

**race.js** is loading. . .



Check out: [cederdorff.github.io/variables-and-outputs](https://cederdorff.github.io/variables-and-outputs)

# Programming knowledge

## JavaScript

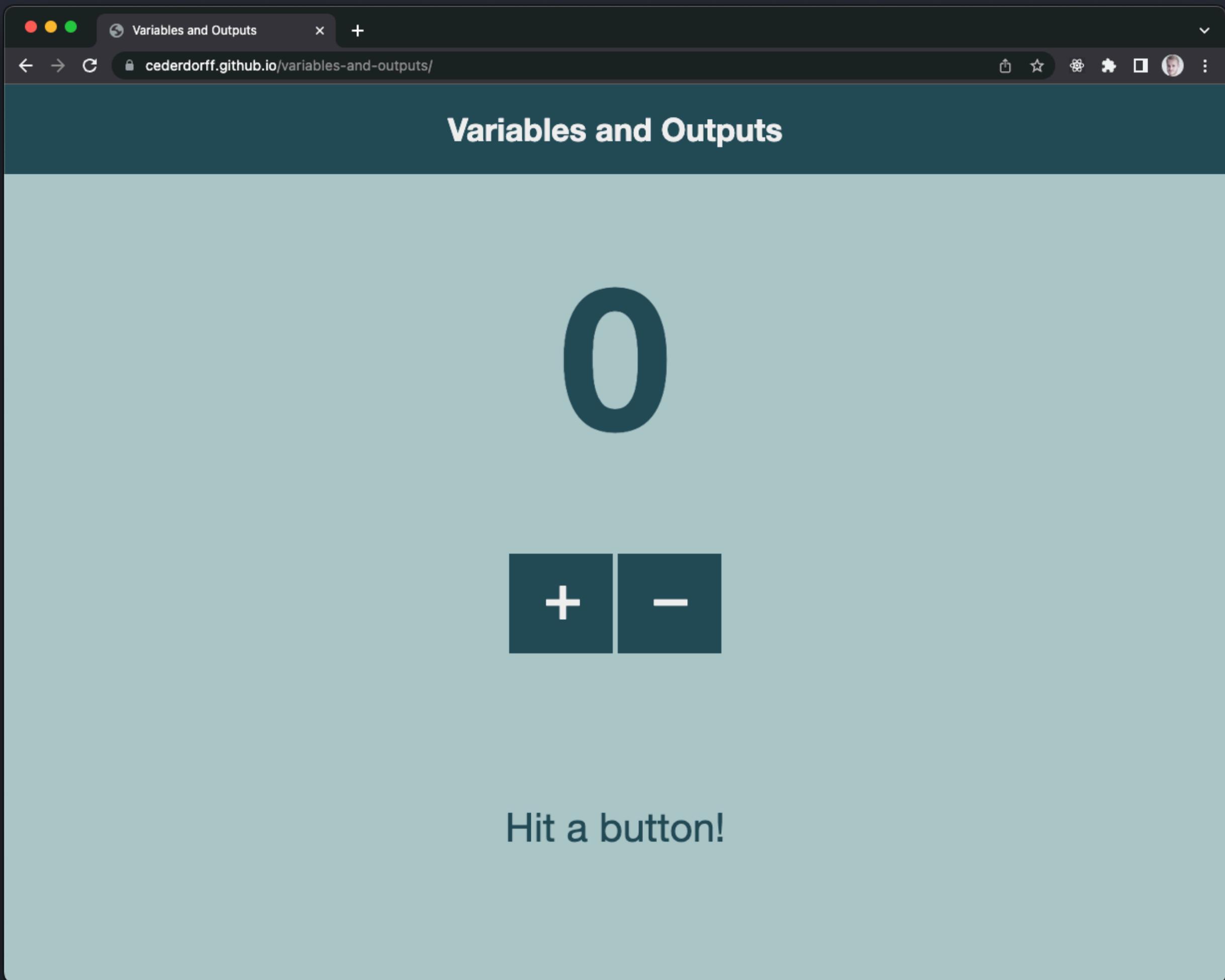
```
index.html
85  }
86
87  /*
88   * Fetches post data from my headless CMS
89  */
90  function getPersons() {
91    fetch('http://headlesscms.cederdorff.com/wp-json/wp/v2/posts?_embed=true')
92      .then(function(response) {
93        return response.json();
94      })
95      .then(function(persons) {
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 
```

# Variables

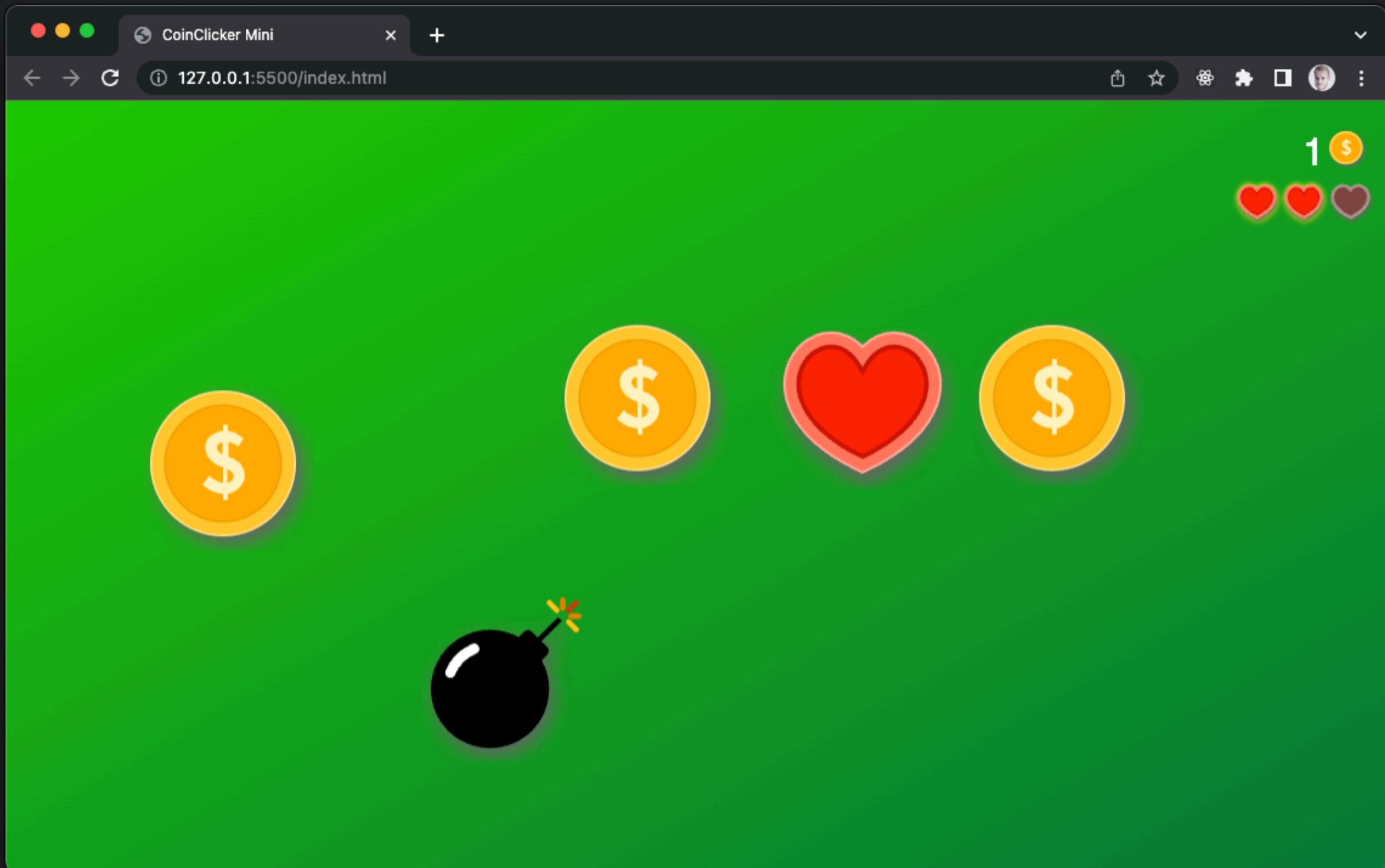
# Agenda

- Formål
- Variabler
- Variabler, UI, Outputs  
og .textContent
- Datetyper, Strings og  
konkatenering
- Tæl point og liv

# Tæl et tal op og ned



# Tæl point og liv (formål)



- Viden om variabler.
- Hvordan anvender vi variabler (generelt)?
- Hvordan kan vi anvende dem i vores spil?
- Outputs, Strings, konkatenering, .textContent, separation of concerns, “use strict”, console, mm.

“

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

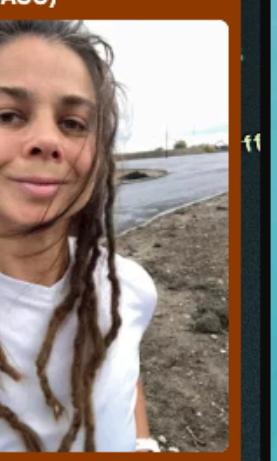
<https://www.youtube.com/watch?v=GLhyjVZp0cw&t=252s>

# 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.

Yaw Boateng  
  
Alder: 32  
By: København NV, Danmark, Jorden, Mælkehøjden.  
Baggrund:  
Levet at fåne folks smerten i led og muskler.

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 hvæld af ekstra

Hvorfor Datamatiker: God chef, tilfældige valg.

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

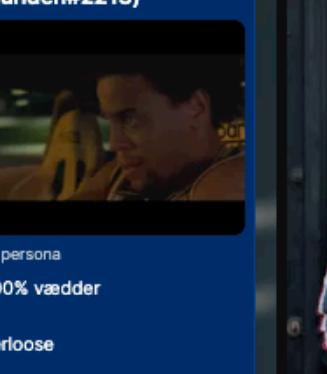
Forventninger og ønsker til Datamatikeruddannelsen: Inspirende med min øjenale viden.

**Hjem er I?**  
Dat23v2

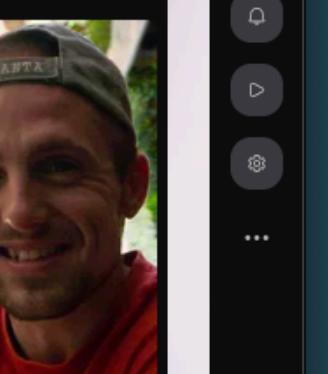
Rasmus Cederdorff (RACE)  
  
Ham med armen  
Alder: Fyldte 30 dage efter Mette lukkede landet 😊🎉  
By: København NV (fra Aarhus)  
Baggrund - uddannelse & Job:  
Datamatiker, PBA Webudvikling & Cand.it - Webarkitektur  
Freelance Web App Developer & Lektor på EAAA  
Har du programmeret (bestemt ikke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

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  
Forventninger og ønsker til Datamatikeruddannelsen: Job  
Drømmejob og fremtid: Cloud Solution Architect  
Hvorfor Datamatiker: Jeg vil gerne undervise folk i at blive gode programmer - programmerer der har det sjovt med at skabe programmer i stedet for bare at

Interesser: Spiller sygt meget WoW - når min pige på 27 år ikke kræver min opmærksomhed.  
Forventninger og ønsker til Datamatikeruddannelsen: Job  
Forventninger og ønsker til Datamatikeruddannelsen: Job  
Drømmejob og fremtid: IT job

full Name

initials

age

city

background

image

...

Rasmus Cederdorff  
(RACE)



Ham med armen

Alder: Fyldte 30 dagen 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):  
Webløsninger og Apps -  
Masser af JavaScript

Hvorfor Datamatiker:  
Kombinationen mellem teori og  
praksis. Praksisnær

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

full Name

initials

age

city

background

image

...

**It's all variables!**

```
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 ...

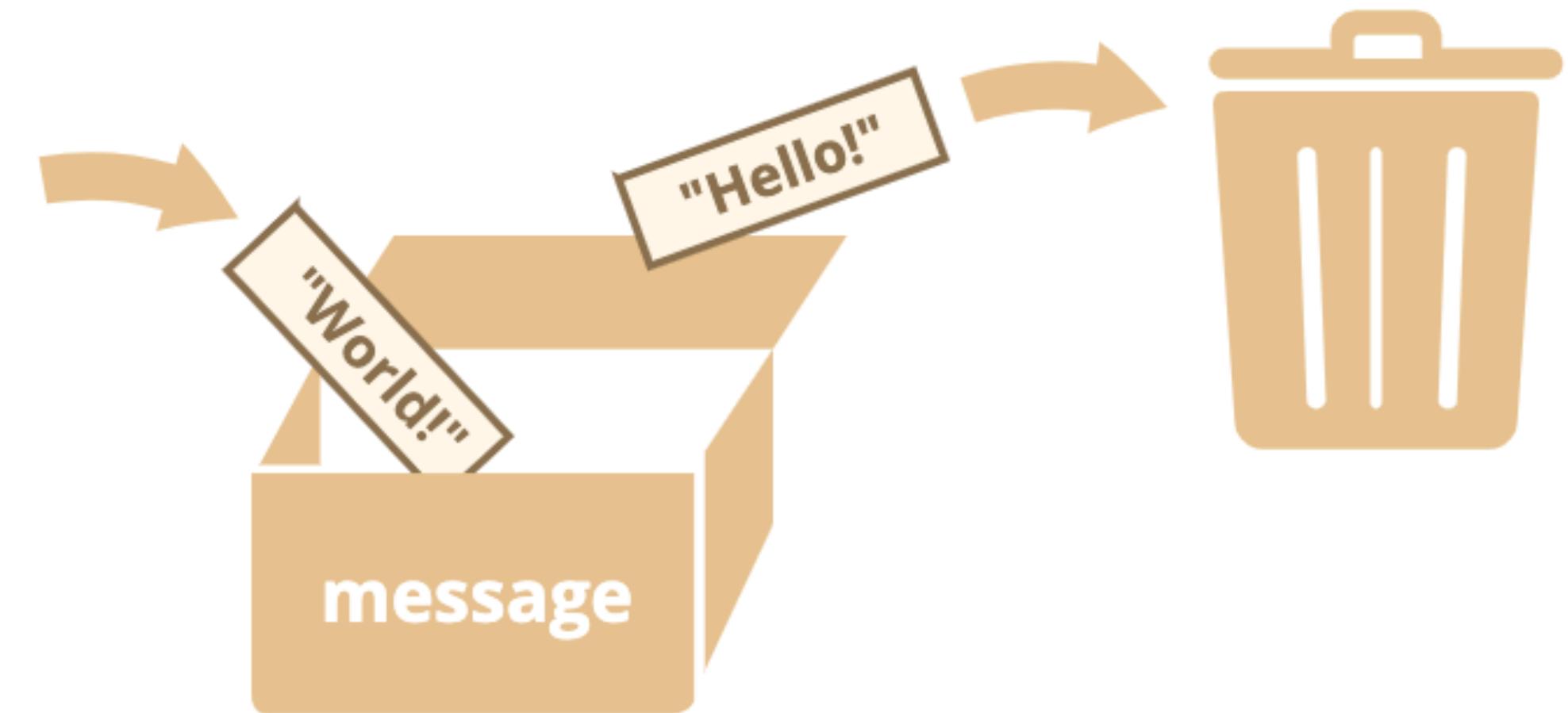
**Har du programmeret**  
**(bestemt ikke et krav): BASIC**  
og Commodore 64 assembler -

# Variables

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

# Variable

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

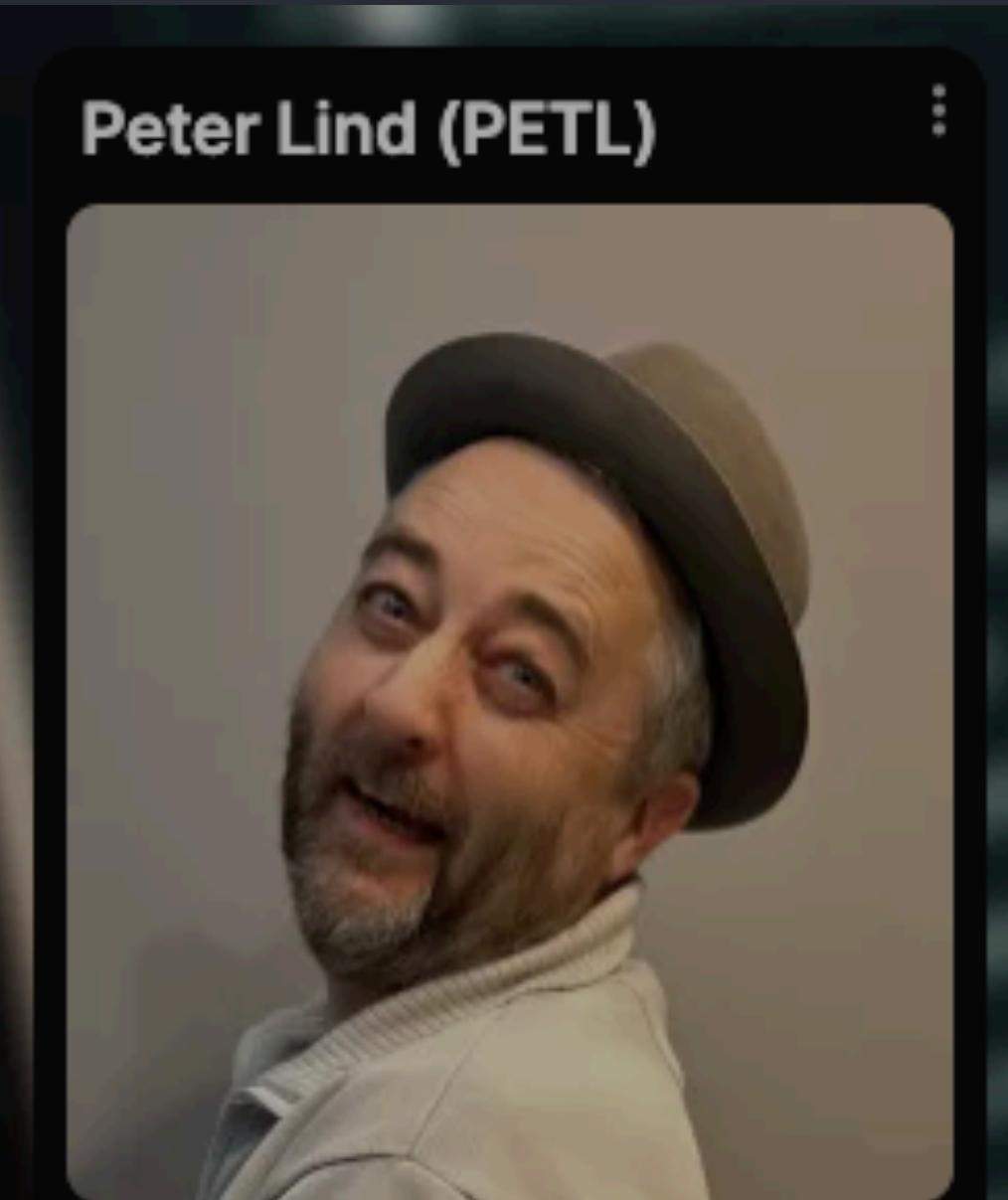


We can change the value of the variables as many times as we want.

# Lad os “lege” med det!

```
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";
```

1. Opret et nyt projekt med en index.html og app.js.
2. Sørg for at index.html loader app.js
3. Skriv console.log("app.js is running"); i toppen af app.js for at teste at det fungerer.
4. Definer variablen fullName med dit eget navn.
5. Brug console.log til at teste værdien af fullName



Peter Lind (PETL)

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

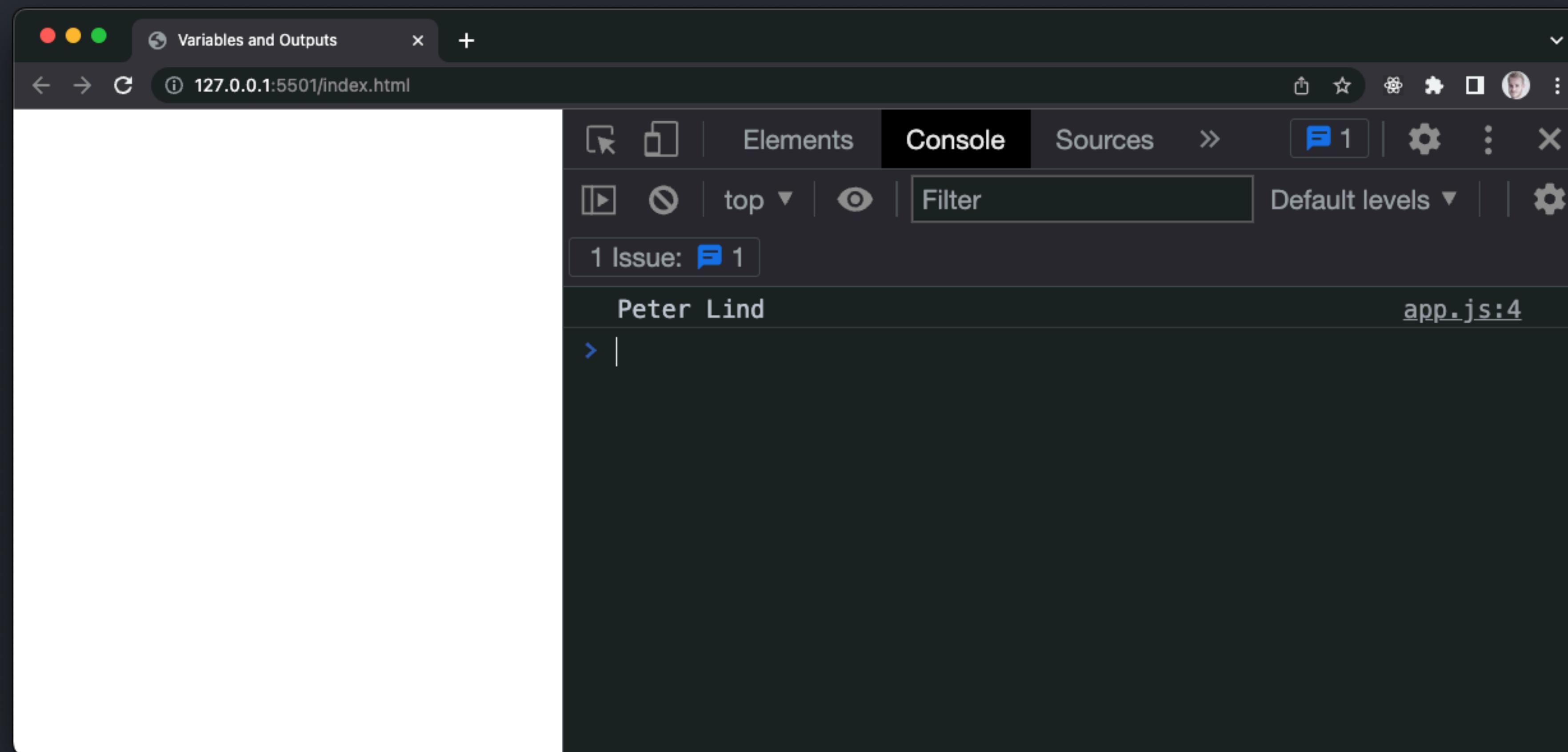
By: Taastrup (fra Kolding a)

**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 -

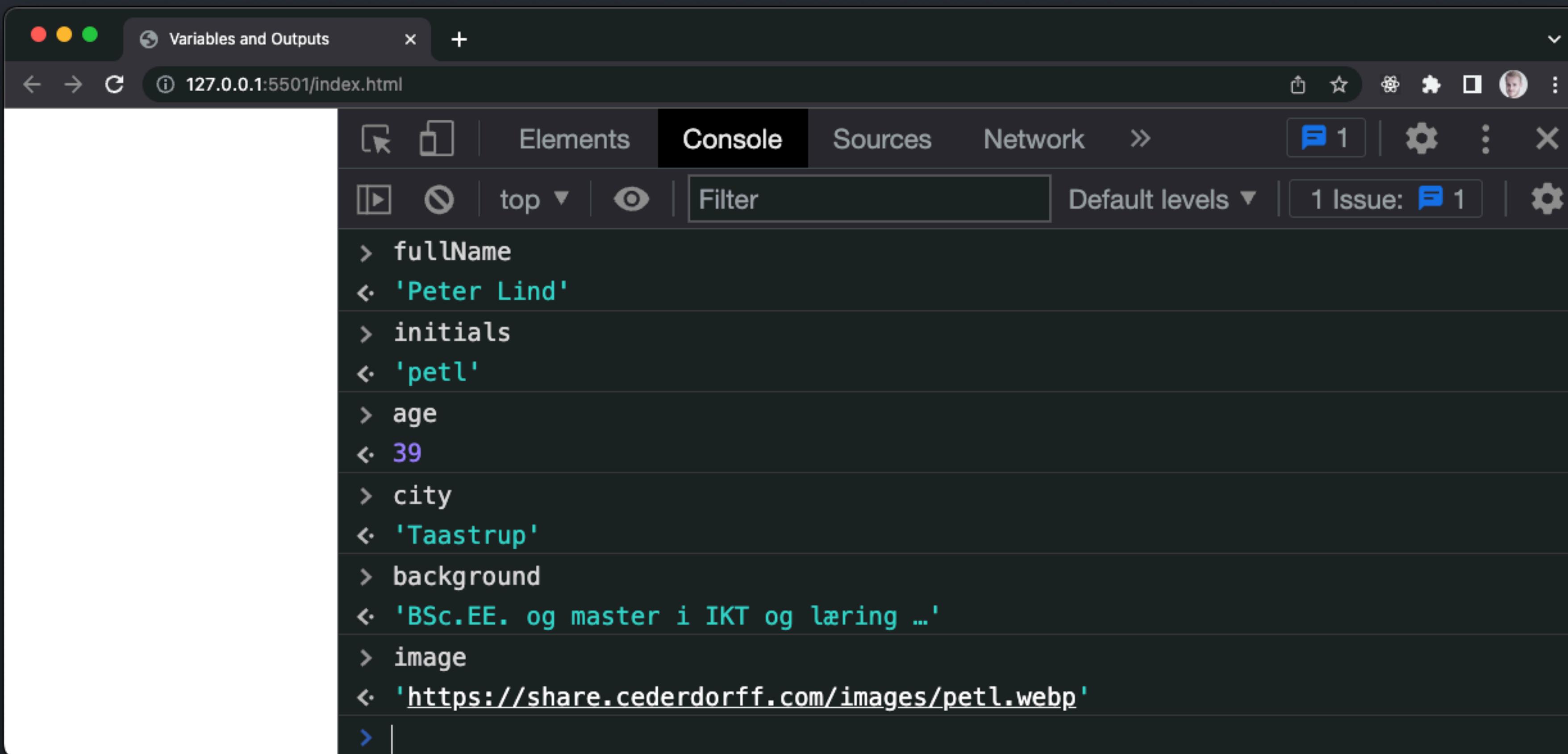
# Hav Console åben - altid!

Og luk den aldrig igen!



# Console/ Konsollen

Skriv navnet på en variabel og tryk enter 🎉



The screenshot shows the 'Variables and Outputs' tab of the developer tools in a browser window. The URL bar indicates the page is 127.0.0.1:5501/index.html. The console tab is active, displaying the following variable outputs:

```
> 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'
```

# Lad os “lege” med det!

```
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";
```

1. Skriv jeres egne variabler med værdier i app.js
2. Brug Console/ Konsollen til at teste dine variabler.
3. Anvend console.log for at teste dine variabler, fx  
console.log(age);
4. Ændr age, fx age = 42;
5. console.log igen for at teste age og brug Console/ Konsollen.



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 -

`"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";

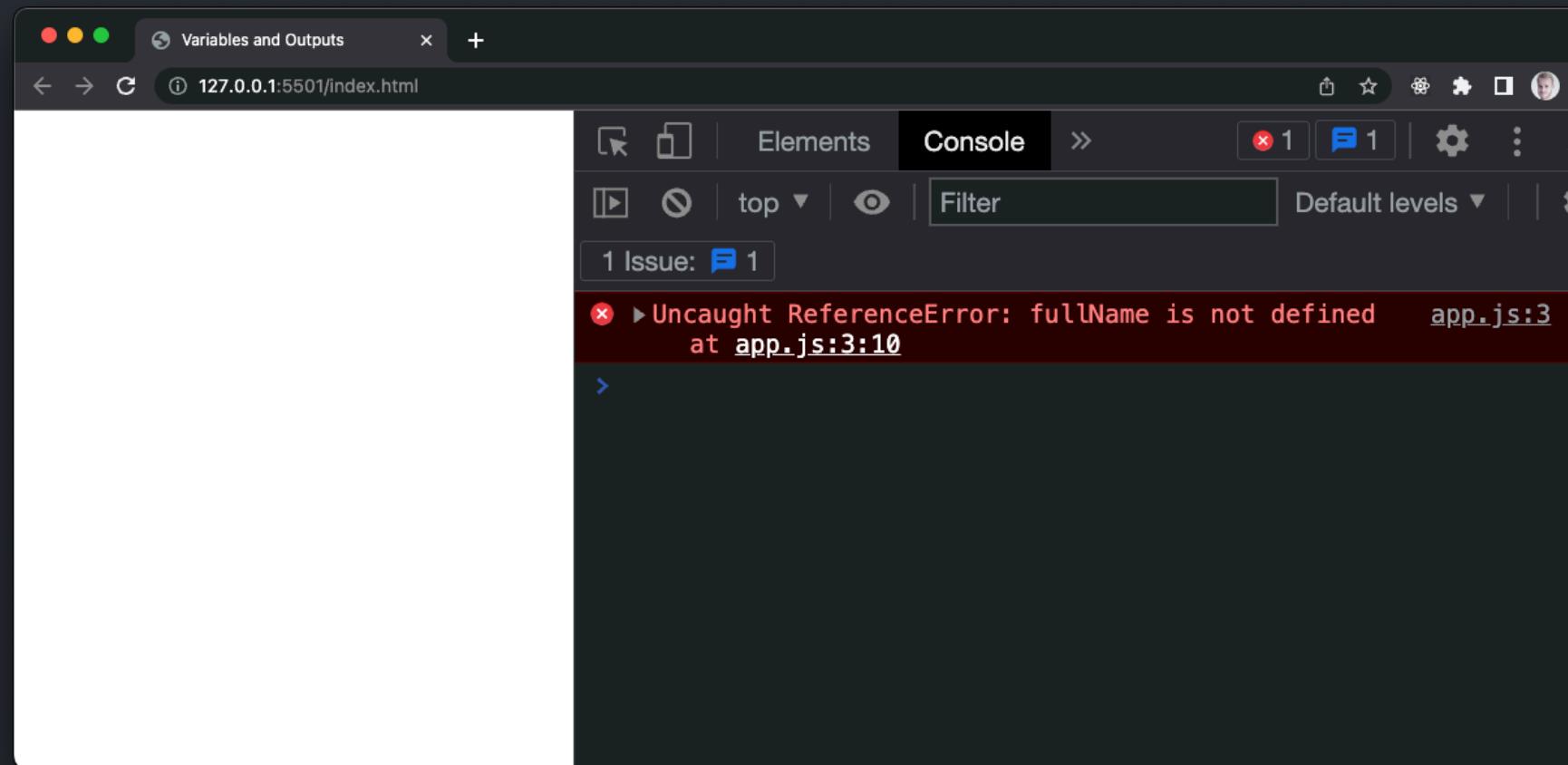
1. Add it to the top of your script.
2. Run your code
3. Check the console
4. What happens?

```
"use strict";  
  
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";
```

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

```
"use strict";
```

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



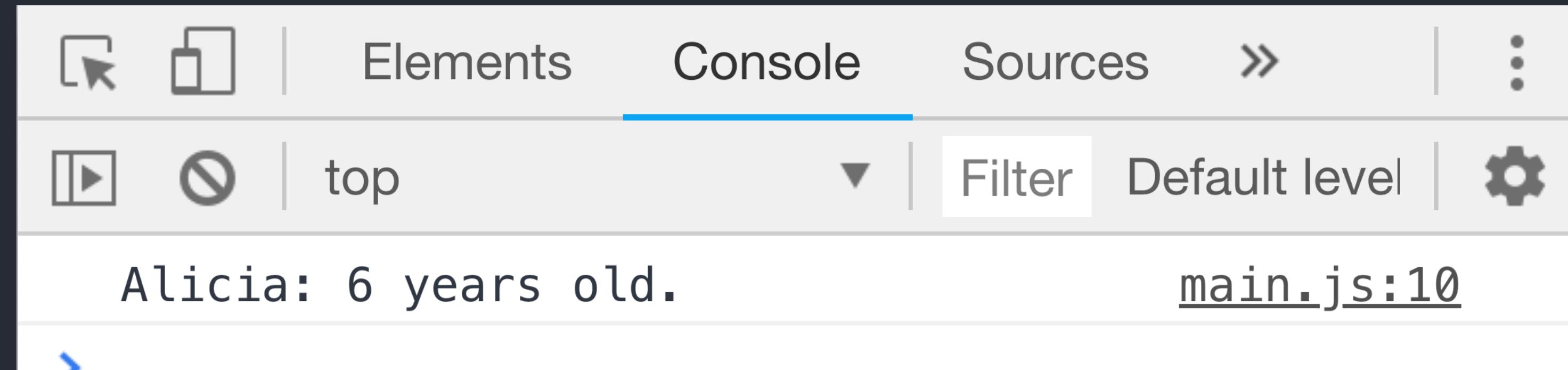
We must define our variables with the keyword `let`, `const` or `var`

# Variables

Store data in the memory

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

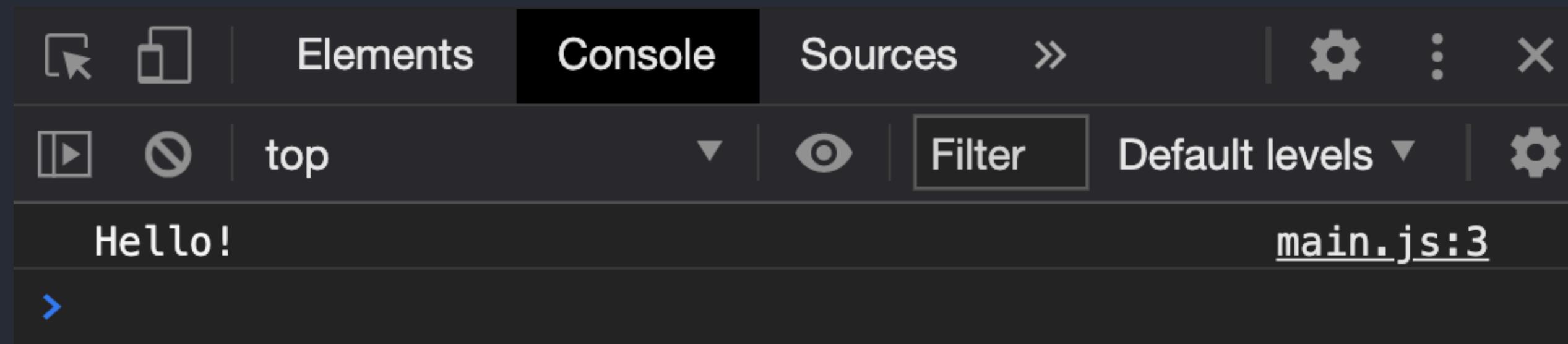
console.log(name + ": " + age + " years old.");
```



# Variables

Store data in the memory

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

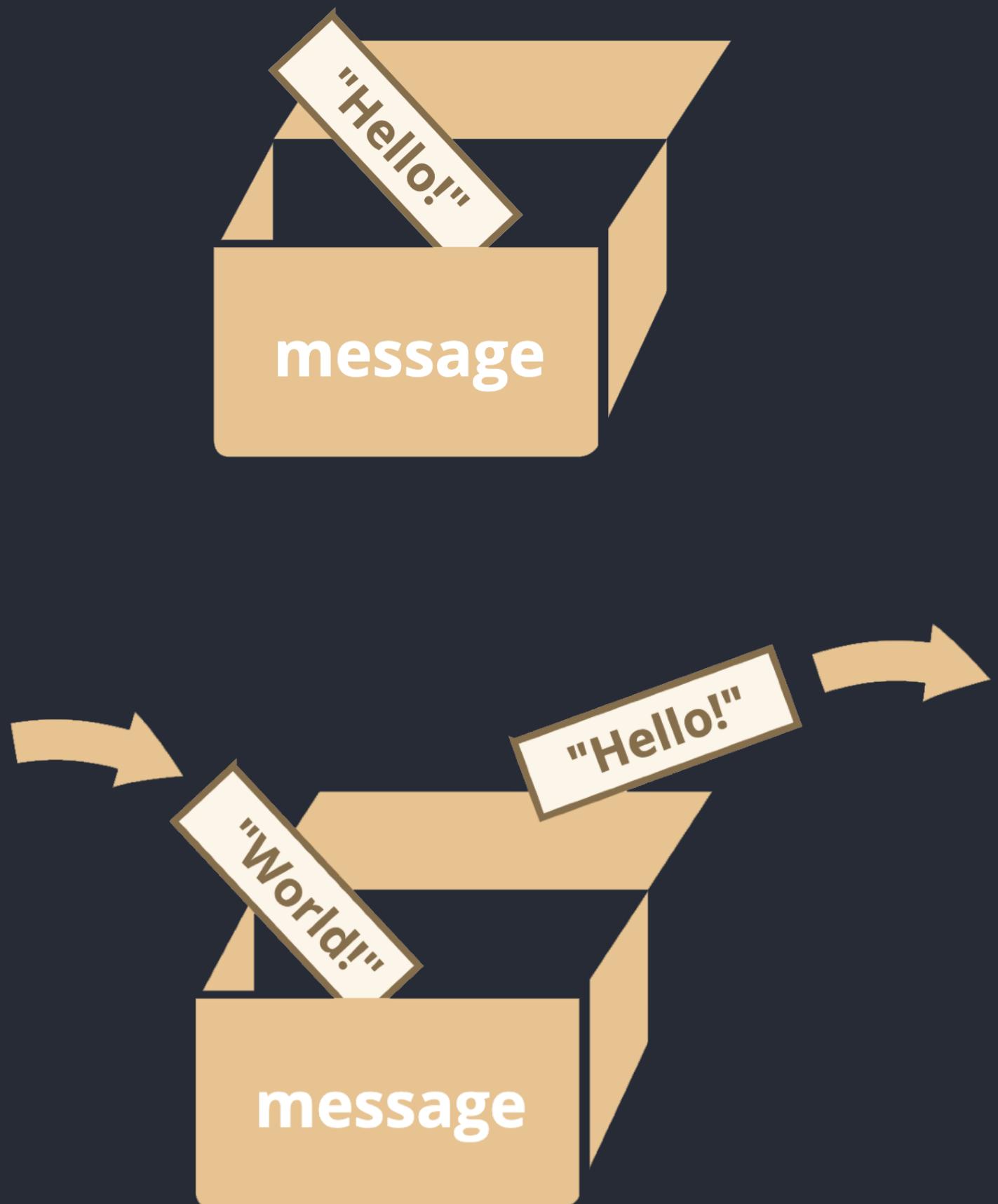
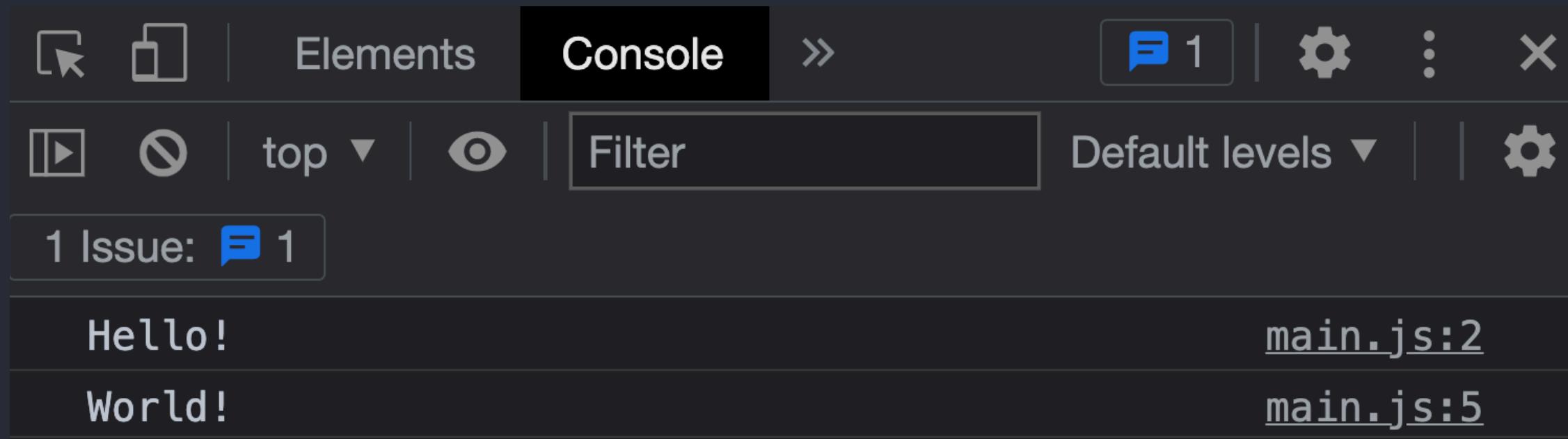


# Variables

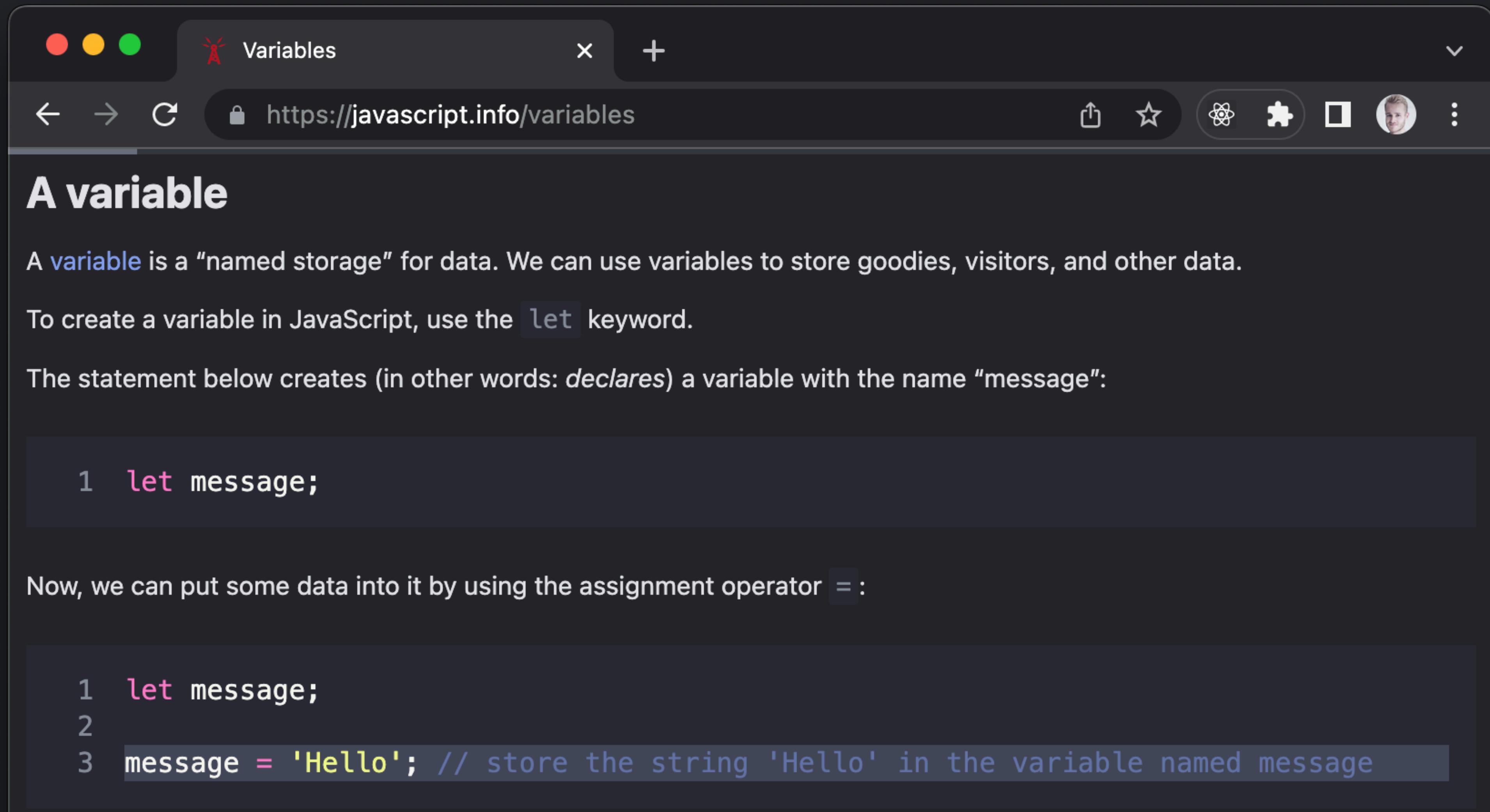
Store data in the memory

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

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



# JavaScript.info/Variables



The screenshot shows a dark-themed web browser window. The title bar reads "Variables". The address bar shows the URL "https://javascript.info/variables". The main content area displays the following text:

## 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”:

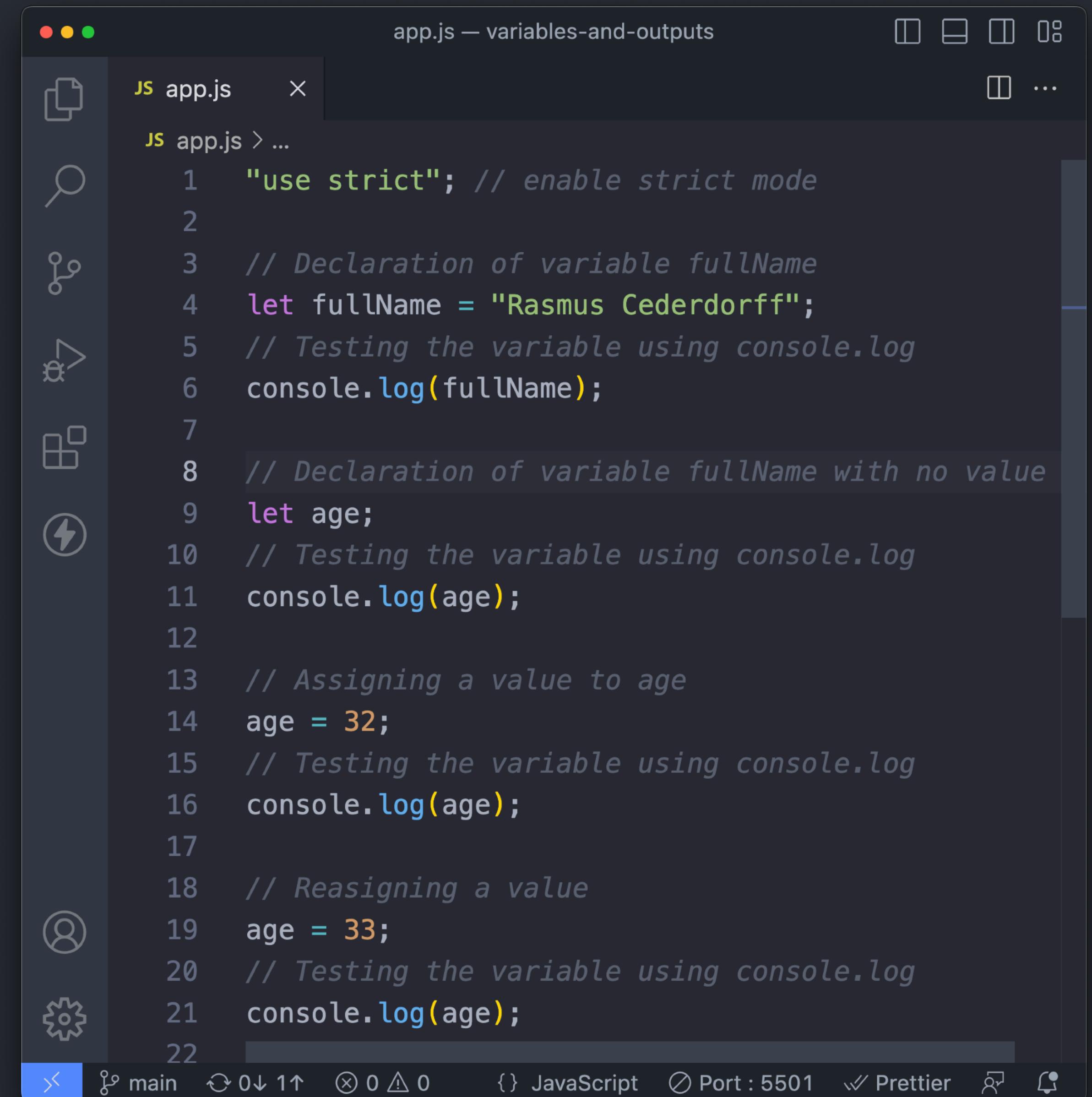
```
1 let message;
```

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 let keyword

1. Declare all your variables with the `let` keyword in front of the variable name.
2. Test your script using `console.log` and/or by typing the variables in the browser Console.
3. Try changing some of the variables and test again.



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

```
JS app.js > ...
1  "use strict"; // enable strict mode
2
3  // Declaration of variable fullName
4  let fullName = "Rasmus Cederdorff";
5  // Testing the variable using console.log
6  console.log(fullName);
7
8  // Declaration of variable fullName with no value
9  let age;
10 // Testing the variable using console.log
11 console.log(age);
12
13 // Assigning a value to age
14 age = 32;
15 // Testing the variable using console.log
16 console.log(age);
17
18 // Reassigning a value
19 age = 33;
20 // Testing the variable using console.log
21 console.log(age);
22
```

The code demonstrates the use of the `let` keyword for variable declaration, testing variables with `console.log`, and changing variable values.

# 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.

# 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

Use let & const  
instead of var

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

For now,

use let



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

## 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.

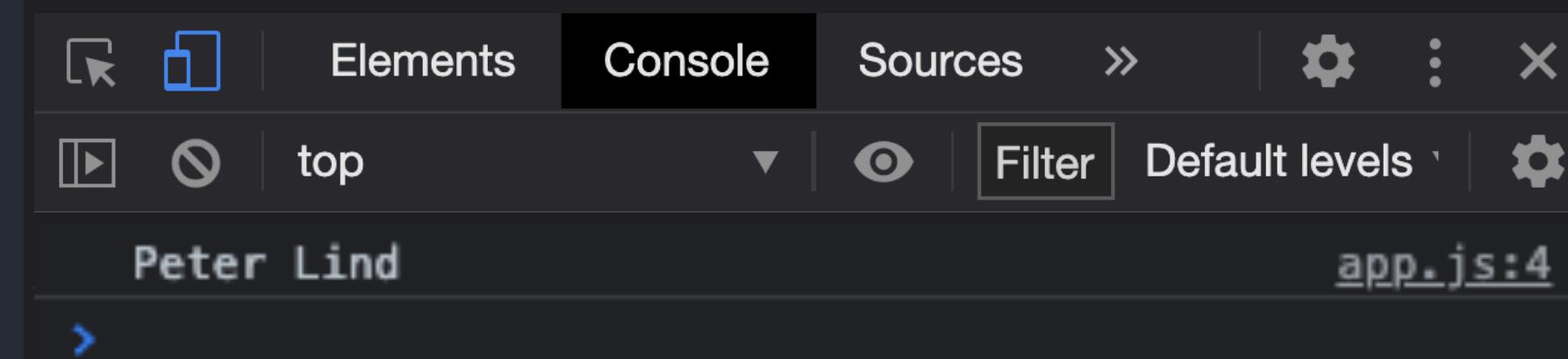
Please spend time thinking about the right name for a variable before declaring it. Doing so will repay you handsomely.

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`.

# 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

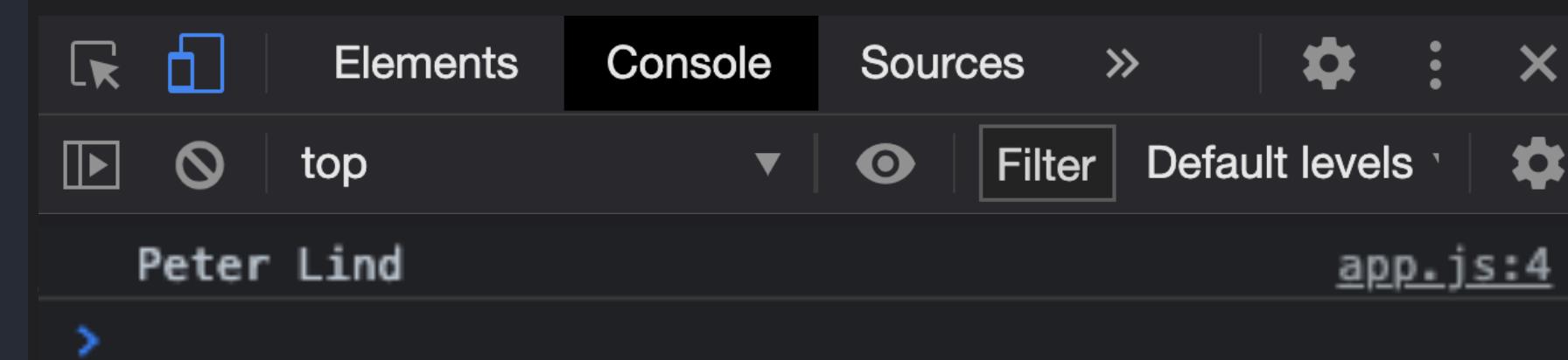
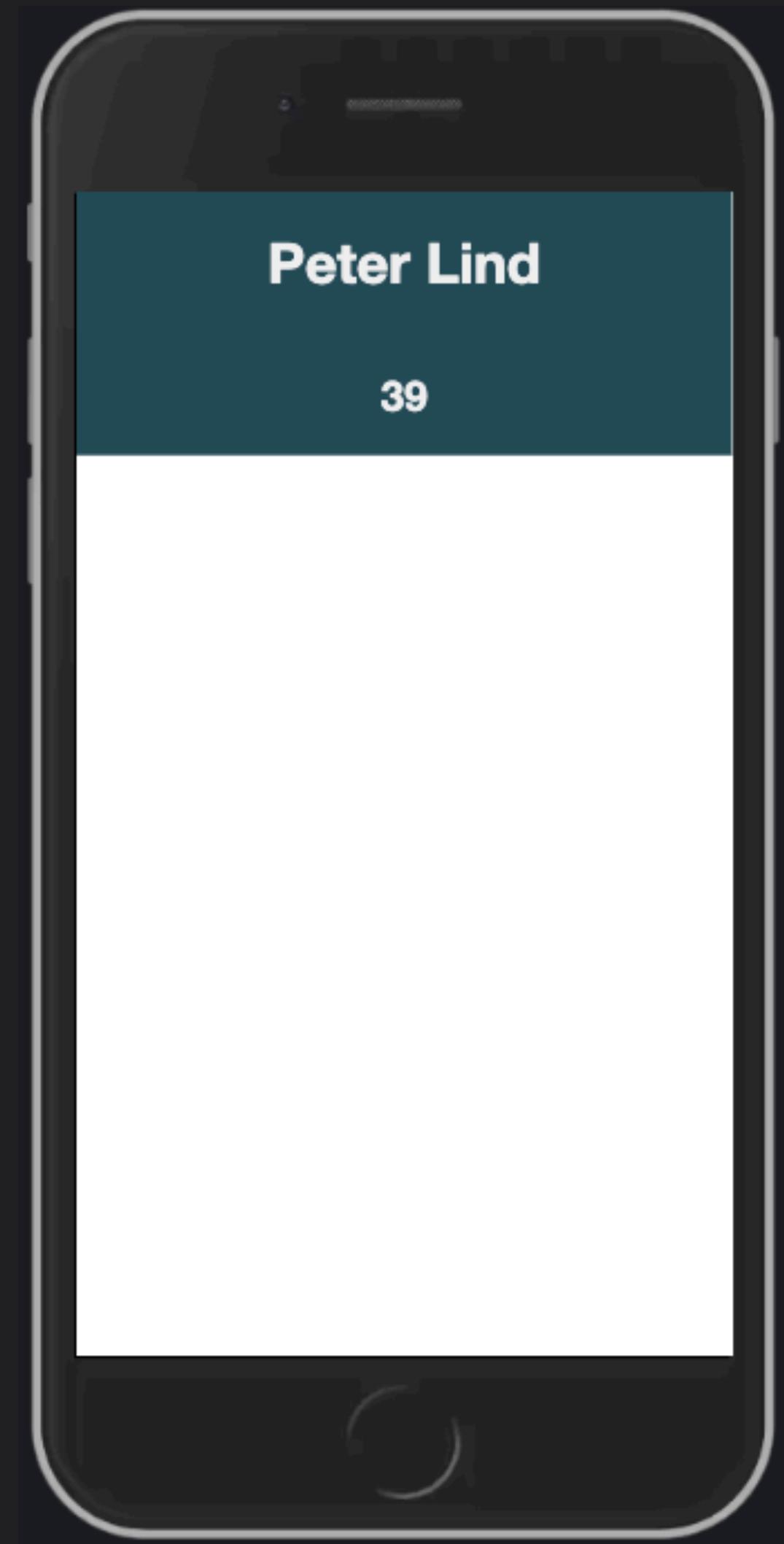
```
let fullName = "Peter Lind";
console.log(fullName);
document.querySelector("#fullName_container").textContent = fullName;
```

## .textContent

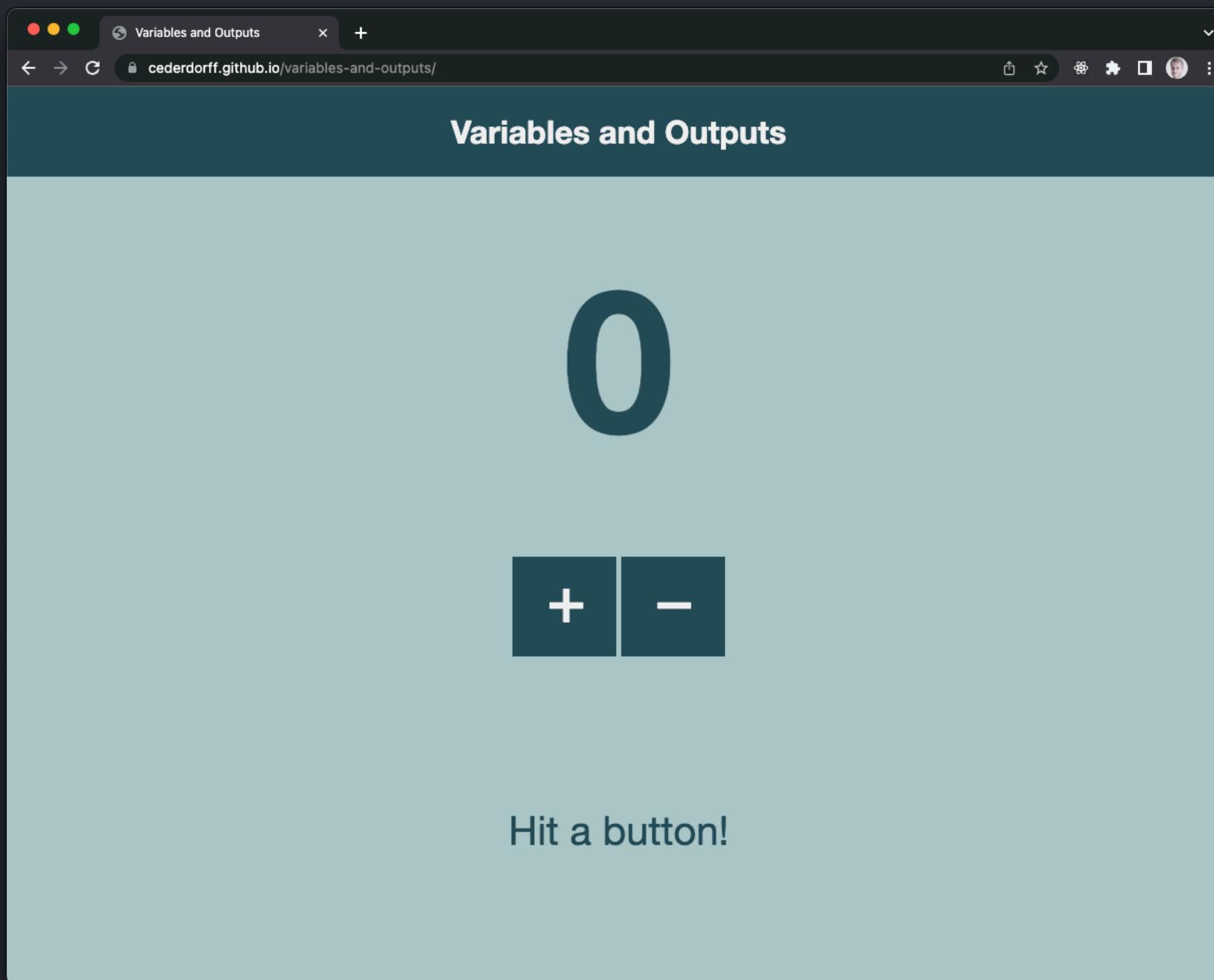
Get or set the text content of a given (HTML) element

# 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;
```



# Tæl et tal op og ned



1. Fork: [variables-and-outputs](#)
2. Kør og test med Live Server
3. Tilføj variablen `number` i toppen af `app.js`. Husk også  
`"use strict";`
4. Definer et `load` event som tilføjer `click` events til plus- og minus-knapperne (kig i HTML for at se deres id'er).
5. Plus skal kalde en funktion, der lægger en til `number`.
6. Minus skal kalde en funktion, der trække en fra `number`.
7. Husk løbende at teste med `console.log`.
8. Lav en funktion, der opdaterer `<p id="number">0</p>` med værdien fra `number`. Sørg for at funktionen kaldes hver gang `number` ændres.

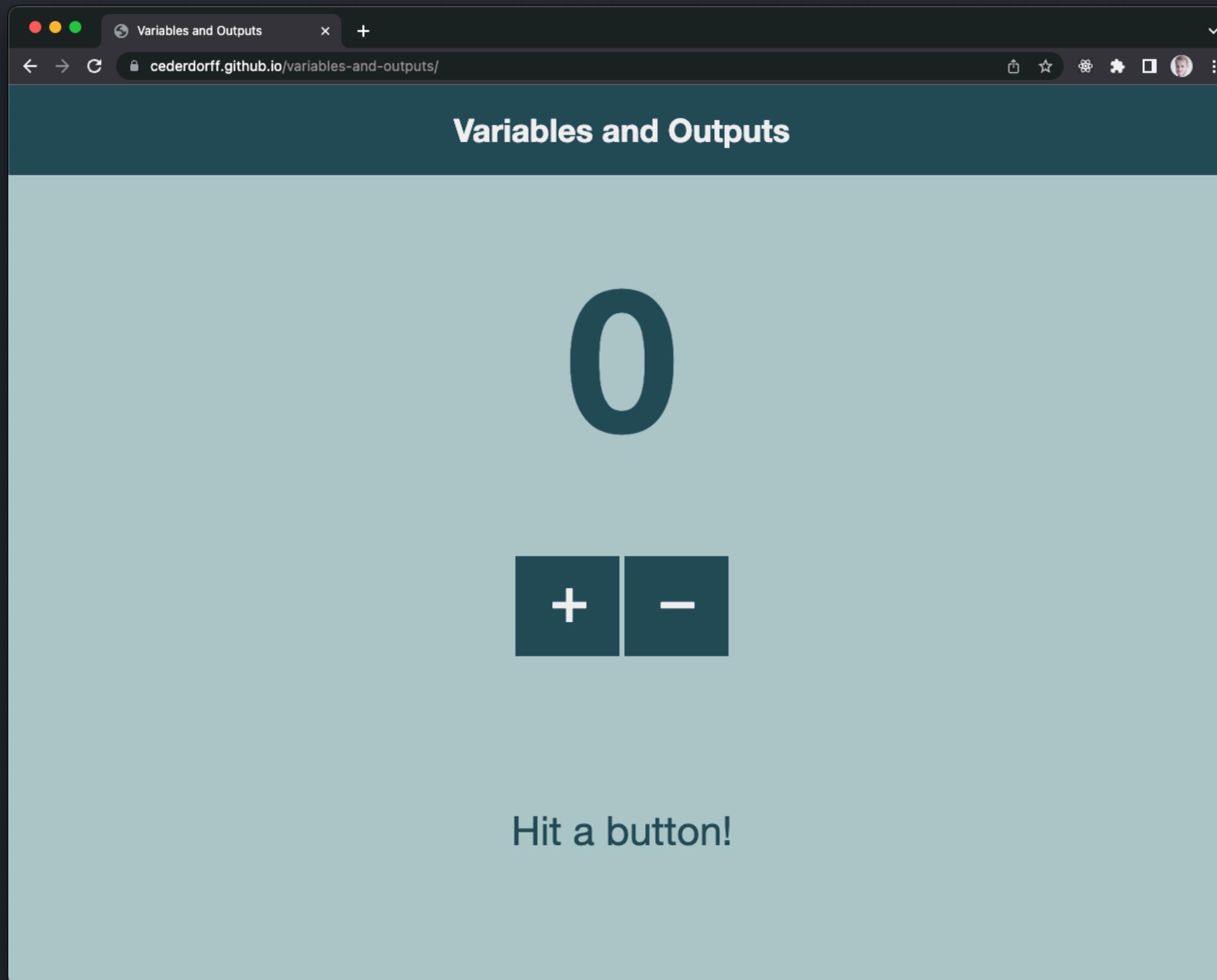
DU SKAL IKKE BARE VIDE.  
DU SKAL KUNNE.

# DU SKAL IKKE VIDE DET HELE FRA BEGYNDELSEN

Accepter det! Du vil kode og gøre ting, som måske ikke giver mening - det kommer!



# Separation of concerns



```
"use strict";
window.addEventListener("load", initApp);

// ===== Variables ===== //
let number = 0;

// ===== Init App Event ===== //
function initApp() { ... }

// ===== Increment and decrement functions ===== //
function incrementNumber() { ... }

function decrementNumber() { ... }

// ===== Display Number function ===== //
function displayNumber() { ... }
```

“

In computer science, separation of concerns is a design principle for separating a computer program into distinct sections.

”

Each section addresses a separate concern, a set of information that affects the code of a computer program.

Wikipedia

Separation of concerns

[https://en.wikipedia.org/wiki/Separation\\_of\\_concerns](https://en.wikipedia.org/wiki/Separation_of_concerns)

# Lad os spørge ...

Adskillelse af bekymringer. x +

← → C chat.openai.com/chat

+ New chat

Adskillelse af bekymri ✎ 🗑️

CSS Grid Layout-modellen

Hvad er separation of concerns?

Separation of concerns (på dansk: adskillelse af bekymringer) er et designprincip inden for softwareudvikling, hvor man opdeler et system i mindre, mere specialiserede og fokuserede komponenter eller moduler. Hvert modul eller komponent er ansvarligt for at håndtere en specifik opgave eller funktionalitet i systemet.

Formålet med separation of concerns er at reducere kompleksiteten i et system ved at opdele det i mindre og mere håndterbare dele. Dette gør det lettere at udvikle, teste, vedligeholde og skalere systemet, da hver del af systemet kan behandles separat.

For eksempel kan man i en webapplikation adskille de forskellige lag af funktionalitet, såsom brugergrænseflade, forretningslogik og databaselag. Ved at adskille disse bekymringer kan man fokusere på hver af dem individuelt og gøre ændringer eller opgraderinger uden at påvirke de andre dele af systemet.

Generelt kan man sige, at separation of concerns handler om at opdele et system i mindre, mere specialiserede dele, så man kan opnå bedre struktur, større fleksibilitet og nemmere vedligeholdelse.

# Separation of concerns

**HTML**



**JS**



**CSS**



STRUCTURE  
CONTENT

FUNCTIONALITY  
BEHAVIOR

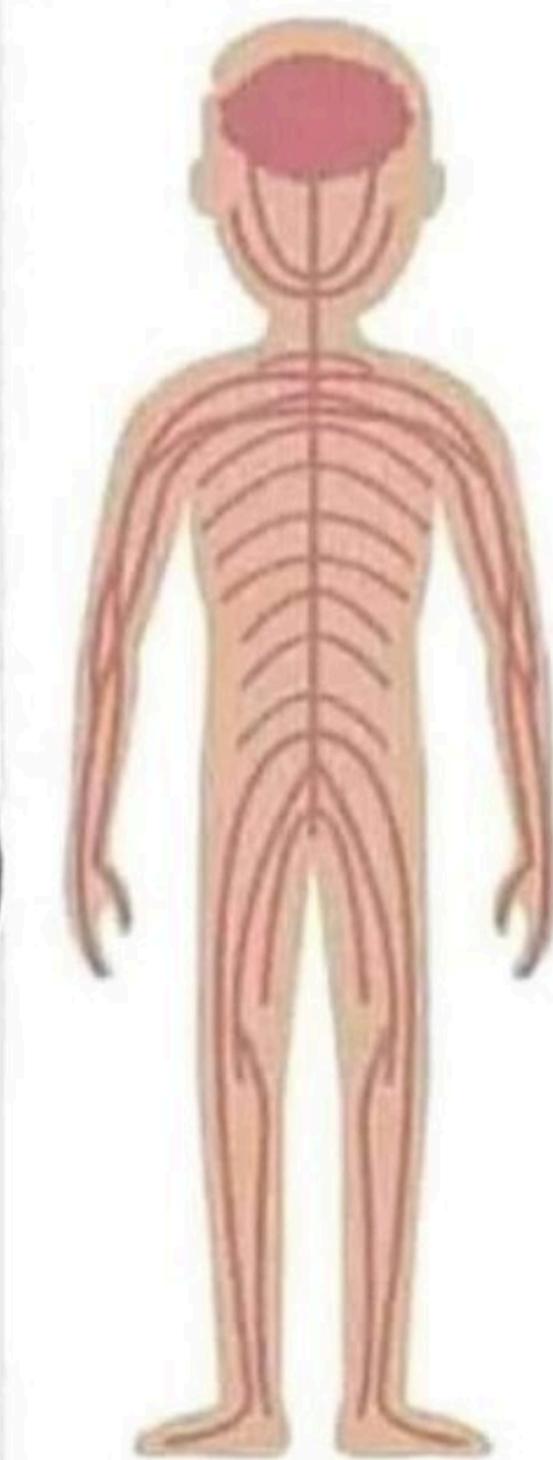
LAYOUT/STYLING  
PRESENTATION

HTML



Struktur  
Skelet

JS



Funktionalitet  
Nervesystem

CSS



Styling  
Tøj & makeup

Adskillelse af bekymringer. + New chat

chat.openai.com/chat

håndtere en specifik opgave eller funktionalitet i systemet.

Formålet med separation of concerns er at reducere kompleksiteten i et system ved at opdele det i mindre og mere håndterbare dele. Dette gør det lettere at udvikle, teste, vedligeholde og skalere systemet, da hver del af systemet kan behandles separat.

For eksempel kan man i en webapplikation adskille de forskellige lag af funktionalitet, såsom brugergrænseflade, forretningslogik og databaselag. Ved at adskille disse bekymringer kan man fokusere på hver af dem individuelt og gøre ændringer eller opgraderinger uden at påvirke de andre dele af systemet.

Generelt kan man sige, at separation of concerns handler om at opdele et system i mindre, mere specialiserede dele, så man kan opnå bedre struktur, større fleksibilitet og nemmere vedligeholdelse.

 Kan du forklare det i forhold til JavaScript?

 Ja, selvfølgelig! Separation of concerns er også relevant i forhold til JavaScript og webudvikling generelt.

I JavaScript kan man anvende separation of concerns ved at opdele koden i mindre, mere specialiserede moduler eller funktioner, der er ansvarlige for specifik funktionalitet eller opgaver. For eksempel kan man adskille koden til at håndtere brugergrænsefladen (DOM-manipulation), forretningslogikken og kommunikationen med serveren.

Ved at adskille disse bekymringer kan man opnå flere fordele. For det første gør det koden mere overskuelig og nemmere at vedligeholde, da man kan fokusere på en specifik del af koden ad gangen. Desuden gør det koden mere genanvendelig, da man kan bruge de samme moduler flere steder i koden.

Her er et eksempel på separation of concerns i JavaScript:

```
javascript
// Brugergrænseflade-modul
```

Copy code Regenerate response

ChatGPT Feb 13 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

# Separation of concerns (MVC)

```
"use strict";
window.addEventListener("load", initApp);
```

```
// ===== Variables ===== //
let number = 0;
```

Logik, der laver variabel

```
// ===== Init App Event ===== //
```

```
function initApp() { ...
}
```

```
// ===== Increment and decrement functions ===== //
```

```
function incrementNumber() { ...
}
```

Logik, der tæller number op og ned

```
function decrementNumber() { ...
}
```

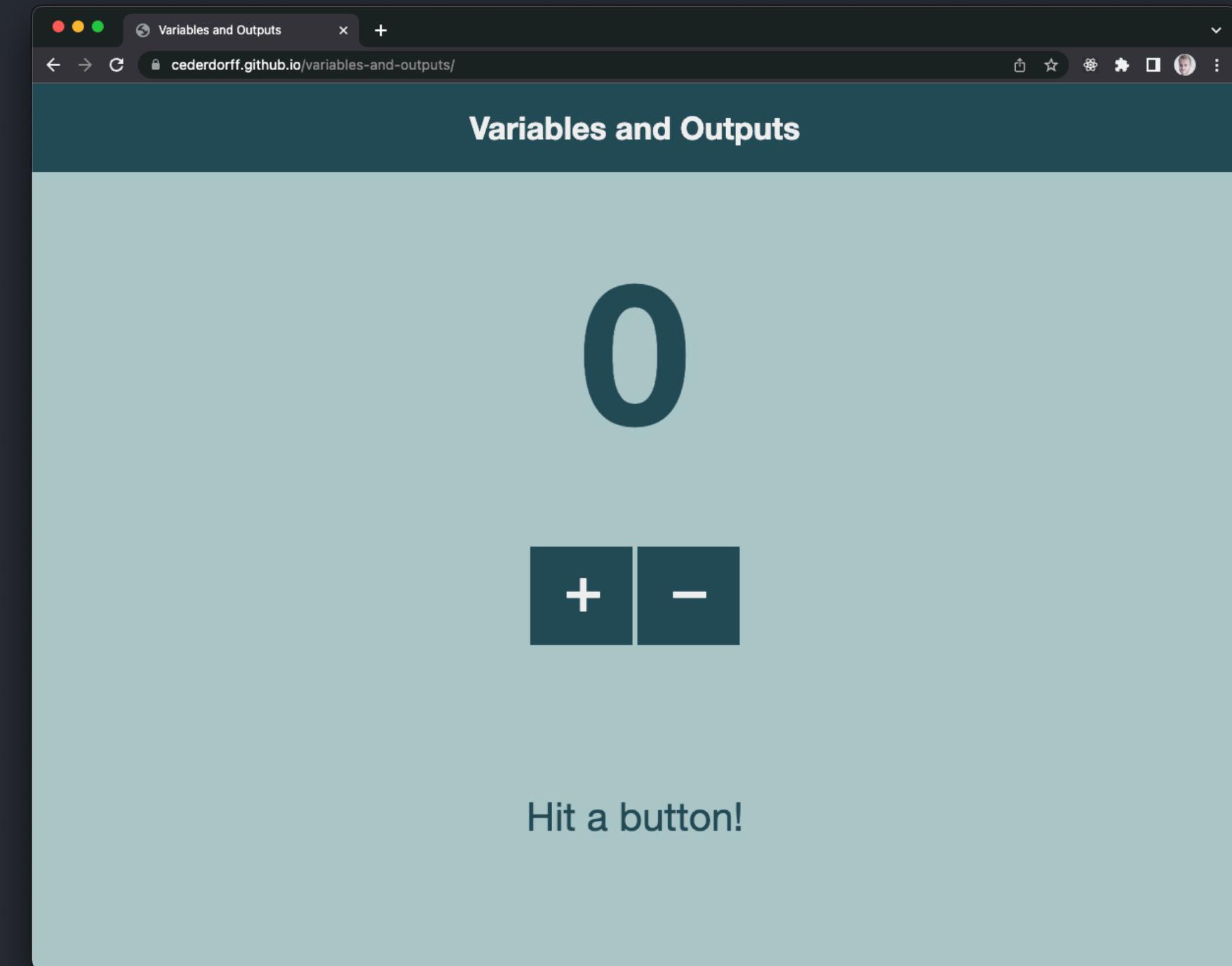
```
// ===== Display Number function ===== //
```

```
function displayNumber() { ...
}
```

Logik, der opdaterer vores UI (HTML)

- Inddel i (mindre) funktioner -> “*at opdele et system i mindre, mere specialiserede dele, så man kan opnå bedre struktur, større fleksibilitet og nemmere vedligeholdelse.*”
- Hver funktion udfører en afgrænset specialiseret opgave.
- Separer logik, der opdaterer fx variabler (model/controller), fra logik der opdaterer vores UI (view). Dvs at logikken, der opdaterer data, er separeret fra logikken, der opdaterer UI (HTML).

# Tæl et tal op og ned

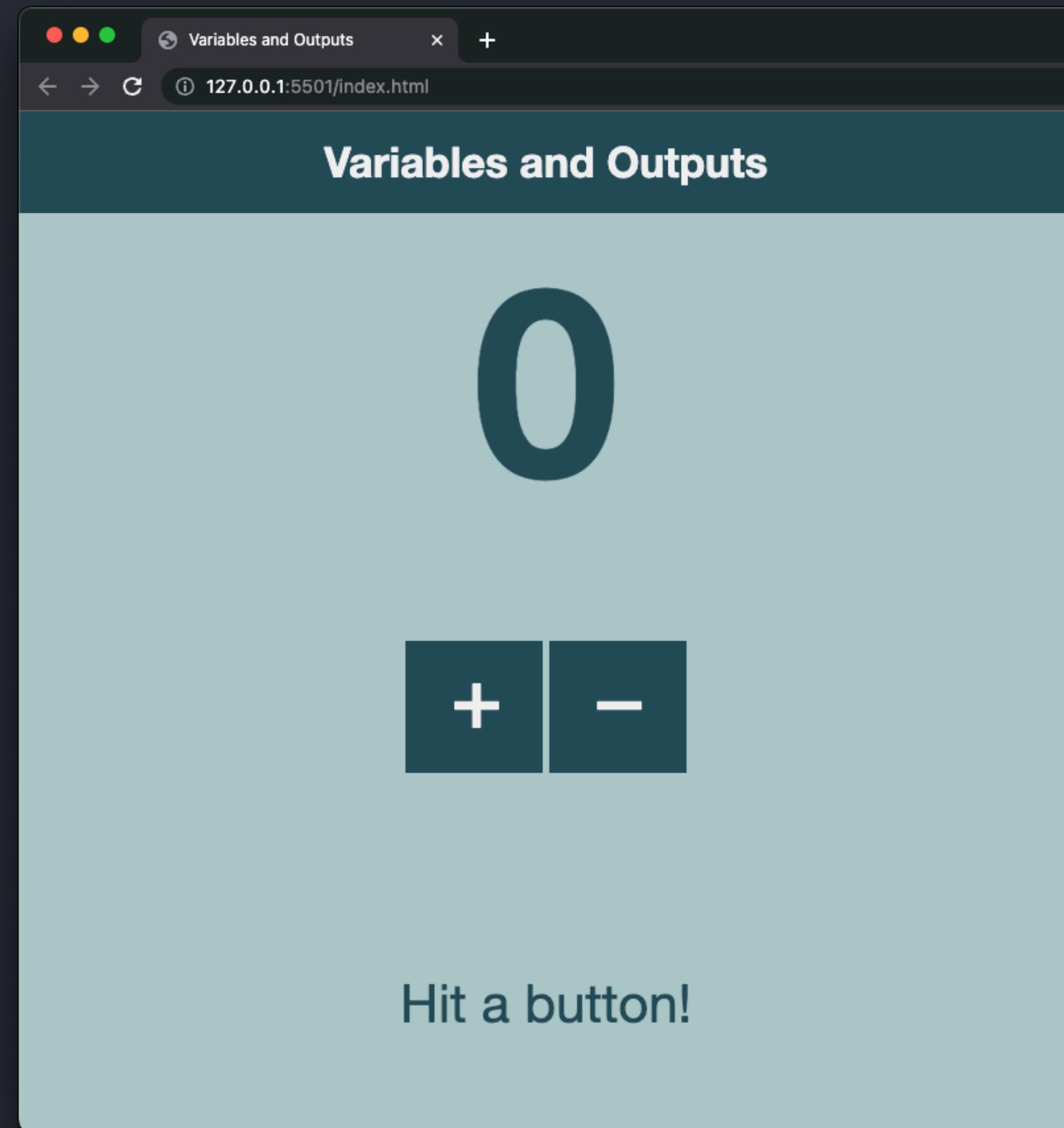


To-og-to:

- “Overholdes” separation of concerns?
- Er logikken, der opdaterer data, separeret fra logikken, der opdaterer UI (HTML)?
- Forbedringsforslag?

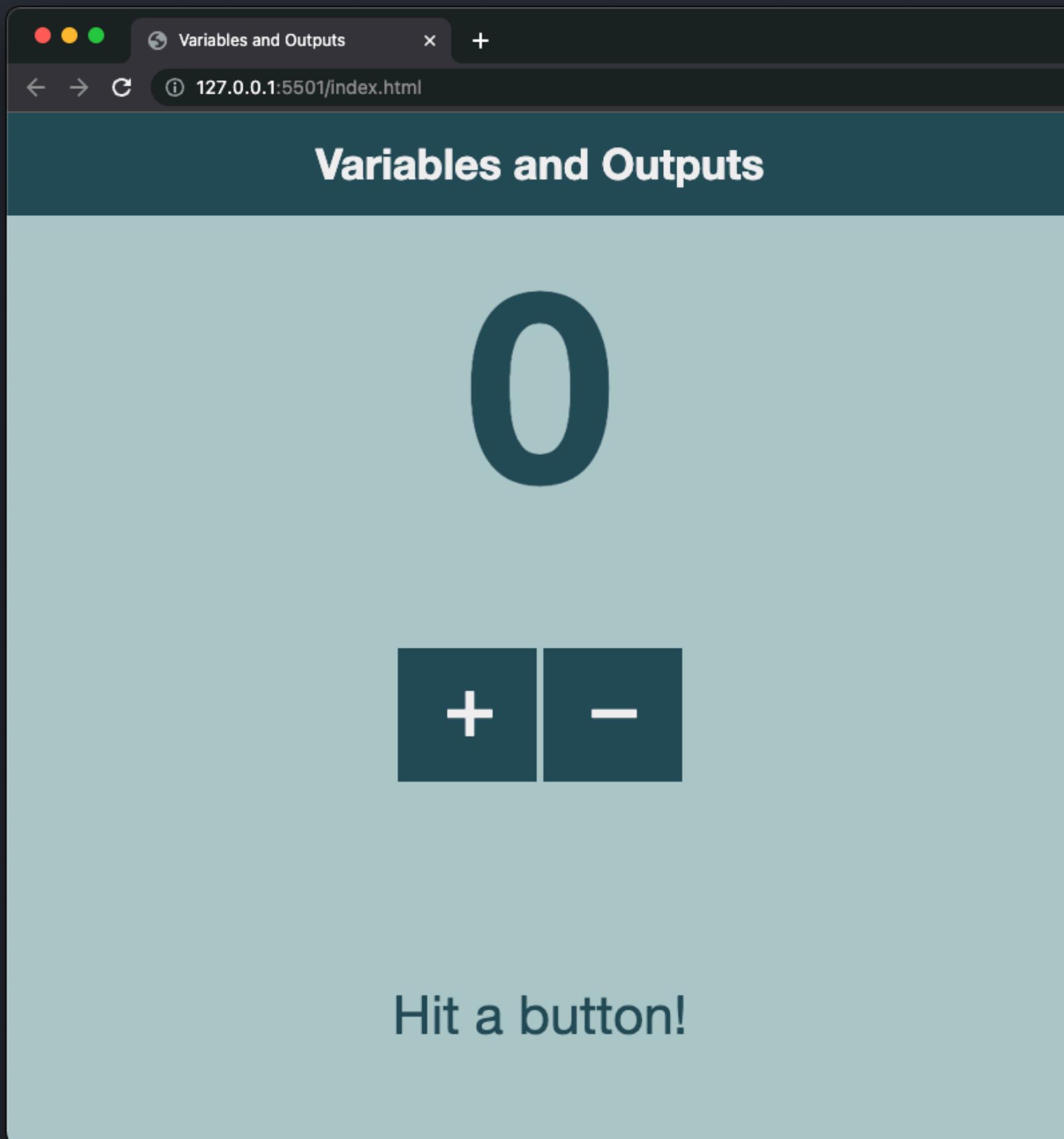
# “Tælle” på en variabel

```
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
```



# Udskrive til DOM'en

```
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  
  
document.querySelector("#mitElement").textContent = variableName;
```



# 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.

# 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 liv = 3;  
document.querySelector("#liv").textContent = "Du har " + liv + " liv tilbage";
```



```
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`.

# Concatenation and UI

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

let message = `${fullName} is ${age} years old.`;
console.log(message);
document.querySelector("#message_container").textContent = message;
```

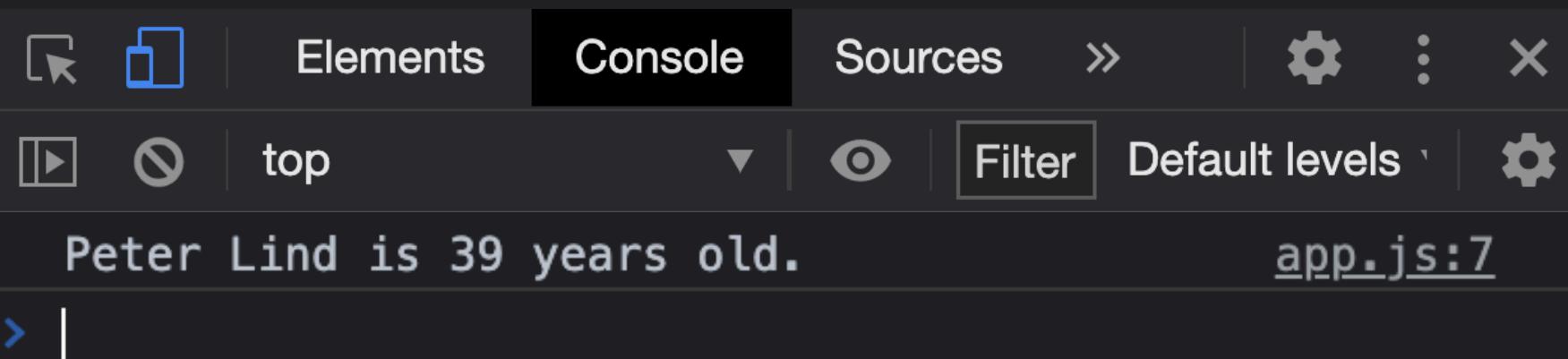
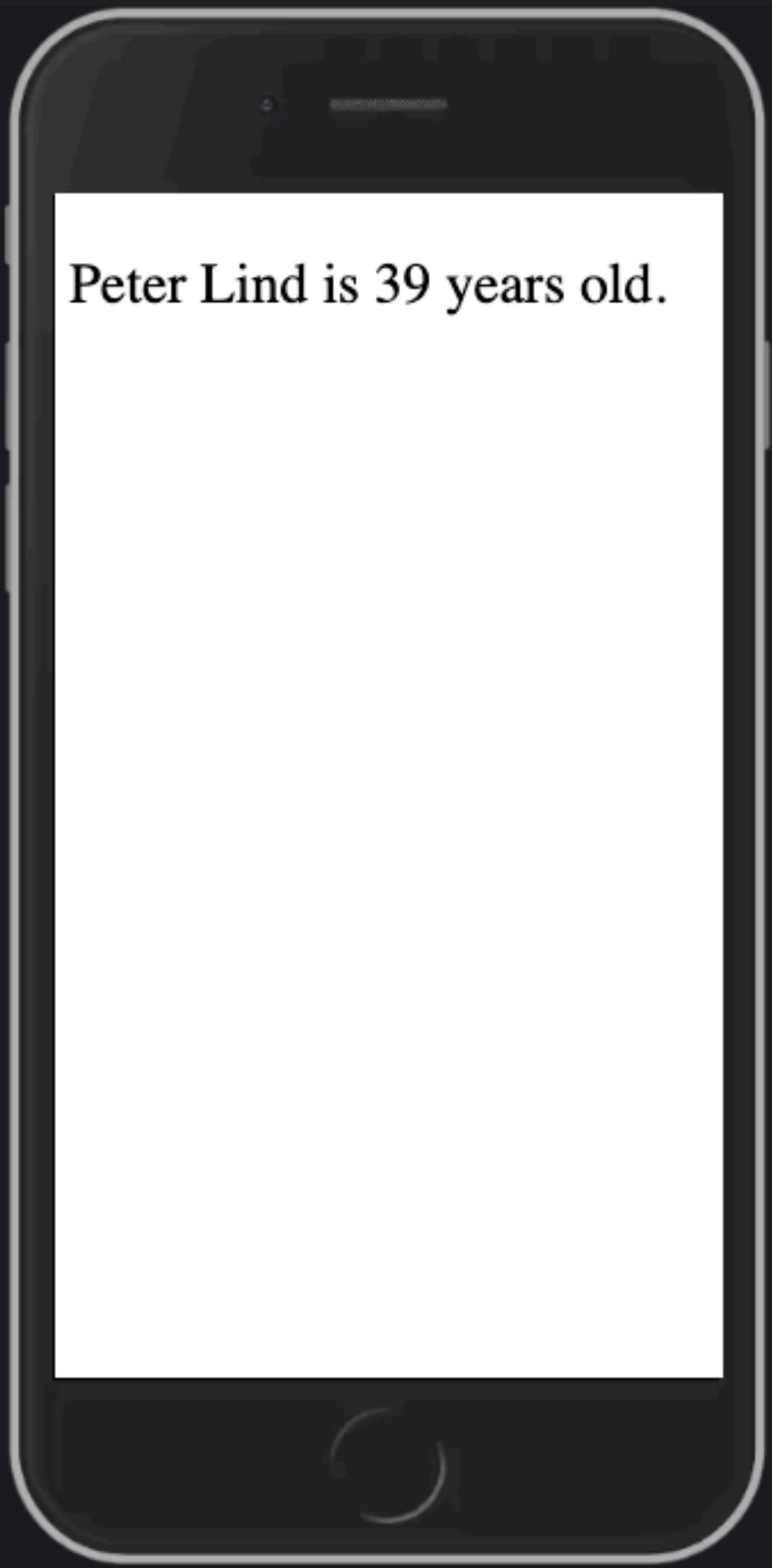
```
<body>
  <main>
    <p id="message_container"></p>
  </main>
  <script src="app.js"></script>
</body>
```



# Concatenation and UI

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

let message = `${fullName} is ${age} years old.`;
console.log(message);
document.querySelector("#message_container").textContent = message;
```



# `backtick string`

*“Template literals are literals delimited with backtick () characters, allowing for multi-line strings, for string interpolation with embedded expressions, and for special constructs called tagged templates.”*

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

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

```
console.log(` ${name} is ${age} years old. `);
```

Alicia is 6 years old.

[main.js:10](#)

Alicia is 6 years old.

[main.js:12](#)

# `backtick string`

## Template String / Template Literals

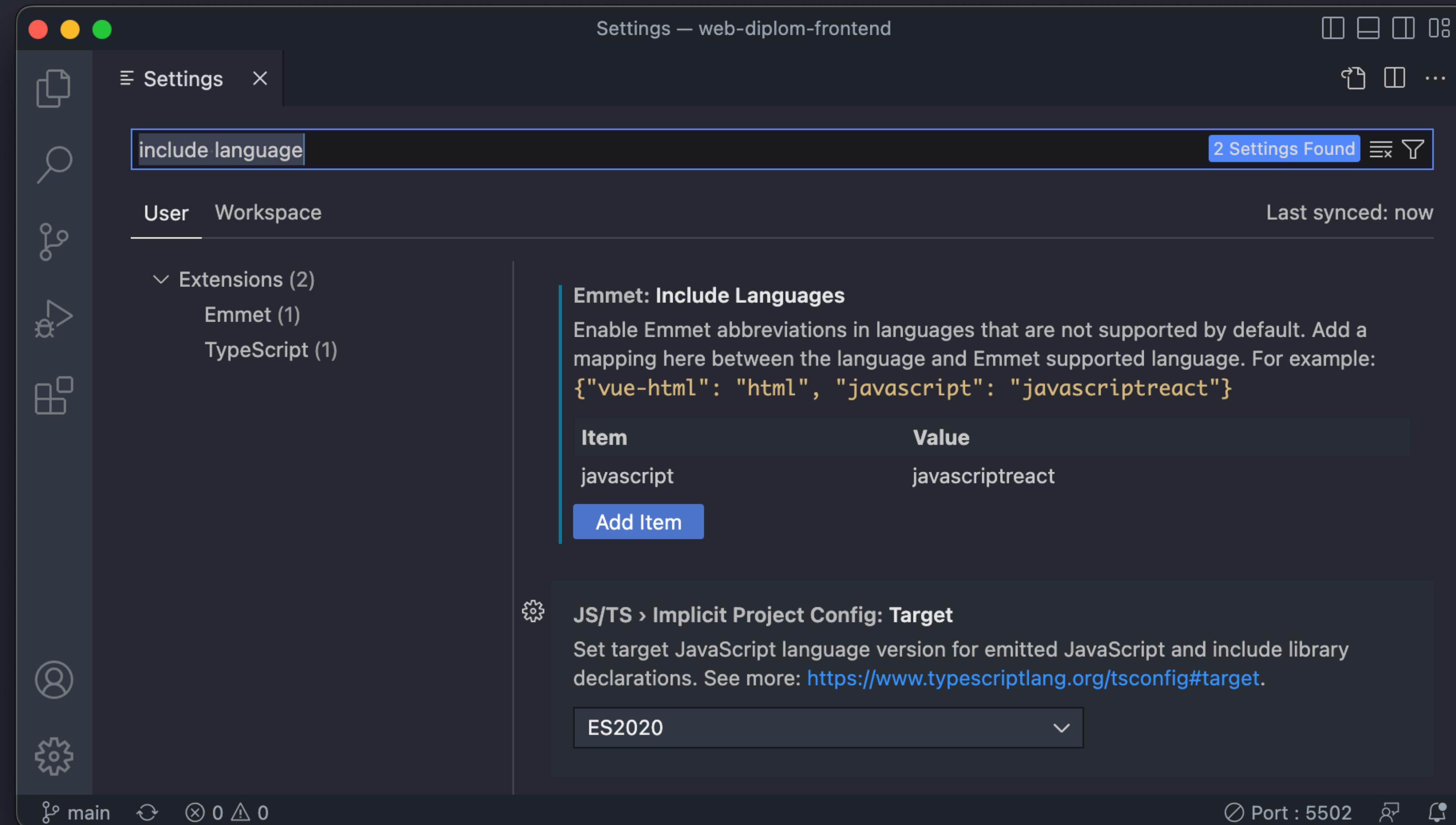
- Extended functionality
- Simplifies concatenating strings
- Embed values and expression into a string with \${...}
- Simplifies the syntax and the reading
- Let us create more readable HTML templates

```
let name = "Alicia";
console.log(`Hello, ${name}`);
```

Hello, Alicia

main.js:8

# Add language support in template string



# Global Variables

Variables outside a function (and scopes) are global variables

# [JavaScript.info/function-basics#local-variables](https://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
```

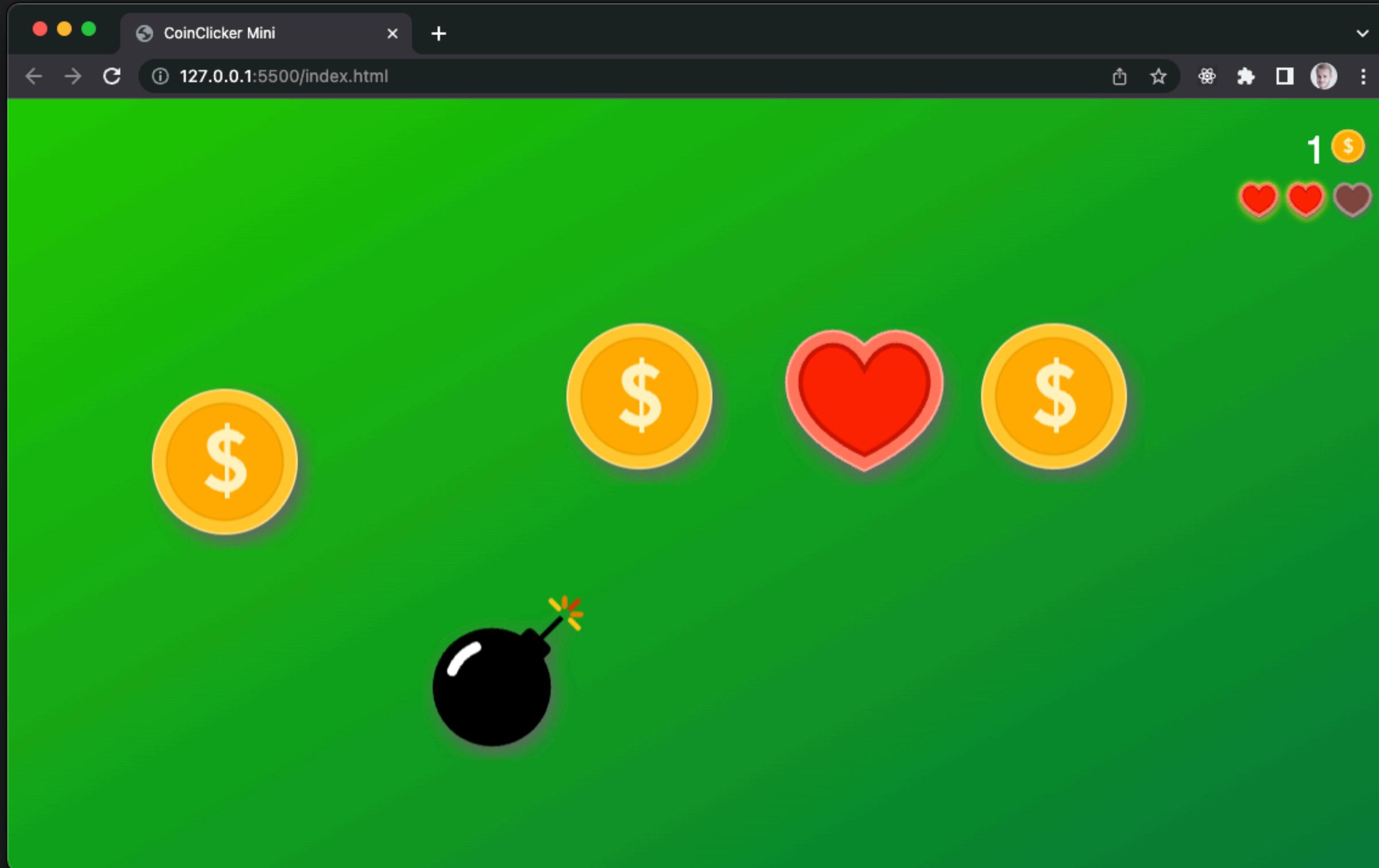
## Global variables

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).

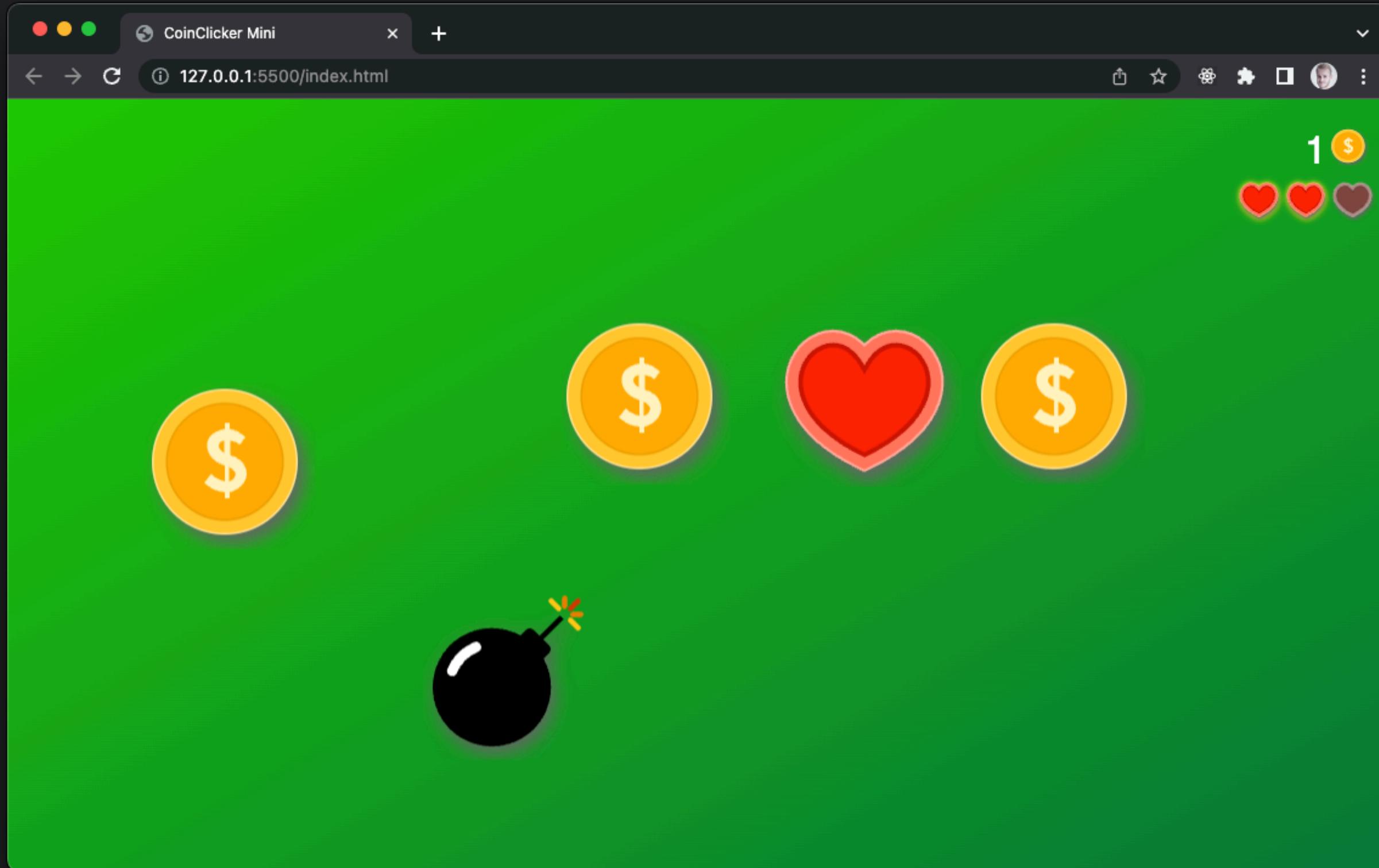
It's a good practice to minimize the use of global variables. Modern code has few or no globals. Most variables reside in their functions. Sometimes though, they can be useful to store project-level data.

# Tæl point og liv



1. Download zip: [Coinclicker-mini-2023 \(userinterface\)](#)
2. Udpak og placer projektmappe i din lokale “kodemappe”.
3. Åben Coinclicker-mini-2023-userinterface i VS Code, orienter dig i koden. Test hvad du har indtilvidere, før du går videre.

# Tæl point og liv



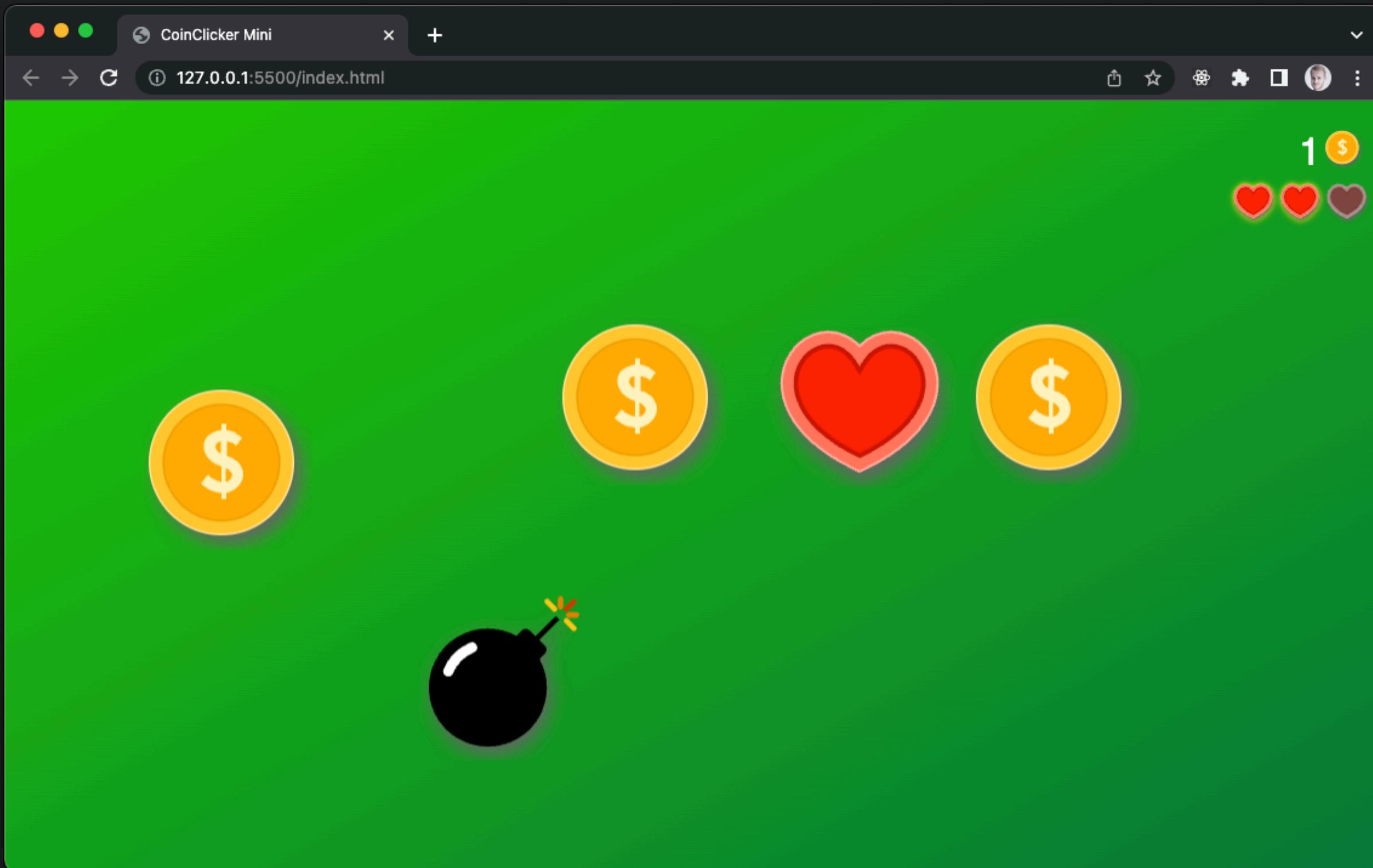
Du skal nu tilføje logik, der kan tælle point, når du trykker på en mønt. Denne del minder meget om foregående øvelse.

Husk løbende at teste med `console.log`.

4. Tilføj variablen `points` i toppen af `script.js`. Husk også `"use strict"`;
5. I `load` eventet tilføjes der allerede et `click` event til `#coin1_container`, som hedder `clickCoin`. Kontroller evt. hvad `clickCoin` gør.
6. `clickCoin` skal kalde en funktion, der tæller `points` op. Lav denne funktion og kald den `incrementPoints`. Test med `console.log`, da vi endnu ikke gør noget for at opdatere UI (HTML'en).
7. Din `incrementPoints` funktion skal, foruden at tælle `points` op, kalde en ny funktion, der viser den opdaterede værdi af `points`. Kald funktionen `displayPoints`.
8. `displayPoints` skal opdatere `#coin_count`'s `.textContent` med `points`.

Husk løbende at teste og brug `console.log`.

# Tæl point og liv

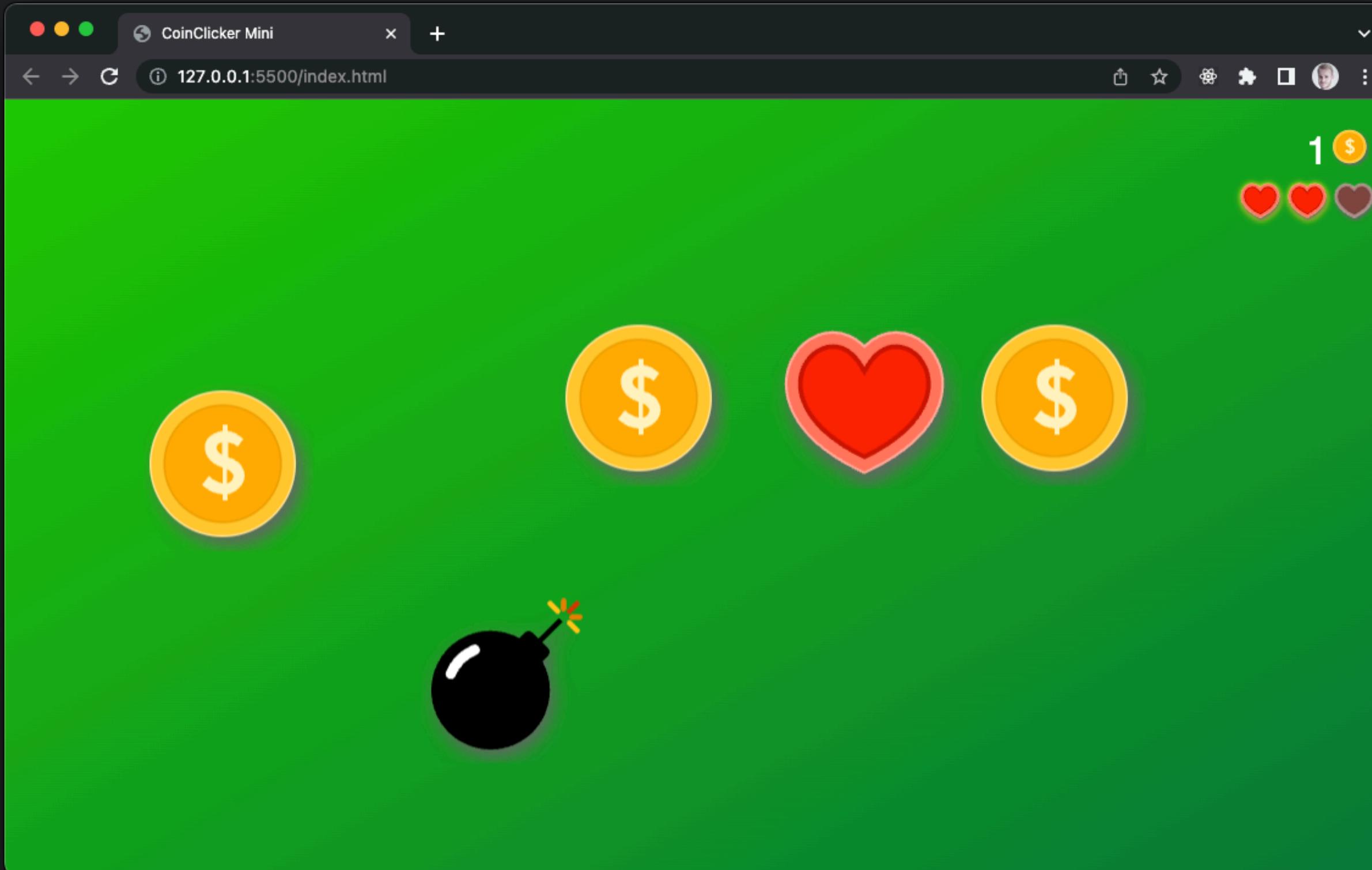


Når du trykker på en bombe, skal du tælle liv ned. Dette minder meget om de foregående øvelser, du har været igennem. Du skal dog ikke vise et tal for antal liv, men hjerter, der er grå, hvis du har mistet liv.

Husk løbende at teste med `console.log`.

9. Tilføj variablen `lives` i toppen af `script.js`, samme sted som du tilføjede `points`.
10. I `load` eventet tilføjes der allerede et `click` event til `#bomb_container`, som hedder `clickBomb`. Kontroller evt. hvad `clickBomb` gør.
11. `clickBomb` skal kalde en funktion, der tæller `lives` ned. Lav denne funktion og kald den `decrementLives`. Test med `console.log`, da vi endnu ikke gør noget for at opdatere UI (HTML'en).
12. Din `decrementLives` funktion skal, foruden at tælle `lives` ned, kalde en ny funktion, der viser den opdaterede værdi af `lives`. Kald funktionen `displayDecrementLives`.
13. `displayDecrementLives` skal opdatere visningen af hjerter. Dette styres med CSS klasserne `active_heart` og `broken_heart`. Med `querySelector` skal du få fat i "`#heart`" + `lives`, fjerne klassen `active_heart` og tilføje klassen `broken_heart`.

# Tæl point og liv



## Ekstraopgaver

14. Click på hjerte skal tælle lives op.
15. Sørg for at alle mønster fungerer og tæller points op.
16. Bombe kan eventuelt fratrække et antal points.
17. Husk at opdatere UI. Du kan muligvis genbruge nogle af dine eksisterende funktioner.



Code  
Every  
Day