

Variabler og datatyper

Datamatiker

```
demo > JavaScript </pre>
<h1>Hello</h1>
```

Agenda

- Formål
- Variabler: const, let & var
- Scopes, globale & lokale variabler
- Datatyper
- Konvertering,
konkatenering &
sammenligning

- “use strict;”
- Working with variables
- Hello JavaScript
- CRUD App
- Name things right (or wrong?)
- Datatypes
- String quotes
- Numeric conversion
- String conversion
- Numeric conversion 2
- Concatenation looks like addition
- Type Conversion



Øvelser

Well done



“

The cool thing about JavaScript is
that you can create stuff that will put
a smile on your face, as a developer.

You can create stuff and it will
visually look like something, and it'll
do stuff, and it makes you feel good,
it makes you fall in love with
programming.

”

Lex Fridman
Computer Scientist

<https://www.youtube.com/watch?v=GLhyjVZp0cw&t=252s>

Computer science student



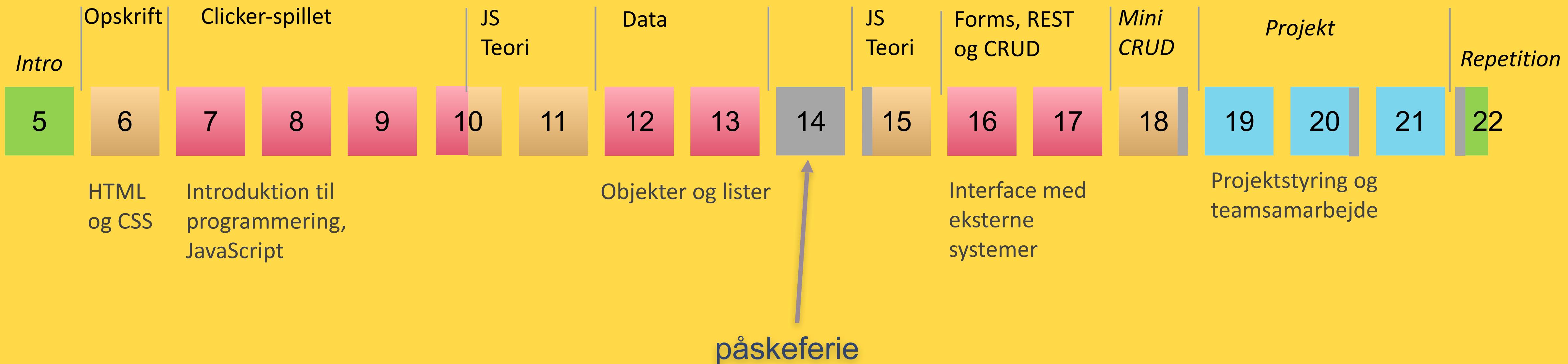
Senior developer, 10+ years experience



<https://www.instagram.com/p/BxWAgatgSmn/>

1. SEMESTERS STRUKTUR

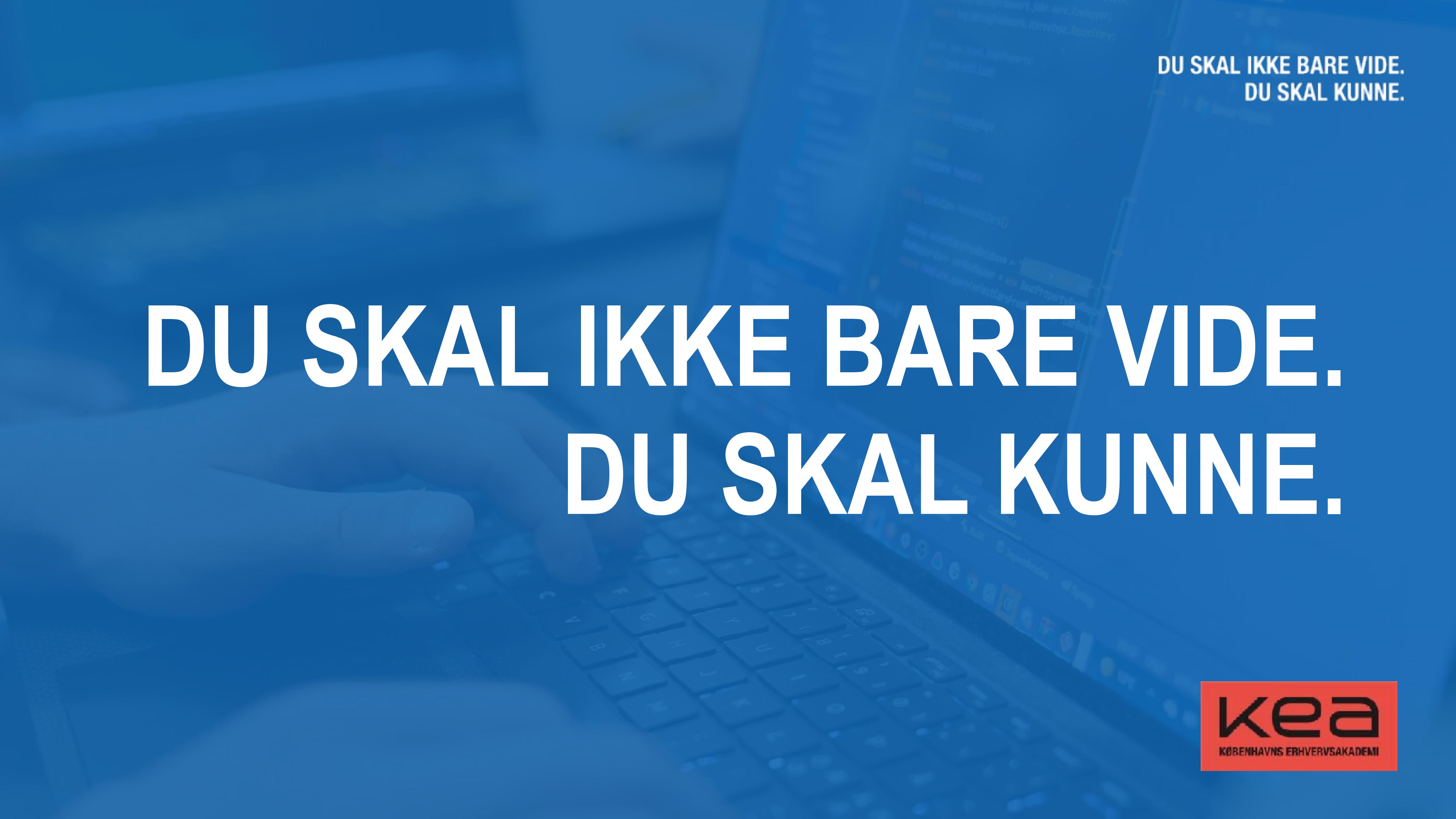
Semesteret består af 16 undervisningsuger (inkl. en masse helligdage)



Eksamens dato: 12.-14. juni

Formål

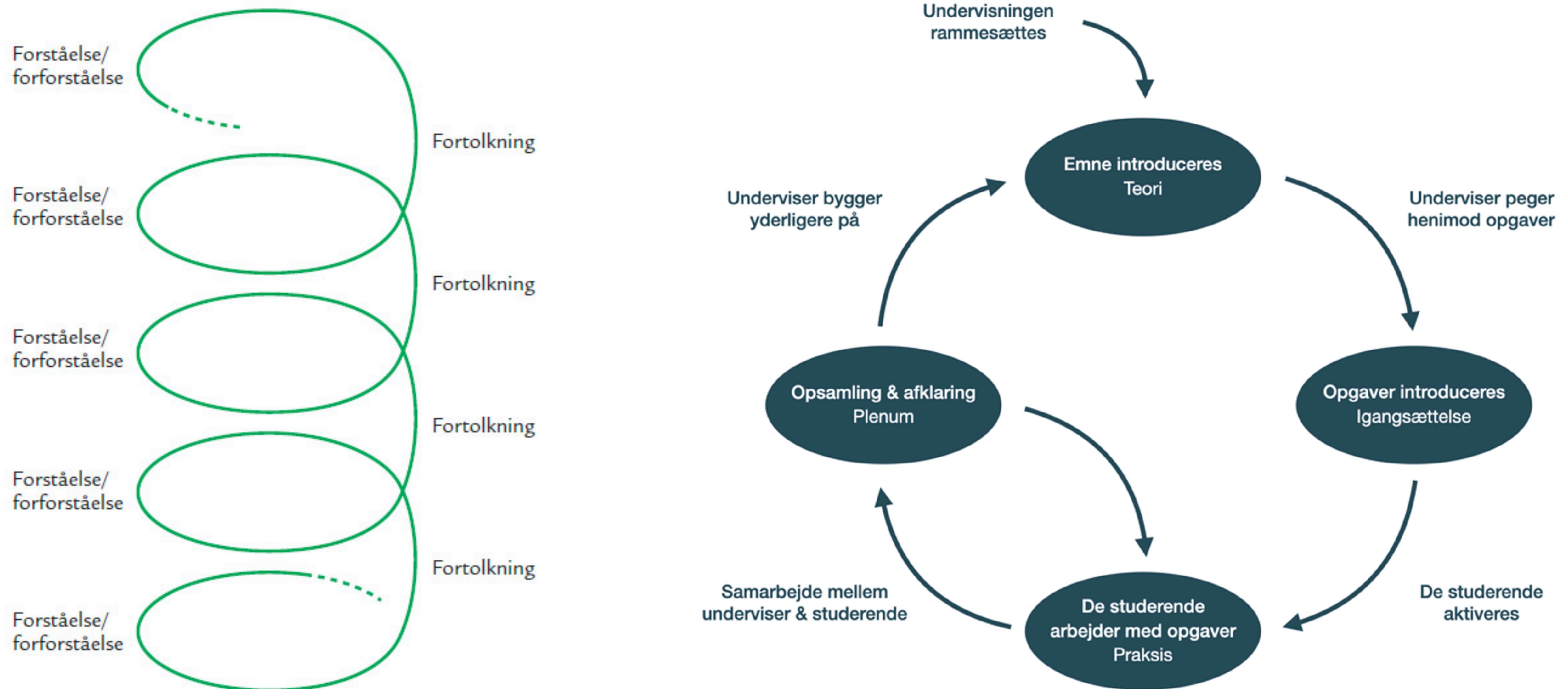
- Opsamling og dybere viden om variabler og datatyper
- Viden om konvertering og konkatenering
- Hands-on på mindre opgaver



**DU SKAL IKKE BARE VIDE.
DU SKAL KUNNE.**

**DU SKAL IKKE BARE VIDE.
DU SKAL KUNNE.**

kea
KØBENHAVNS ERHVERVSAKADEMII



Programming knowledge

JavaScript

```
index.html
85  }
86
87  /*
88   * Fetches post data from my headless CMS
89  */
90  function getPersons() {
91    fetch('http://headlesscms.cederdorff.com/wp-json/wp/v2/posts?_embed=true')
92      .then(function(response) {
93        return response.json();
94      })
95      .then(function(persons) {
96        appendPersons(persons);
97      });
98  }
99  /*
100  Appends json data to the DOM
101 */
102 function appendPersons(persons) {
103   let htmlTemplate = '';
104   for (let person of persons) {
105     console.log();
106     htmlTemplate += `
107       <article>
108         
109         <h4>${person.title.rendered}</h4>
110         <p>${person.acf.age} years old</p>
111         <p>Hair color: ${person.acf.hairColor}</p>
112         <p>Relation: ${person.acf.relation}</p>
113       </article>
114     `;
115   }
116   document.querySelector("#family-members").innerHTML += htmlTemplate;
117 }
118
119 // Main entry point
120 main();
```

Variabler?

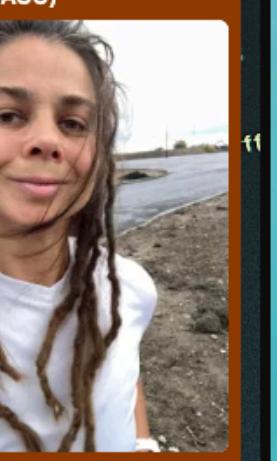
Hjem er I?
Dat23v1

Rasmus Cederdorff (RACE)

Ham med armen
Alder: Fyldte 30 dage efter Mette lukkede landet 😊🎉
By: København NV (fra Aarhus)
Baggrund - uddannelse & Job:
Datamatiker, PBA Webudvikling & Cand.it - Webarkitektur
Freelance Web App Developer & Lektor på EAAA
Har du programmeret (bestemt ikke et krav): Webslinger og Apps - Masser af JavaScript
Hvorfor Datamatiker:
Kombinationen mellem teori og praksis. Praksisnær undervisning.
Forventninger og ønsker til Datamatikeruddannelsen:
Klæde jer bedst muligt på til den virkelige verden.
Drømmejob og fremtid: Senior Web Developer at Apple Inc
Interesser: Når jeg ikke tænker i JavaScript, forsøger jeg at lære

Peter Lind (PETL)

Alder: Fylder 50 inden I er halvejs færdige med uddannelsen 😊🎉
By: Tastrup (fra Kolding a)
Baggrund - uddannelse & Job:
BSc.EE, og master i IKT og læring - arbejdet med webudvikling siden 1996, med lidt pauser med et væld af halve uddannelser undervejs. Undervist på KEA i snart 10 år ... chok ...
Har du programmeret (bestemt ikke et krav): Webslinger og Apps - Masser af JavaScript
Hvorfor Datamatiker: Jeg vil gerne undervise folk i at blive gode programmerer - programmerer der har det sjovt med at skabe programmer i stedet for bare at

Lenka (Magdalena Maria Otap / MAGO)

Alder: Fylder 50 inden I er halvejs færdige med uddannelsen 😊🎉
By: Tastrup (fra Kolding a)
Baggrund - uddannelse & Job:
Den dag jeg løb alle veje Københavns kommune færdig
Alder: 14+ (det er ingen alder for en drage)
By: kbh, østerbro
Baggrund:
Levet at fåne folks smerten i led og muskler.
Har du tidligere programmeret?: Ja. Mest PHP og Python, men været inde over lidt af hvert.
Hvorfor Datamatiker: Jeg har kun ligge dyptet tærene hos Code Academy og noget python på Udemy.com

Yaw Boateng

Alder: 32
By: København NV, Danmark, Jorden, Mælkehøjene.
Baggrund:
Levet at fåne folks smerten i led og muskler.
Har du tidligere programmeret?: Ja. Mest PHP og Python, men været inde over lidt af hvert.
Hvorfor Datamatiker: God chef, tilfældige valg.

Ziggy Lange (ZIGMIGHSTER)

Alder: 28 år (sommerbarn - Cancer)
By: København F (Frederiksberg hehe)
Baggrund - uddannelse & Job:
Spansk sprog og Kultur, KU Risteassistent i et kaffefirma Tidligere cykelbud i DKS to største byer uha uha

Malte Mørkeberg Sørensen

Alder: 26
By: København NV, Tingbjerg
Baggrund - uddannelse & Job:
HF, kundeservice i HBO Max og tidligere cykelbud i Wien, Østrig.
Har du programmeret?: Nope, har lavet lidt opgaver på Python, men intet jeg som sådan kan huske nu.
Hvorfor Datamatiker?: Fordi jeg er interesseret i computere og gerne vil lære at programmere. Og så gør det heller ikke noget at der er gode muligheder for arbejde efter.

Forventninger og ønsker til Datamatikeruddannelsen: Forventer at kunne programmere efter - i hvert fald en smule.

Dremmejob og fremtid: Jeg er lidt for praktisk anlagt til at gå at 'drømme' om fremtiden, men jeg tror

Forventninger og ønsker til

Rasmus Cederdorff + 35 + 3d
Hjem er I?
Dat23v1

Hjem er I?
Dat23v2

Rasmus Cederdorff (RACE)

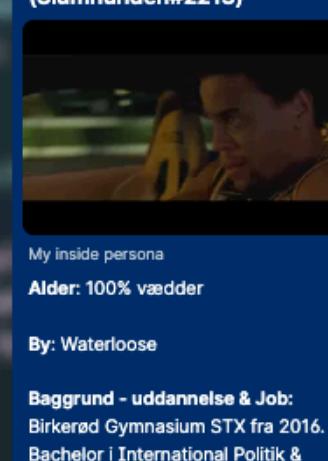
Ham med armen
Alder: Fyldte 30 dage efter Mette lukkede landet 😊🎉
By: København NV (fra Aarhus)
Baggrund - uddannelse & Job:
Datamatiker, PBA Webudvikling & Cand.it - Webarkitektur
Freelance Web App Developer & Lektor på EAAA
Har du programmeret (bestemt ikke et krav): Webslinger og Apps - Masser af JavaScript
Hvorfor Datamatiker:
Kombinationen mellem teori og praksis. Praksisnær undervisning.
Forventninger og ønsker til Datamatikeruddannelsen:
Klæde jer bedst muligt på til den virkelige verden.

Peter Lind (PETL)

Alder: Fylder 50 inden I er halvejs færdige med uddannelsen 😊🎉
By: Tastrup (fra Kolding a)
Baggrund - uddannelse & Job:
BSc.EE, og master i IKT og læring - arbejdet med webudvikling siden 1996, med lidt pauser med et væld af halve uddannelser undervejs. Undervist på KEA i snart 10 år ... chok ...
Har du programmeret (bestemt ikke et krav): Webslinger og Apps - Masser af JavaScript
Hvorfor Datamatiker: BASIC og Commodore 64 assembler - og Pascal, C, Ada, VHDL, C++, Java, PHP, ActionScript, ASP, C#, Python, JavaScript, Rust ... og lidt ekstra

Lenka (Magdalena Maria Otap / MAGO)

Alder: 100% vædder
By: Waterloose
Baggrund - uddannelse & Job:
Birkerd Gymnasium STX fra 2016. Bachelor i International Politik & Økonomi 2020. Tidligere arbejdet som Onboarding & Training Specialist hos 3 Mobil
Har du programmeret (bestemt ikke et krav): Ikke en eneste linje kode.
Hvorfor Datamatiker: Dollars, fleksibilitet ift. arbejde.

Lucas Bastin (Slumhunden#2213)

Alder: 100% vædder
By: Waterloose
Baggrund - uddannelse & Job:
Birkerd Gymnasium STX fra 2016. Bachelor i International Politik & Økonomi 2020. Tidligere arbejdet som Onboarding & Training Specialist hos 3 Mobil
Har du programmeret (bestemt ikke et krav): Ikke en eneste linje kode.
Hvorfor Datamatiker: Dollars, fleksibilitet ift. arbejde.

Nikolaj Christian Møller

Alder: 31 i april 2022
By: København SV (kommer fra Virum)
Baggrund - uddannelse & Job:
Har arbejdet i mange år i servicebranchen, læst fys i ét semester og læst påruc i 2 år.
Har du programmeret (bestemt ikke et krav): Aldrig
Hvorfor Datamatiker: Har altid været ved en computer, og er god til at holde ting i systemer. Vil have en uddannelse som jeg kan blive god til.

Jack Valentin Winther Hansen

Alder: 33
By: Slangerup
Baggrund - uddannelse & Job:
Anlægsgartner
Har du programmeret (bestemt ikke et krav): Lidt HTML, CSS, python og C
Hvorfor Datamatiker: Erhvervsorienteret IT ud.
Forventninger og ønsker til Datamatikeruddannelsen: Job
Dremmejob og fremtid: Cloud Solution Architect
Interesser: Spiller sygt meget WoW - når min pige på 27 år ikke kræver min opmærksomhed.
Forventninger og ønsker til

Rasmus Cederdorff + 28 + 3d
Hjem er I?
Dat23v2

```
fullName = "Peter Lind"
```

```
initials = "petl"
```

```
age = 39
```

```
city = "Taastrup"
```

```
background = "BSc.EE. og master  
i IKT og læring ..."
```

```
image = "https://  
share.cederdorff.com/images/  
petl.webp"
```

```
...
```



Peter Lind (PETL)

Alder: Fylder 50 inden I er
halvvejs færdige med
uddannelsen 😊🎉

By: Taastrup (fra Kolding a)

Baggrund - uddannelse & Job:
BSc.EE. og master i IKT og
læring - arbejdet med
webudvikling siden 1996, med
lidt pauser med et væld af
halve uddannelser undervejs.
Undervist på KEA i snart 10 år
... chok ...

Har du programmeret
(bestemt ikke et krav): BASIC
og Commodore 64 assembler -

Variables

... are used to store data (values, objects, collections) in the memory

Variables

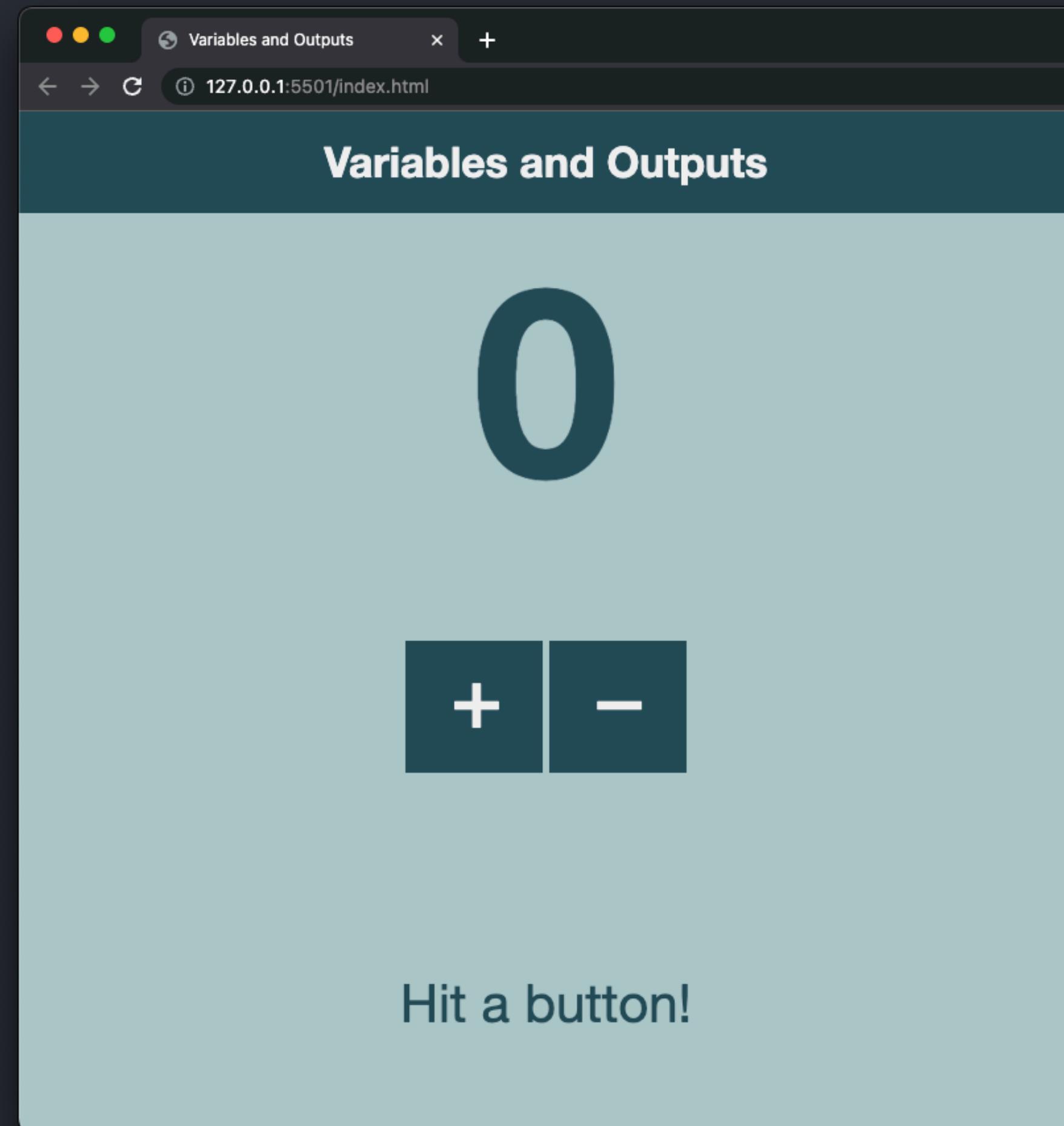
... a simple way to store, get and set data in
your code.

Variables

- Hvilke variabler har du anvendt i Clicker-spillet? Og hvor?
- Brug to minutter på at gennemse din kode og noter dig anvendelsen af variabler.
- Forklar dernæst sidemakkeren om din anvendelse af variabler og hvorfor du havde brug for variabler.

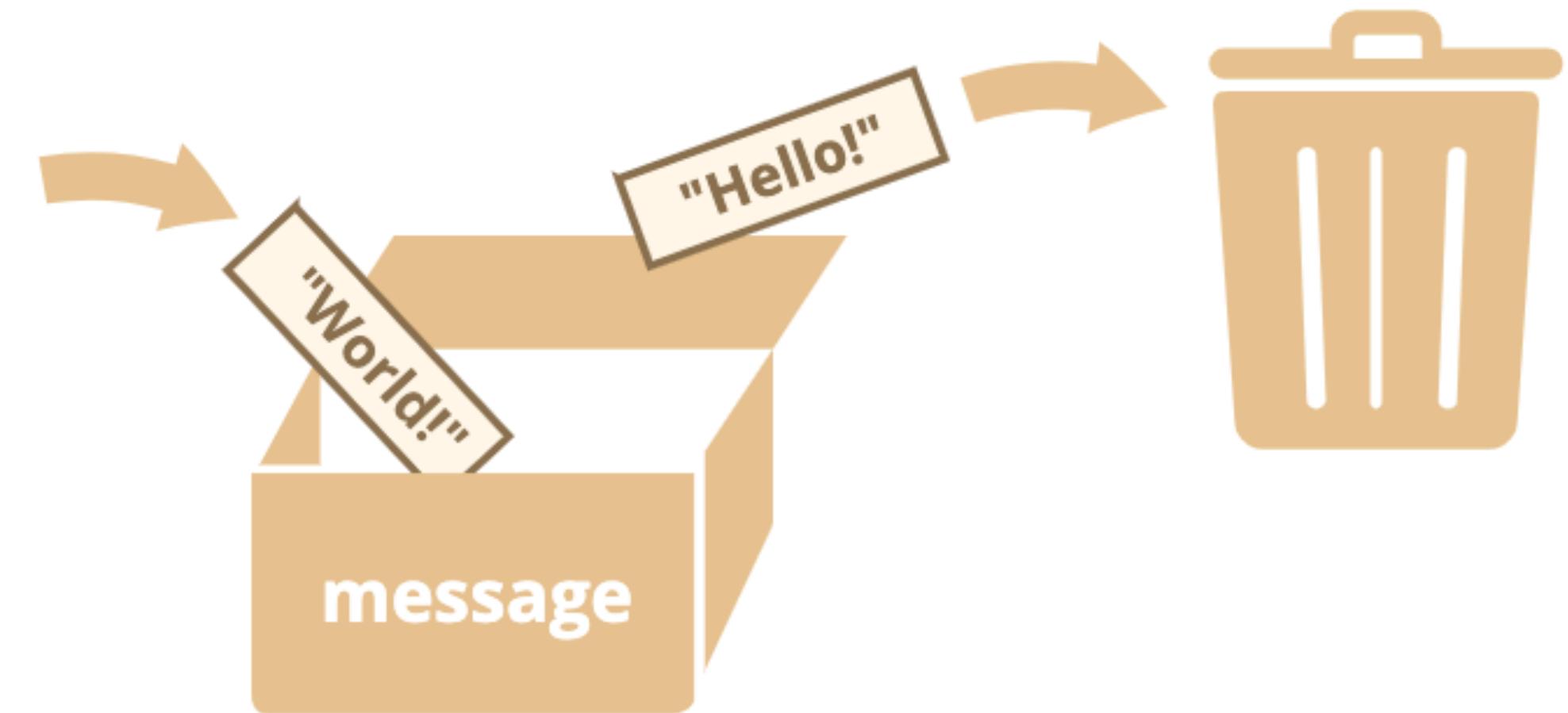
“Tælle” på en variabel

```
let variableName = 0;  
variableName++; //lægger 1 til den nuværende værdi  
variableName--; //trækker 1 fra den nuværende værdi  
variableName += x; //lægger værdien x til den nuværende værdi  
variableName -= x; //trækker værdien x fra den nuværende værdi
```



Variable

A variable is a “named storage” and is stored in the memory of the browser.



We can change the value of the variables as many times as we want.

`"use strict";`

Enables all features of modern JavaScript

<https://javascript.info/strict-mode>

```
"use strict";  
  
// your awesome JavaScript  
// code goes here ...
```

- “use strict”;**
- Add it to the top of your script.
 - ALWAYS!
 - It helps us
 - Gives us more errors and helps us to debug our code.

"use strict";

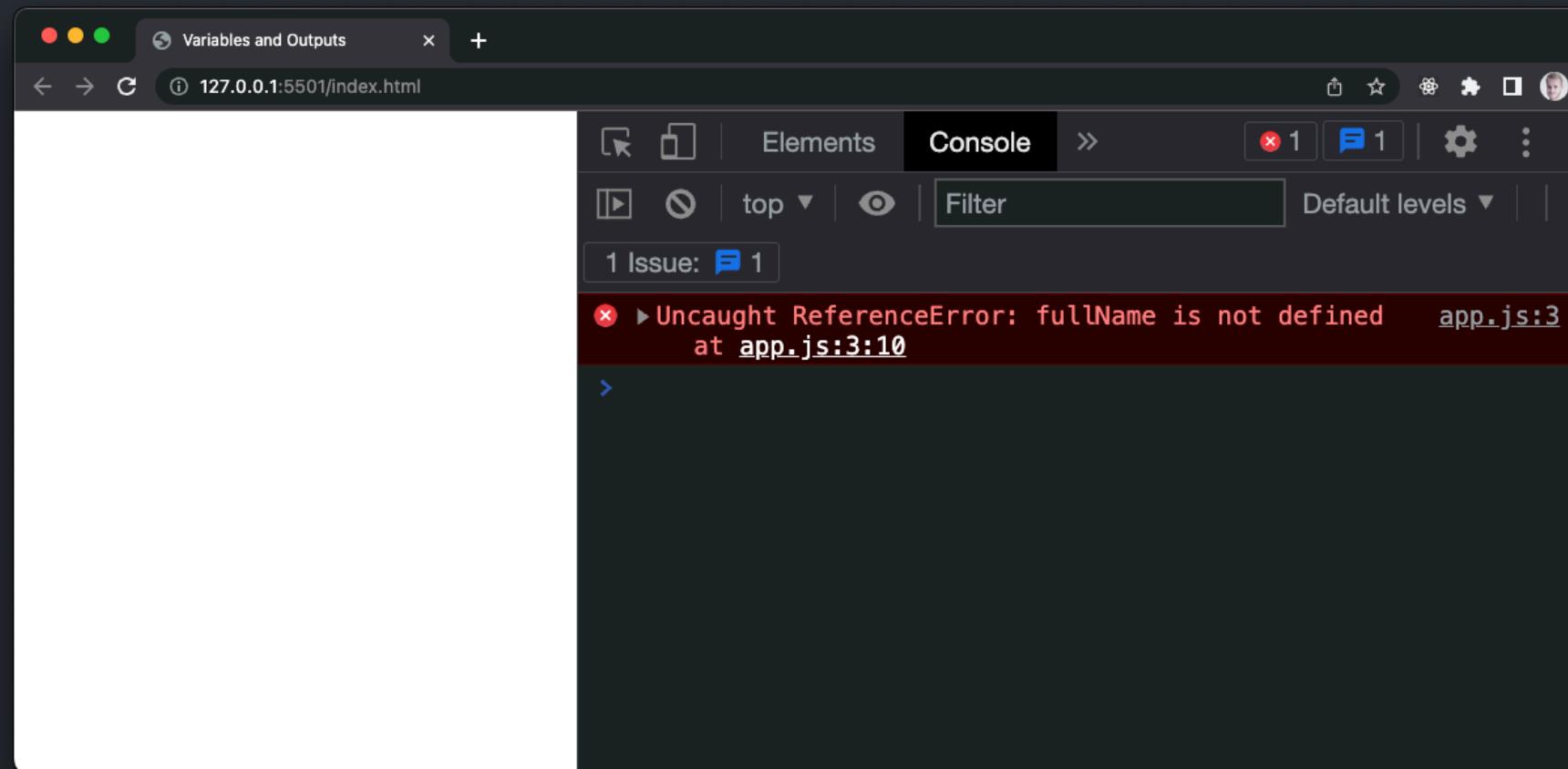
1. Run the code and see what happens.
2. Remove "use strict"; and see what happens.
3. Insert "use strict"; and see what happens.
4. Remove the keyword let and see what happens

```
"use strict";
```

```
let fullName = "Peter Lind";  
console.log(fullName);
```

```
"use strict";
```

```
// your awesome JavaScript  
// code goes here ...
```



We must declare our variables using keyword `let`, `const` or `var`

The keyword is only used for declaration



app.js — variables

JS app.js X

JS app.js > ...

...

```
1 "use strict";
2
3 let fullName = "Peter Lind"; // declare with value
4 console.log(fullName); // test in console
5 fullName = "Peter Hansen"; // assign a new value
6 console.log(fullName); // test new value
7
```

⊗ 0 ⚠ 0 Spaces: 4 UTF-8 LF {} JavaScript ⚡ Port : 5501 ✓ Prettier ⌂ ⌂

The screenshot shows a code editor window titled "app.js — variables". The file contains the following JavaScript code:

```
"use strict";
let fullName = "Peter Lind"; // declare with value
console.log(fullName); // test in console
fullName = "Peter Hansen"; // assign a new value
console.log(fullName); // test new value
```

The code editor has a dark theme with syntax highlighting. The variable declaration "let" is highlighted in purple, while the assignment operator "=" is highlighted in green. The code is annotated with comments in gray text:

- // declare with value
- // test in console
- // assign a new value
- // test new value

The status bar at the bottom of the editor shows various settings: "Spaces: 4", "UTF-8", "LF", "{} JavaScript", "Port : 5501", "Prettier", and icons for file operations.

The keyword is only used for declaration



The screenshot shows a code editor window titled "app.js — variables". The file contains the following code:

```
JS app.js
JS app.js > ...
...
1 "use strict";
2
3 let fullName; // declare with no value
4 console.log(fullName); // test in console
5 fullName = "Peter Lind"; // assign a value
6 console.log(fullName); // test value
7
```

The code editor interface includes a sidebar with icons for file operations, a status bar at the bottom with file statistics and encoding information, and a toolbar with various icons.

Why let and why “use strict”

The screenshot shows a dark-themed code editor window titled "app.js — variables". The code editor displays the following JavaScript code:

```
1 "use strict";
2
3 let fullName = "Peter Lind";    Cannot redeclare block-scoped variable 'fullName'.
4 console.log(fullName);
5 let fullName = "Hanne Hansen";  Cannot redeclare block-scoped variable 'fullName'.
6
```

The third and fifth lines both declare a variable named `fullName`, which is highlighted in purple and underlined with a red wavy line, indicating a syntax error. The error message "Cannot redeclare block-scoped variable 'fullName'." is displayed in red next to each declaration.

The status bar at the bottom of the code editor shows the following information: Ln 5, Col 29, Spaces: 4, UTF-8, LF, {} JavaScript, Port : 5501, Prettier checked. There are 2 issues and 0 warnings.

To the right of the code editor is a "Console" tab showing one error message:

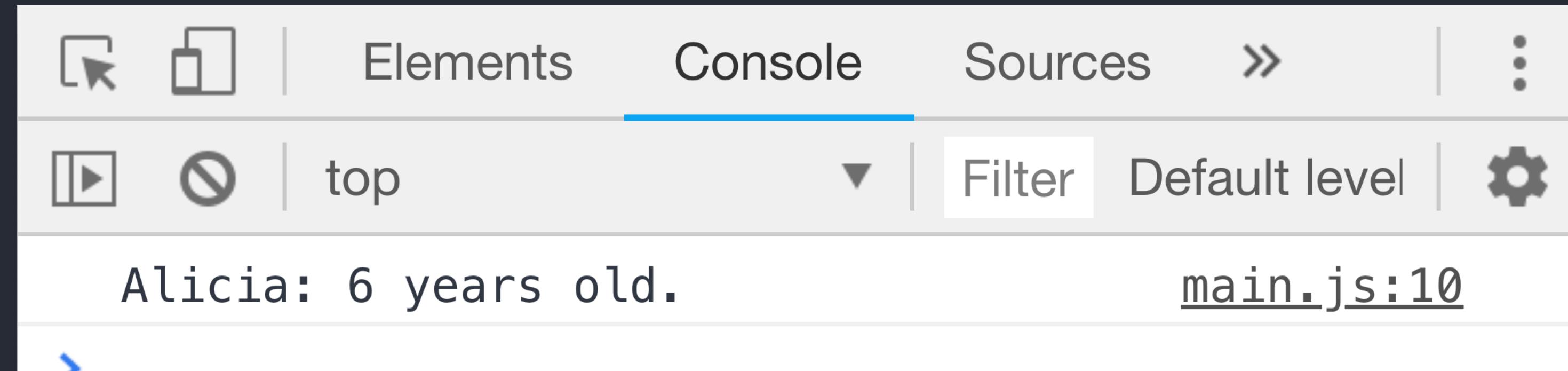
```
✖ Uncaught SyntaxError: Identifier 'fullName' has already been declared (at app.js:5)
```

Variables

Store data in the memory

```
let name = "Alicia";
let age = 6;

console.log(name + ": " + age + " years old.");
```

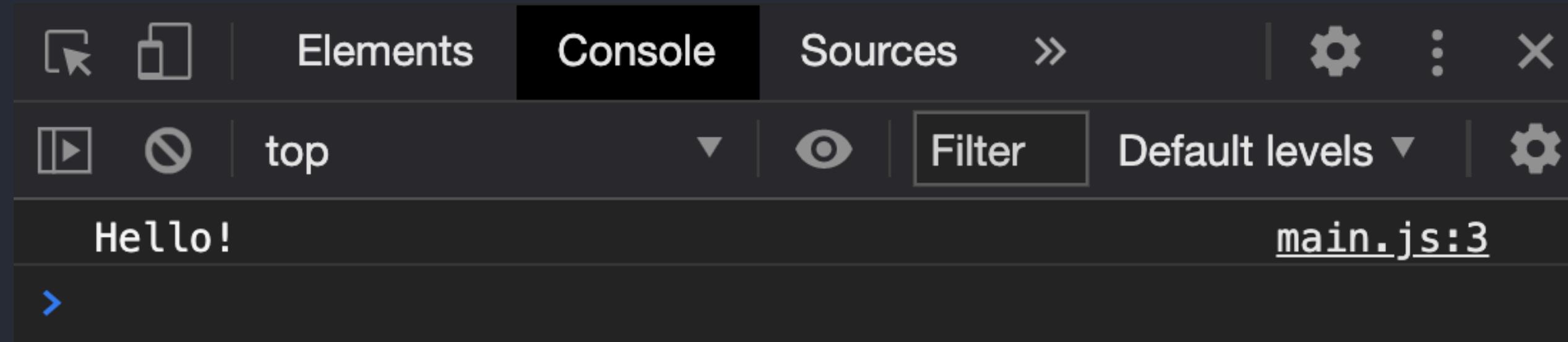


Variables

Store data in the memory

```
let message = "Hello!";
```

```
console.log(message);
```

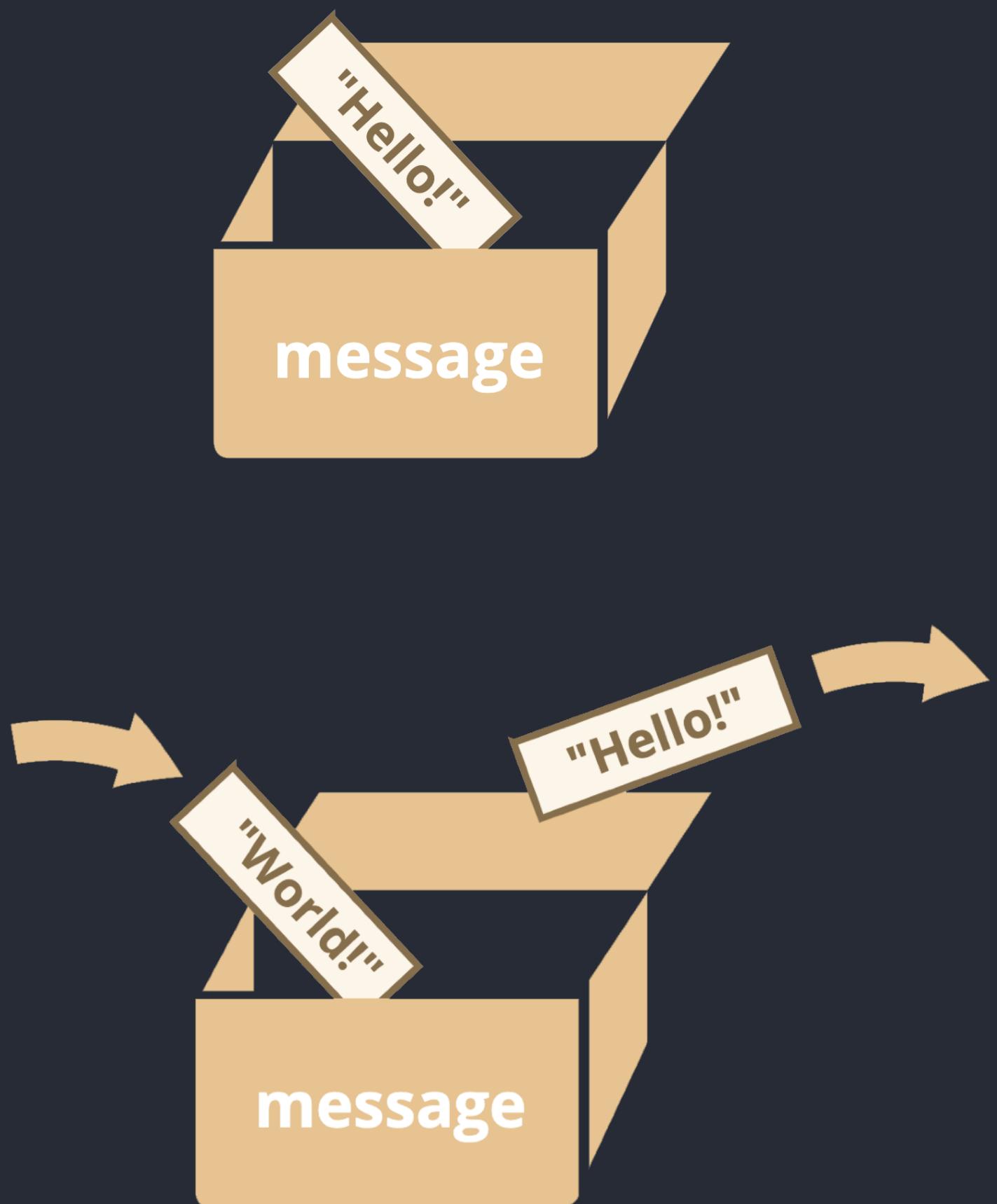
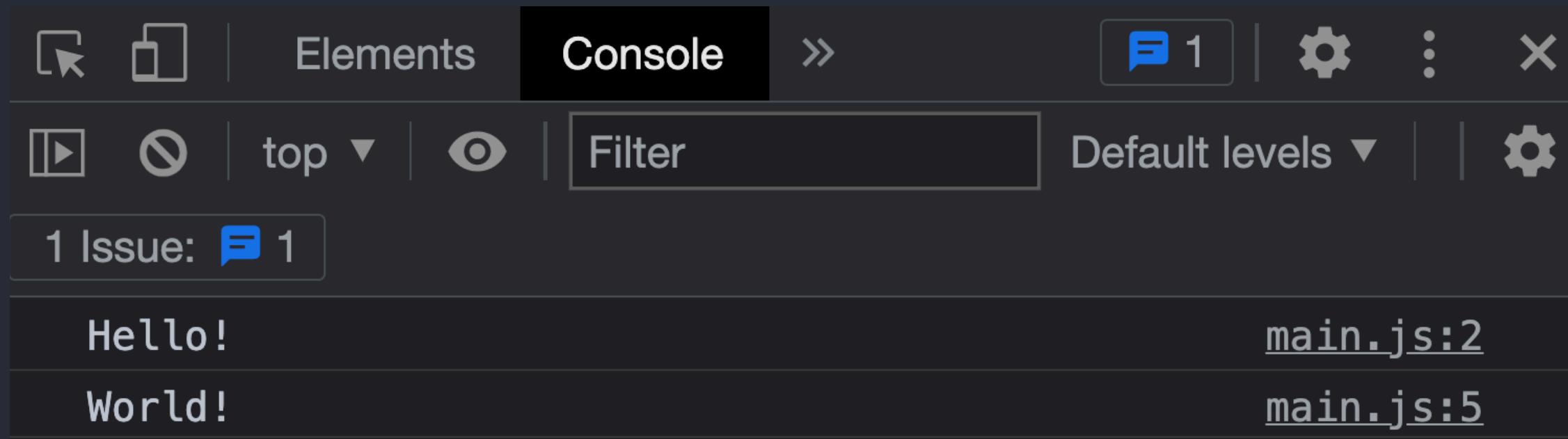


Variables

Store data in the memory

```
let message = "Hello!";
console.log(message);
```

```
message = "World!";
console.log(message);
```

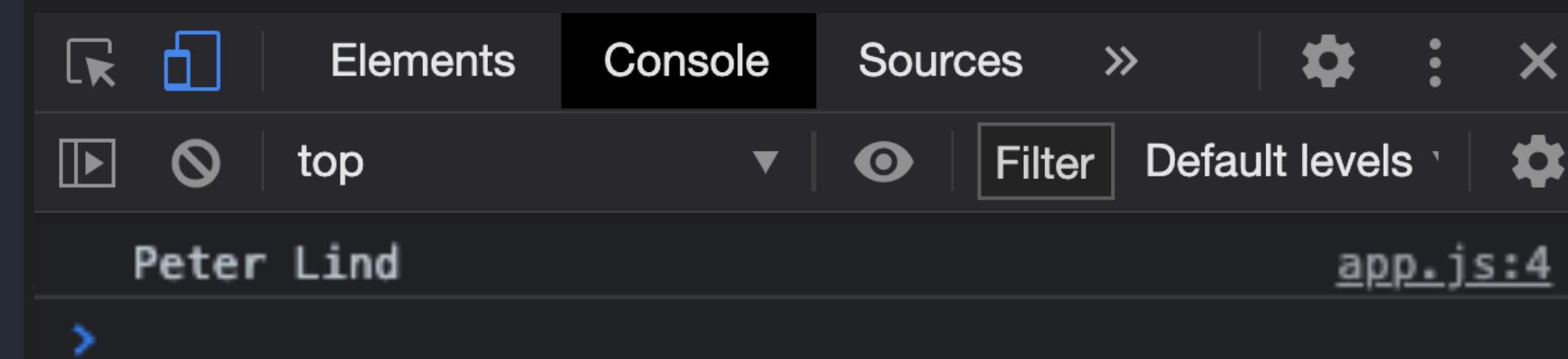


Working with variables

1. Declare two variables: admin and firstName.
2. Assign the value "John" to firstName.
3. Copy the value from firstName to admin.
4. Show the value of admin using console.log (must output "John").

Variables & UI

```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```



Variables & UI

```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```

```
<body>
  <header>
    <h1 id="fullName_container"></h1>
  </header>
  <script src="app.js"></script>
</body>
```



Variables & UI

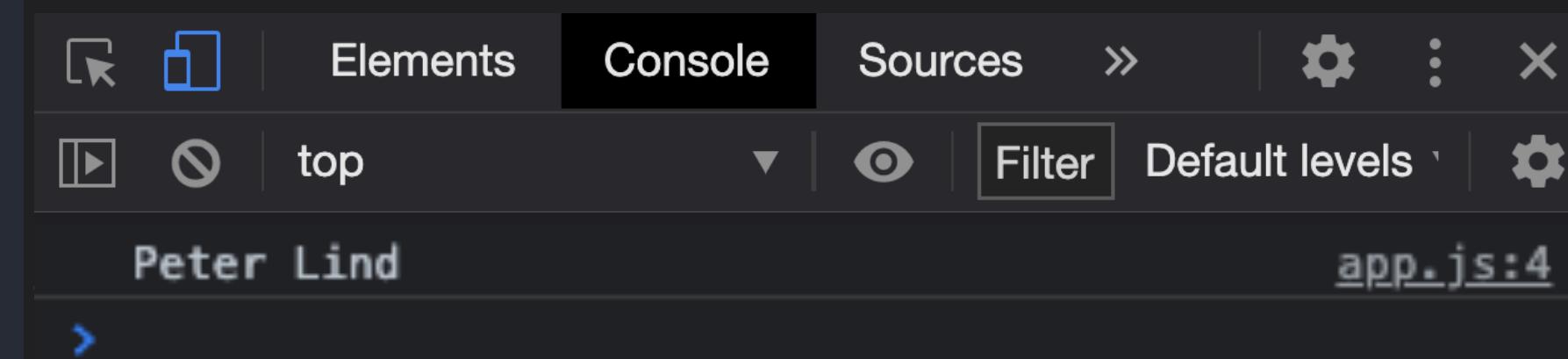
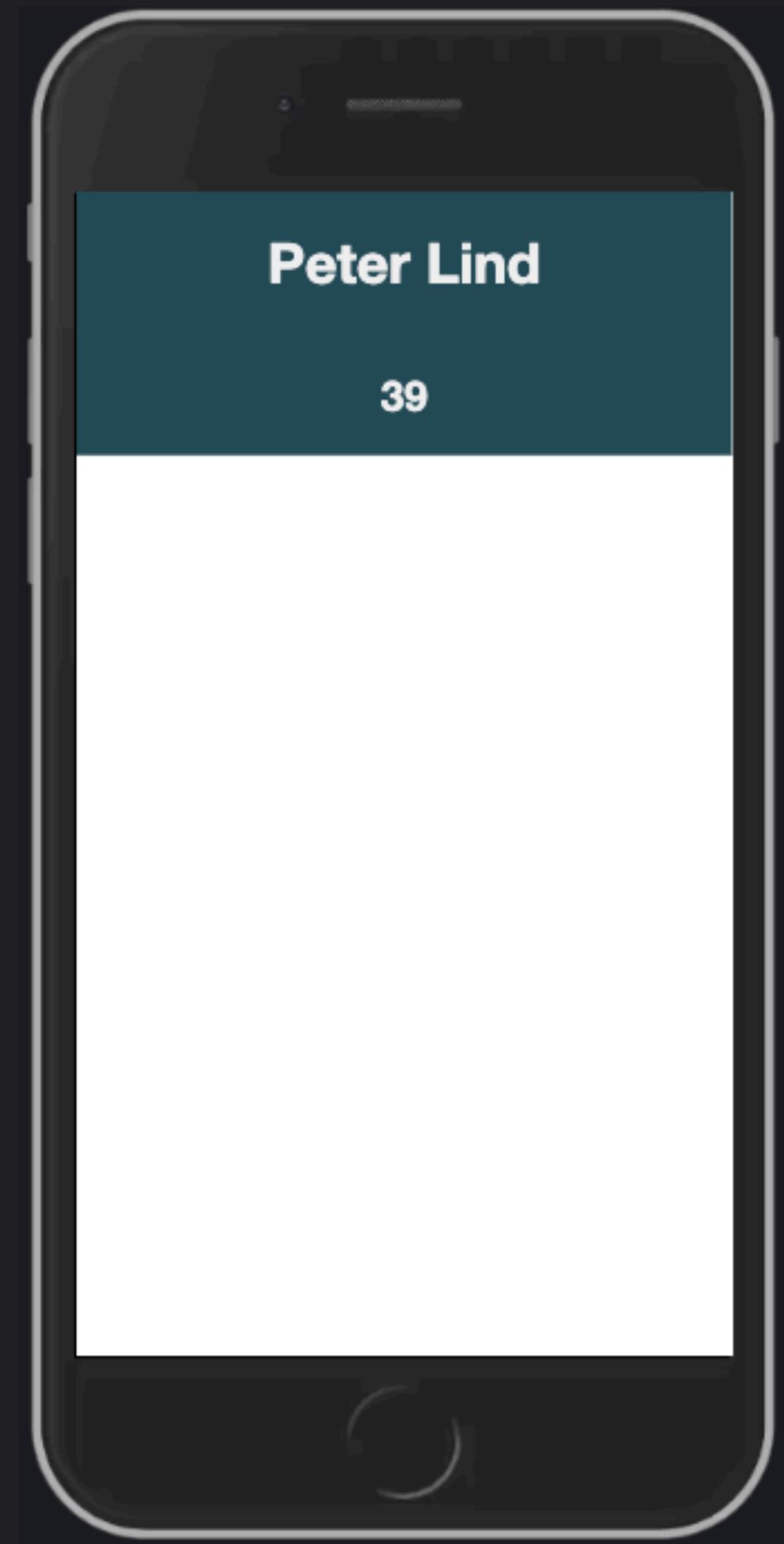
```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```

.textContent

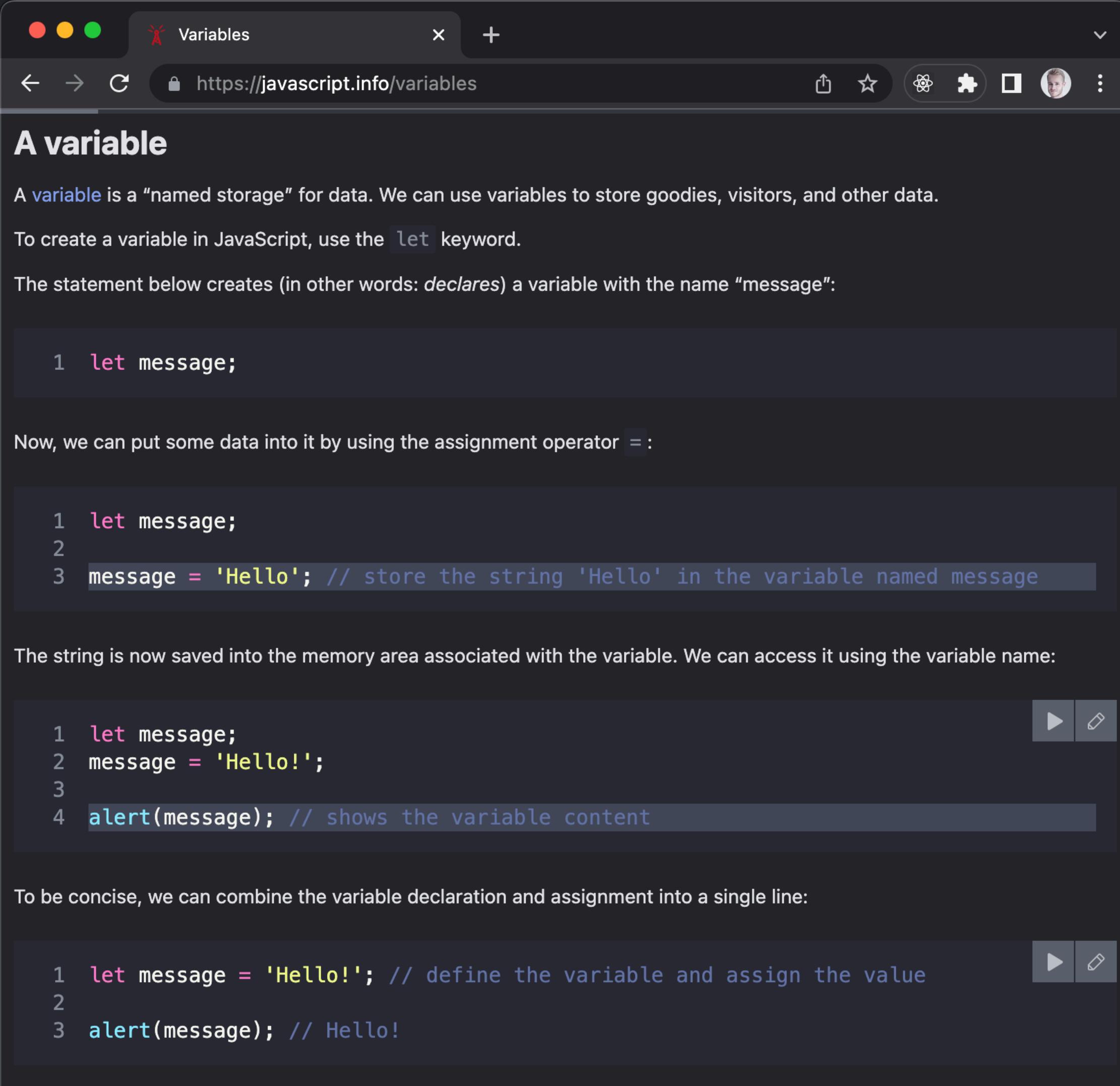
Get or set the text content of a given (HTML) element

Variables & UI

```
let fullName = "Peter Lind";
let age = 39;
console.log(fullName, age);
document.querySelector("#fullName_container").textContent = fullName;
document.querySelector("#age_container").textContent = age;
```



JavaScript.info/Variables



A screenshot of a web browser window titled "Variables". The URL in the address bar is <https://javascript.info/variables>. The page content is as follows:

A variable

A [variable](#) is a “named storage” for data. We can use variables to store goodies, visitors, and other data.

To create a variable in JavaScript, use the `let` keyword.

The statement below creates (in other words: *declares*) a variable with the name “message”:

```
1 let message;
```

Now, we can put some data into it by using the assignment operator `=`:

```
1 let message;
2
3 message = 'Hello'; // store the string 'Hello' in the variable named message
```

The string is now saved into the memory area associated with the variable. We can access it using the variable name:

```
1 let message;
2 message = 'Hello!';
3
4 alert(message); // shows the variable content
```

To be concise, we can combine the variable declaration and assignment into a single line:

```
1 let message = 'Hello!'; // define the variable and assign the value
2
3 alert(message); // Hello!
```

Read, read,
read,
The docs



Variables

Template: [project-template](#) (GitHub) eller egen løsning

+ :::

1. Definer en variabel

- Definer en variabel af typen `let`.
- Gem dit eller et andet navn i variablen.

```
let name = "Rasmus";
```

- Brug `console.log(...)` til at udskrive/ logge værdien til konsollen i browser dev tool.

```
console.log(name);
```

- Ændrer `name` variablen til en et andet navn.

```
name = "Ida";
```

- Log værdien igen.

```
console.log(name);
```

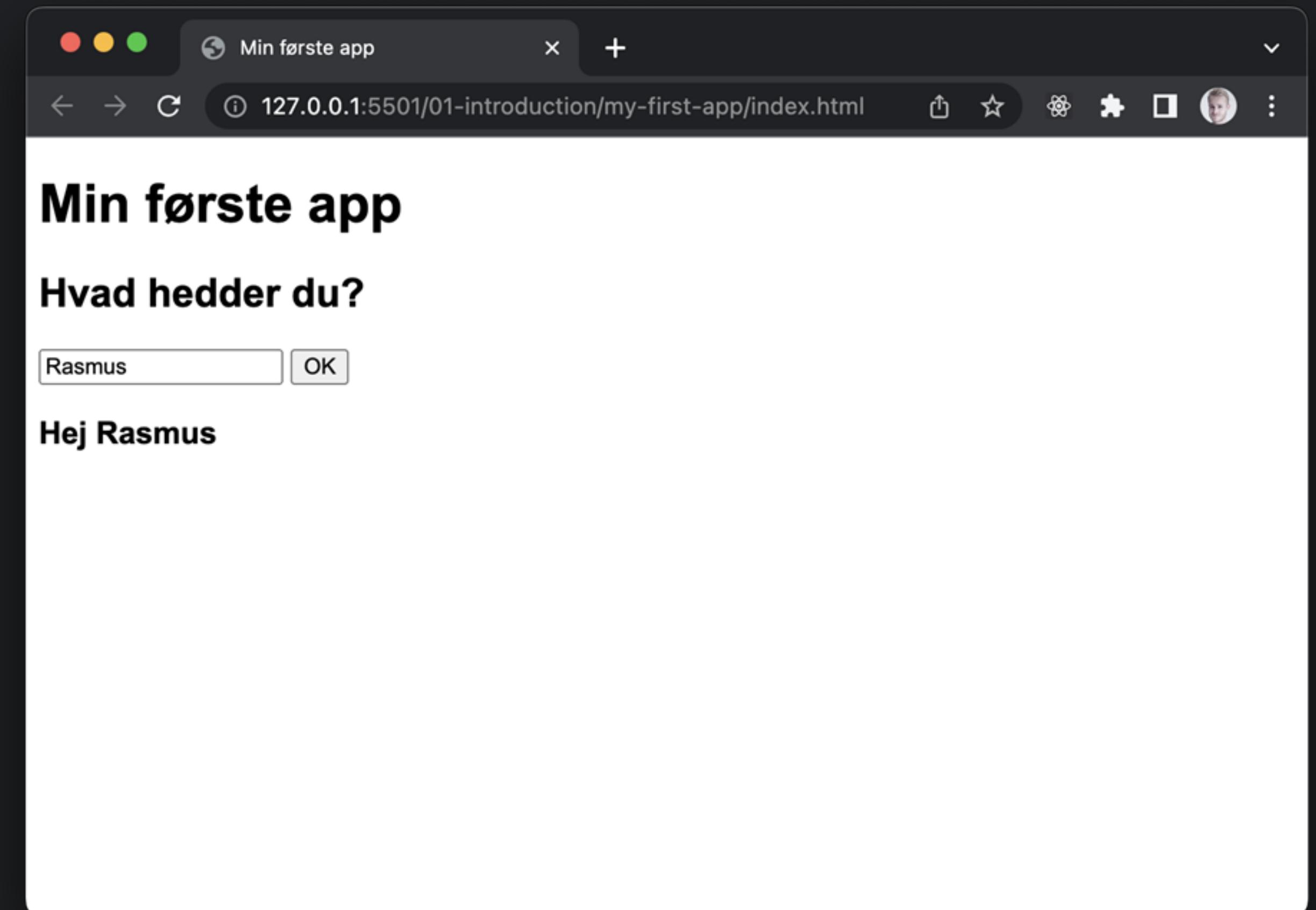
- Vis `name` variablen på websitet (tilføj værdien til DOMen ved hjælp af `document.querySelector` :

```
document.querySelector("#content").innerHTML = name;
```



Hello JavaScript

1. Find øvelsen fra 2. introdag:
[Hello JavaScript](#)
2. Genbesøg din kode.
3. Overvej hvordan du kan anvende variabler.
4. Implementer ændringerne.
5. Forbedrer, simplificerer eller gør det gin kode lettere at læse? Hvordan?
6. Er der anden ny viden, du kan anvende til at forbedre din løsning?



var vs let

THE DIFFERENCE IS THE SCOPING

VAR IS FUNCTION-WIDE OR GLOBAL SCOPE

LET IS BLOCK SCOPED

VAR TOLERATES REDECLARATION

<https://javascript.info/variables>

<https://javascript.info/var>

```
// Example 1
// "var" has no block scope
if (true) {
| var test1 = true; // use "var" instead of "let"
}
console.log(test1); // true, the variable lives after if

// Example 2
if (true) {
| let test2 = true; // use "let"
}
console.log(test2); // Error: test is not defined

// Example 3
for (var i = 0; i < 10; i++) {
| // ...
}
console.log(i); // 10, "i" is visible after loop, it's a global variable
```

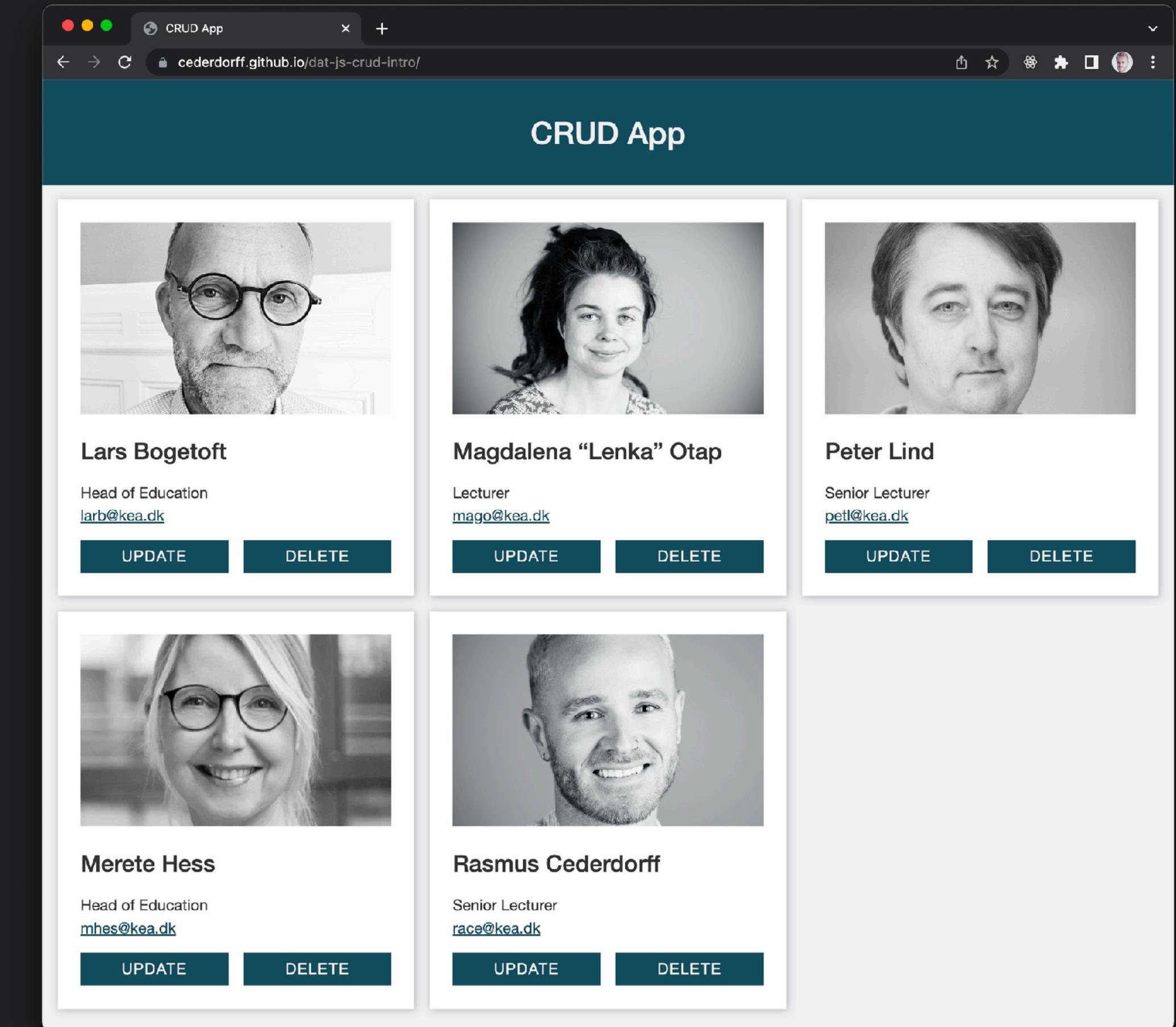
```
// "var" tolerates redeclarations
var user1 = "Pete";
var user1 = "John"; // this "var" does nothing (already declared)
// ...it doesn't trigger an error
console.log(user1); // John

let user2;
let user2; // SyntaxError: 'user' has already been declared
```

var-vs-let

CRUD App

1. Find øvelsen fra [Intro til Programmering - CRUD App](#)
2. Genbesøg din kode.
3. Overvej hvordan du anvender variabler.
4. Hvilke variabeltyper anvender du? Hvorfor?



Const

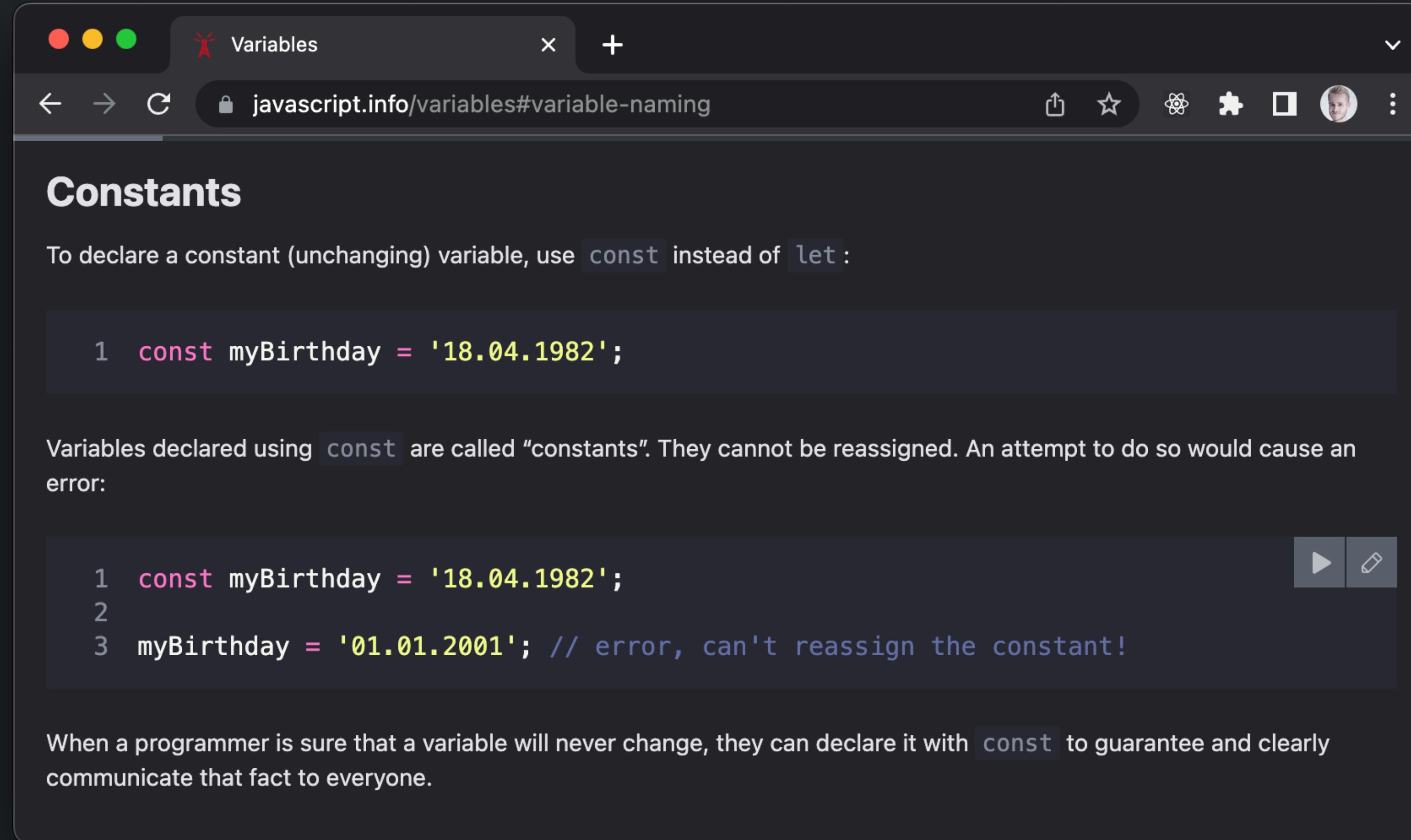
Const is an unchanging variable.

```
const myBirthday = "12-03-1990";
myBirthday = "12-03-1989";
// Uncaught TypeError: can't reassign the constant!
```

const cannot be reassigned.

If you try to, an error will be thrown.

And the doc says...



The screenshot shows a dark-themed web browser window with the title bar "Variables". The address bar contains the URL "javascript.info/variables#variable-naming". The main content area displays the following text and code examples:

Constants

To declare a constant (unchanging) variable, use `const` instead of `let`:

```
1 const myBirthday = '18.04.1982';
```

Variables declared using `const` are called "constants". They cannot be reassigned. An attempt to do so would cause an error:

```
1 const myBirthday = '18.04.1982';
2
3 myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

When a programmer is sure that a variable will never change, they can declare it with `const` to guarantee and clearly communicate that fact to everyone.

Const can't be reassigned

```
const myBirthday = "12-03-1990";
myBirthday = "12-03-1989"; // Uncaught TypeError: can't reassign the constant!

const person = {
    name: "Kasper",
    mail: "kato@eaaa.dk",
    age: 32
};

person.age = 33; // no error

person = {
    name: "Rasmus",
    mail: "race@eaaa.dk",
    age: 31
}; // Uncaught TypeError: can't reassign the constant!
```

Const

Hvis du igen tænker på Clicker-Spillet, havde det givet mening at anvende const? Har du anvendt const?

Uppercase?

The screenshot shows a dark-themed web browser window. The title bar says "Variables". The address bar shows "javascript.info/variables#variable-naming". The main content area has a heading "Uppercase constants". It discusses the widespread practice of using constants as aliases for difficult-to-remember values known prior to execution. It notes that such constants are named using capital letters and underscores. An example code block is shown:

```
1 const COLOR_RED = "#F00";
2 const COLOR_GREEN = "#0F0";
3 const COLOR_BLUE = "#00F";
4 const COLOR_ORANGE = "#FF7F00";
5
6 // ...when we need to pick a color
7 let color = COLOR_ORANGE;
8 alert(color); // #FF7F00
```

Below the code, a section titled "Benefits:" lists three points:

- `COLOR_ORANGE` is much easier to remember than `"#FF7F00"`.
- It is much easier to mistype `"#FF7F00"` than `COLOR_ORANGE`.
- When reading the code, `COLOR_ORANGE` is much more meaningful than `#FF7F00`.

Benefits:

- `COLOR_ORANGE` is much easier to remember than `"#FF7F00"`.
- It is much easier to mistype `"#FF7F00"` than `COLOR_ORANGE`.
- When reading the code, `COLOR_ORANGE` is much more meaningful than `#FF7F00`.

When should we use capitals for a constant and when should we name it normally? Let's make that clear.

Being a "constant" just means that a variable's value never changes. But there are constants that are known prior to execution (like a hexadecimal value for red) and there are constants that are *calculated* in run-time, during the execution, but do not change after their initial assignment.

A screenshot of a web browser window displaying a page from javascript.info/variables#variable-naming. The page title is "Variables". The main content is titled "Uppercase const?" with a link icon. It has an "importance: 4" rating. The text asks to examine the following code:

```
1 const birthday = '18.04.1982';
2
3 const age = someCode(birthday);
```

The text explains that there is a constant `birthday` for the date and a constant `age` calculated from `birthday` using `someCode()`. It asks if it would be right to use uppercase for `birthday`, `age`, or both. Below is another code snippet with comments:

```
1 const BIRTHDAY = '18.04.1982'; // make birthday uppercase?
2
3 const AGE = someCode(BIRTHDAY); // make age uppercase?
```

<https://javascript.info/variables>

Use let & const
instead of var

<https://javascript.info/variables>
<https://javascript.info/var>

But which one?

<https://javascript.info/variables>
<https://javascript.info/var>

Start with const

If needed, change to let

Never use var!

<https://javascript.info/variables>

<https://javascript.info/var>

The screenshot shows a web browser window with a dark theme. The title bar says "Variables". The address bar shows the URL "javascript.info/variables#name-things-right". The main content area has a sidebar on the left with a list of navigation items: Chapter, JavaScript Fundamentals, Lesson navigation, A variable, A real-life analogy, Variable naming, Constants, Name things right (which is the current page), Summary, Tasks (3), Comments, Share, and Edit on GitHub. The main content area has a header "Name things right" and a sub-header "Talking about variables, there's one more extremely important thing." It discusses the importance of variable naming for readability and maintainability. It also provides some good-to-follow rules for naming variables.

Name things right

Talking about variables, there's one more extremely important thing.

A variable name should have a clean, obvious meaning, describing the data that it stores.

Variable naming is one of the most important and complex skills in programming. A quick glance at variable names can reveal which code was written by a beginner versus an experienced developer.

In a real project, most of the time is spent modifying and extending an existing code base rather than writing something completely separate from scratch. When we return to some code after doing something else for a while, it's much easier to find information that is well-labeled. Or, in other words, when the variables have good names.

Please spend time thinking about the right name for a variable before declaring it. Doing so will repay you handsomely.

Some good-to-follow rules are:

- Use human-readable names like `userName` or `shoppingCart`.
- Stay away from abbreviations or short names like `a`, `b`, `c`, unless you really know what you're doing.
- Make names maximally descriptive and concise. Examples of bad names are `data` and `value`. Such names say nothing. It's only okay to use them if the context of the code makes it exceptionally obvious which data or value the variable is referencing.
- Agree on terms within your team and in your own mind. If a site visitor is called a "user" then we should name related variables `currentUser` or `newUser` instead of `currentVisitor` or `newManInTown`.

Sounds simple? Indeed it is, but creating descriptive and concise variable names in practice is not. Go for it.

<https://javascript.info/variables#name-things-right>

The screenshot shows a dark-themed web browser window with the title bar 'Variables'. The address bar displays the URL 'javascript.info/variables'. The main content area is titled 'Variable naming'.

There are two limitations on variable names in JavaScript:

1. The name must contain only letters, digits, or the symbols `$` and `_`.
2. The first character must not be a digit.

Examples of valid names:

```
1 let userName;
2 let test123;
```

When the name contains multiple words, **camelCase** is commonly used. That is: words go one after another, each word except first starting with a capital letter: `myVeryLongName`.

What's interesting – the dollar sign `'$'` and the underscore `'_'` can also be used in names. They are regular symbols, just like letters, without any special meaning.

These names are valid:

```
1 let $ = 1; // declared a variable with the name "$"
2 let _ = 2; // and now a variable with the name "_"
3
4 alert($ + _); // 3
```

Examples of incorrect variable names:

```
1 let 1a; // cannot start with a digit
2
3 let my-name; // hyphens '-' aren't allowed in the name
```

<https://javascript.info/variables#variable-naming>

Name things right (or wrong?)

1. Create a variable with the name of our planet. How would you name such a variable?
2. Create a variable to store the name of a current visitor to a website. How would you name that variable?

<https://javascript.info/variables>

Global Variables

Variables outside a function (and scopes) are global variables

JavaScript.info/function-basics#local-variables

Local variables

A variable declared inside a function is only visible inside that function.

For example:

```
1 function showMessage() {  
2   let message = "Hello, I'm JavaScript!"; // local variable  
3  
4   alert( message );  
5 }  
6  
7 showMessage(); // Hello, I'm JavaScript!  
8  
9 alert( message ); // <-- Error! The variable is local to the function
```

Scopes:

- Local Variable
- Global Variable

Outer variables

A function can access an outer variable as well, for example:

```
1 let userName = 'John';  
2  
3 function showMessage() {  
4   let message = 'Hello, ' + userName;  
5   alert(message);  
6 }  
7  
8 showMessage(); // Hello, John
```

Global variables

Variables declared outside of any function, such as the outer `userName` in the code above, are called *global*.

Global variables are visible from any function (unless shadowed by locals).

It's a good practice to minimize the use of global variables. Modern code has few or no globals. Most variables reside in their functions. Sometimes though, they can be useful to store project-level data.

Global & local Variables

Eksempler fra Clicker-Spillet?

Hvorfor er det vigtigt?

Data Types

In JavaScript there are two main data types:

- **Primitive values** like strings, numbers and booleans.
- **Objects** with properties.

```
1 let str = "Hello";
2 let str2 = 'Single quotes are ok too';
3 let phrase = `can embed another ${str}`;
```

```
1 let n = 123;
2 n = 12.345;
```

```
1 let user = new Object(); // "object constructor" syntax
2 let user = {}; // "object literal" syntax
```

In JavaScript, a value always has a certain type like a string, number, boolean, object, array, etc.

Data Types

Variables can hold 7 different kinds of values (data types)

- only one at a time ...

Boolean

Number

String

Object

Null

Undefined

Symbol

Datatypes - when to use which one?

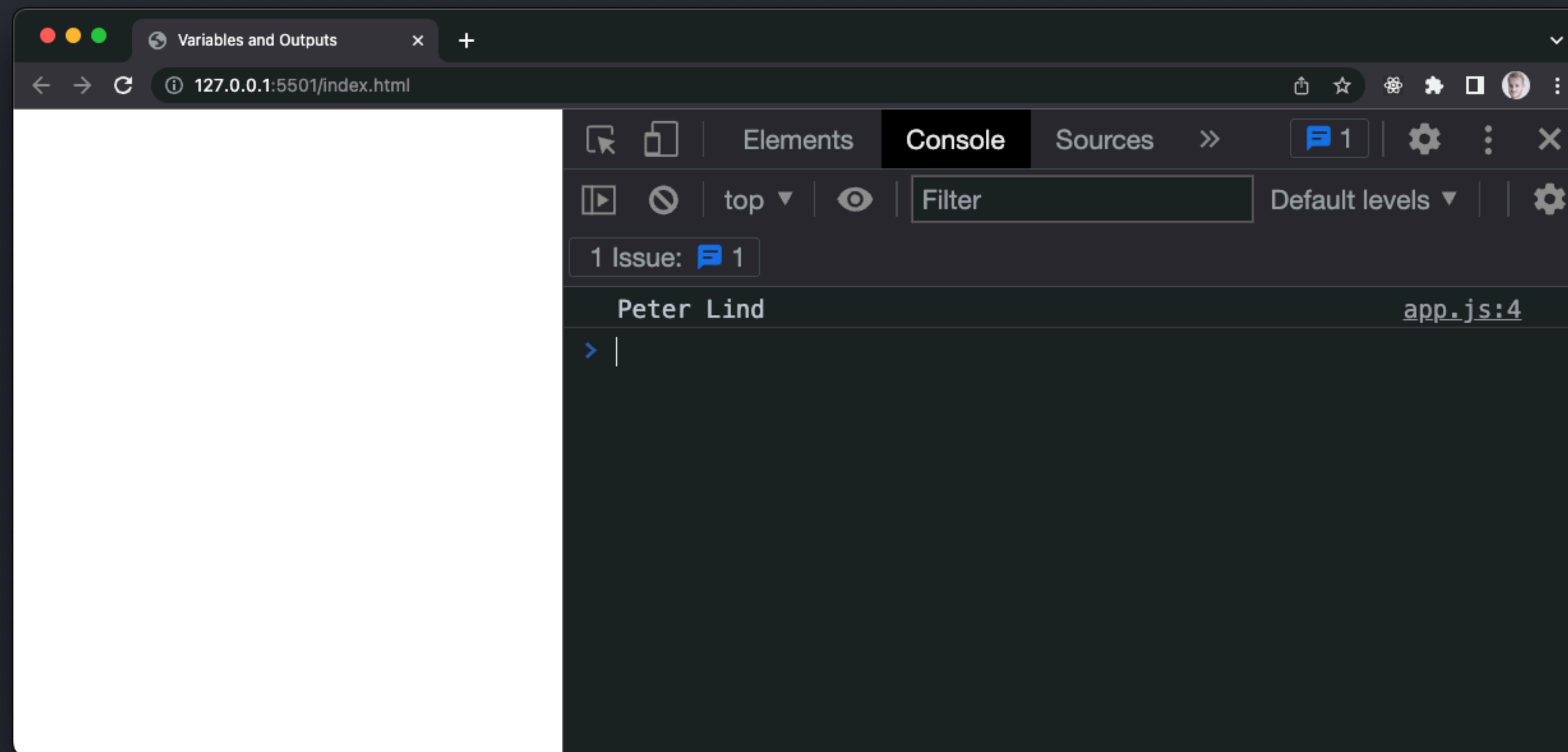
Boolean	Values that are true or false – conditions
Number	Numbers! Points, counting, math-operations , measuring ...
String	Text! Input-fields – HTML and CSS-attributes ... URLs (Phone-number)
Object	Everything (Arrays, Functions) Combined/collected data (of the other types)
Null	No datatype yet – no value!
Undefined	Something that hasn't defined before – <u>don't set things to undefined!</u>
Symbol	Something complicated

Datatyper

Hvilke datatyper har du anvendt i Clicker-spillet?

Hav Console åben - altid!

Så... hvor kan se datatyperne?



Datetyper

1. Prøv disse og test med `console.log`
2. Hvordan ser vi værdien?
3. Hvordan ser vi typen?
4. Hvordan definerer du, hvilken type, du vil gemme i en variabel?

```
const bool = true;
const num = 41;
const str = "Peter";
const obj = {
    cats: 2,
    cars: 1
};
const nothing = null;
let undf;
const symbol = Symbol("symbol");
```

Strings

In JavaScript textual data is stored as strings.

Strings are defined with either ‘single quotes’, “double quotes” or `backticks`.

```
let single = 'single-quoted';
```

```
let double = "double-quoted";
```

```
let backticks = `backticks`;
```

```
let fullName = "Peter Lind";
let age = 39;

let message = fullName + " is " + age + " years old.";
console.log(message); // Peter Lind is 39 years old.
```

Concatenation

We can combine two or more variables in one string (variable).

```
let fullName = "Peter Lind";
let age = 39;

// single
console.log(fullName + ' is ' + age + ' years old.');
// double
console.log(fullName + " is " + age + " years old.");
// backticks
console.log(` ${fullName} is ${age} years old.`);
```

Concatenation

Strings are defined with either ‘single quotes’, “double quotes” or `backticks`.

String quotes

1. Test the code to the right.
2. What is the output of the script to the right?

```
"use strict";  
  
let fullName = "Alicia Keys";  
  
console.log(`hello ${1}`); // ?  
  
console.log(`hello ${"name"} `); // ?  
  
console.log(`hello ${fullName}`); // ?
```

Data Type Conversion

In JavaScript, we never specify the type of a variable (String, Number, Boolean, Object, etc.).

We set the value and use the value as the type we expect it to be.

```
const firstName = "Rasmus";
const lastName = "Cederdorff";

let age = 32;

let isSeniorLecture = true;

const obj = {
  cats: 0,
  cars: 1
};

const kids = ["Alicia", "Ida"];
```

It happens when we do:

- Concatenation (+)
- Comparisons (==, !=, <, >, <=, >=)
- Calculations (-, *, /, %)

Automatic conversion

So often JavaScript has to do automatic conversion.

Everything in HTML is strings, so when writing or reading numbers, they have to be converted.

Most of the time this happens automatically.

“Semi-automatic” conversion

`String(value)` converts a value to a string, usually by calling `value.toString()`

`Number(value)` converts a value to a number, usually by calling `value.valueOf()`

`Boolean(value)` converts a value to a boolean, by unknown magic means!

Note:

`Boolean(0)` is false, but
`Boolean("0")` is true

It is always possible to convert something into a string –
but it might not be possible to convert everything into a number!

Numeric conversion

1. How do you convert to Number?

```
let string = "1234";  
console.log(string);  
// number??
```

String conversion

1. How do you convert to String?

```
let number = 1234;  
console.log(number);  
// string??
```

Numeric conversion 2

1. Try yourself!

2. What is the output in
the console?

```
console.log(Number(" 123  ")); // ?  
console.log(Number("123z")); // ?  
console.log(Number(true)); // ?  
console.log(Number(false)); // ?
```

Type Conversions

← → C 🔒 javascript.info/type-conversions

EN AJJS Buy EPUB/PDF 🚙 ☰ ⚡

Chapter

JavaScript Fundamentals

Lesson navigation

String Conversion

Numeric Conversion

Boolean Conversion

Summary

Comments

Share

[Twitter](#) [Facebook](#)

Edit on GitHub

Type Conversions

Most of the time, operators and functions automatically convert the values given to them to the right type.

For example, `alert` automatically converts any value to a string to show it. Mathematical operations convert values to numbers.

There are also cases when we need to explicitly convert a value to the expected type.

ⓘ Not talking about objects yet

In this chapter, we won't cover objects. For now, we'll just be talking about primitives.

Later, after we learn about objects, in the chapter [Object to primitive conversion](#) we'll see how objects fit in.

String Conversion

String conversion happens when we need the string form of a value.

For example, `alert(value)` does it to show the value.

We can also call the `String(value)` function to convert a value to a string:

```
1 let value = true;
2 alert(typeof value); // boolean
3
4 value = String(value); // now value is a string "true"
5 alert(typeof value); // string
```

String conversion is mostly obvious. A `false` becomes "false", `null` becomes "null", etc.

Numeric Conversion

Numeric conversion in mathematical functions and expressions happens automatically.

Read, read,
read,
The docs



Manual conversion

You can omit
the prefix
`Number.` if you
want

`Number.parseInt(string, radix)`

Converts a string **to a number**, using the specified numerical system ([MDN](#))

`Number.prototype.toString(radix)`

Converts a number **to a string**, in the specified numerical system ([MDN](#))

There's also: `parseFloat`, `toExponential`, `toFixed`, `toPrecision` for other kinds of conversion

```
const n1 = 1;  
const n2 = 2;  
  
const s1 = "1";  
const s2 = "2";  
  
n1 + n2;  
s1 + s2;  
  
n1 + s2;  
s1 + n2;
```

Concatenation looks like addition

- What will be the result of these “calculations”?
- Test them in the console!

Automatic data type conversion with concatenation

String + string is a **concatenation**

Number + number is a **calculation**

But if the types are different, one gets converted to the other!

The challenge is to figure out which one gets converted to what ...

```
"" + 1 + 0;  
"" - 1 + 0;  
true + false;  
6 / "3";  
"2" * "3";  
4 + 5 + "px";  
"$" + 4 + 5;  
"4" - 2;  
"4px" - 2;  
" -9 " + 5;  
" -9 " - 5;  
null + 1;  
undefined + 1;  
" \t \n" - 2;
```

Type Conversion

1. Test these expressions.
2. First: Try to guess the result, then check your guess in the console!
3. Explain to each other why your guess was correct/wrong.



Code
Every
Day