 ⓘ http://127.0.0.1:3000/nextround

Configure Game:

Name

Number of Dice:

Play Game:

Yatzy Scores

**Brian Nielsen**

Round Name	Dice	Score
1s		2
2s		2
3s		3
4s		0
5s		0
6s		0
Sum		7
Bonus		0
1 Pair		0
2 Pairs		0
Three Identical		0
Four Identical		0
Little Straight		0
Big Straight		0
House		0
Chance		0
Yatzy		0
Total Score		7

# Multi-Yatzy V2

- V2: Som dynamisk HTML baseret "Web-site"
  - Spillet foretages på server-siden, som genererer HTML sider
  - Bruger input data konfigureres gennem "HTML Formularer"
  - Spillet styres af en formular

← → ↻ 🏠 ⓘ http://127.0.0.1:3000/nextround

Configure Game:

Name

Number of Dice:

Play Game:

Yatzy Scores

**Brian Nielsen**

Round Name	Dice	Score
1s		2
2s		2
3s		3
4s		0
5s		0
6s		0
Sum		7
Bonus		0
1 Pair		0
2 Pairs		0
Three Identical		0
Four Identical		0
Little Straight		0
Big Straight		0
House		0
Chance		0
Yatzy		0
Total Score		7

# Multi-Yatzy V3

- V2: Som interaktiv "Web-app"
  - Spillet foretages på server-siden
  - Server udstiller et API som klienten bruger til spillet
  - Spil data overføres via "HTTP+JSON"
  - Klienten opdaterer siden, fx drag'n drop af terninger


← → ↻ ⌂ ⓘ http://127.0.0.1:3000

## IWP Yatzy Game

Configure Game:

Name  Number of Dice:

Play Game:



Yatzy Scores

Peter		
Round Name	Dice	Score
1s		0
2s		
3s		
4s		
5s		
6s		
Sum		0
Bonus		0
1 Pair		
2 Pairs		
Three Identical		
Four Identical		
Little Straight		
Big Straight		
House		
Chance		
Yatzy		
Total Score		0

This program is inspired by Kurt Normarks exam assignment for C++

Lidt intro til JS sproget

# Variabel Erklæringer - Bindinger

```
let x=0; // der laves en binding mellem navnet x og værdien 0
let y;   //uden initialisering bindes til den specielle værdi undefined
console.log(y); //undefined

y=x+1;   //y bindes til resultatet af udtrykket x+1
console.log(y); //1

const minDice=1; //en konstant
```

- Brug nøgleordet **let** til at definere en "binding" (oprette variabel)
- En **const** forbliver bundet til samme værdi
- Bemærk: ingen eksplicitte type erklæringer!!
- Typer udledes på køretid.
- Primitive typer: Tal, Booleans, Streng

# Lidt om Streng

```
let roundName = "Yatzy";
let cell1=<td class=\"left-text\"> + roundName + '</td>';
let cell2= `<td class="left-text"> ${roundName}</td>`;
//<td class="left-text">Yatzy</td>
cell1[4]; //"c"
cell1.includes("text"); //true
let textStart=cell1.indexOf("text"); //16

let smiley="\u263A"; //☺
```

- En streng er indbygget type i JS (ikke som i C et array)
- Både " " og single tick ' ' angiver en streng værdi (bør anvende samme stil i samme program)
- Special tegn escapes med \
- + giver en konkatenering
- Back-tics ` ` angiver en *template literal*
  - kan indeholde pladsholdere til program generet tekst i `${expr}`
  - kan deles over flere linier, og bruge visse specialtegn uden escape af fx " (tegnene \${} skal dog)
- Tegn kodes i "UTF-16"-format
  - "Teknikalitet": nogle eksotiske tegn kræver 2 koder ('A\uD87E\uDC04Z' er 4 koder lang, men giver 3 tegn tekst A你Z)
- Et enormt arsenal af metoder på streng værdier, se pensum

# Typer og type konvertering

- Hvis operanders type ikke matcher, forsøger JS at lave en meningsfyldt type konvertering
  - Fx: `+` mellem tal og streng giver en `+` (konkatenering) mellem tallet som streng
- Særlige funktioner findes til eksplicit konvertering
- Brug normalt `===` til sammenligning, checker for matchende type og værdi
- Operatoren `==` sammenligner værdier efter typekonvertering ("deprecated")

```
let x=0;
let s="Hej";
s=s+x;
console.log(s); //"Hej0"

let a; //undefined
let b=a+1; //får den særlige værdi NaN ("not a number")

//eksplicit konvertering
x=Number("6"); //heltalsværdien 6
s=String(7); //strengværdien "7"

if( 6 == "6") //true
  console.log("true");

if(6==="6") //false: types doesn't match
  console.log("true");
else
  console.log("false");
```



# Køretidsfejl

- Et JS program oversættes ikke, men fortolkes
- Kun grove syntax fejl findes inden programmet startes, fx manglende } eller ”
- Smart:
  - hurtig udvikling,
  - kørsel af ufærdige programmer
  - Interaktive ”skriv-og-evaluer”
- Ulempe:

```
let roundNo=0;  
let q=roundno;      //Køretidsfejl: Reference Error: roundno is not defined
```

- Programmet stopper med TRÆLSE fejl, fx som følge af simple stavefejl
- Kører videre med ”undefined” værdier
  - IntelliSense hjælper dog lidt
  - [ESLint analyse værktøj](#) (kræver installation af værktøjet og evt. extension til VSCode)

# Et fragment fra Multi Yatzy V1 (C-lign. JS)

- En terning er et tal 1..6
- Et "kast" er et array af terning-værdier
- Bemærk funktions-erklæring
- Bemærk at arrays er dynamiske!
  - Elementer oprettes som de indsættes!
- JS har mange andre former for iteration ud over `for`, `while`

```
const minDice=1;
const maxDice=6; //min and max value of a normal dice

//returns an array that represents the outcome of rolling M dice
///e.g diceRoll [1,6,5,5,2]
function roll(M){
    let diceRoll=[]; //empty array
    for(let i=0;i<M;i++){
        diceRoll[i]= random(minDice,maxDice);
    }
    return diceRoll;
}
```