

Conditional Statements

if, if-else & else-if

Agenda

- if og if-else
- Operatorer og sammenligning
- else-if
- Vis “Game Over” eller “Level Complete”
- Genstart animationer
- Opsamling og vejledning

- Tæl et tal op og ned (1)
- Tæl et tal op og ned (2)
- Tæl et tal op og ned (3)
- Child, adult or senior? (1)
- Child, adult or senior? (2)
- Vis “Game Over” eller “Level Complete”
- Clicker-spil - Game Over og Level Complete

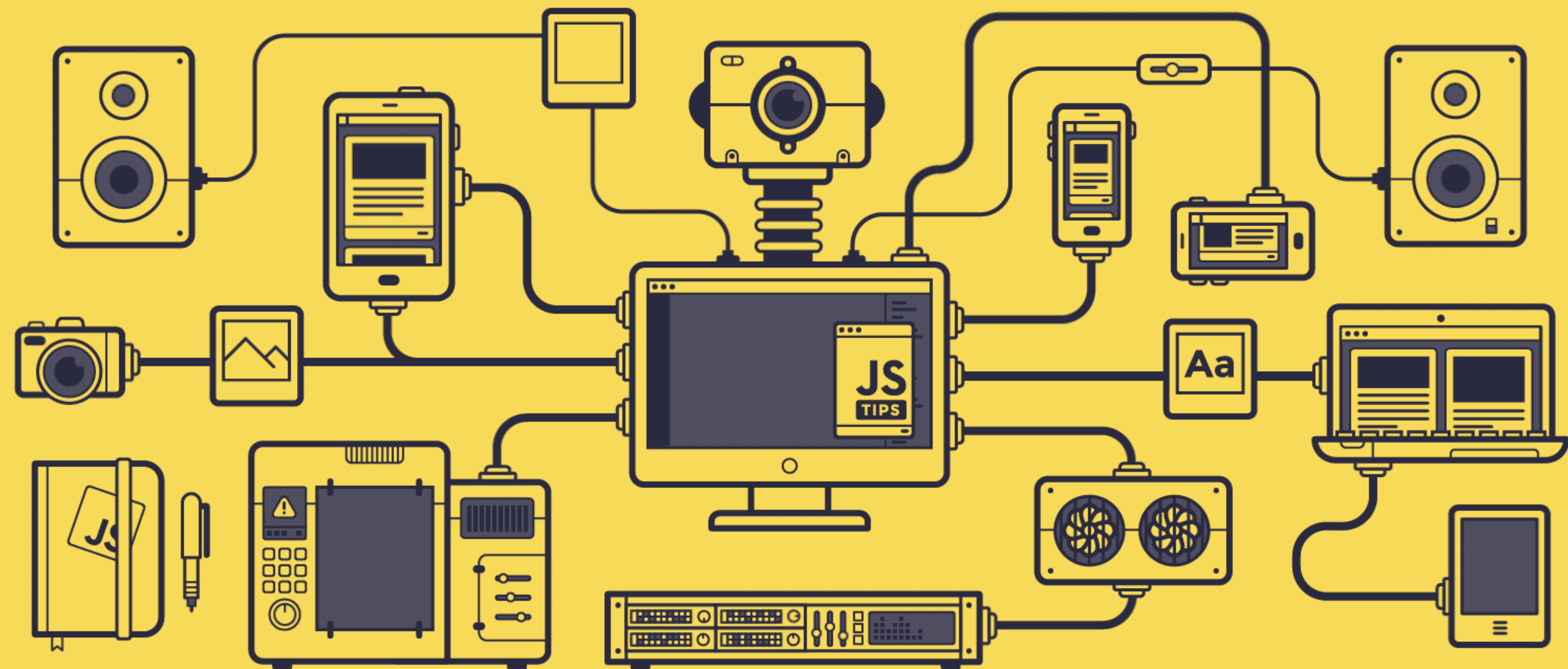


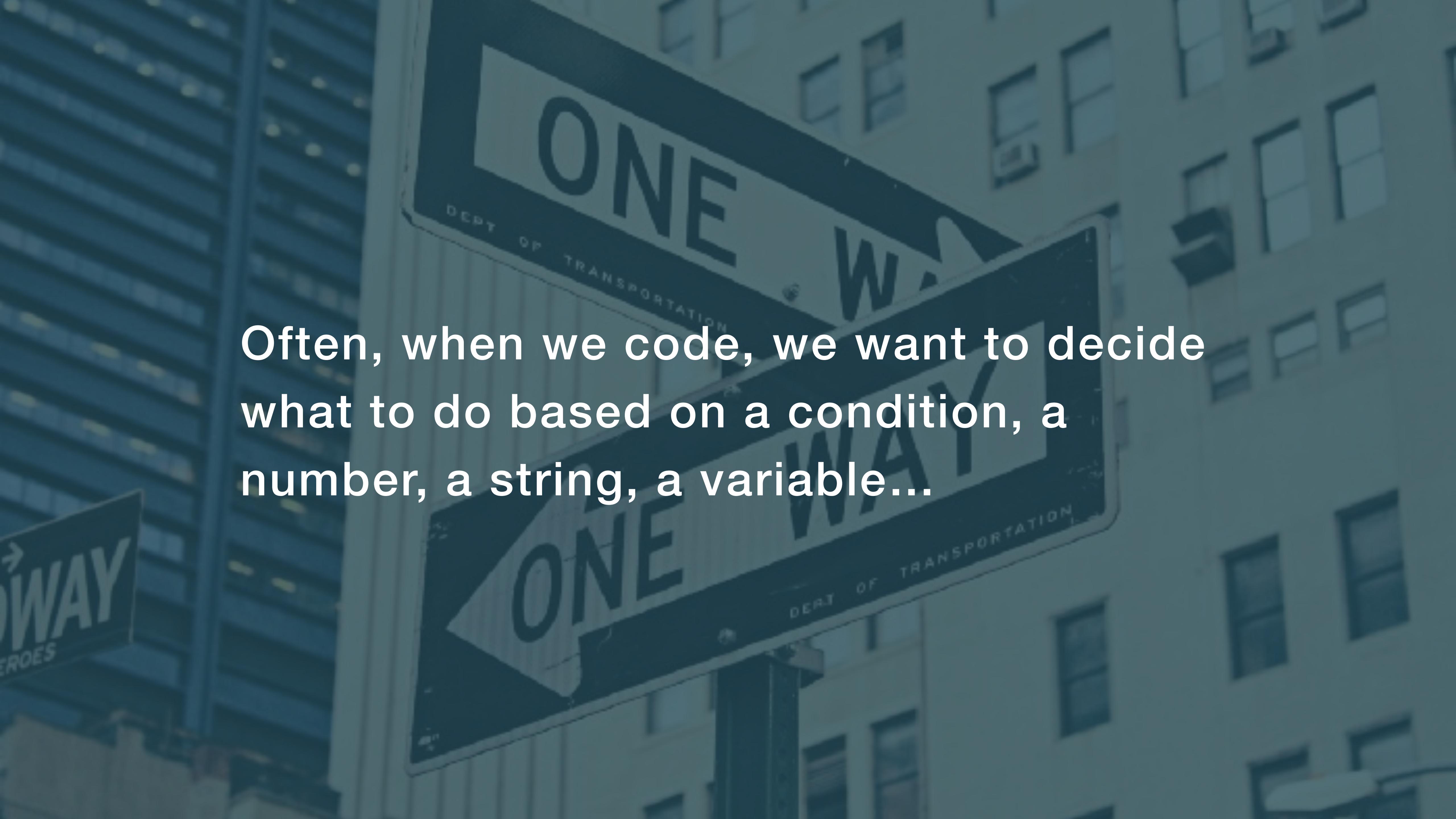
Øvelser

Programming knowledge

JavaScript

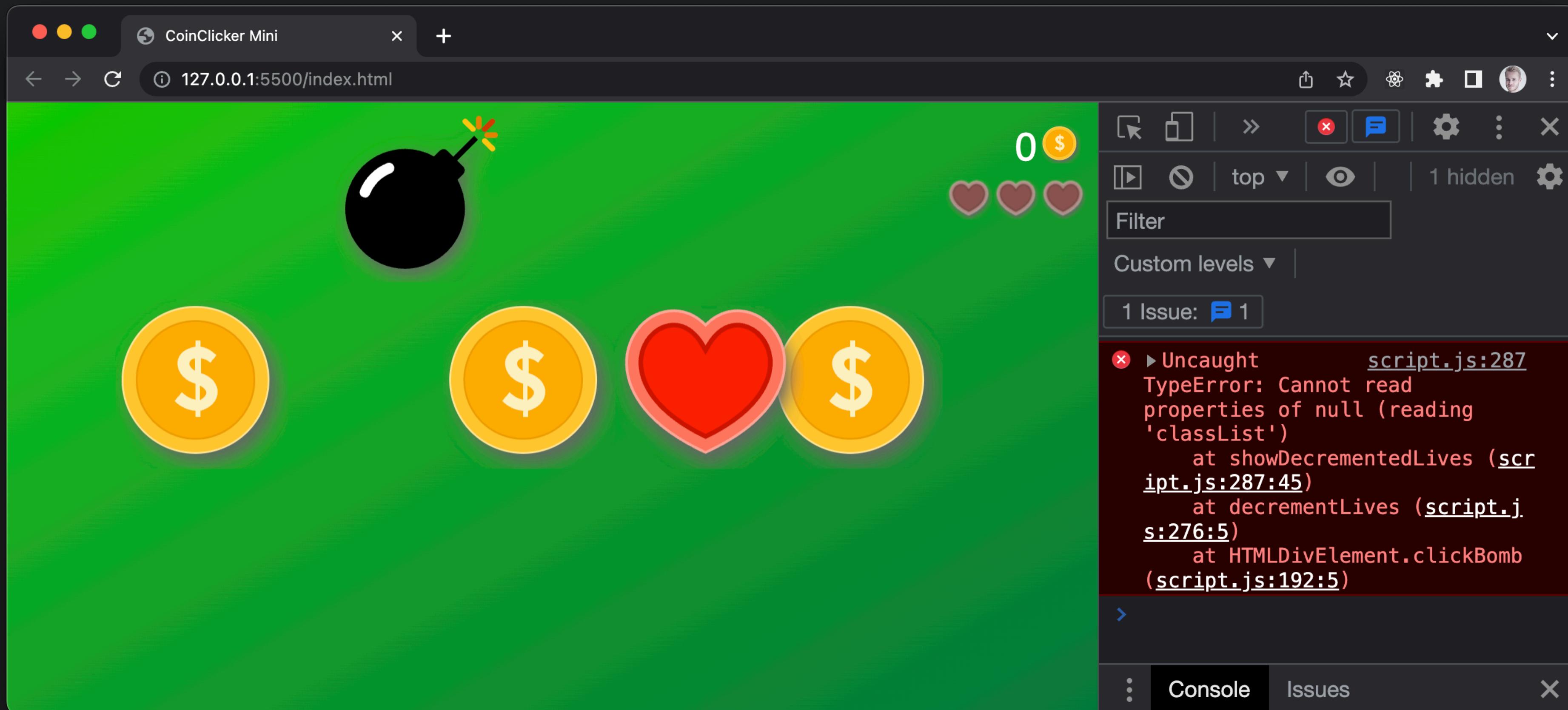
```
index.html
85  }
86
87  /*
88   * Fetches post data from my headless CMS
89  */
90  function getPersons() {
91    fetch('http://headlesscms.cederdorff.com/wp-json/wp/v2/posts?_embed=true')
92      .then(function(response) {
93        return response.json();
94      })
95      .then(function(persons)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



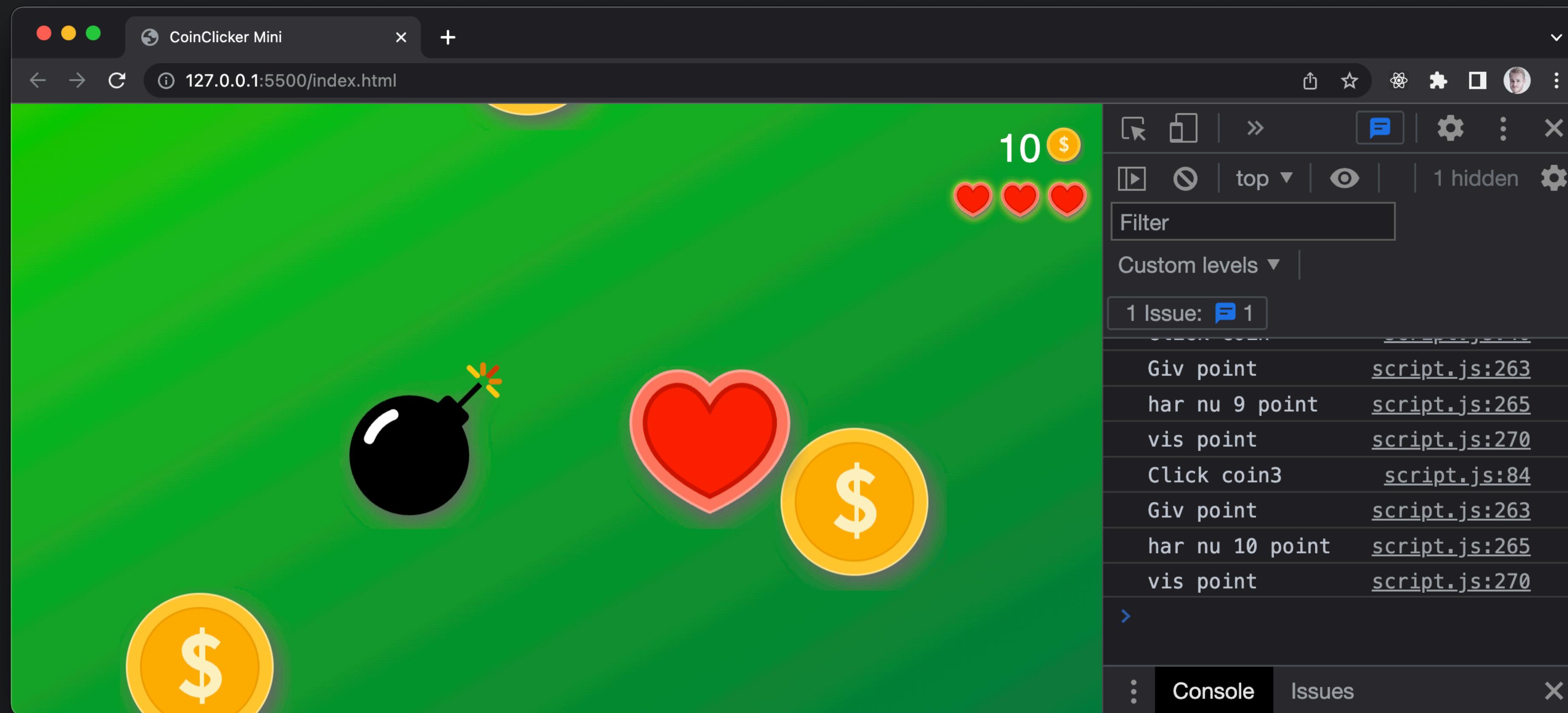


Often, when we code, we want to decide what to do based on a condition, a number, a string, a variable...

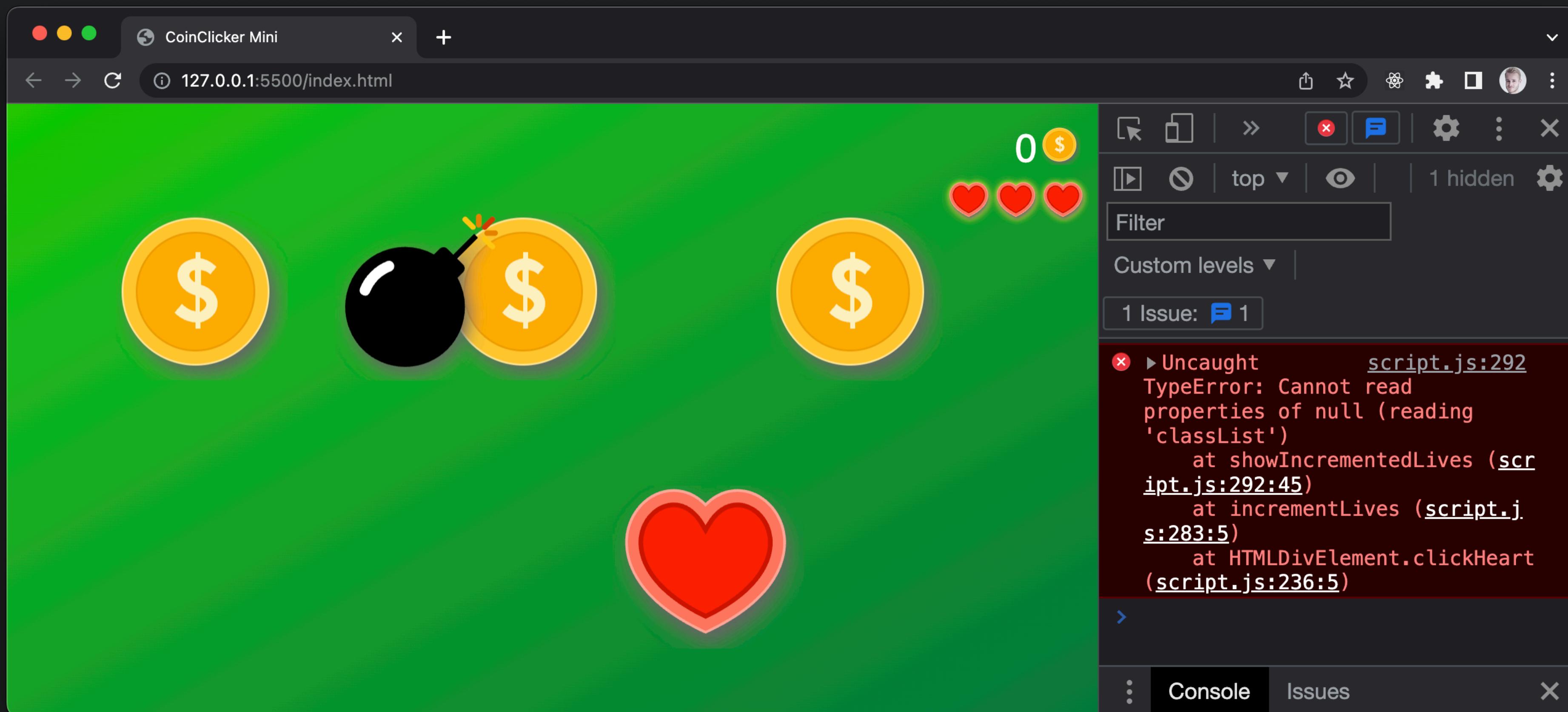
Når vi ikke har flere liv?



Når vi fx har fået 10 point?



Når vi får mere end 3 liv?



Conditional Statements

“

... are used to perform different actions based on different conditions.

”

Conditional Statement

```
if (the condition is met) {  
    Then do this...  
}
```

Conditional Statement

```
if (lives < 0) {  
    || gameOver();  
}
```

Conditional Statement

```
if (lives <= 0) {  
    | | gameOver();  
}  
| |
```

Lad os lege med det!

```
let num = 10;  
  
if (num > 5) {  
    console.log("Tallet er større end 5");  
}
```

“

I dette eksempel definerer vi en variabel `num` og checker så om det er større end 5. Hvis betingelsen er sand, vil koden inden i krølleparenteserne blive udført, i dette tilfælde at logge beskeden "Tallet er større end 5". Hvis betingelsen er falsk, vil koden inde i krølleparenteserne blive sprunget over.

ChatGBT

”

Lad os lege med det!

```
let num = 2;  
if (num > 5) {  
    console.log("Tallet er større end 5");  
} else {  
    console.log("Tallet er mindre end eller lig med 5");  
}
```

“

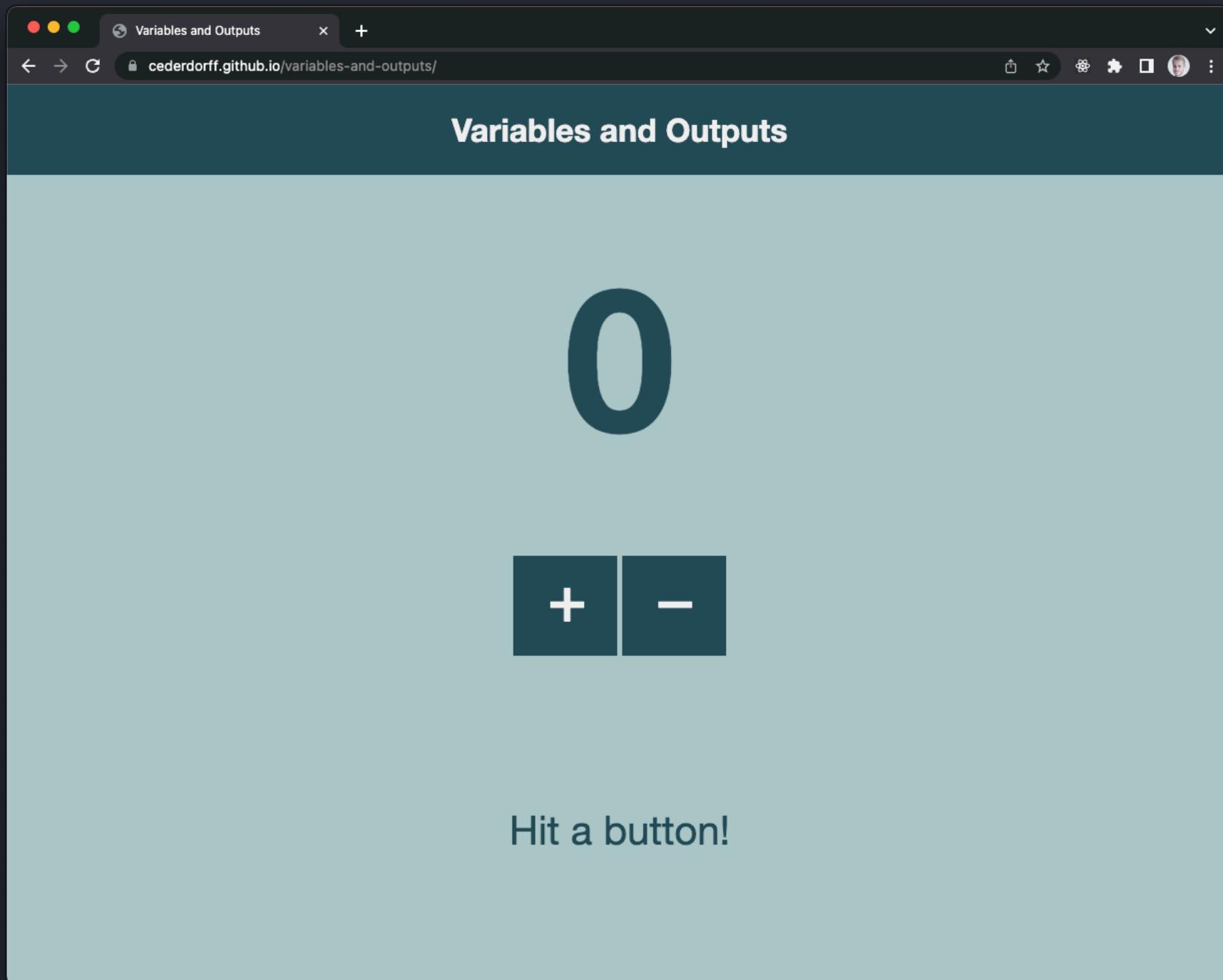
Du kan også inkludere en "else" del i din if-sætning for at definere, hvad der skal ske, hvis betingelsen ikke er opfyldt.

I dette eksempel vil koden inde i "if" blokken ikke blive udført, fordi betingelsen ikke er opfyldt. Koden i "else" blokken vil blive udført i stedet, og beskeden "Tallet er mindre end eller lig med 5" vil blive logget.

ChatGBT

”

Tæl et tal op og ned (1)



- Vis teksten “The number is above 10, når tallet er større end 10.

The use of

`== === < > >= <= !=`

Operatorer

<code>==</code>	Er lig med (værdien)	<code>x == 5</code>
<code>!=</code>	Er ikke lig med (er forskellig fra)	<code>2 != 5</code>
<code>></code>	Er større end	<code>5 > 2</code>
<code><</code>	Er mindre end	<code>2 < 5</code>
<code>>=</code>	Er større end eller lig med	<code>5 >= 2</code>
<code><=</code>	Er mindre end eller lig med	<code>2 <= 5</code>

JavaScript Comparison and Logical Operators

w3schools.com/js/js_comparisons.asp

HTML CSS JAVASCRIPT SQL PYTHON JAVA

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that `x = 5`, the table below explains the comparison operators:

Operator	Description	Comparing	Returns	Try it
<code>==</code>	equal to	<code>x == 8</code>	false	Try it »
		<code>x == 5</code>	true	Try it »
		<code>x == "5"</code>	true	Try it »
<code>==</code>	equal value and equal type	<code>x === 5</code>	true	Try it »
		<code>x === "5"</code>	false	Try it »
<code>!=</code>	not equal	<code>x != 8</code>	true	Try it »
<code>!==</code>	not equal value or not equal type	<code>x !== 5</code>	false	Try it »
		<code>x !== "5"</code>	true	Try it »
		<code>x !== 8</code>	true	Try it »
<code>></code>	greater than	<code>x > 8</code>	false	Try it »
<code><</code>	less than	<code>x < 8</code>	true	Try it »
<code>>=</code>	greater than or equal to	<code>x >= 8</code>	false	Try it »
<code><=</code>	less than or equal to	<code>x <= 8</code>	true	Try it »

JavaScript Operators

w3schools.com/js/js_operators.asp

HTML CSS JAVASCRIPT

JavaScript Comparison Operators

Operator	Description
<code>==</code>	equal to
<code>==</code>	equal value and equal type
<code>!=</code>	not equal
<code>!=</code>	not equal value or not equal type
<code>></code>	greater than
<code><</code>	less than
<code>>=</code>	greater than or equal to
<code><=</code>	less than or equal to
<code>?</code>	ternary operator

= = VS == =

== IS MORE STRICT

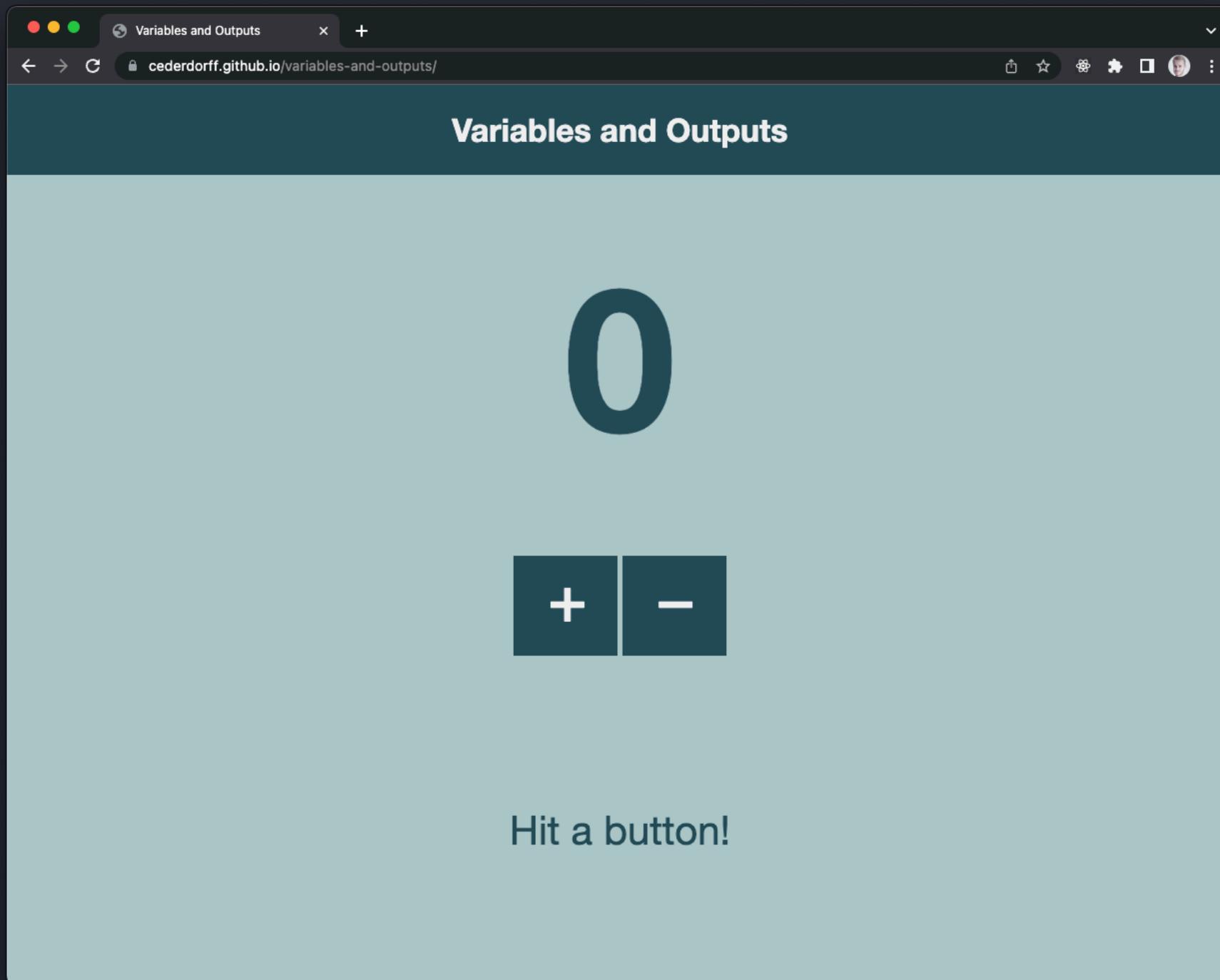
**== compares both types and values
== compares values**

```
"32" == 32 is true  
"32" === 32 is false
```

Conditional Statement

```
if (the condition is met) {  
    Then do this...  
} else {  
    Do this...  
}
```

Tæl et tal op og ned (2)



- Vis teksten “The number is above 10, når tallet er større end 10.
- Vis teksten “The number is below 10”, når tallet er under 10.

Conditional Statement

```
if (lives <= 0) {  
    gameOver();  
} else {  
    levelComplete();  
}
```

Conditional Statement

```
if (lives <= 0) {  
    gameOver();  
} else {  
    levelComplete();  
}  
  
function gameOver() {  
    console.log("Game Over");  
}  
  
function levelComplete() {  
    console.log("Level Complete");  
}
```

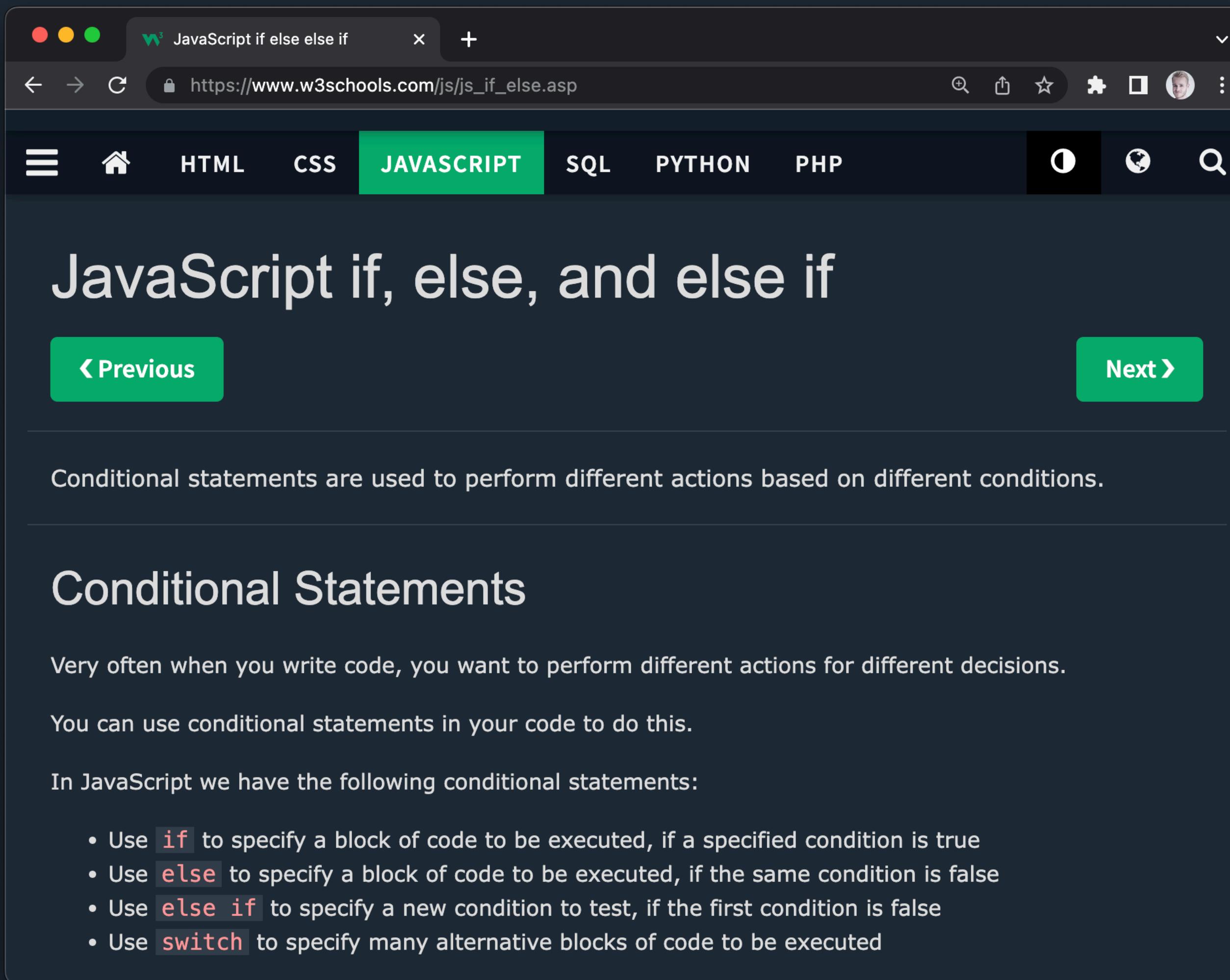
- Split your code into functions

“Separation of Concerns”

```
if (lives <= 0) {  
    | | gameOver();  
} else {  
    | | levelComplete();  
}  
  
function gameOver() {  
    | | console.log("Game Over");  
}  
  
function levelComplete() {  
    | | console.log("Level Complete");  
}
```

- Call one function inside of your if statement
- And one function inside of your else statement.
- ALWAYS!

Conditional Statements



The screenshot shows a web browser window with the title bar "JavaScript if else else if" and the URL "https://www.w3schools.com/js/js_if_else.asp". The page content is titled "JavaScript if, else, and else if". It includes navigation buttons for "Previous" and "Next", and a section about conditional statements. The text states: "Conditional statements are used to perform different actions based on different conditions." Below this, a heading "Conditional Statements" is followed by text explaining that conditional statements allow performing different actions for different decisions, and listing the four types of conditional statements in JavaScript: if, else, else if, and switch.

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

https://www.w3schools.com/js/js_if_else.asp

if to specify a block of code to be executed, if a specified condition is true

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

else to specify a block of code to be executed, if the same condition is false

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

else if to specify a new condition to test, if the first condition is false

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false  
    // and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false  
    // and condition2 is false  
}
```

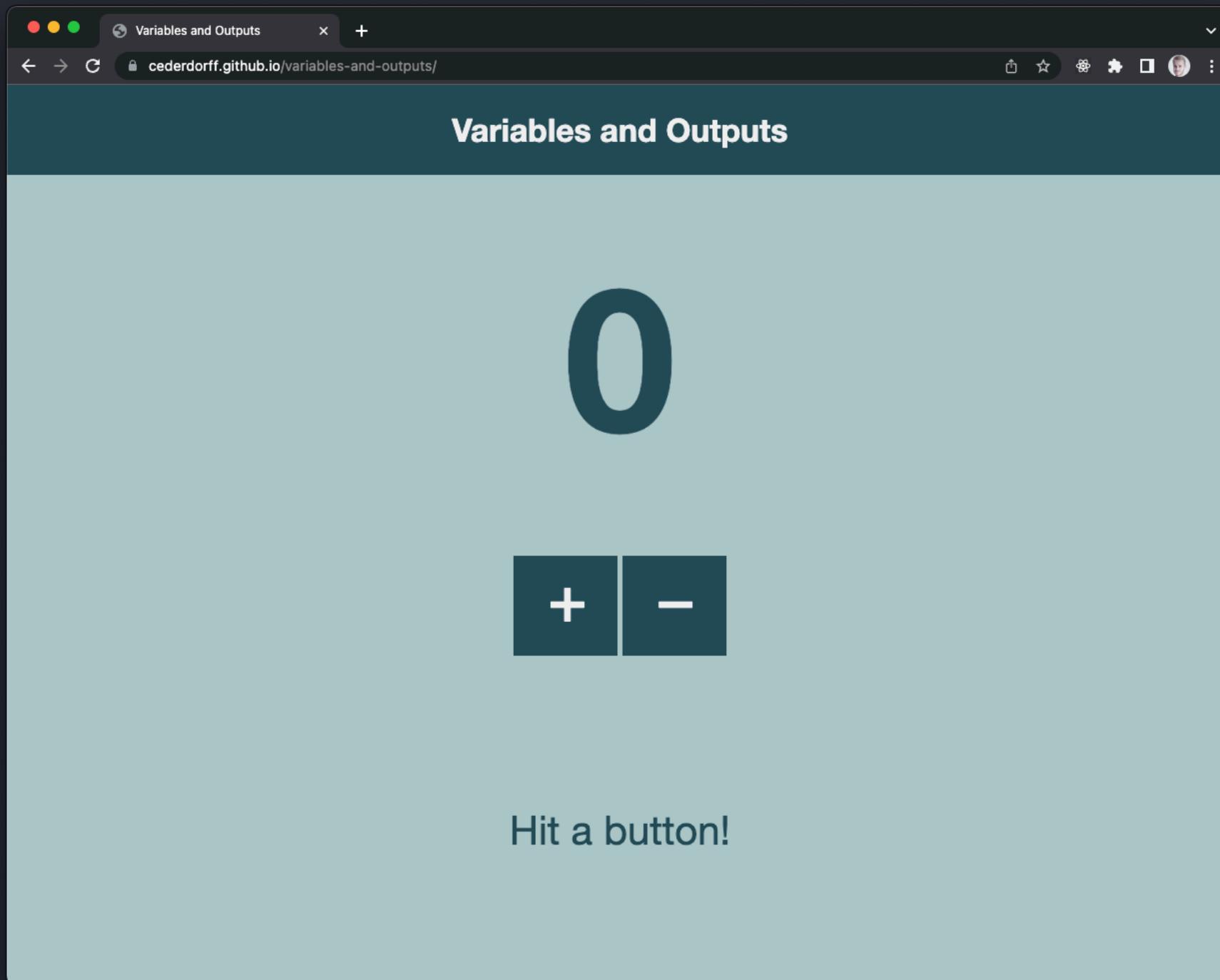
else if

```
if (the condition is met) {  
    Then do this...  
} else if (another condition) {  
    Do this...  
} else {  
    Do this...  
}
```

else if

```
if (lives <= 0) {  
    | | gameOver();  
} else if (points >= 10) {  
    | | levelComplete();  
} else {  
    | | gameOver();  
}
```

Tæl et tal op og ned (3)



- Vis teksten “The number is above 10”, når tallet er større end 10.
- Vis teksten “The number is below 10”, når tallet er under 10.
- Vis teksten “The number is 10”, når tallet er lig med 10.

Child, adult or senior? (1)

1. Lav et program, der kan afgøre om en person er voksen eller barn.
2. Anvend en variable til at holde alder.
3. Brug dernæst if-else til at afgøre om personen er voksen eller barn:
 1. Du skal kalde en funktion der hedder `isAdult`, hvis betingelsen mødes, ellers skal du kalde en funktion, der hedder `isChild`. Funktionerne skal anvende `console.log()` til at teste. Fx `console.log("Du er voksen")`.
4. Tilføj logik, der gør at funktionerne kan vise en tekst i DOM'en, så du ikke blot får `console.log`'s. Du skal bruge `.textContent`.

```
if (the condition is met) {  
    Then do this...  
} else {  
    Do this...  
}
```

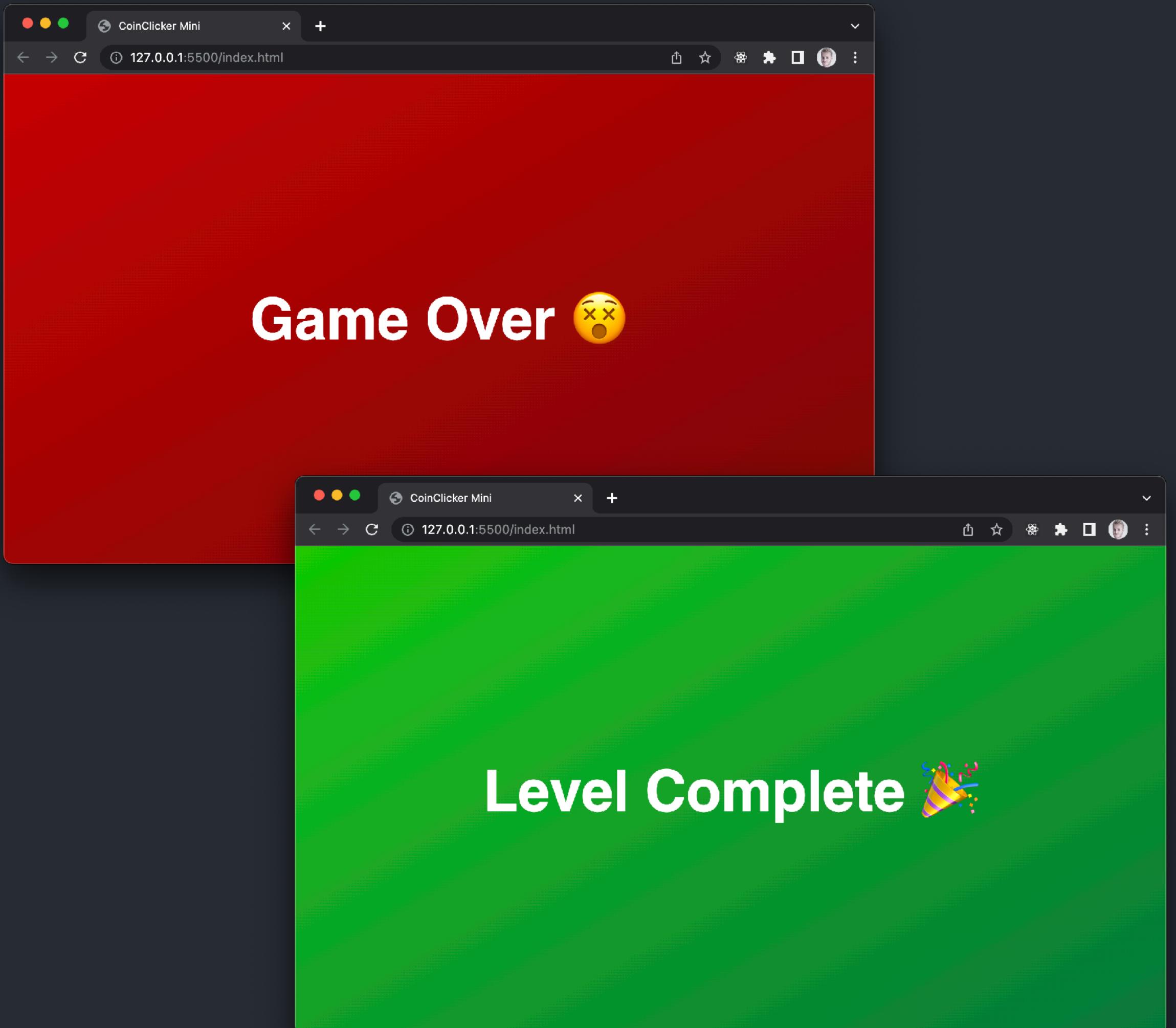
Child, adult or senior? (2)

5. Udvid dit program, så du kan afgøre om en person er barn, voksen eller pensionist.
6. Anvend fortsat en variable til at holde alder.
7. Brug dernæst if-elseif-else til at afgøre om personen er barn, voksen eller pensionist. Pensionistalderen kan sættes til 72 år.
8. Du skal fortsat anvende funktioner der console.log'er.
9. Tilføj logik, der gør at funktionerne kan vise en tekst i DOM'en, så du ikke blot får console.log's. Du skal bruge .textContent.

```
if (the condition is met) {  
    Then do this...  
} else if (another condition) {  
    Do this...  
} else {  
    Do this...  
}
```

Vis “Game Over” eller “Level Complete”

1. Download: [Coinclicker-mini-2023 \(lives-and-points\)](#)
2. I index.html, tilføj indhold til #game_over og #level_complete - test ved at fjerne klassen hidden.
3. I script.js, lav to funktioner, gameOver() og levelComplete(), der console.log'er hhv “Game Over” og “Level Complete”.
4. I decrementedLives()
Hvis lives er mindre end eller lig med 0, kald gameOver(). Ellers showDecrementedLives()
5. I incrementPoints()
Hvis points er større end eller lig med fx 10, kald levelComplete()
6. Husk fortsat at separere logik i funktioner (hvis du har fulgt ovenstående, har du gjort det).
7. Funktionerne gameOver() og levelComplete() skal nu fjerne CSS-klassen hidden på hhv #game_over og #level_complete, så vi får vist “Game Over”- eller “Level Complete”-skærmen.
8. Sørg for at du ikke får en fejl, når du klikker på et hjerte og allerede har 3 liv.
Hint: det er også noget med if!
9. Ekstra: Når spillet er slut, skal du kalde en funktion, der fjerne alle animationer og events (dvs det modsatte af hvad start() gør).



Clicker-spil - Game Over og Level Complete

- Brug de beslutninger, du har fastlagt i dit aktivitetsdiagram.
- Game Over eller Level Complete skærmen skal vises – og de skal som minimum fortælle bruger om deres spil var en succes eller ej.
- Spillet skal stoppe, når det er slut – animationerne skal ophøre, og det må ikke længere være muligt at klikke på noget. Lav fx en funktion, der fjerner alle animationer og alle events (det modsatte af hvad start() gør).
- Spillet skal kun kunne spilles én gang – der behøver ikke være nogen knap til at prøve igen (kommer senere).
- Der er ingen timer i spillet endnu (kommer senere).
- Tip: Lav så mange funktioner (små) som muligt, der tager sig af en afgrænset logik!

Clicker-spil – Game Over og Level Complete

Bare rolig, det er ikke game over endnu, men i dit spil har du fået tilføjet mulighed for at man kan miste liv – og det crasher sikkert når man mister sit sidste liv. Det skal fikses i denne udgave, for nu skal spillet kunne slutte og vise enten en Game Over eller en Level Complete skærm.

Det kan kun slutte hvis man mister alle sine liv, eller hvis man opnår et tilstrækkeligt antal point (hvis du altså har defineret dit aktivitetsdiagram sådan at det kan slutte før tid). Der er ikke sat nogen timer på endnu, så spillet bliver bare ved indtil en af de to andre ting sker.

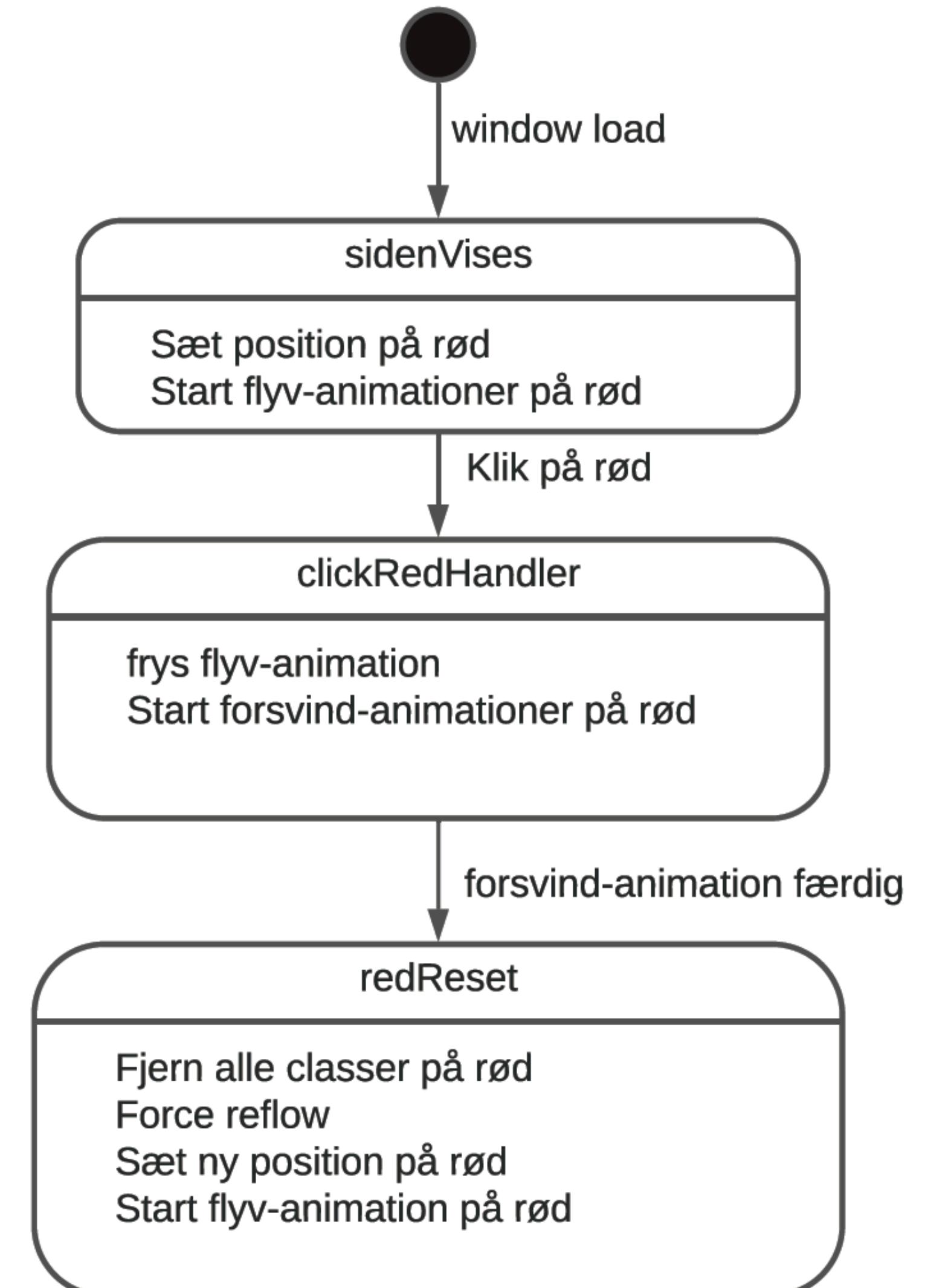
Krav

Du skal bruge de beslutninger du har fastlagt i dit aktivitetsdiagram – eventuelt det reviderede aktivitetsdiagram. Det vil sige at hvis du har fastlagt at spillet skal slutte når man har opnået et vist antal point, så skal det også slutte når man har opnået det antal point!

Spillet skal stoppe når det er slut – animationerne skal ophøre, og det må ikke længere være muligt at klikke på noget.

Game Over eller Level Complete skærmen skal vises – og de skal som minimum fortælle bruger om deres spil var en succes eller ej.

Genstart Animationer og force reflow



Force reflow

For at genstarte en keyframe-animation, skal man tage animationsklassen af, og sætte den på igen:

```
document.querySelector("selector").classList.remove("class");
document.querySelector("selector").classList.add("class");
```

JavaScript er dog så hurtig til at udskifte animationen at browseren ikke opfatter det. Man blive derfor nødt til at tvinge den til at “se” at noget er ændret.

Det gør man ved at bede den om at finde ud af hvor et element befinner sig. Det kaldes at gennemtvinge (force) et reflow:

```
document.querySelector("selector").classList.remove("class");
document.querySelector("selector").offsetLeft;
document.querySelector("selector").classList.add("class");
```

Force reflow

```
function bombGone() {  
    // fjern event der bringer os herind  
    document.querySelector("#bomb_container").removeEventListener("animationend", bombGone);  
  
    // fjern forsvind-animation  
    document.querySelector("#bomb_sprite").classList.remove("zoom_in");  
  
    // fjern pause  
    document.querySelector("#bomb_container").classList.remove("paused");  
  
    // genstart falling animation  
    document.querySelector("#bomb_container").classList.remove("falling");  
    document.querySelector("#bomb_container").offsetWidth;  
    document.querySelector("#bomb_container").classList.add("falling");  
  
    // gør det muligt at klikke på bomb igen  
    document.querySelector("#bomb_container").addEventListener("click", clickBomb);  
}
```

Genstart animationer

Basic Animation

Restarting
animations

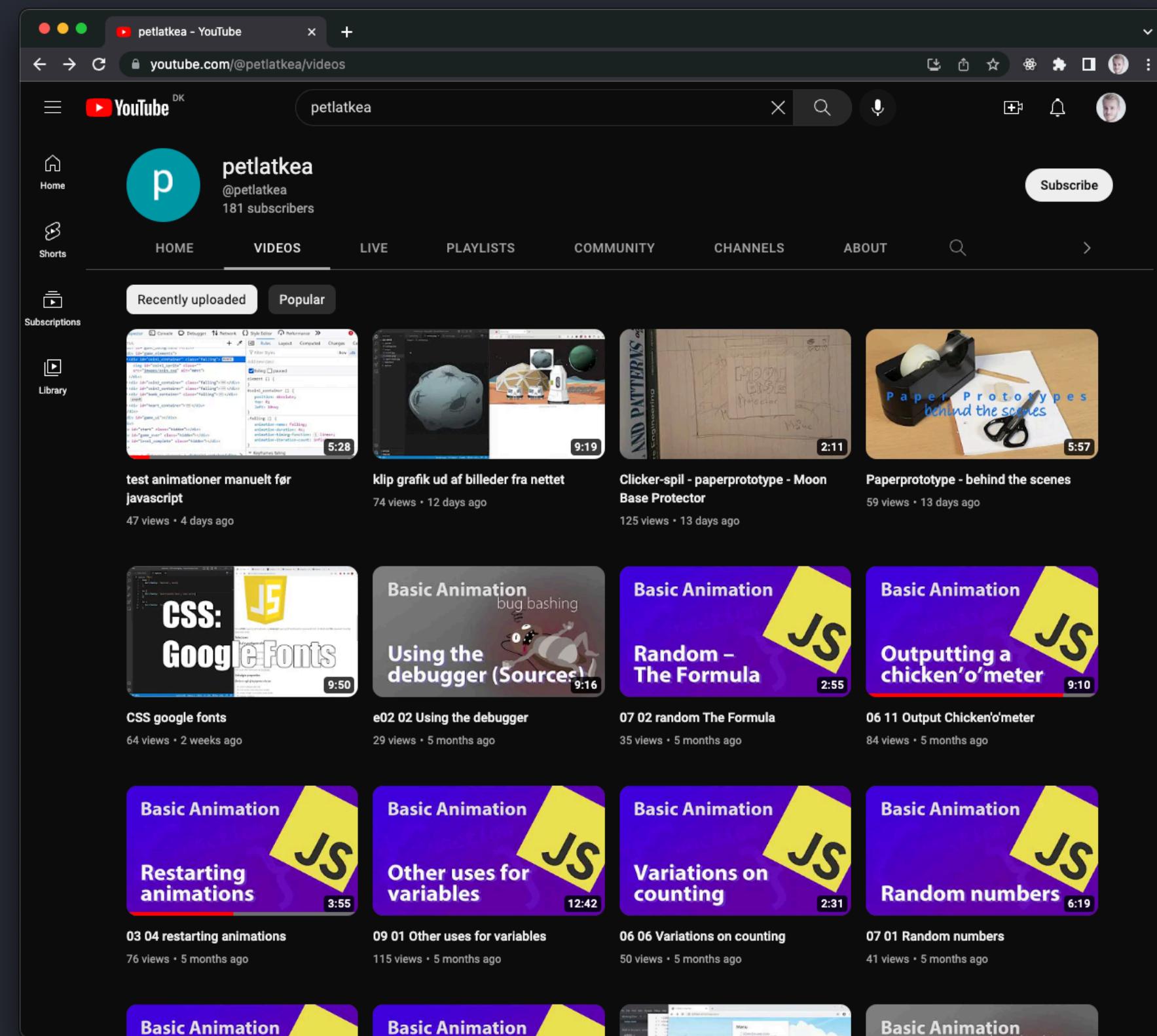


JS

Inspiration til brug af if, if-else og else-if



Inspiration, inspiration, inspiration



... eller stjæl med arm(e) og ben 🤷



Code
Every
Day

... and have fun!