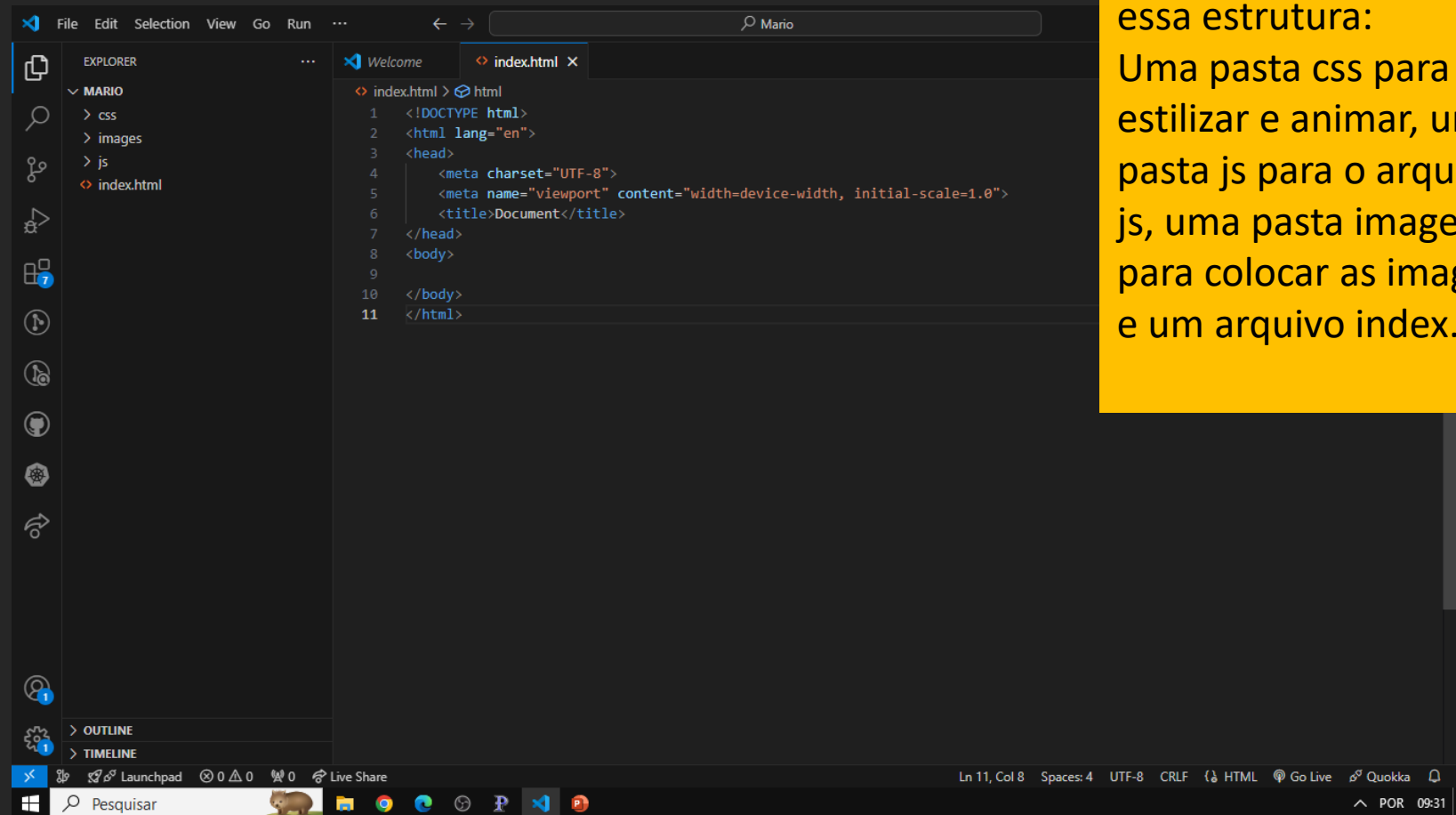


Jogo do mario

JavaScript e CSS



Vamos começar criando essa estrutura:

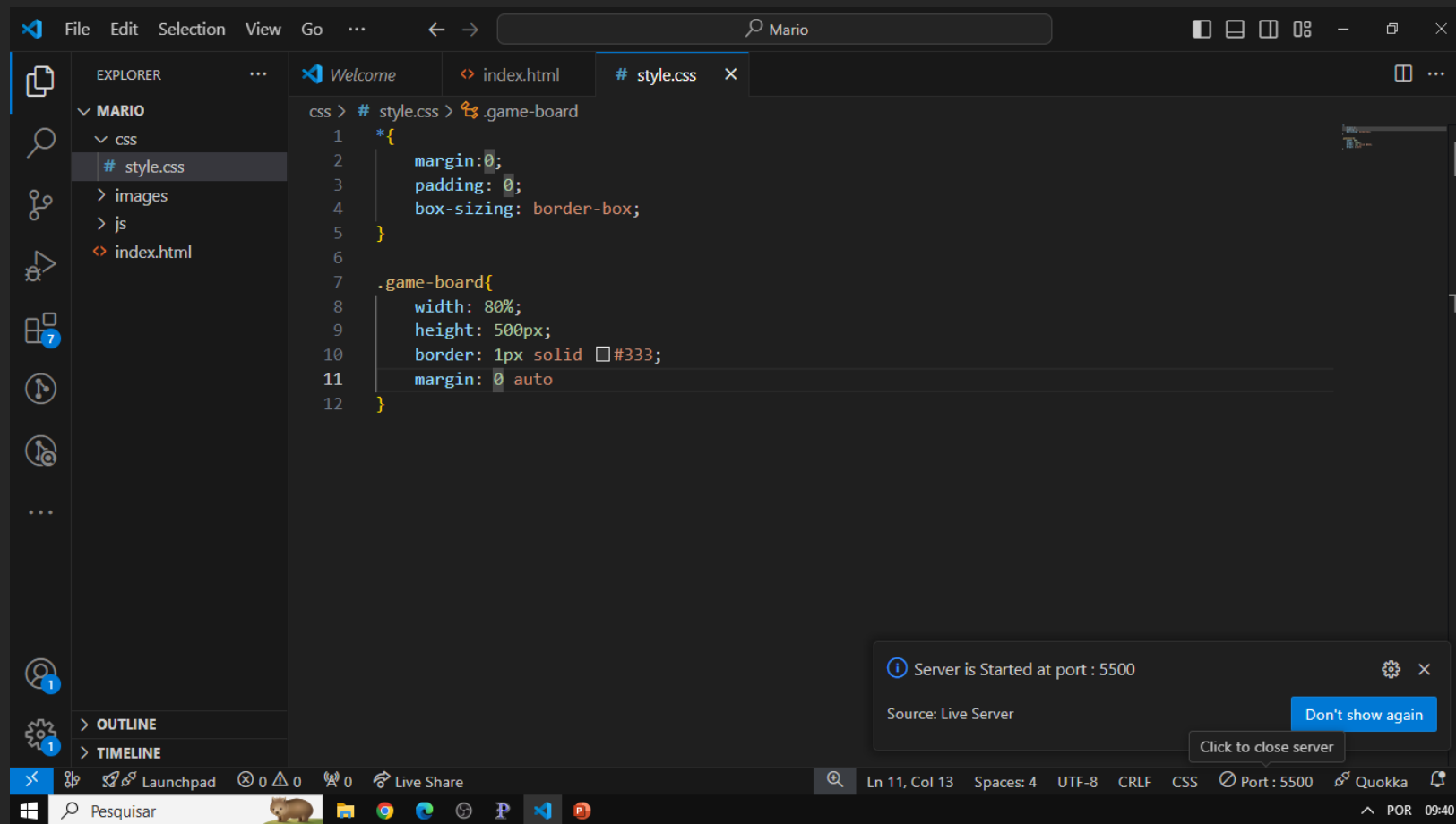
Uma pasta css para estilizar e animar, uma pasta js para o arquivo js, uma pasta images para colocar as imagens e um arquivo index.html

```
<!-->
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href="js/script.js">
7      <link rel="stylesheet" href="css/style.css">
8      <title>Document</title>
9  </head>
10 <body>
11
12 </body>
13 </html>
```

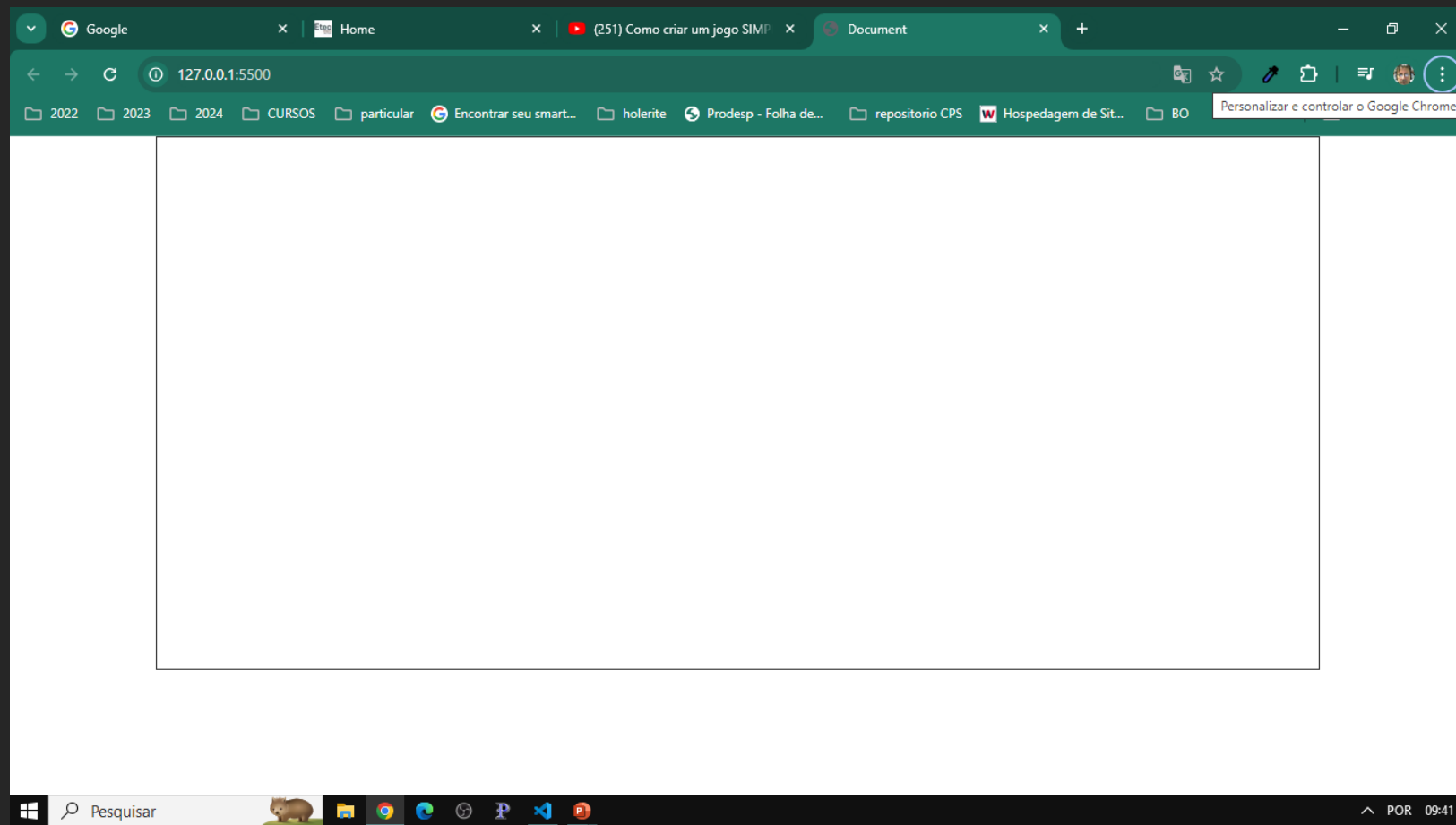
Vamos criar os links para o js e o css

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="js/script.js">
7   <link rel="stylesheet" href="css/style.css">
8   <title>Document</title>
9 </head>
10 <body>
11
12 </body>
13   <div class="game-board">
14
15   </div>
16 </html>
```

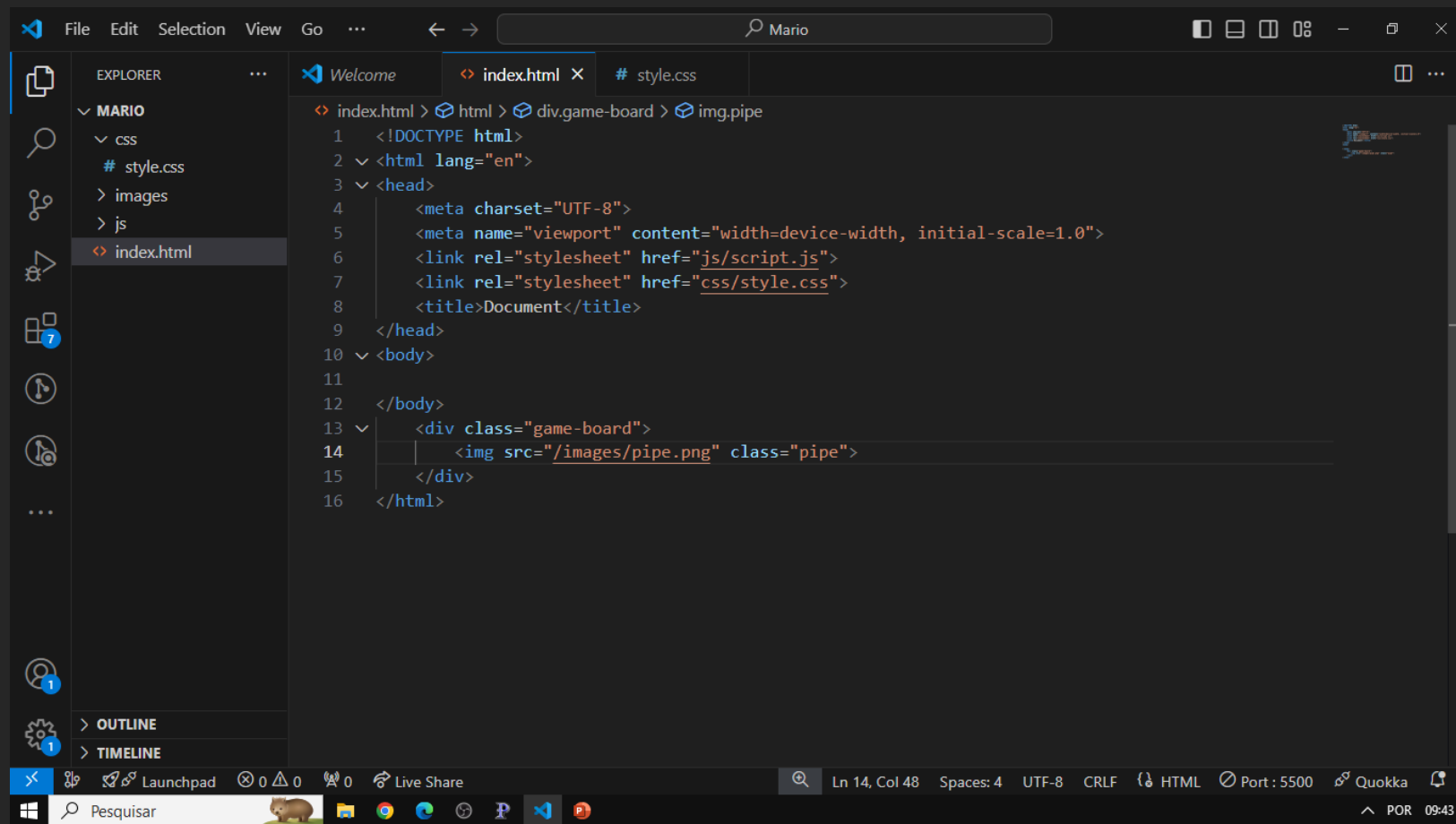
Criamos uma div game-board que é onde o jogo vai rodar.



No arquivo css, vamos zerar todas as margens e criar a nossa classe game-board com altura de 500, largura de 80%, e centralizar. Vamos colocar uma borda pra ver a classe.

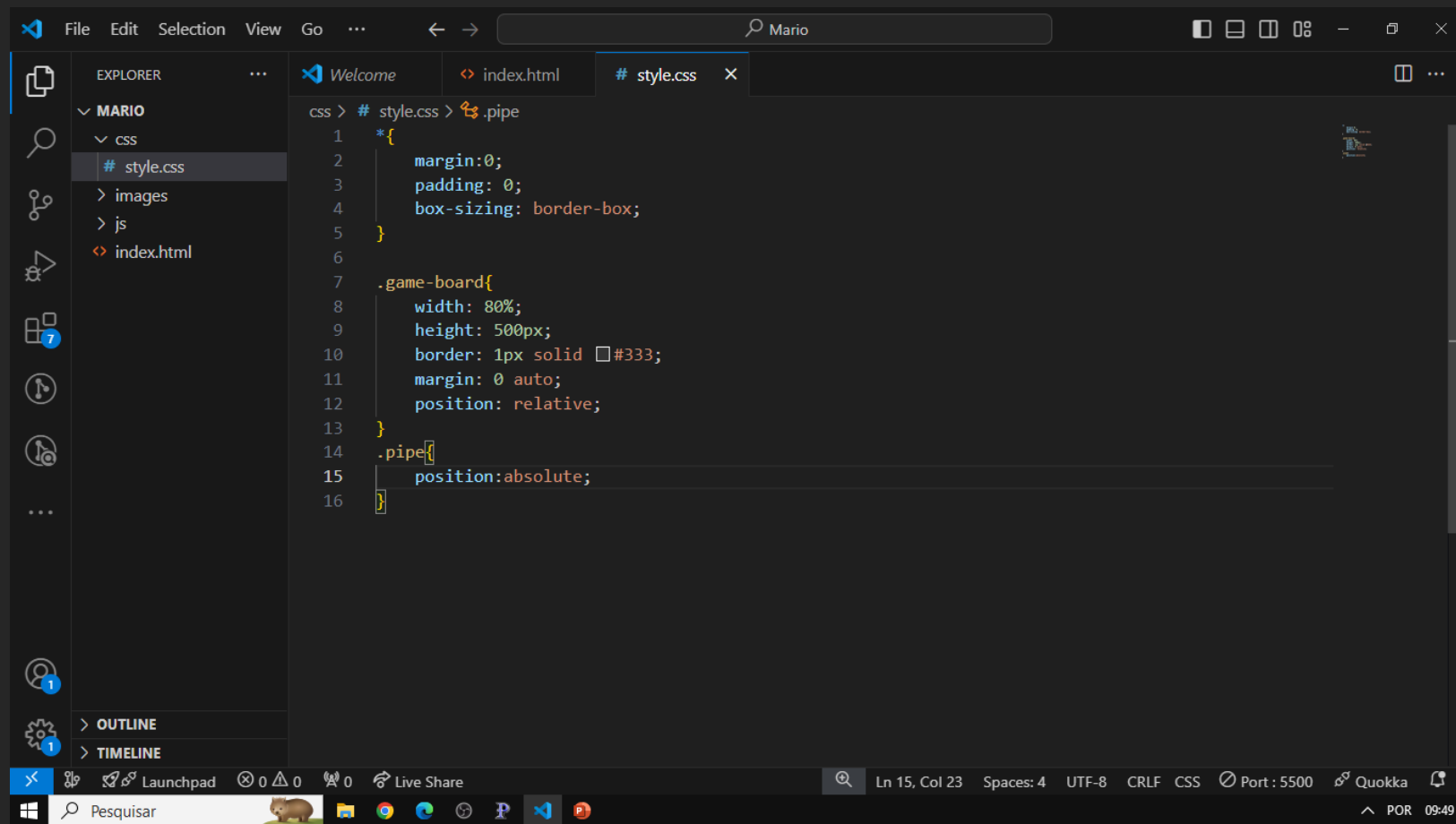


Aqui nossa game-board



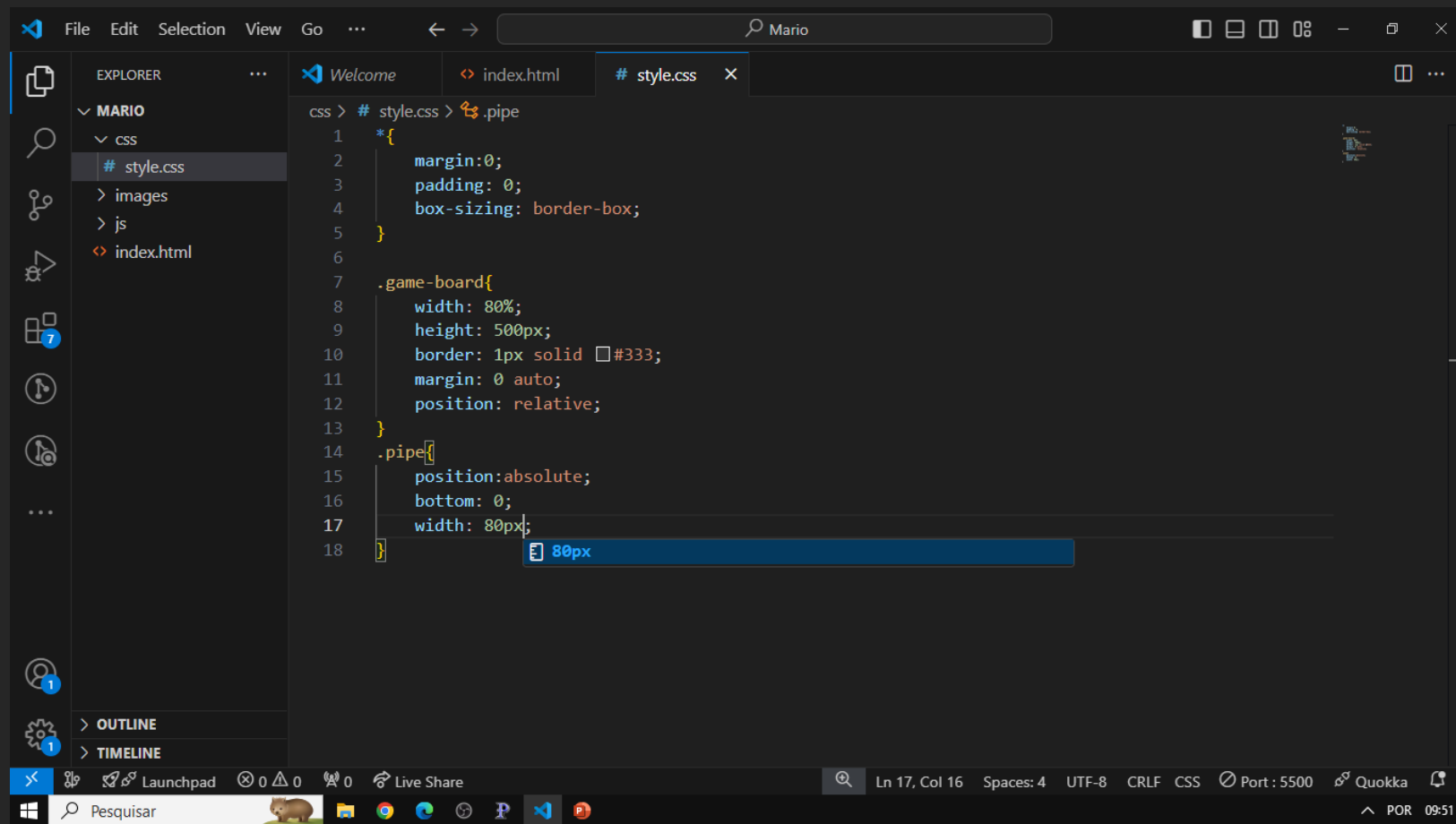
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="js/script.js">
7   <link rel="stylesheet" href="css/style.css">
8   <title>Document</title>
9 </head>
10 <body>
11
12 </body>
13 <div class="game-board">
14   
15 </div>
16 </html>
```

Vamos no index e acrescentamos o tubo, e colocamos uma classe para ela pra poder trabalhar com ele.

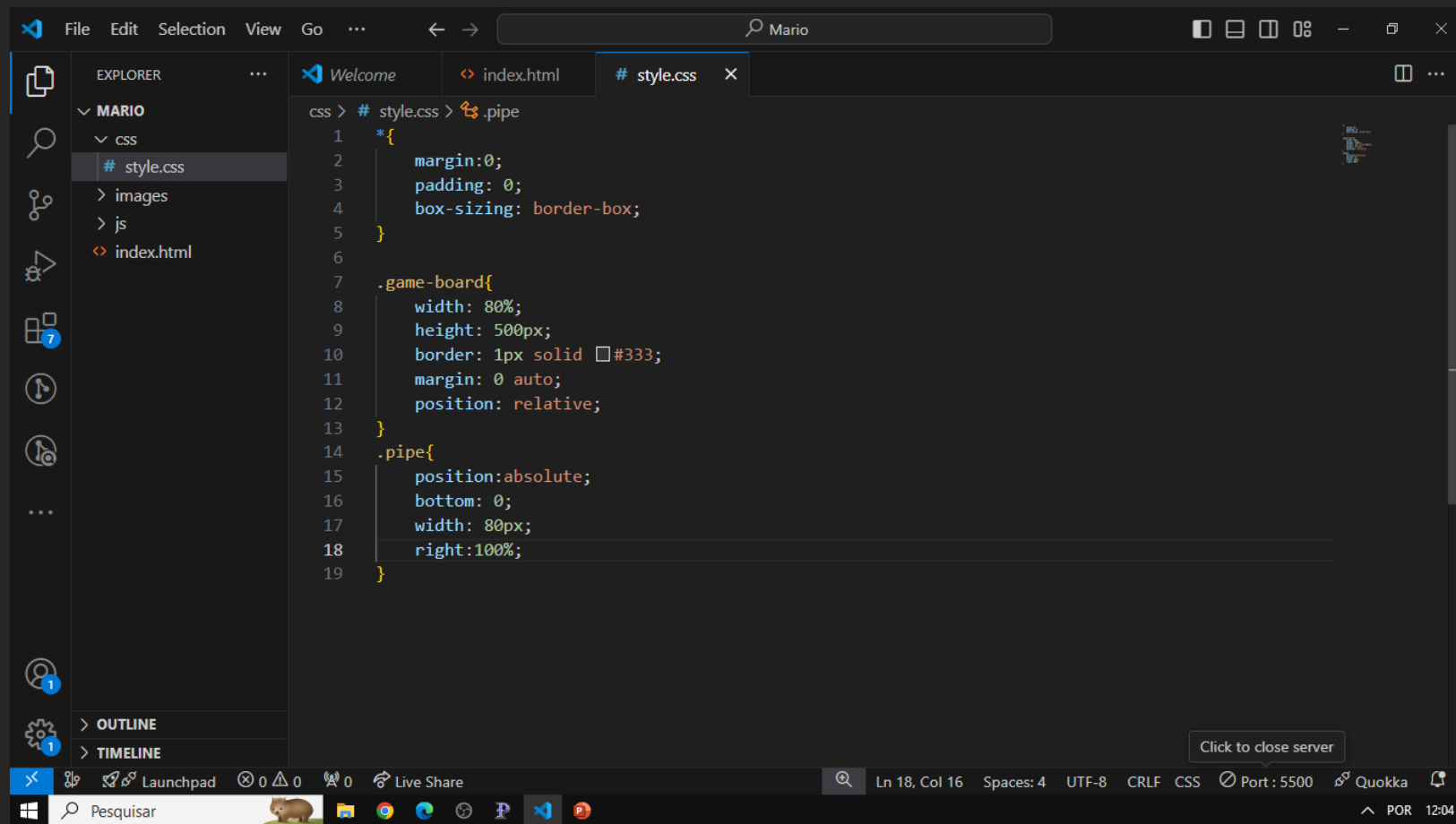


Primeiro de tudo vamos colocar o tubo em baixo, e como vamos trabalhar com varias imagens, vamos colocar um position Absolute para que nenhuma interfira no comportamento uma da outra.

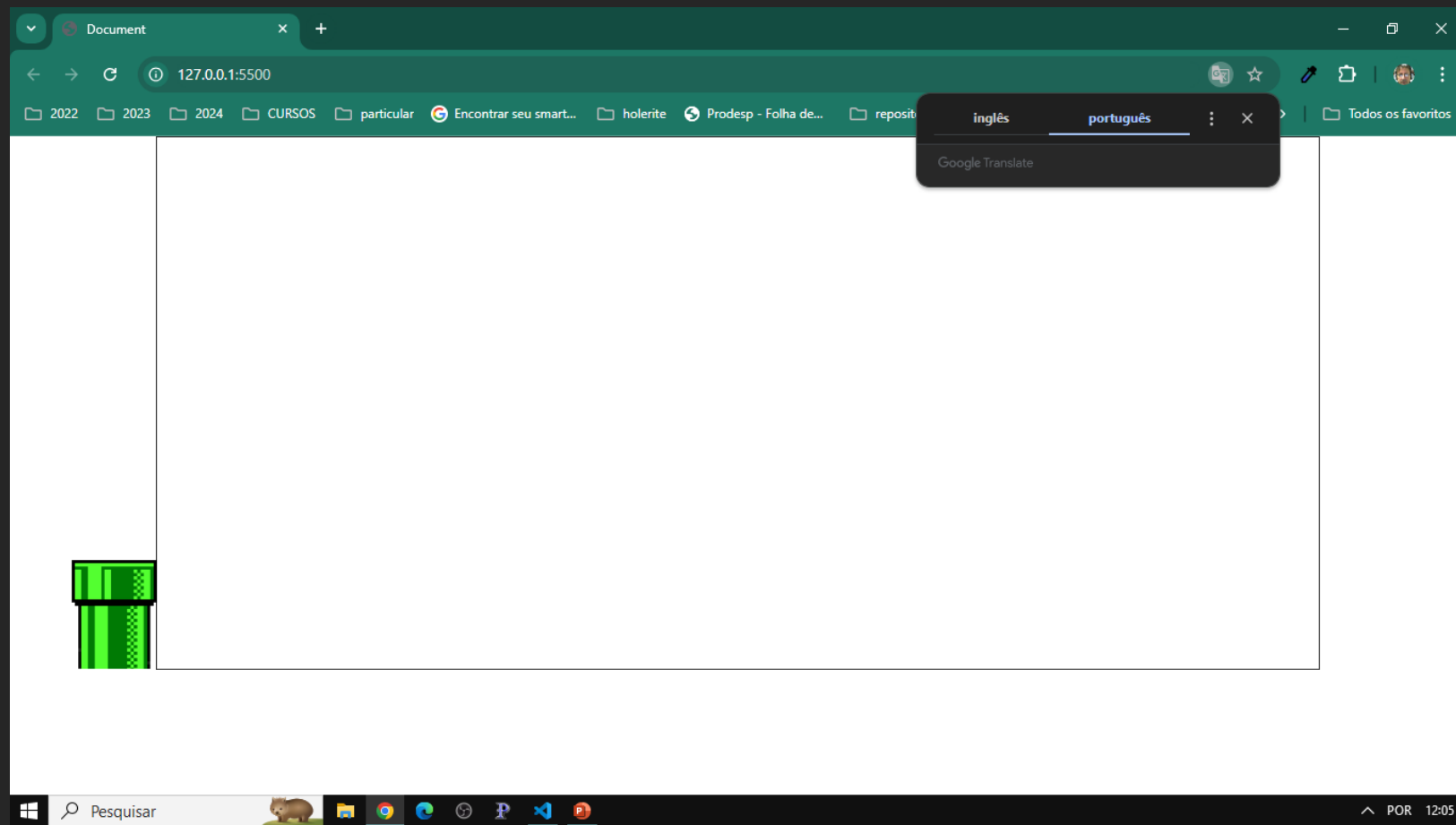
Como vou usar um absolute nas imagens, eu quero que elas respeitem a delimitação do quadro então colocamos um position-relative no quadro



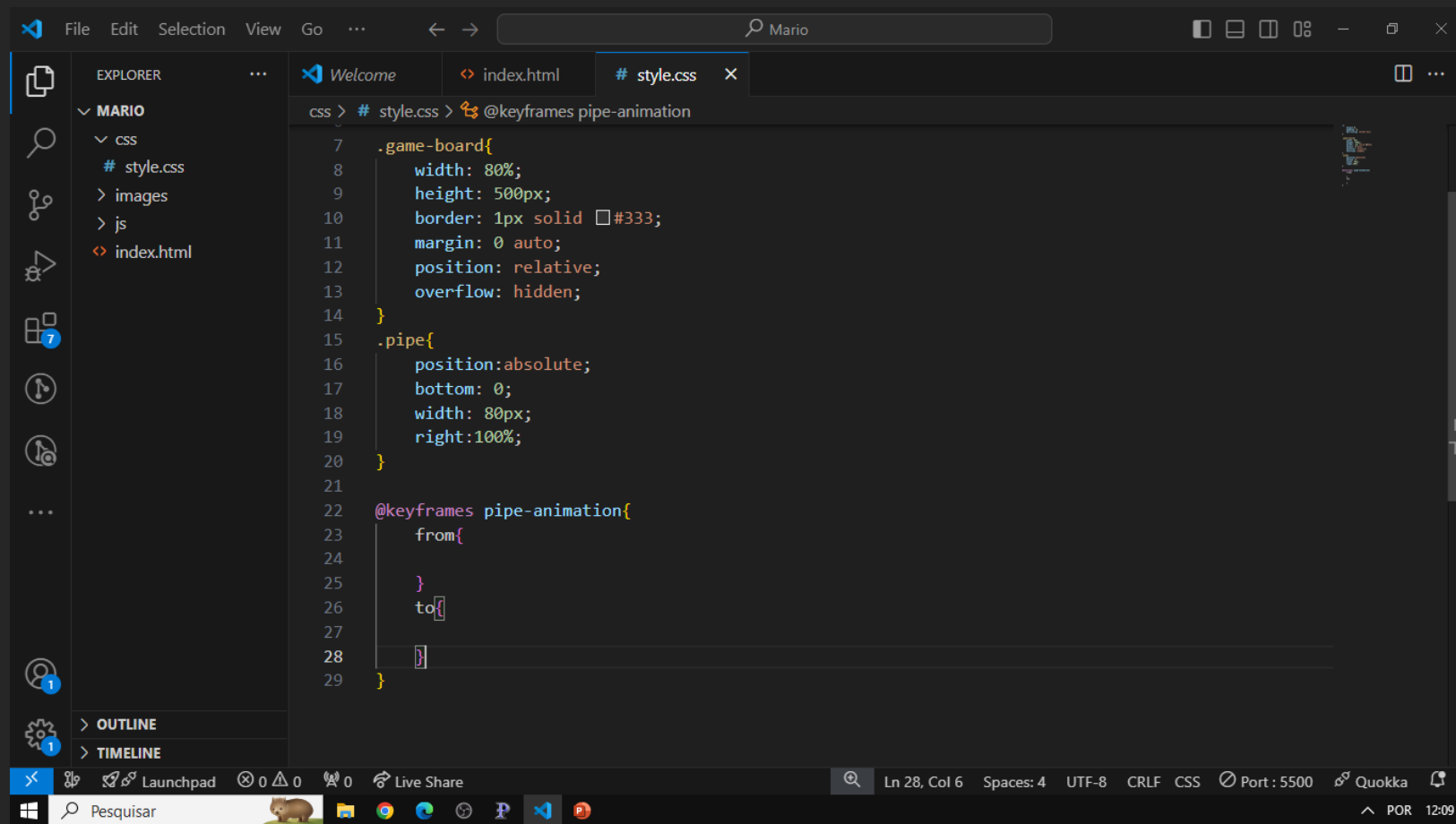
Ajustamos para ficar em baixo e o tamanho dele.



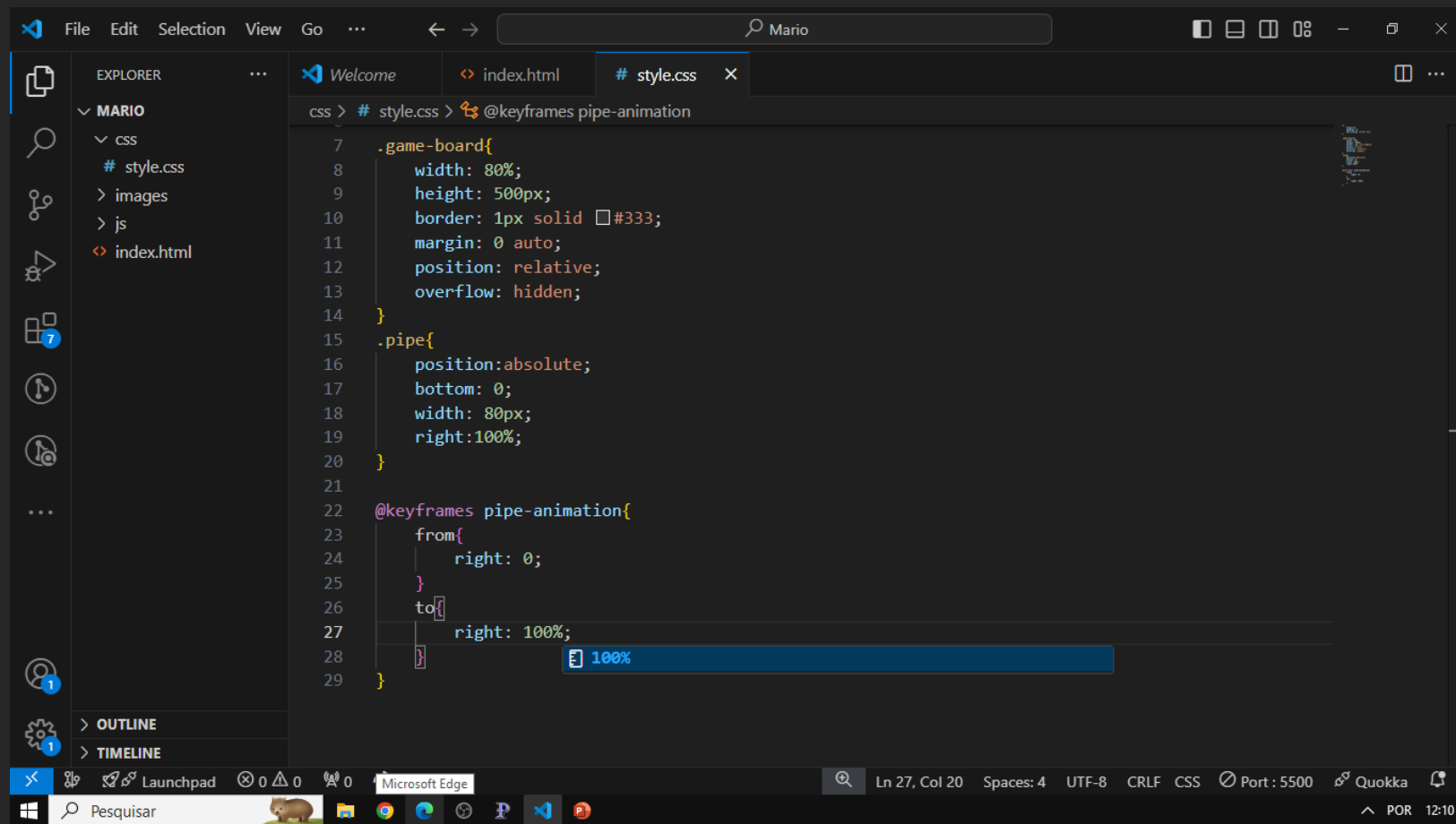
Agora vamos fazer a animação do tubo, para que ele entre pela direita e saia pela esquerda. Vamos mexer na propriedade right. Quando eu coloco 0 ele fica na direita, e com 100% ele vai sair do quadro a esquerda. Então vamos começar a animação com right 0 ate o final da tela a esquerda



Quando ele sair da tela,
a gente não quer que
ele apareça então
vamos esconder tudo
que estiver fora ta tela
com overflow:hidden



Para a animação ,
usamos keyframes. O
keyframes ter um
começo e um fim
Temos também que
colocar um nome para
ele

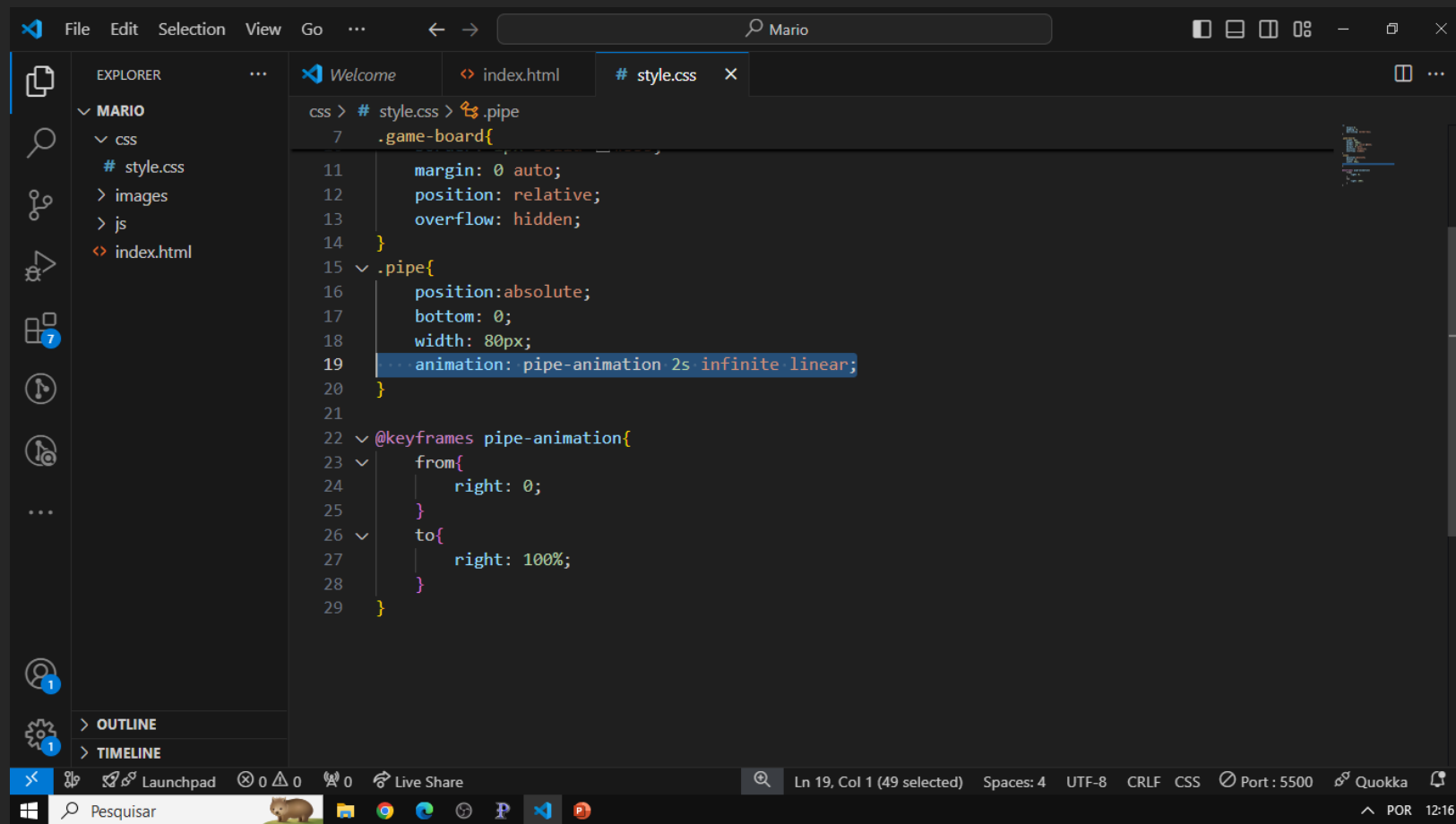


Entao ele começa com zero e termina com 100%

Agora a gente so precisa chamar essa animação. Para isso a gente vai la no tubo, na classe pipe e colocamos a animacao

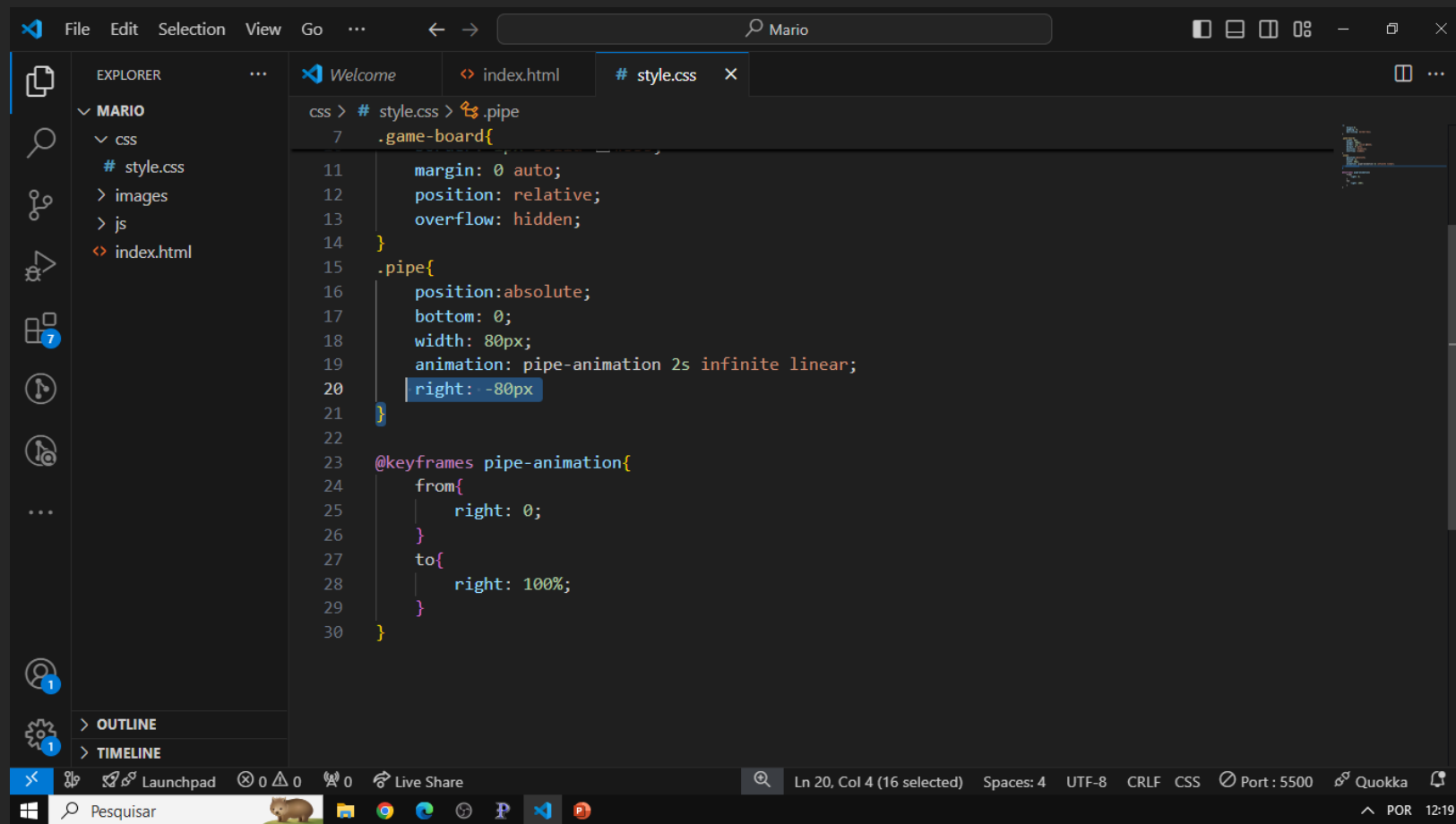
```
7  .game-board{
8      width: 80%;
9      height: 500px;
10     border: 1px solid #333;
11     margin: 0 auto;
12     position: relative;
13     overflow: hidden;
14 }
15 .pipe{
16     position: absolute;
17     bottom: 0;
18     width: 80px;
19     right: 100%;
20     animation: pipe_animation 2s infinite;
21 }
22
23 @keyframes pipe-animation{
24     from{
25         right: 0;
26     }
27     to{
28         right: 100%;
29     }
30 }
```

A animação precisa de um tempo e um modo, o tempo colocamos 2segundos e no modo infinite ele ai ficar em loop
Tirar o right:100% do pipe



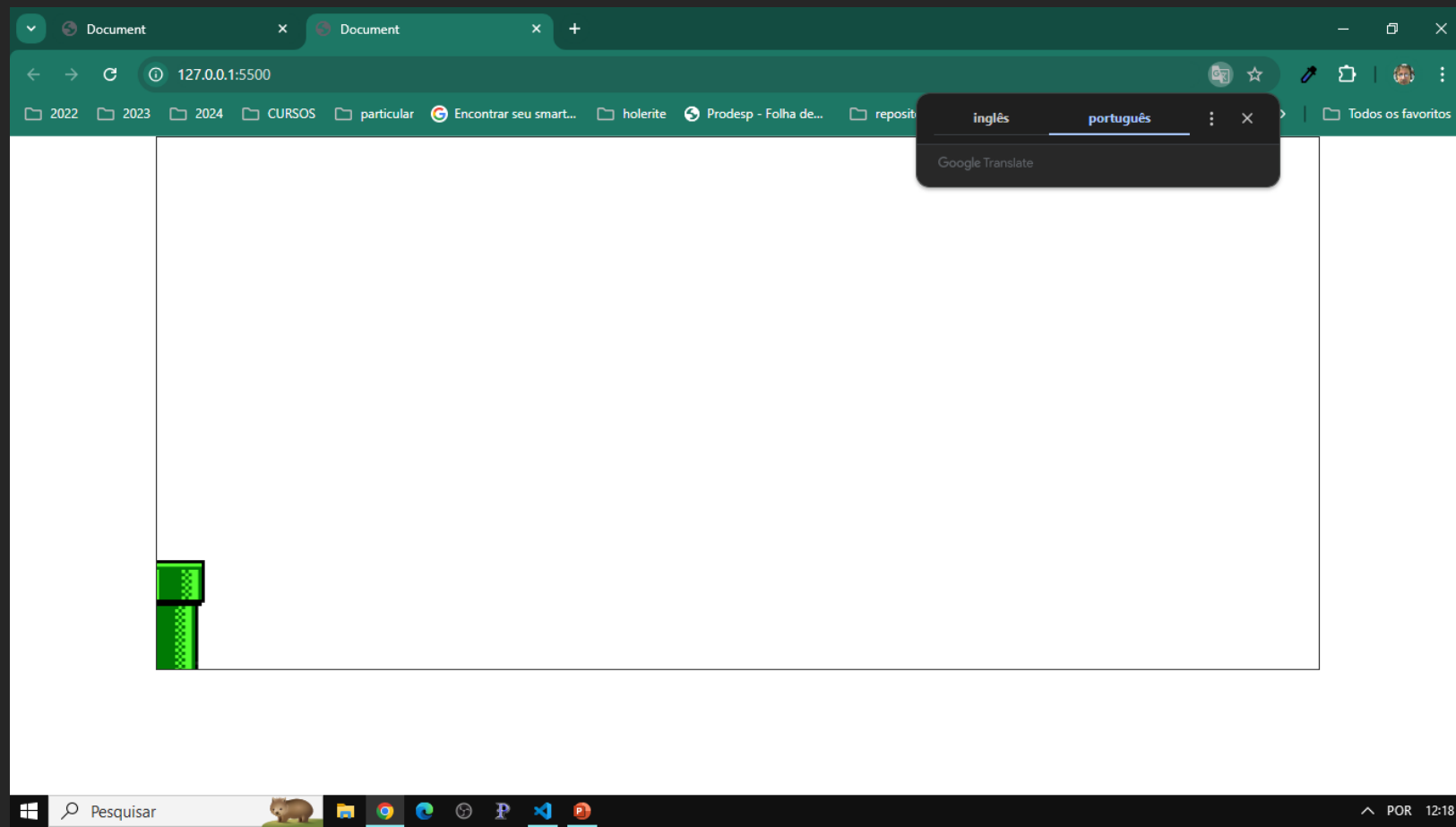
```
css > # style.css > .pipe
7  .game-board{
11     margin: 0 auto;
12     position: relative;
13     overflow: hidden;
14 }
15  .pipe{
16     position: absolute;
17     bottom: 0;
18     width: 80px;
19     animation: pipe-animation 2s infinite linear;
20 }
21
22  @keyframes pipe-animation{
23  from{
24     right: 0;
25 }
26 to{
27     right: 100%;
28 }
29 }
```

A animação começa devagar e vai aumentando a velocidade depois diminui. Pra ficar constante colocamos a propriedade linear

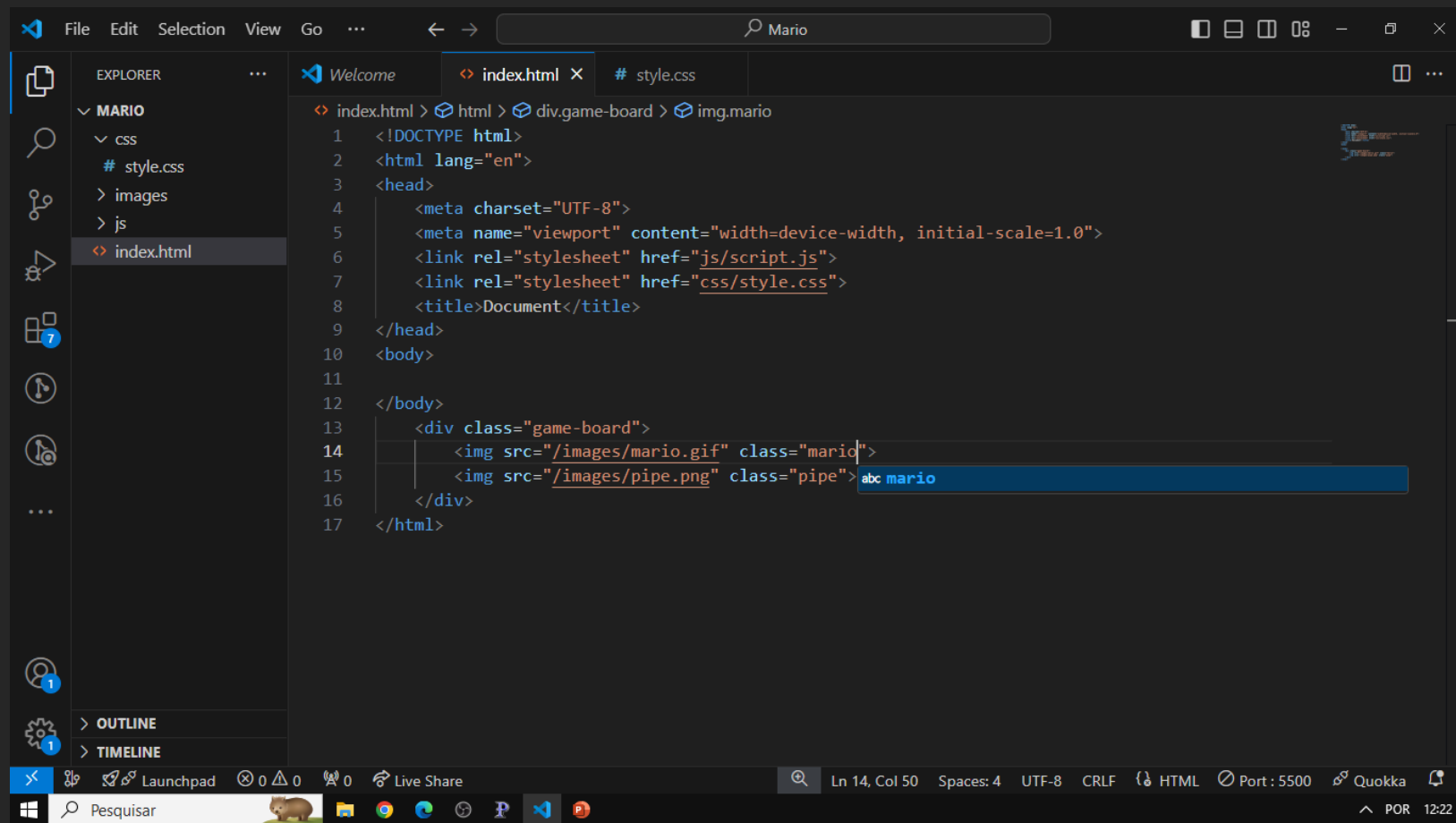


```
css > # style.css > .pipe
7  .game-board{
11     margin: 0 auto;
12     position: relative;
13     overflow: hidden;
14 }
15 .pipe{
16     position: absolute;
17     bottom: 0;
18     width: 80px;
19     animation: pipe-animation 2s infinite linear;
20     right: -80px;
21 }
22
23 @keyframes pipe-animation{
24     from{
25         right: 0;
26     }
27     to{
28         right: 100%;
29     }
30 }
```

Testando a gente ve que na direita o tubo aparece do nada, mas queremos que ele apareça entrando na tela. Entao temos que fazer ele começar antes da tela, colocando um right negativo com a largura do tubo. Agora ele começa fora ta tela, mesmo escondido e entra na tela



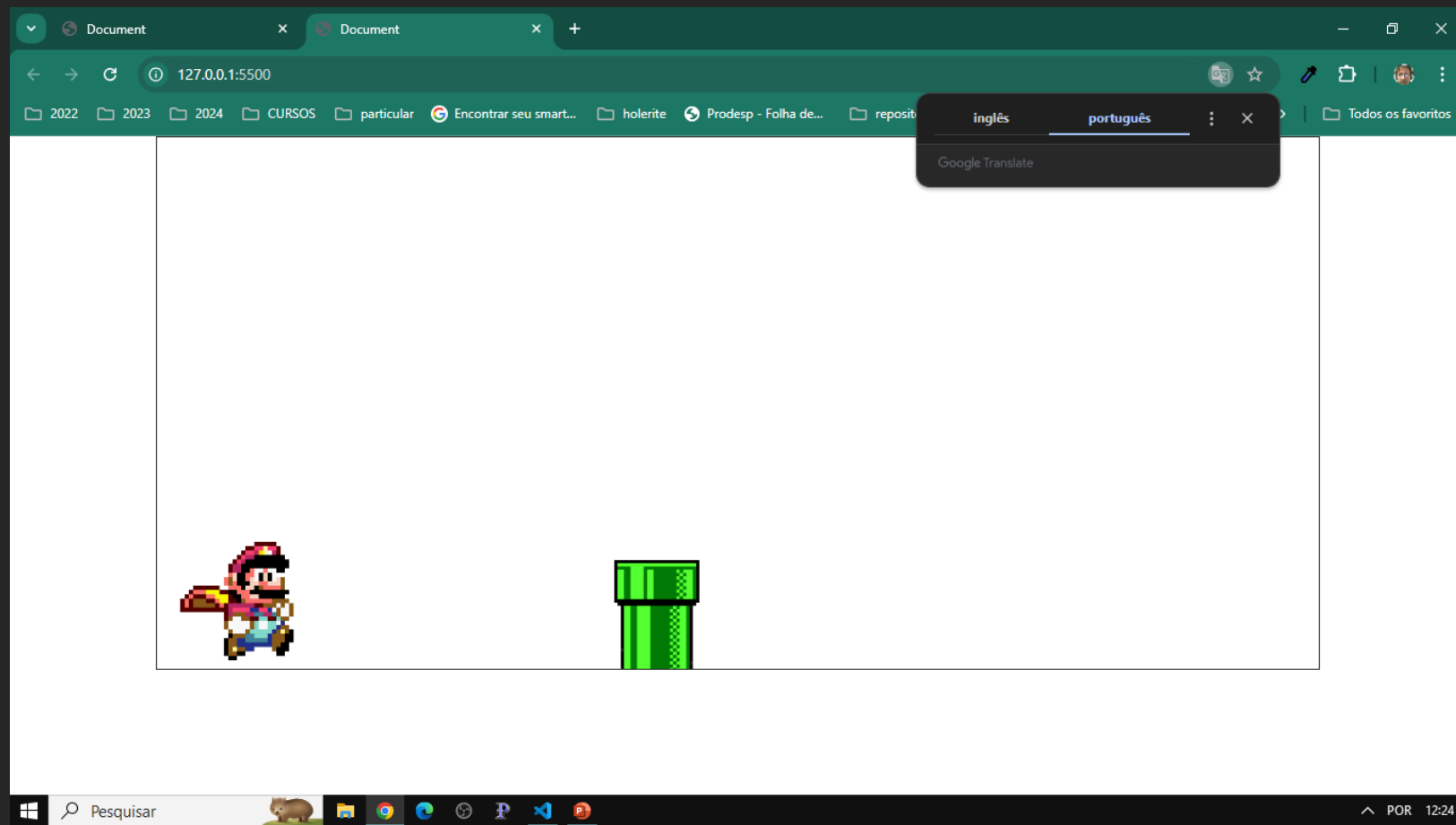
Agora testando...



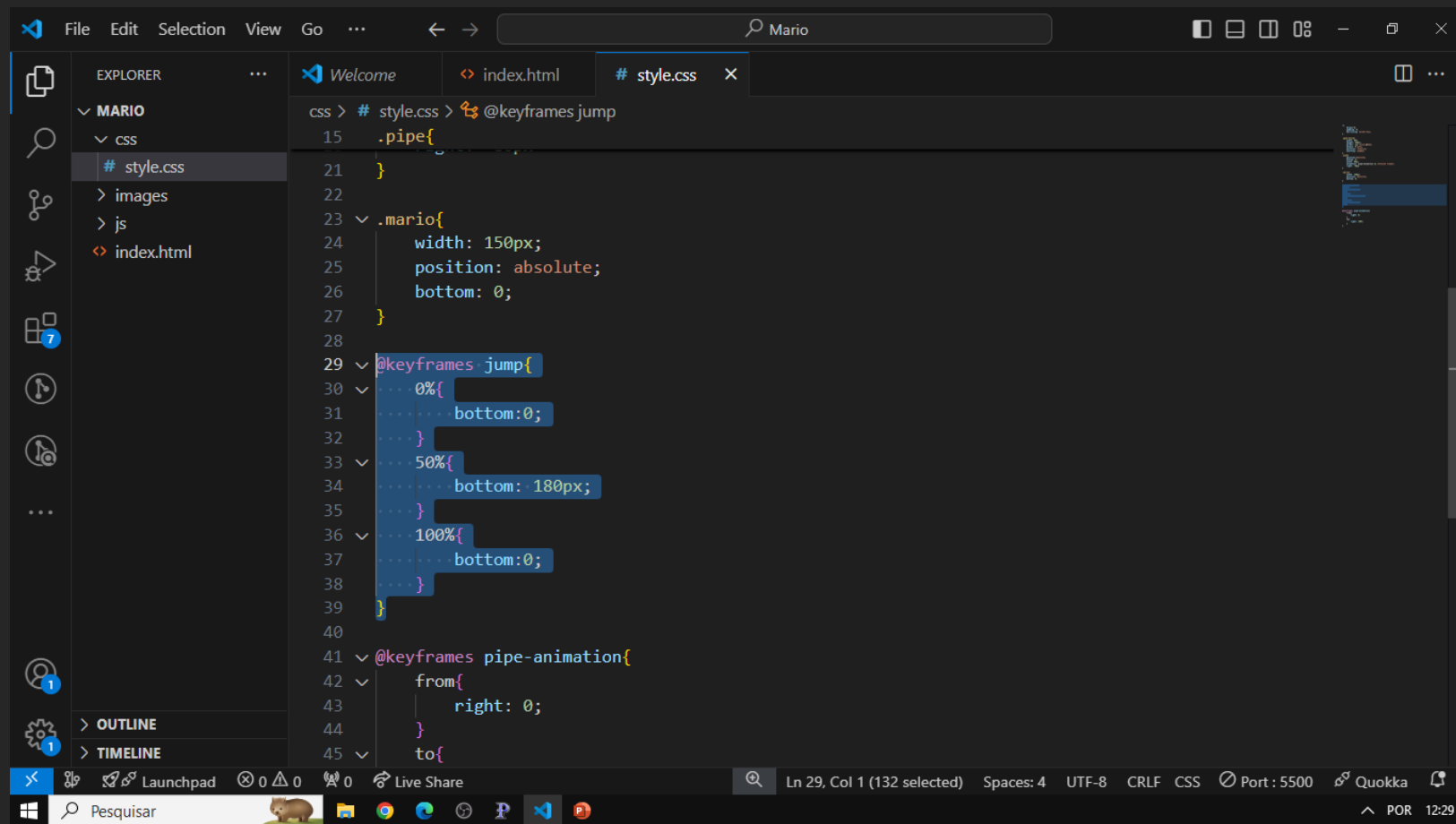
Vamos colocar agora o Mario, la no html
Ele eh um gif animado e vamos colocar uma classe para ele

```
File Edit Selection View Go ... Mario
EXPLORER
MARIO
  css
  # style.css
  images
  js
  index.html
css > # style.css > .mario
7 .game-board{
11   margin: 0 auto;
12   position: relative;
13   overflow: hidden;
14 }
15 .pipe{
16   position: absolute;
17   bottom: 0;
18   width: 80px;
19   animation: pipe-animation 2s infinite linear;
20   right: -80px;
21 }
22
23 .mario{
24   width: 150px;
25   position: absolute;
26   bottom: 0;
27 }
28 @keyframes pipe-animation{
29   from{
30     right: 0;
31   }
32   to{
33     right: 100%;
34   }
35 }
Ln 26, Col 14 Spaces: 4 UTF-8 CRLF CSS Port: 5500 Quokka
```

No css vamos colocar um tamanho, um position Absolute, e um bottom 0 para ficar em baixo



Aqui o Mario já aparece em baixo

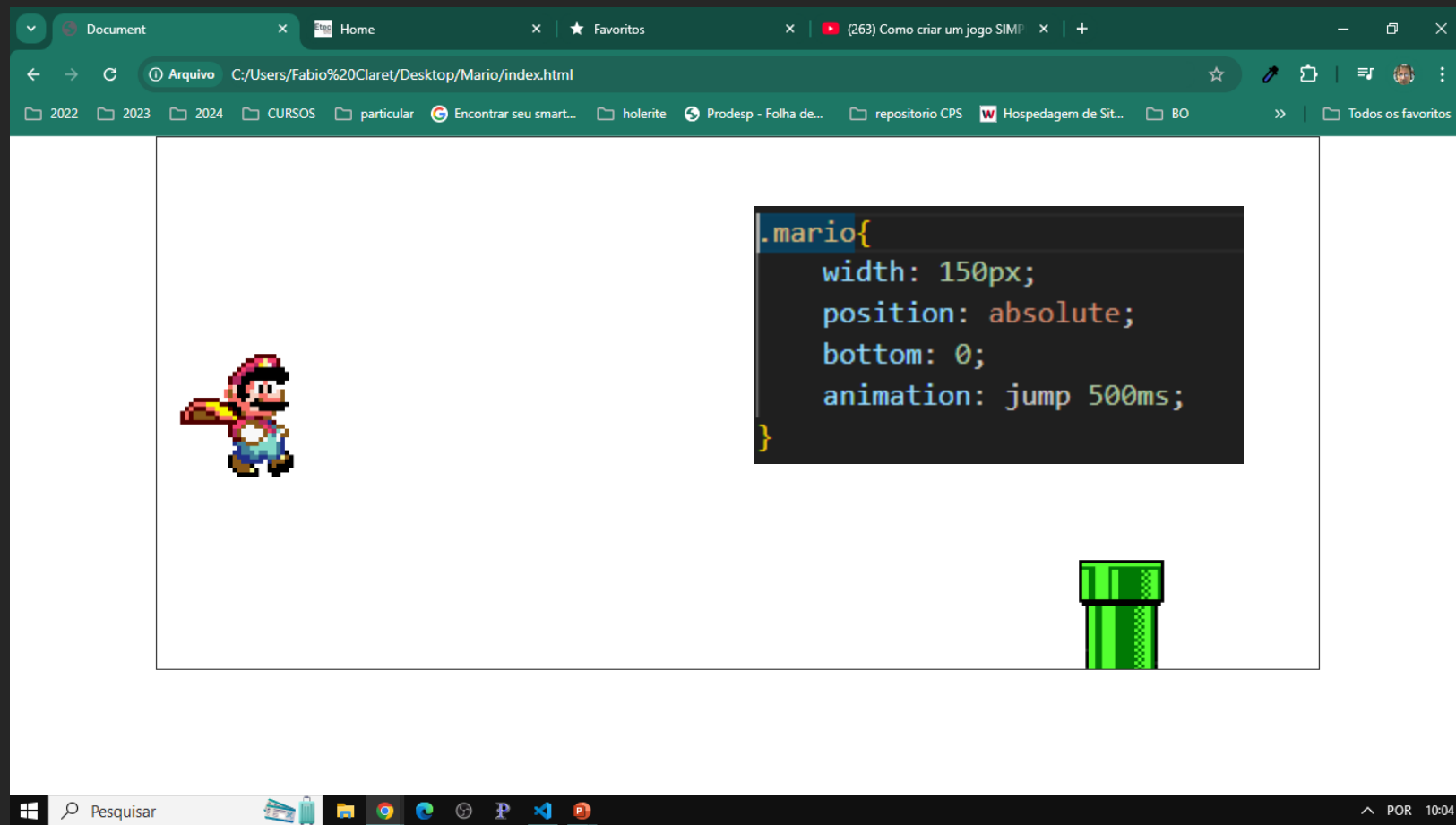


```
css > # style.css > @keyframes jump
15 .pipe{
21 }
22
23 .mario{
24     width: 150px;
25     position: absolute;
26     bottom: 0;
27 }
28
29 @keyframes jump{
30     0%{
31         bottom:0;
32     }
33     50%{
34         bottom: 180px;
35     }
36     100%{
37         bottom:0;
38     }
39 }
40
41 @keyframes pipe-animation{
42     from{
43         right: 0;
44     }
45     to{
```

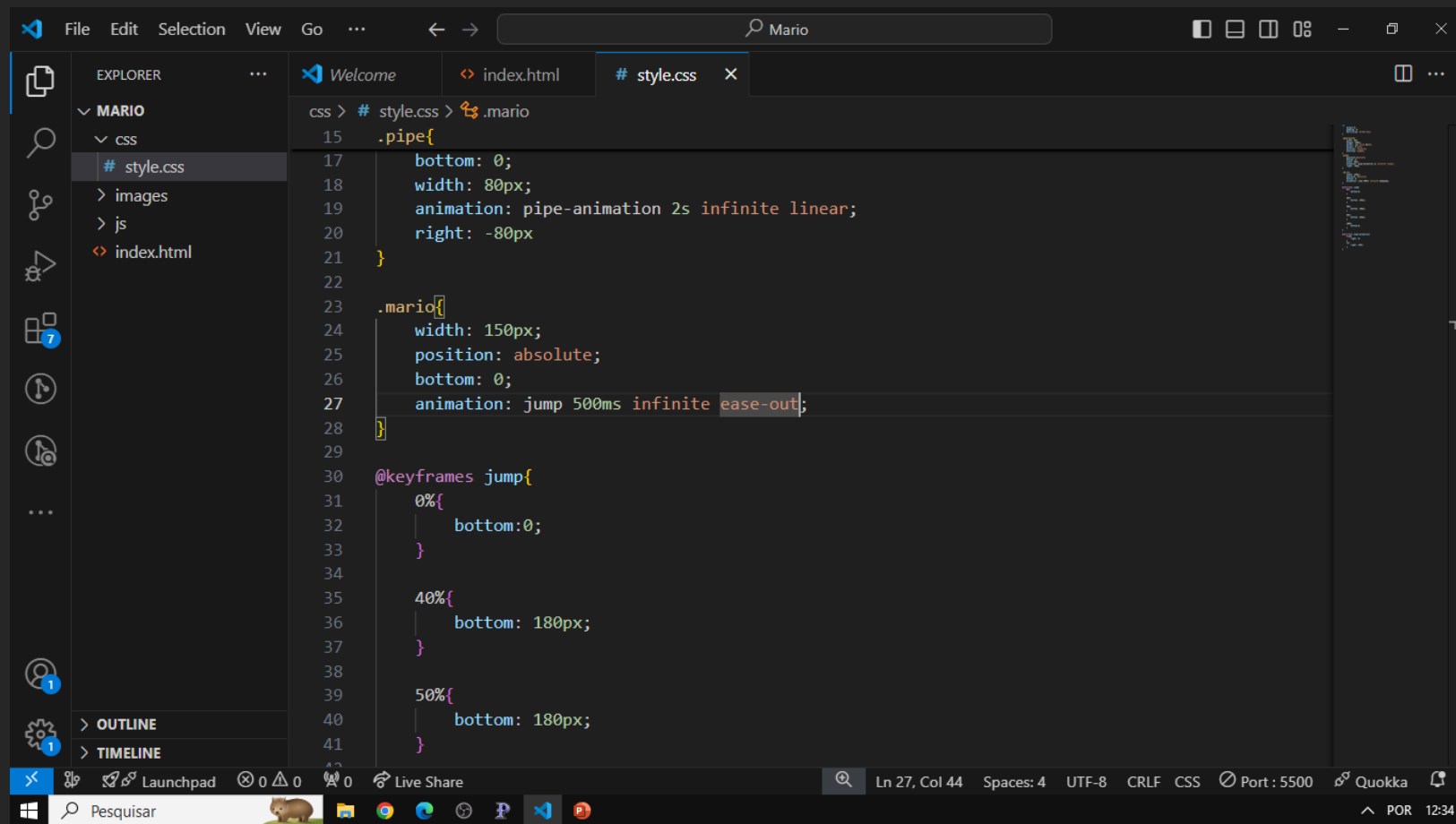
Vamos criar uma animação para fazer o mario pular. Agora a animação sera na propriedade bottom, ele vai começar do bottom 0 vai subir uns 180px e depois vai voltar para o zero. Mas aqui vai ter três ciclos, vai começar , vai ter um meio e vai ter um fim, inicio bottom 0, meio vai ser 180px e no final zero.

```
css > # style.css > .mario
15  .pipe{
21  }
22
23  .mario{
24      width: 150px;
25      position: absolute;
26      bottom: 0;
27      animation: jump 500ms;
28  }
29
30  @keyframes jump{
31      0%{
32          bottom: 0;
33      }
34      50%{
35          bottom: 180px;
36      }
37      100%{
38          bottom: 0;
39      }
40  }
41
42  @keyframes pipe-animation{
43      from{
44          right: 0;
45  }
```

Depois a gente coloca o mario para pular chamando a animacao



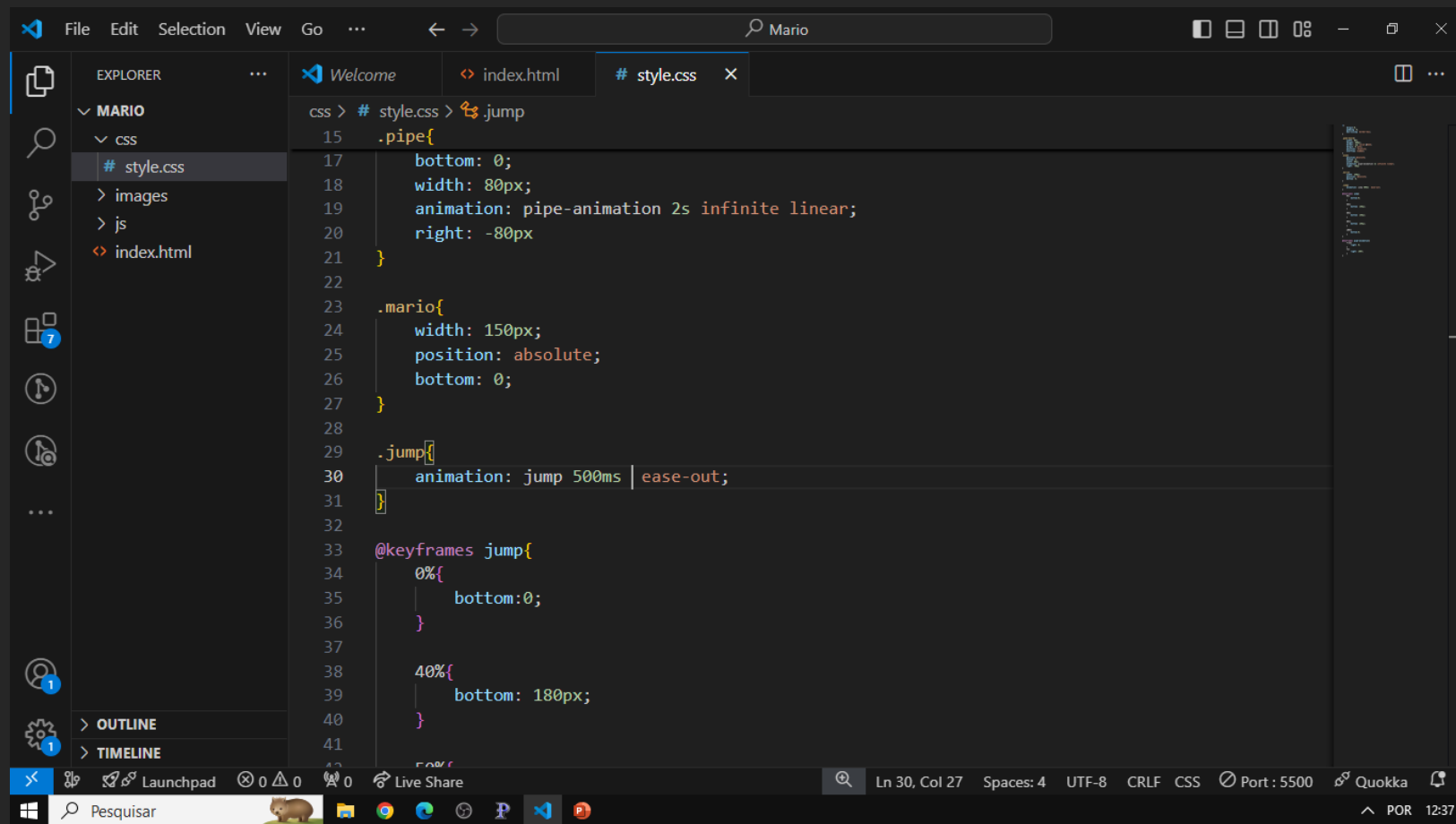
Agora temos a classe mario que eh o gif, e dentro dela a gente colocou a animação. Ao atualizar a pagina, ele vai pular. Mas depois disso, ele para, porque a animação já aconteceu. Como podemos fazer para que ele pule sempre que eu clicar em uma tecla por exemplo?



```
css > # style.css > .mario
15  .pipe{
17      bottom: 0;
18      width: 80px;
19      animation: pipe-animation 2s infinite linear;
20      right: -80px;
21  }
22
23  .mario{
24      width: 150px;
25      position: absolute;
26      bottom: 0;
27      animation: jump 500ms infinite ease-out;
28  }
29
30  @keyframes jump{
31      0%{
32          bottom:0;
33      }
34
35      40%{
36          bottom: 180px;
37      }
38
39      50%{
40          bottom: 180px;
41      }
42  }
```

Vamos fazer ele pular sempre com infinite e colocar um delay na caída com easy-out. Agora ele vai descer mais devagar dando impressão de pulo

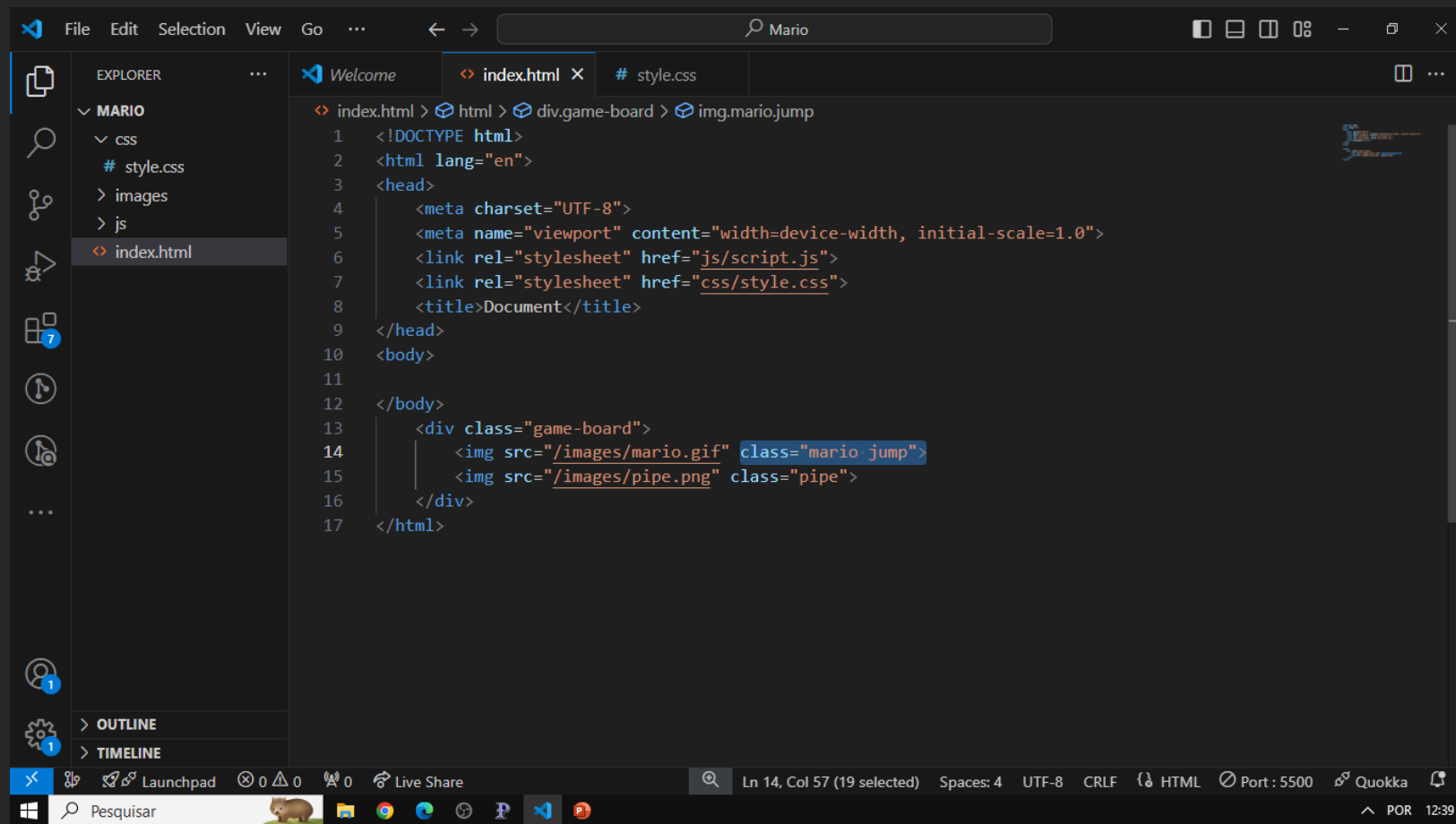




```
css > # style.css > jump
15  .pipe{
16      bottom: 0;
17      width: 80px;
18      animation: pipe-animation 2s infinite linear;
19      right: -80px;
20  }
21
22
23  .mario{
24      width: 150px;
25      position: absolute;
26      bottom: 0;
27  }
28
29  .jump{
30      animation: jump 500ms ease-out;
31  }
32
33  @keyframes jump{
34      0%{
35          bottom:0;
36      }
37
38      40%{
39          bottom: 180px;
40      }
41  }
```

Ln 30, Col 27 Spaces: 4 UTF-8 CRLF CSS Port : 5500 Quokka

Mas a gente não quer que ele fique pulando infinito, e queremos que ele pule apenas quando clicarmos em uma tecla, então tiramos a animação do mario e colocamos em uma nova classe jump



Quando a gente quiser que o mario pule, a gente vai na imagem do mario la no html e coloca a classe jump. Mas vamos criar essa logica no JS. Vamos fazer ele pular quando a gente clicar em qualquer tecla

Vamos começar adicionando um listener para quando a gente clicar em uma tecla, ele chamar a função jump que vamos criar.

O listener fica ouvindo o teclado para capturar quando a gente apertar uma tecla.

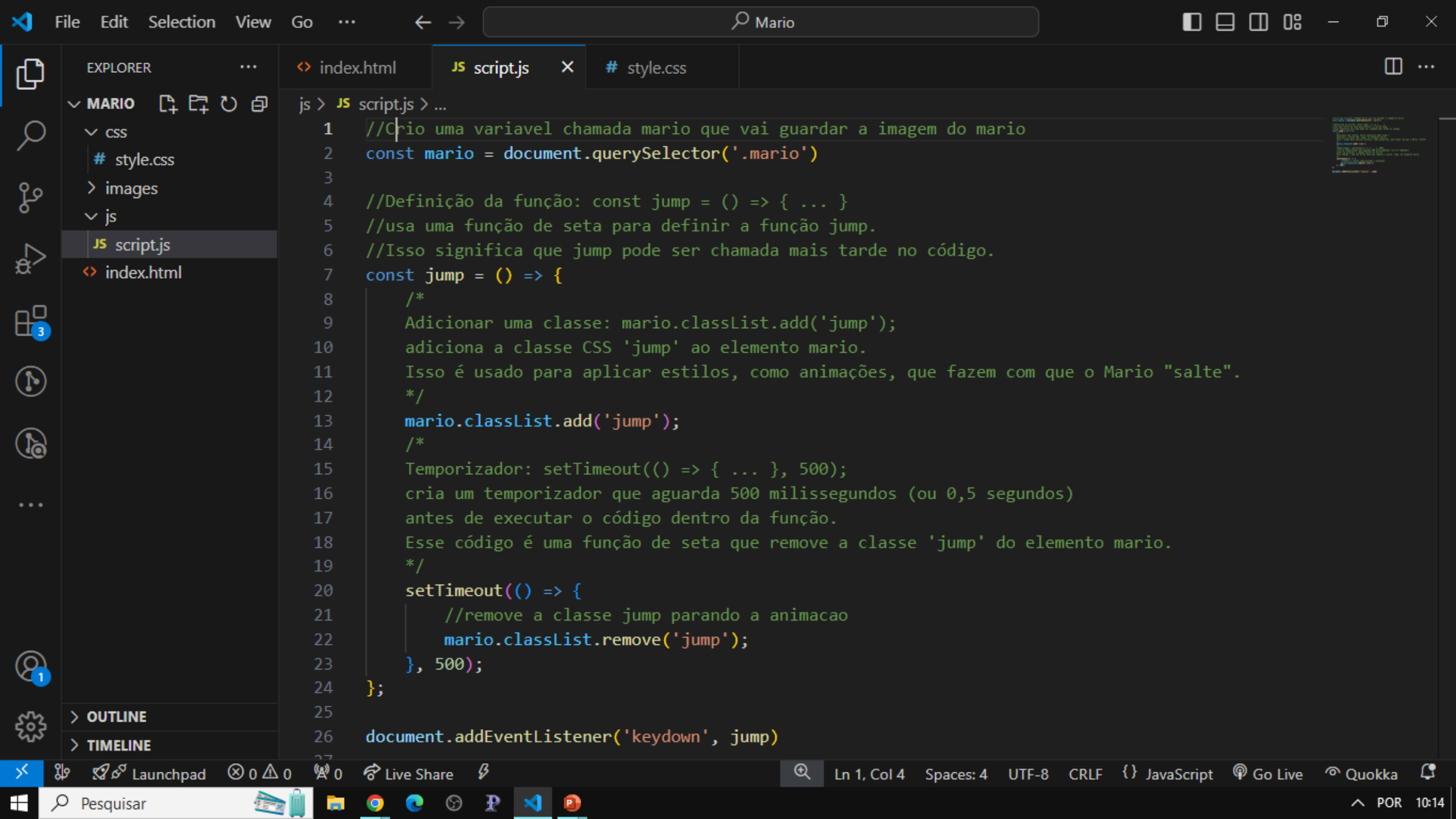
Agora vamos escrever a função jump, pois quando a gente chamar a função, a gente vai adicionar a classe criada lá no css, na imagem do mario.

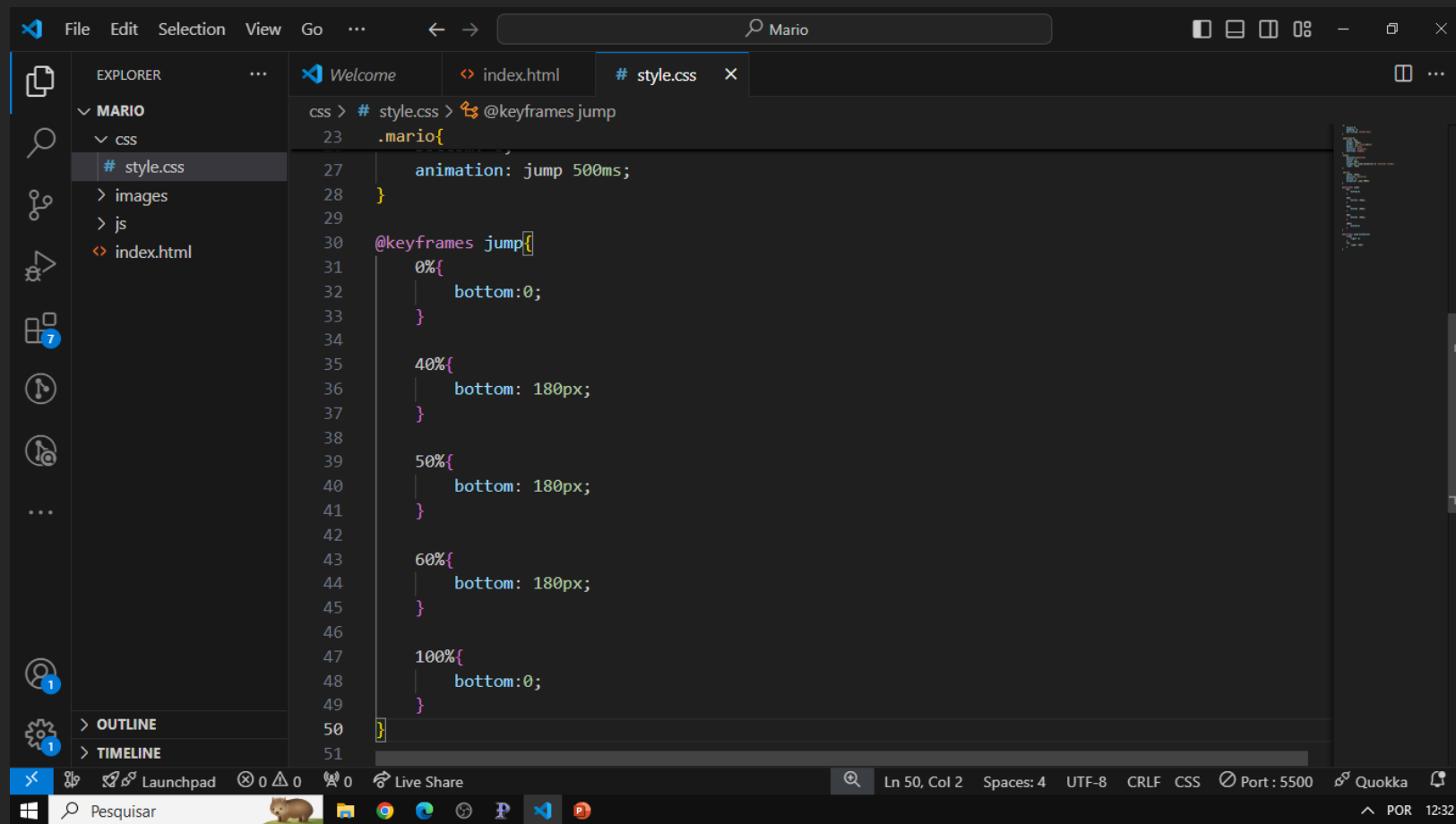
Então antes de tudo temos que pegar o elemento da imagem do mario, com o comando:
`const mario = queryselector` passando a classe para ele.

Esse comando vai guardar a imagem do mario dentro de uma variável chamada mario.

Agora a gente precisa adicionar a classe na função jump.

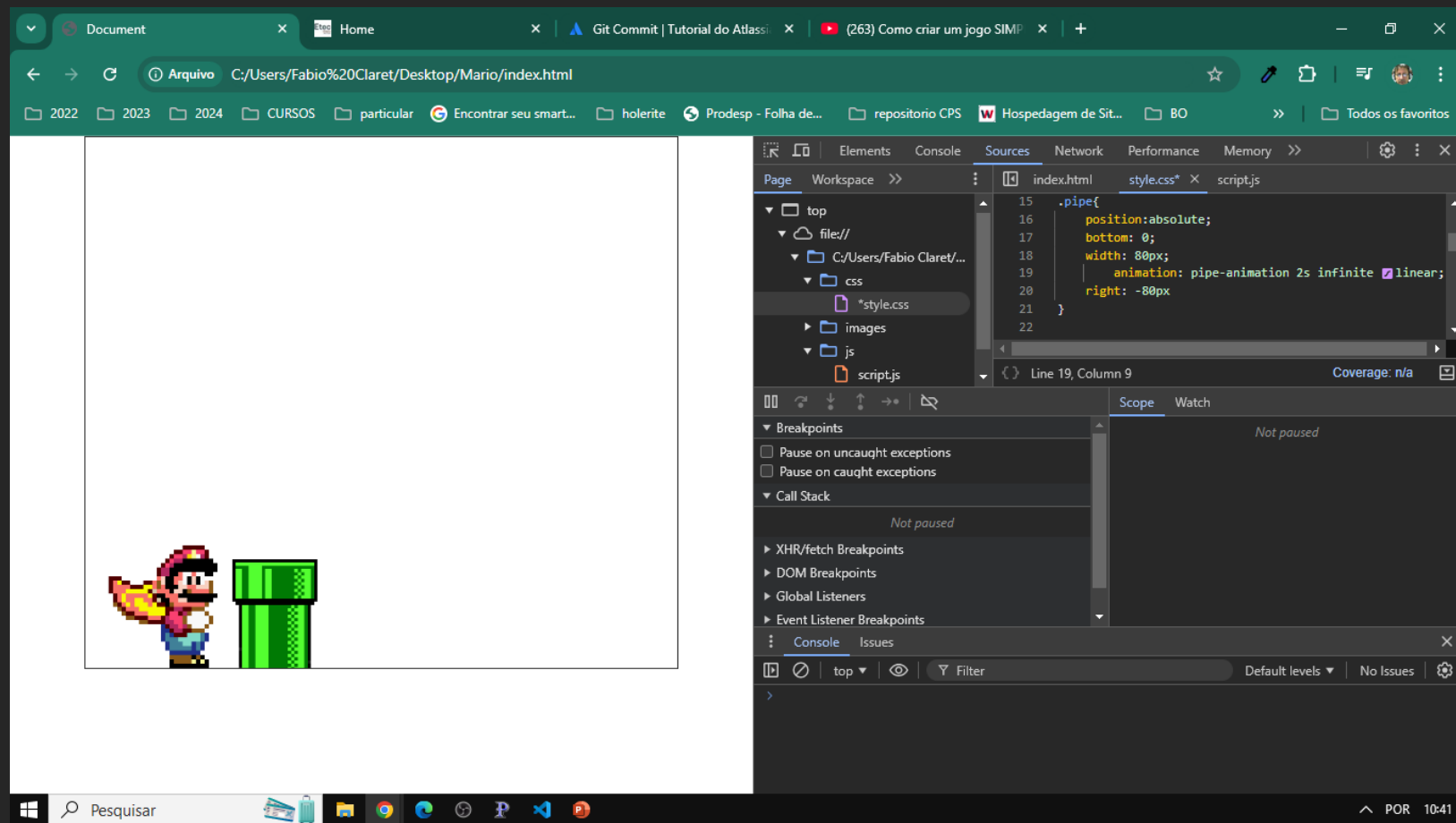
Então o que acontece aqui: quando eu pressionar uma tecla, ele vai chamar a função jump, e a função jump vai adicionar a classe jump que criamos lá no css, na imagem mario.





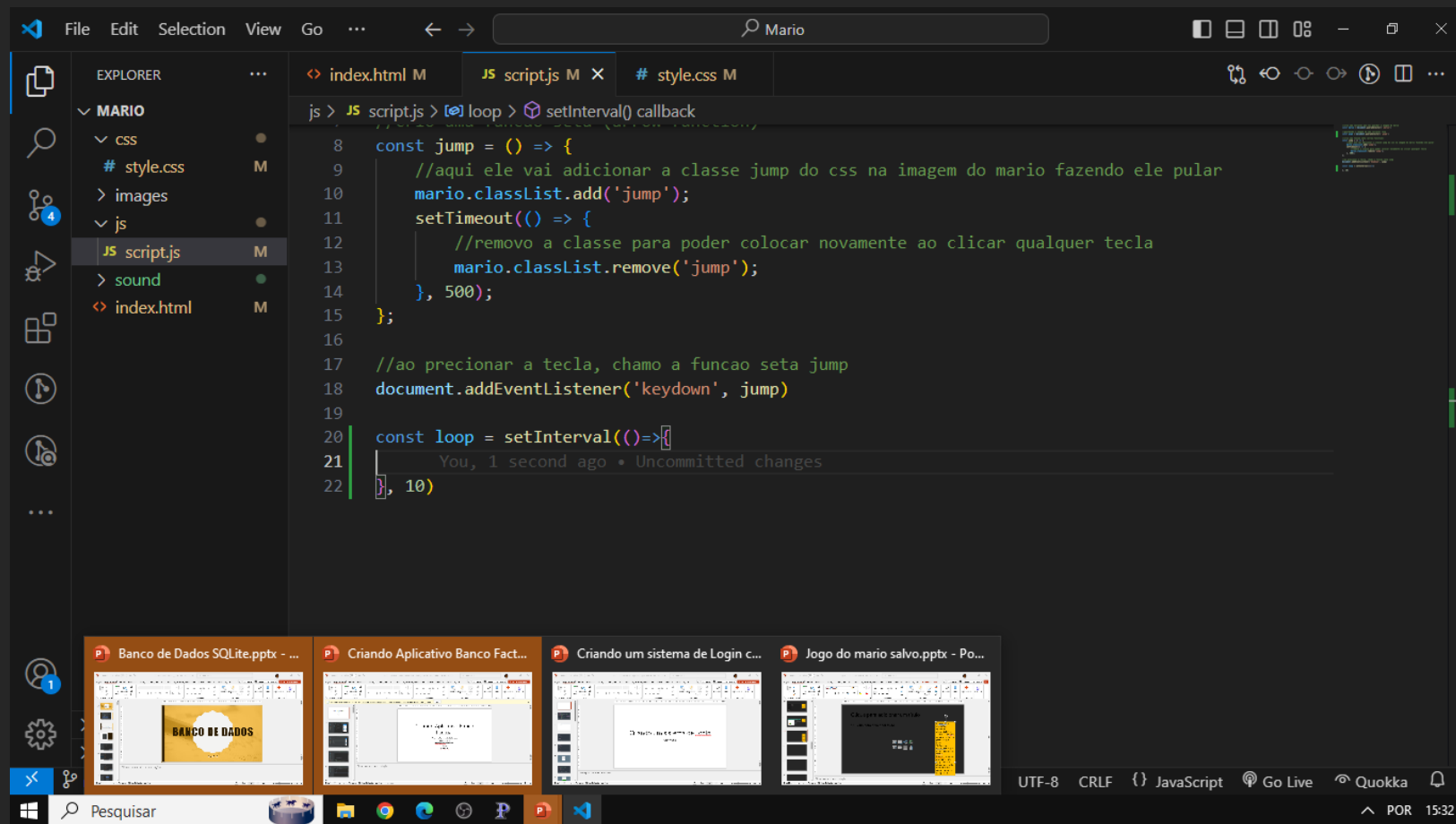
Desafio

Mas podemos melhorar essa animação, fazendo ele pular, demorar um pouco no ar depois cair pra dar uma sensação de pulo mesmo, colocando pontos intermediários.



Agora a gente precisa criar uma logica para quando o jogador não pular o tubo e ai acabar o jogo.

Lembrando que o mario esta sempre na posição 0, e o que se move eh o tubo



A logica eh a seguinte...

Quando a distancia left for 120px, se o mario não tiver pulado, o jogo tem que parar.

120px eh a distancia em que o tufo alcança o mario...

Primeiro, a gente precisa ter um loop para ficar checando a todo momento se a gente perdeu ou não perdeu..

A função que fica checando eh a setInterval..

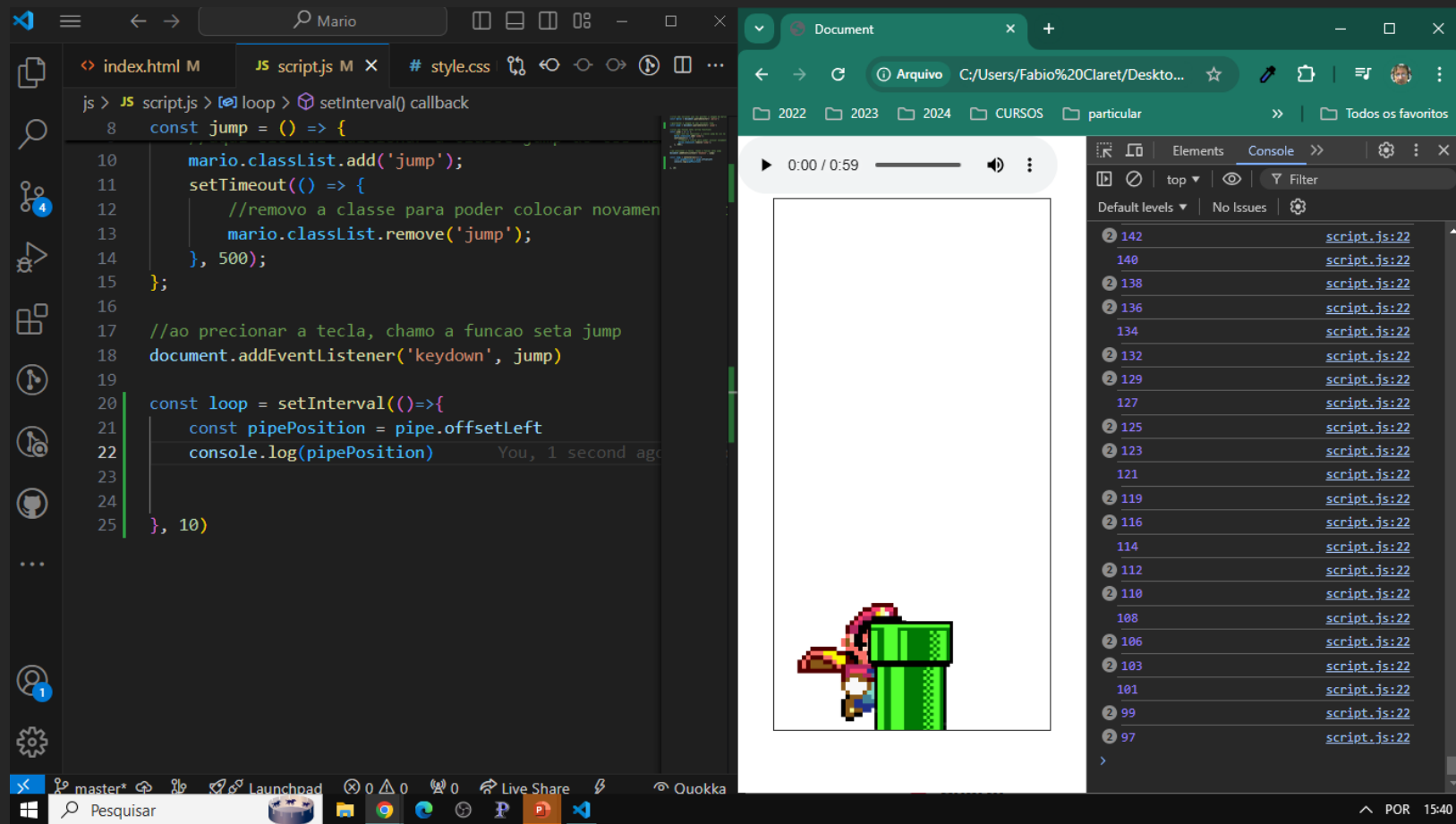
Dentro do setInterval a gente passa uma função e um tempo em que ela vai ficar repetindo

```
js > JS script.js > ...
You, 2 minutes ago | 1 author (You)
1 //crio uma variavel que vai guardar a imagem do mario
2 const mario = document.querySelector('.mario')
3
4 //guardando a imagem em uma variavel loop
5 const pipe = document.querySelector('.pipe')
6
7 You, 2 minutes ago • Uncommitted changes
7 //crio uma funcao seta (arrow function)
8 const jump = () => {
9     //aqui ele vai adicionar a classe jump do css na imagem do mario fazendo ele pular
10     mario.classList.add('jump');
11     setTimeout(() => {
12         //remove a classe para poder colocar novamente ao clicar qualquer tecla
13         mario.classList.remove('jump');
14     }, 500);
15 };
16
17 //ao precionar a tecla, chamo a funcao seta jump
18 document.addEventListener('keydown', jump)
19
20 const loop = setInterval(()=>{
21     const pipePosition = pipe.offsetLeft
22
23     if(pipePosition <= 120){
24         pipe.style.animation = 'none'
25         //template literals = 0 cifrão($) seguido de chaves ({}))
26         //define que o que está dentro é um valor e não uma string
```

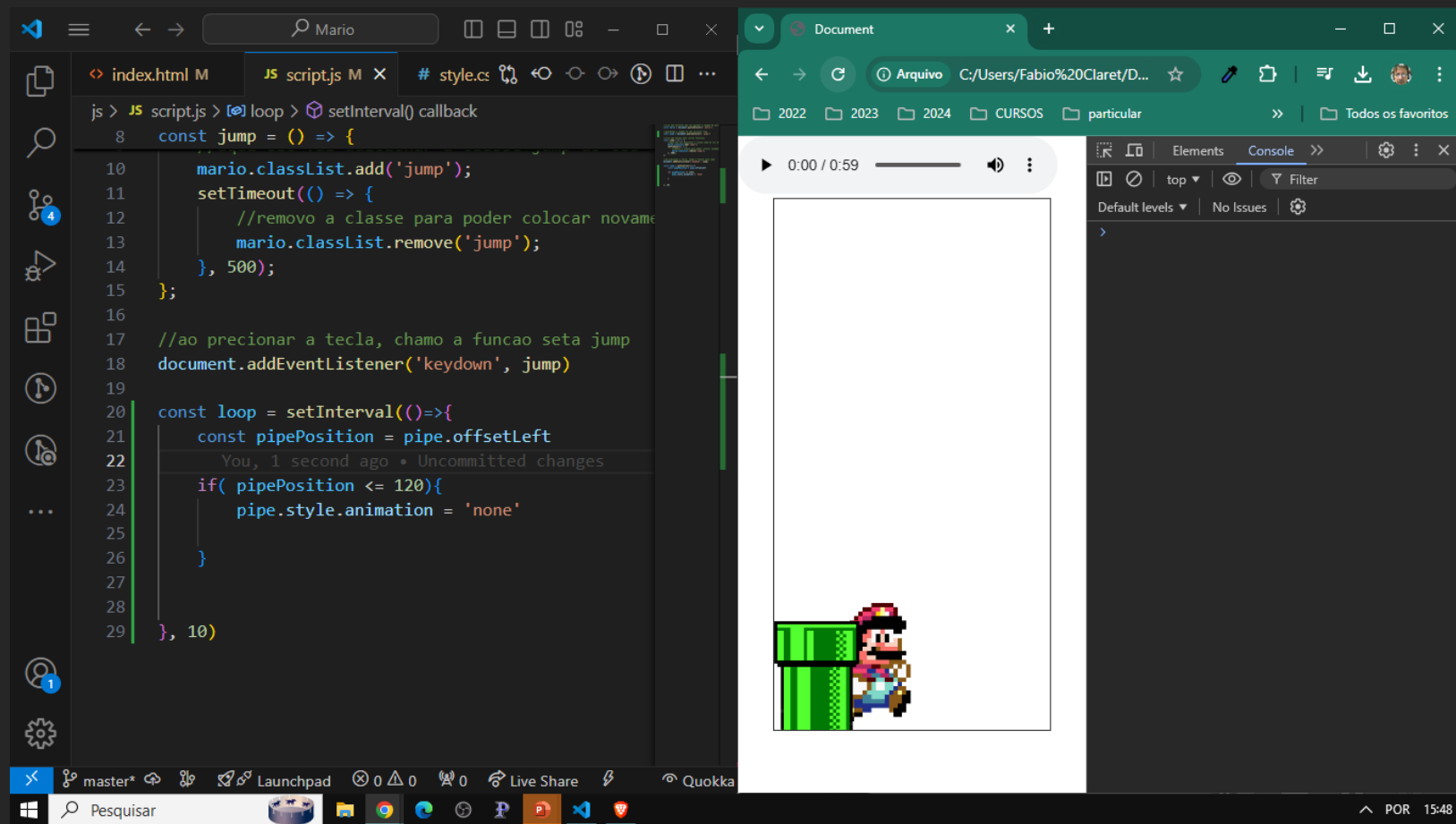
Se o deslocamento do tubo for menor ou igual a 120, quer dizer que o tubo já chegou aqui. Pra saber a posição do tubo, a primeira coisa que precisamos é pegar a imagem do tubo como fizemos com a imagem do mario. Vamos criar uma const então com a imagem do tubo


```
js > JS script.js > ...
You, 2 minutes ago | 1 author (You)
1 //crio uma variavel que vai guardar a imagem do mario
2 const mario = document.querySelector('.mario')
3
4 //guardando a imagem em uma variavel loop
5 const pipe = document.querySelector('.pipe')
6
7 You, 2 minutes ago • Uncommitted changes
7 //crio uma funcao seta (arrow function)
8 const jump = () => {
9     //aqui ele vai adicionar a classe jump do css na imagem do mario fazendo ele pular
10     mario.classList.add('jump');
11     setTimeout(() => {
12         //removo a classe para poder colocar novamente ao clicar qualquer tecla
13         mario.classList.remove('jump');
14     }, 500);
15 };
16
17 //ao precionar a tecla, chamo a funcao seta jump
18 document.addEventListener('keydown', jump)
19
20 const loop = setInterval(()=>{
21     const pipePosition = pipe.offsetLeft
22
23     if(pipePosition <= 120){
24         pipe.style.animation = 'none'
25         //template literals = 0 cifrão($) seguido de chaves ({}))
26         //define que o que está dentro é um valor e não uma string
```

Dentro da função, a gente vai acessar a propriedade deslocamento esquerdo.

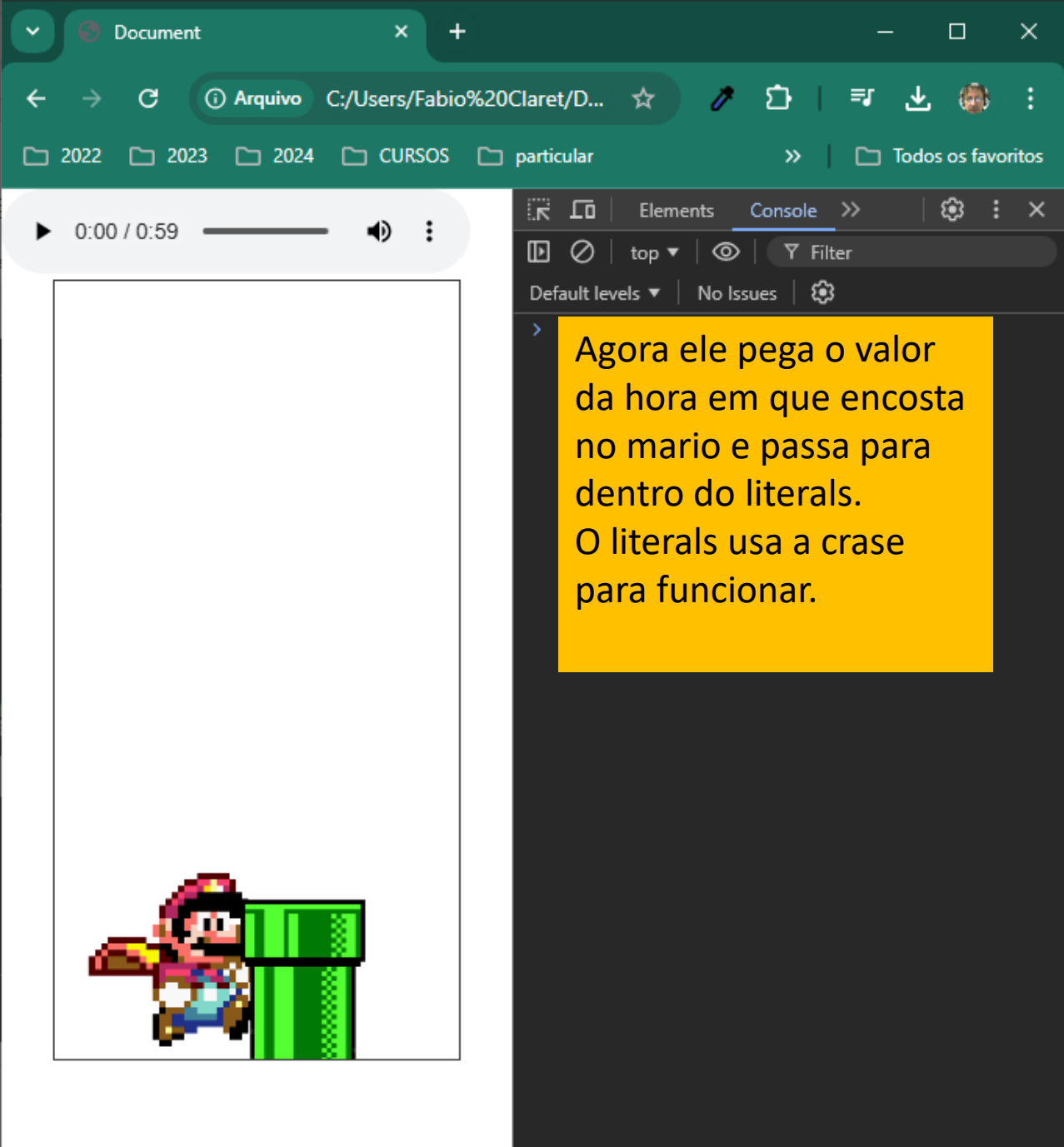


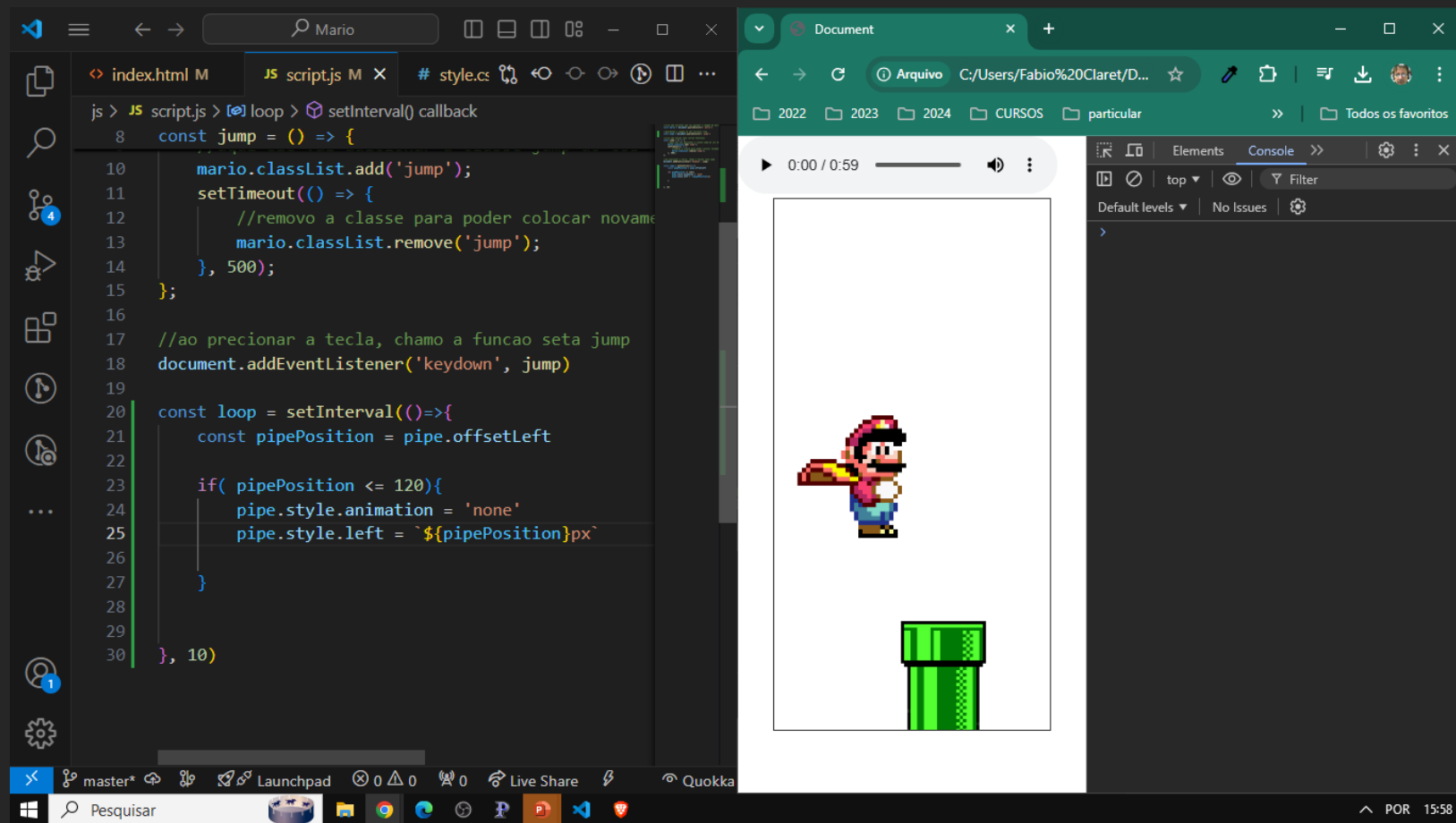
Vamos dar um console.log pra ver o que esta acontecendo. E o que esta acontecendo é que ele esta mostrando a posição do tubo o tempo todo. E quando chegar a 120 a gente para o jogo



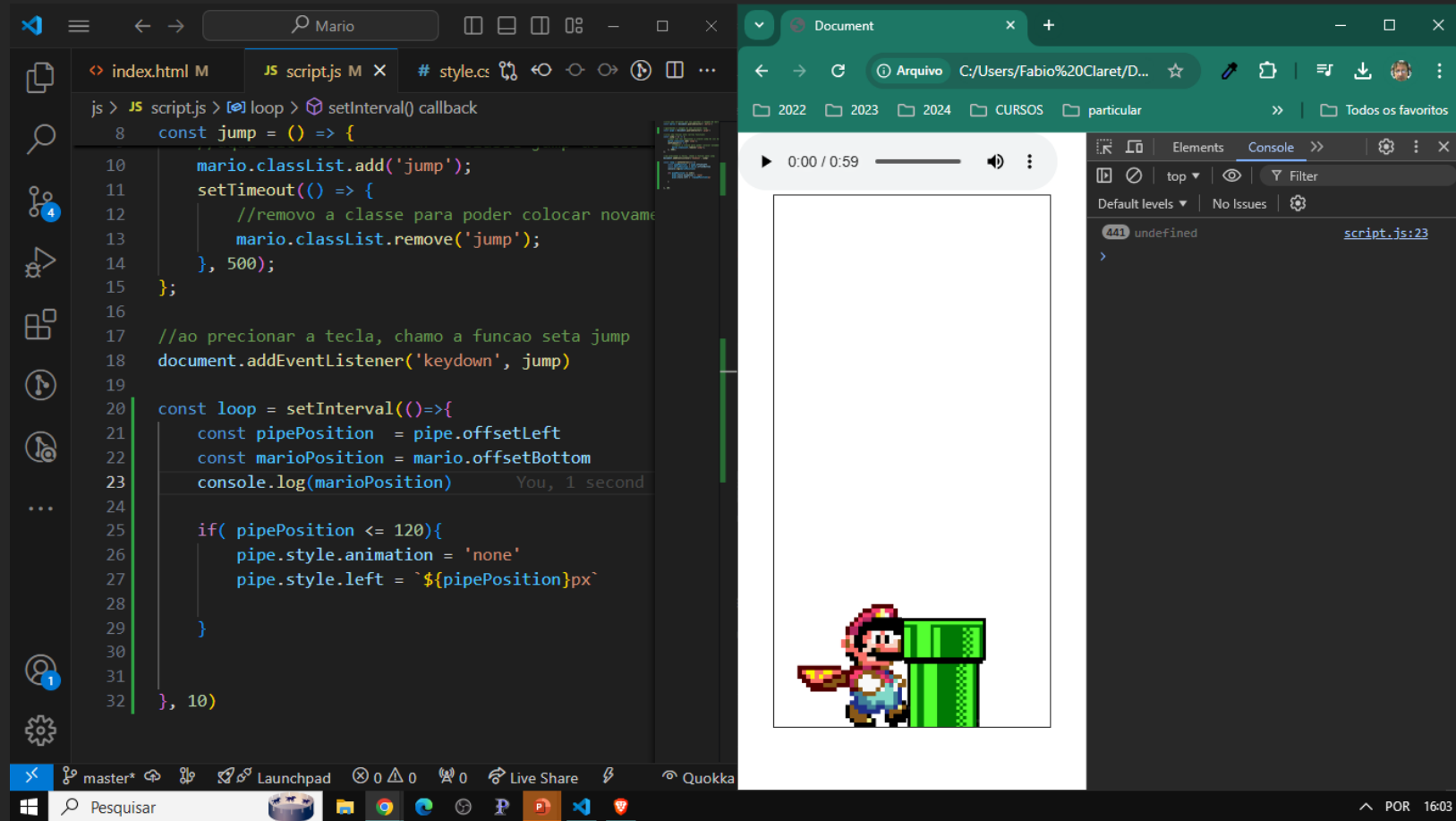
Entao com um if, eu vou checar se a posição eh menor ou igual 120. E a primeira coisa que fazemos se for, é parar a animação. E ele para a animação so que ele fica aqui na posição inicial, mas queremos que ele fique parado onde bateu.

```
index.html M JS script.js M X # style.css ↺ ↻ ↺ ↻ ↺ ↻ ↺ ↻ ...
js > JS script.js > [?] loop > [?] setInterval() callback
8 const jump = () => {
10     mario.classList.add('jump');
11     setTimeout(() => {
12         //removo a classe para poder colocar novame
13         mario.classList.remove('jump');
14     }, 500);
15 };
16
17 //ao precionar a tecla, chamo a funcao seta jump
18 document.addEventListener('keydown', jump)
19
20 const loop = setInterval(()=>{
21     const pipePosition = pipe.offsetLeft
22
23     if( pipePosition <= 120){
24         pipe.style.animation = 'none'
25         pipe.style.left = `${pipePosition}px`
26     }
27 }
28
29
30 }, 10)
```



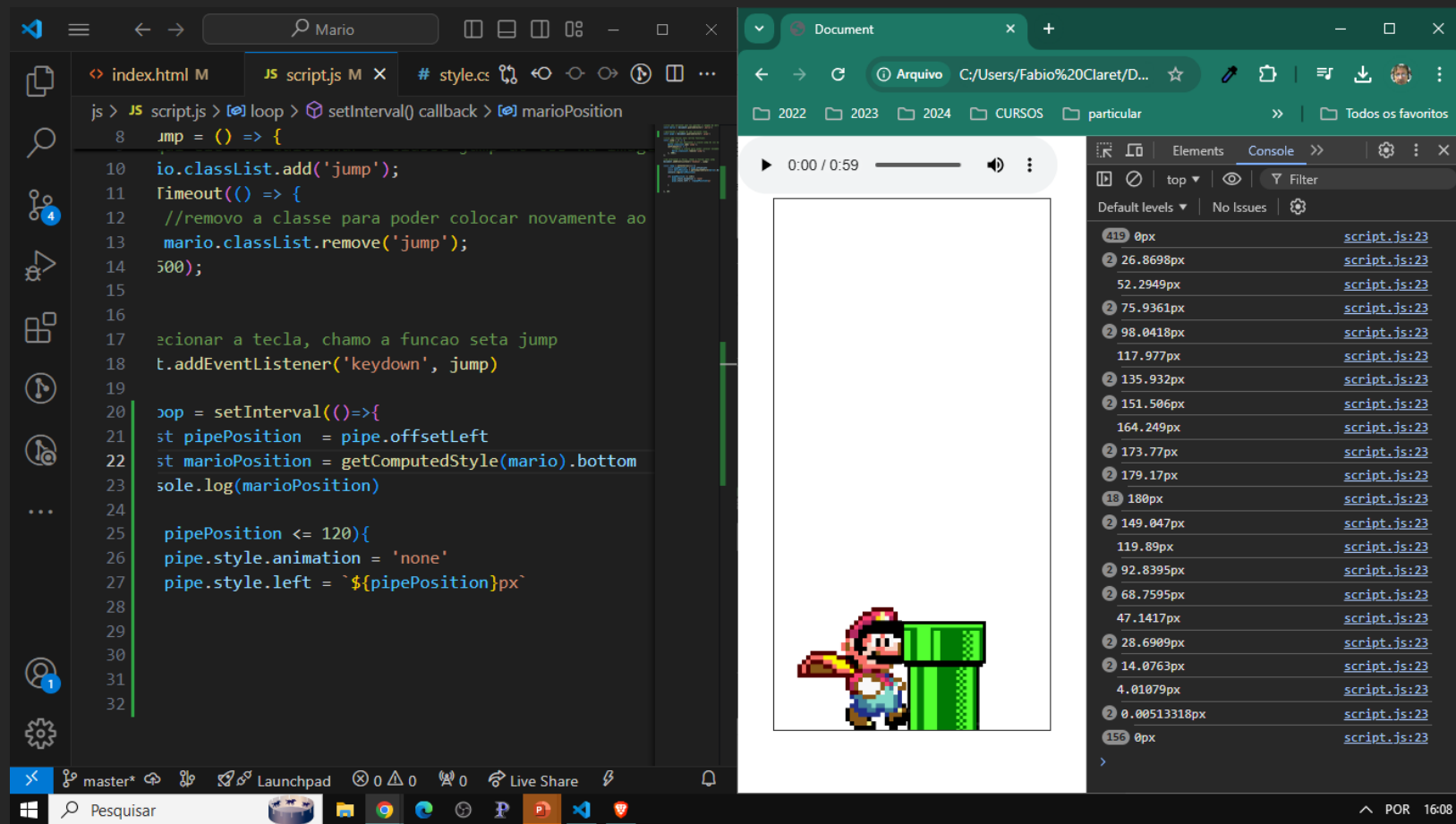


Ele já esta parando...
Mas esta parando
mesmo quando a gente
pula. Porque a única
condição que colocamos
pra acabar o jogo, é a
posição. Mas a ideia eh
so acabar o jogo se o
mario não tiver pulado.
E pra saber se o mario
não pulou, basta saber a
altura que o mario esta.

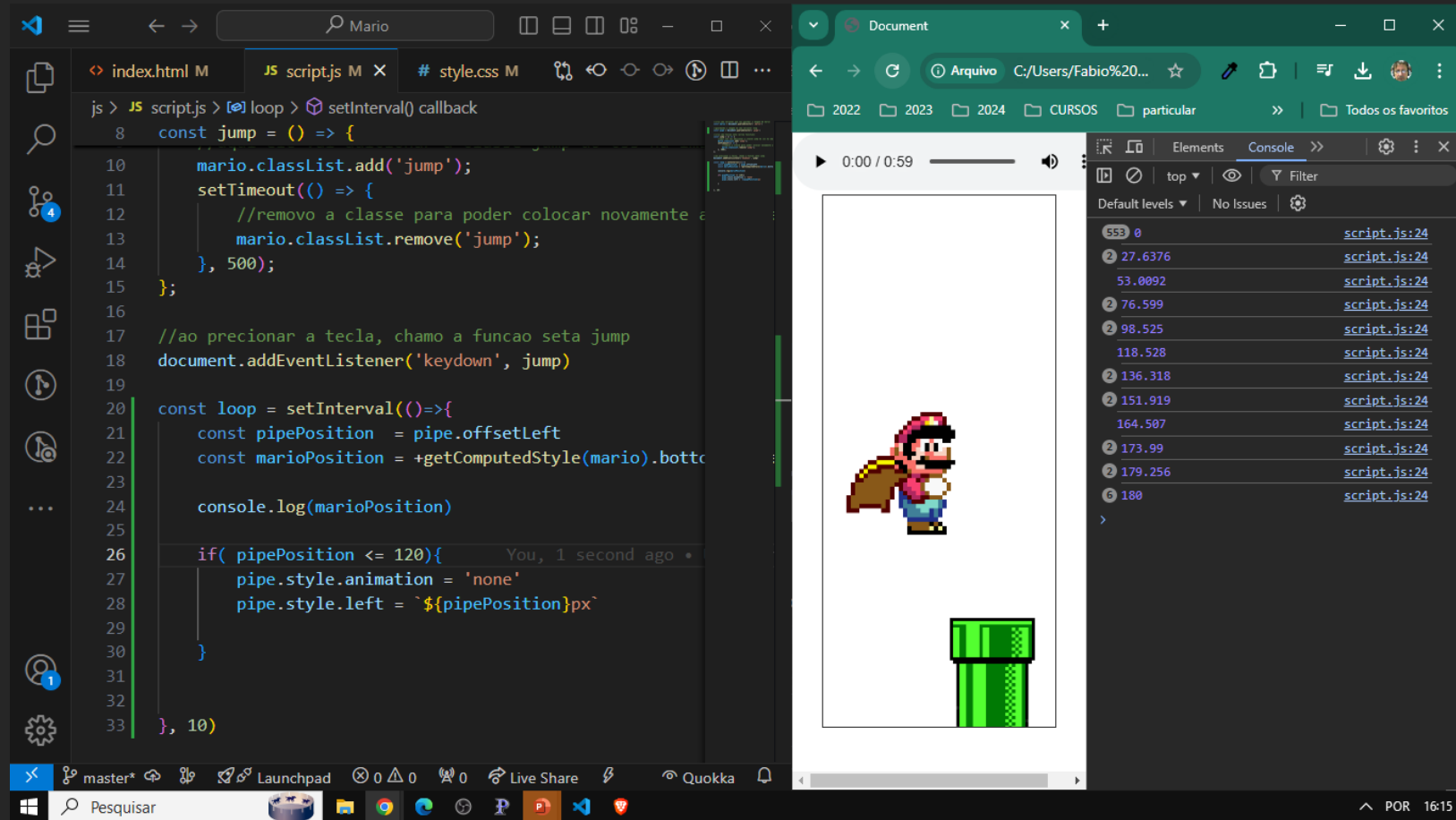


Vamos então verificar se o mario pulou, se a posição dele esta maior que 120, ai o jogo não vai acabar.

A gente tem que pegar a altura do mario, mas não podemos pegar do mesmo jeito que fizemos com o pipe. Vamos fazer um teste para ver que vai dar undefined

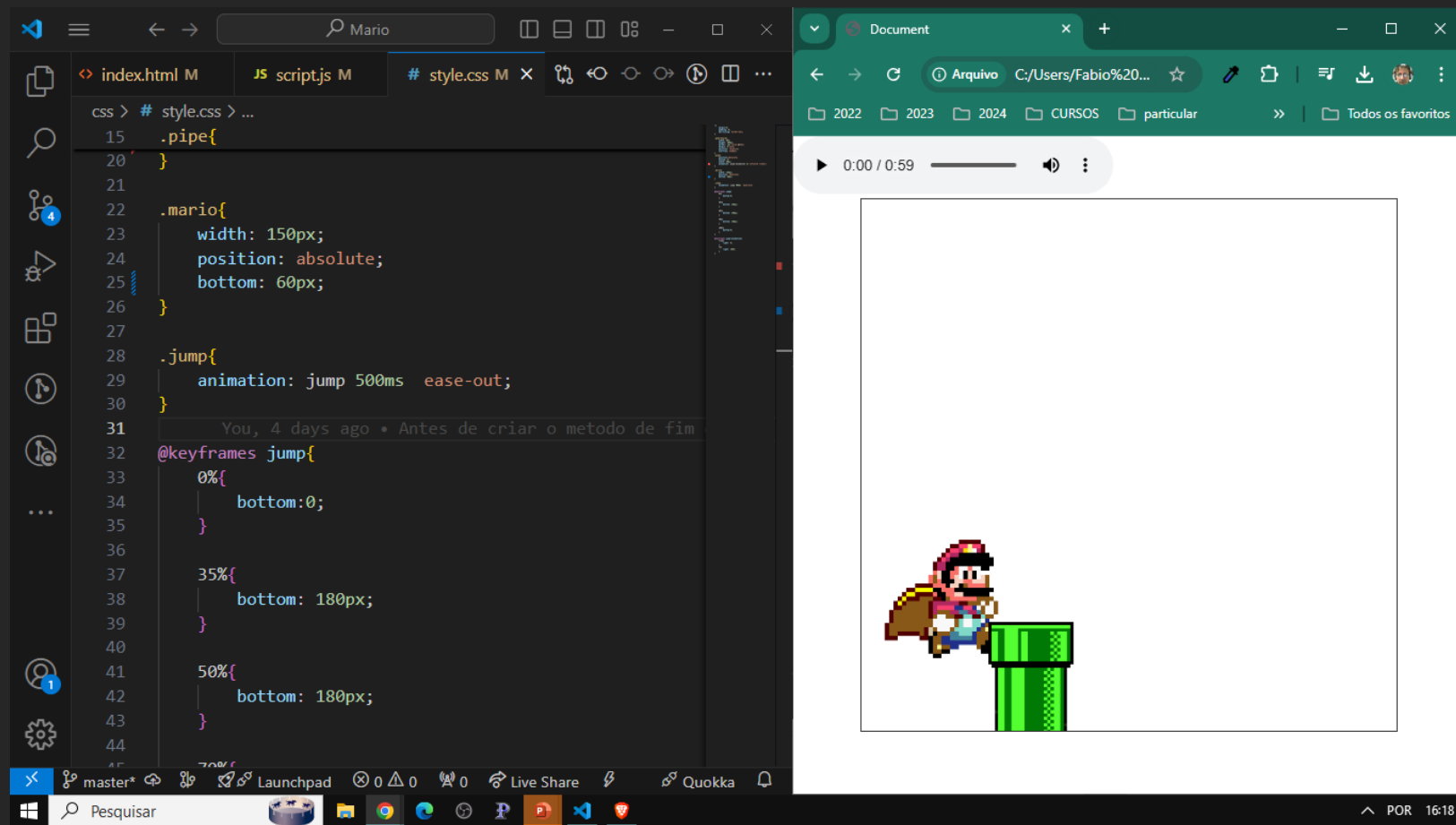


Entao temos que pegar a altura de outra forma. E a forma eh pegar com `getComputedStyle`. Passamos a classe mario, e podemos agora pegar qualquer valor imputado a classe. Agora ao pular, conseguimos pegar a altura em relação a parte de baixo. Mas o detalhe eh que ele esta retornando uma string, por causa do px. E precisamos do valor numérico dele

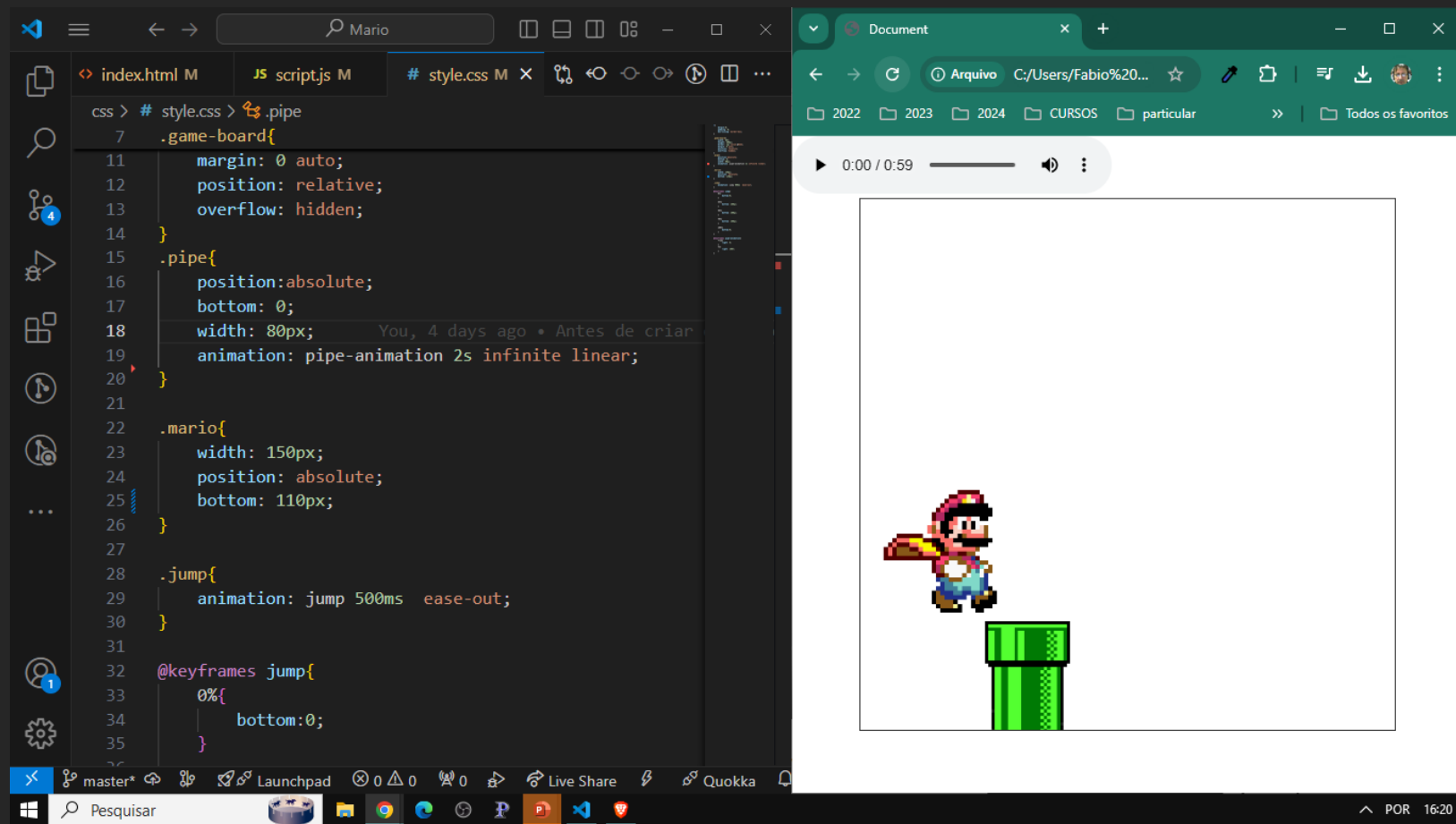


Temos que retirar o px, com replace, e mesmo assim converter para um numero.

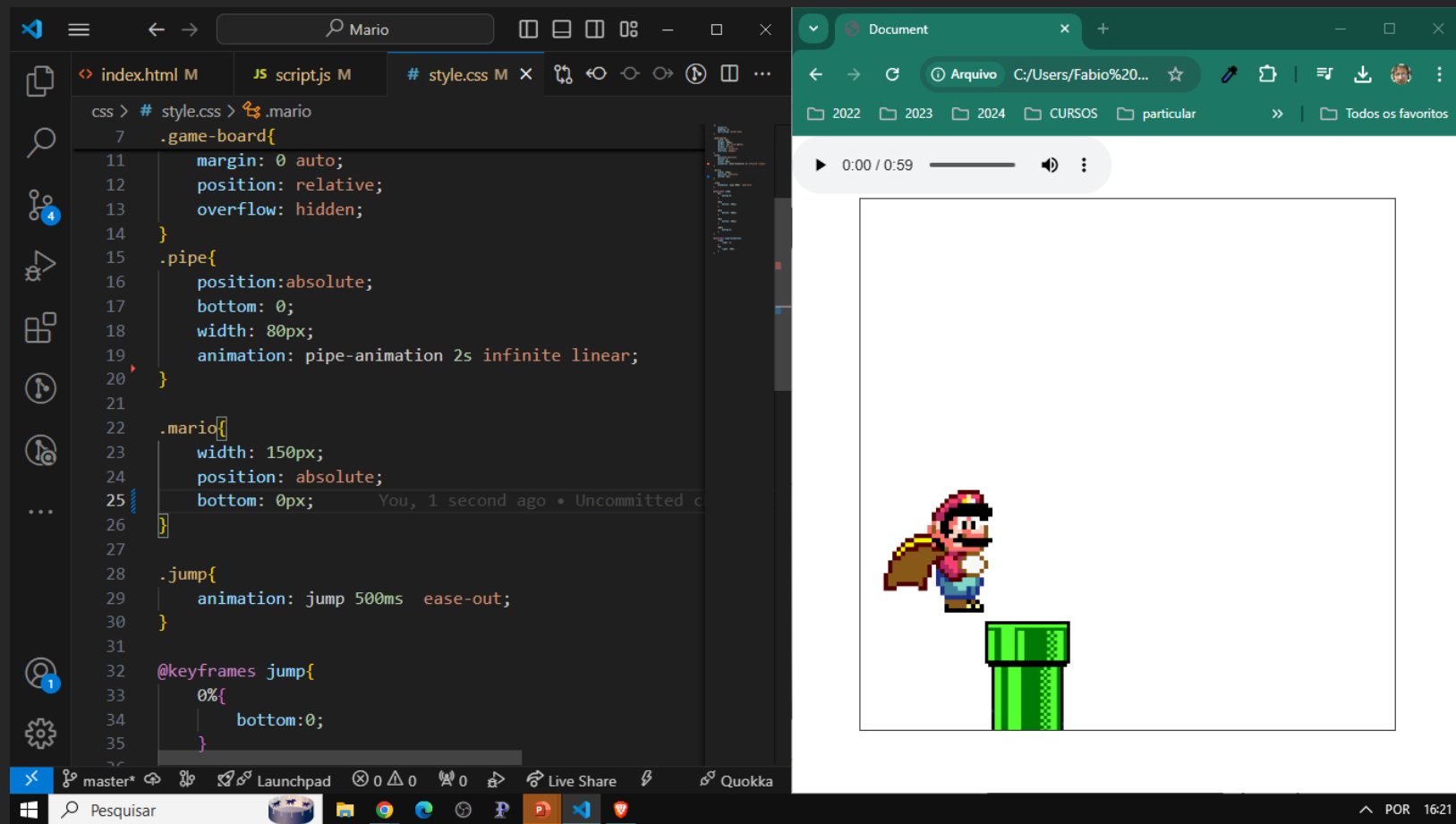
Fazemos isso ou com uma função number para converter, ou colocando o sinal de mais na frente. Esse mais vai tentar converter para numero caso seja numero mesmo



Agora temos que saber qual a altura mínima que o mario não encosta no tubo. Vamos fazer la no estilo para ver qual eh esse valor. Aqui vemos que 60 ainda bate.



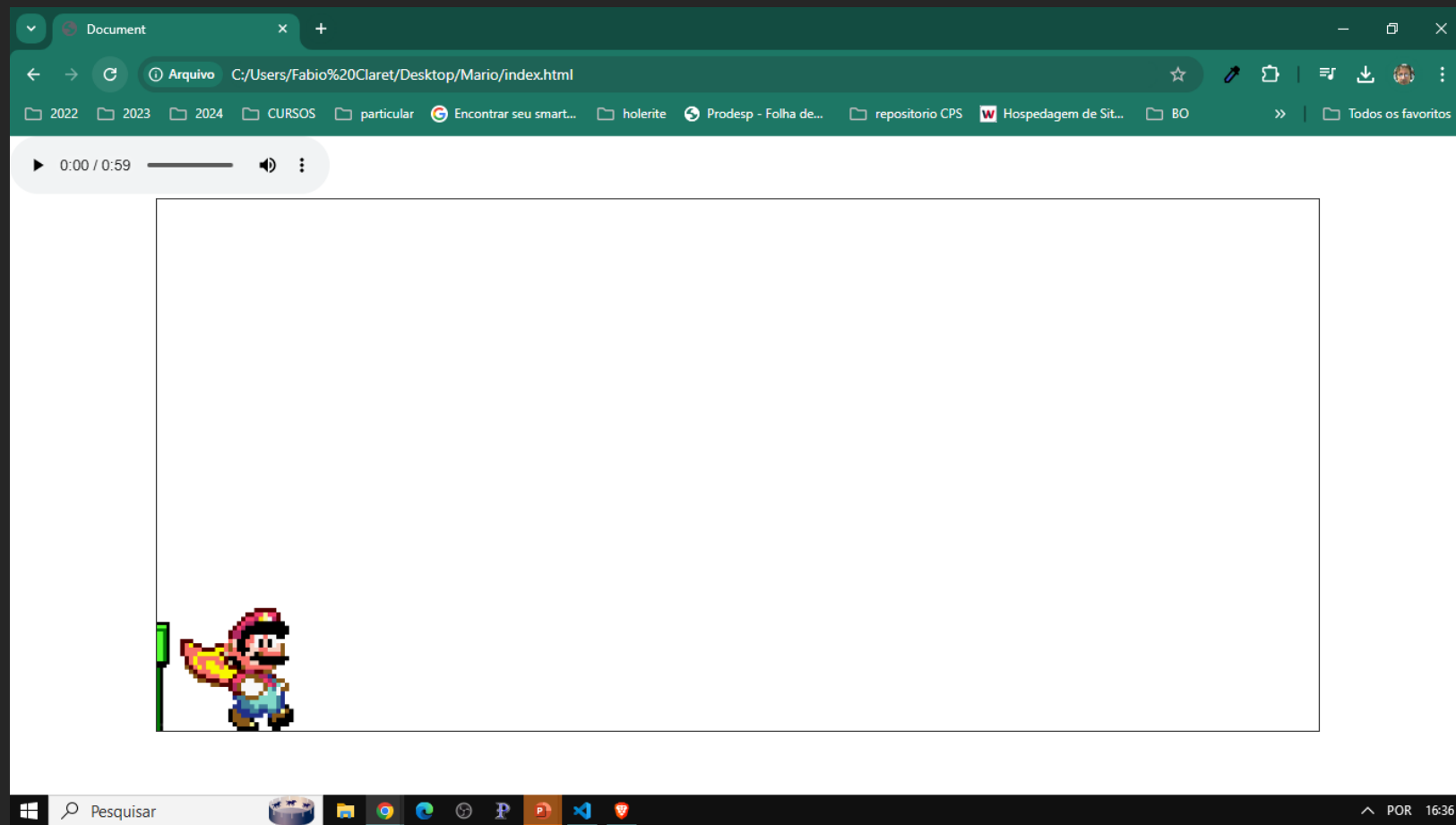
Aqui vemos que 100px
eh um valor que ele já
não bate.
Vamos voltar para zero
la no mario, e vamos
adicionar uma nova
condição la no javascript



Voltamos para zero a posição do mario

```
js > JS script.js > [1] loop > setInterval() callback
20 const loop = setInterval(()=>{
21   const pipePosition = pipe.offsetLeft
22   const marioPosition = +getComputedStyle(mario).bottom.replace('px','')
23
24   console.log(marioPosition)
25
26   if( pipePosition <= 120 && marioPosition < 110){
27     pipe.style.animation = 'none'
28     pipe.style.left = `${pipePosition}px`
29   }
30 }, 10)
```

E aqui na função, vamos adicionar uma nova condição, que é verificar se o tubo chegou e a posição do mario for menor que 110px. Ou seja, se o tubo chegou e a altura eh menor que 110, ai acaba o jogo;



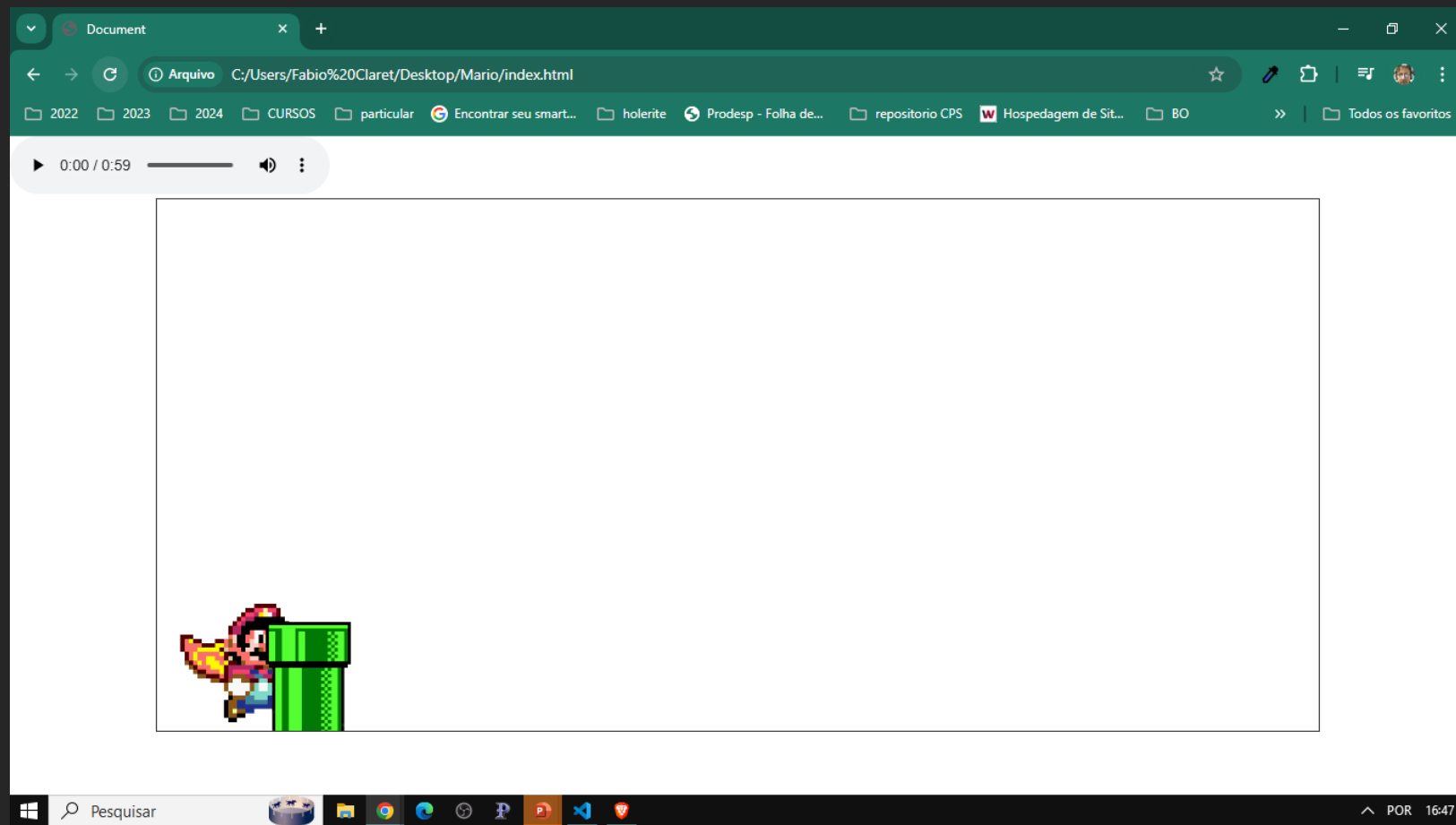
Agora, quando o mario cai, independente de onde caiu, o tubo para. Mas isso não é pra acontecer. Se o mario passar do tubo, o tubo tem que continuar. Pra resolver isso, a gente tem que pensar que se a posição do tubo for menor que zero, quer dizer que o tubo já passou do mario, e não tem como o mario encostar nele.

```
js > JS script.js > [?] loop
20 const loop = setInterval(()=>{
21   const pipePosition = pipe.offsetLeft
22   const marioPosition = +getComputedStyle(mario).bottom.replace('px','')
23
24   console.log(marioPosition)
25
26   if( pipePosition <= 120 && pipePosition > 0 && marioPosition < 110){
27     pipe.style.animation = 'none'
28     pipe.style.left = `${pipePosition}px`
29   }
30 }, 10)
```

You, 28 seconds ago • Uncommitted changes

Ln 30, Col 7 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Quokka

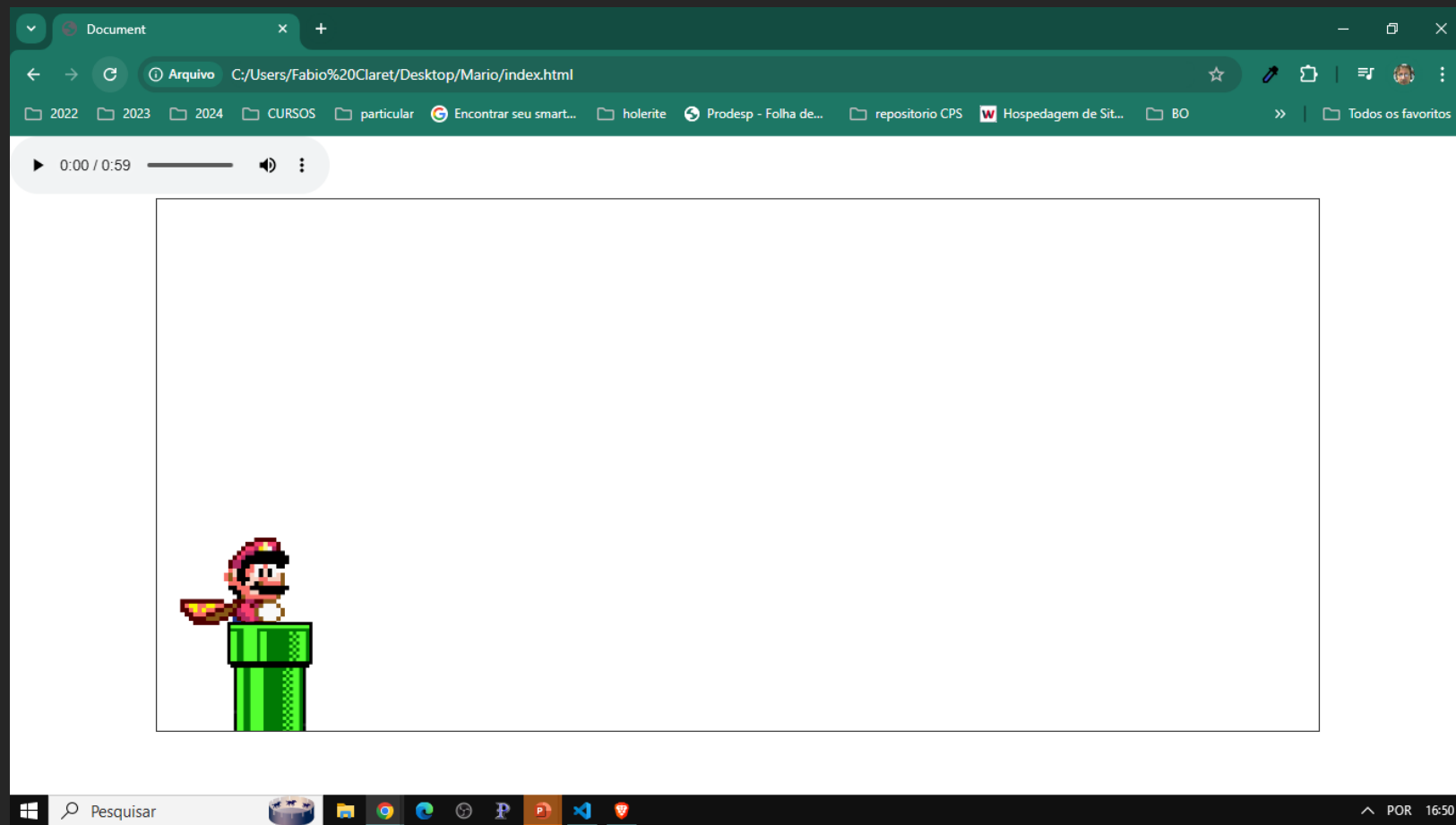
Entao agora colocamos mais uma condiçao. Se o deslocamento do tubo já chegou aqui no 120 e a posição do pipe ainda eh maior que zero, quer dizer que o tubo ainda esta em baixo do mario. E o pulo do mario esta menor que 80, ai acaba o jogo.



Agora se o mario pula, a animação não para e o jogo continua.

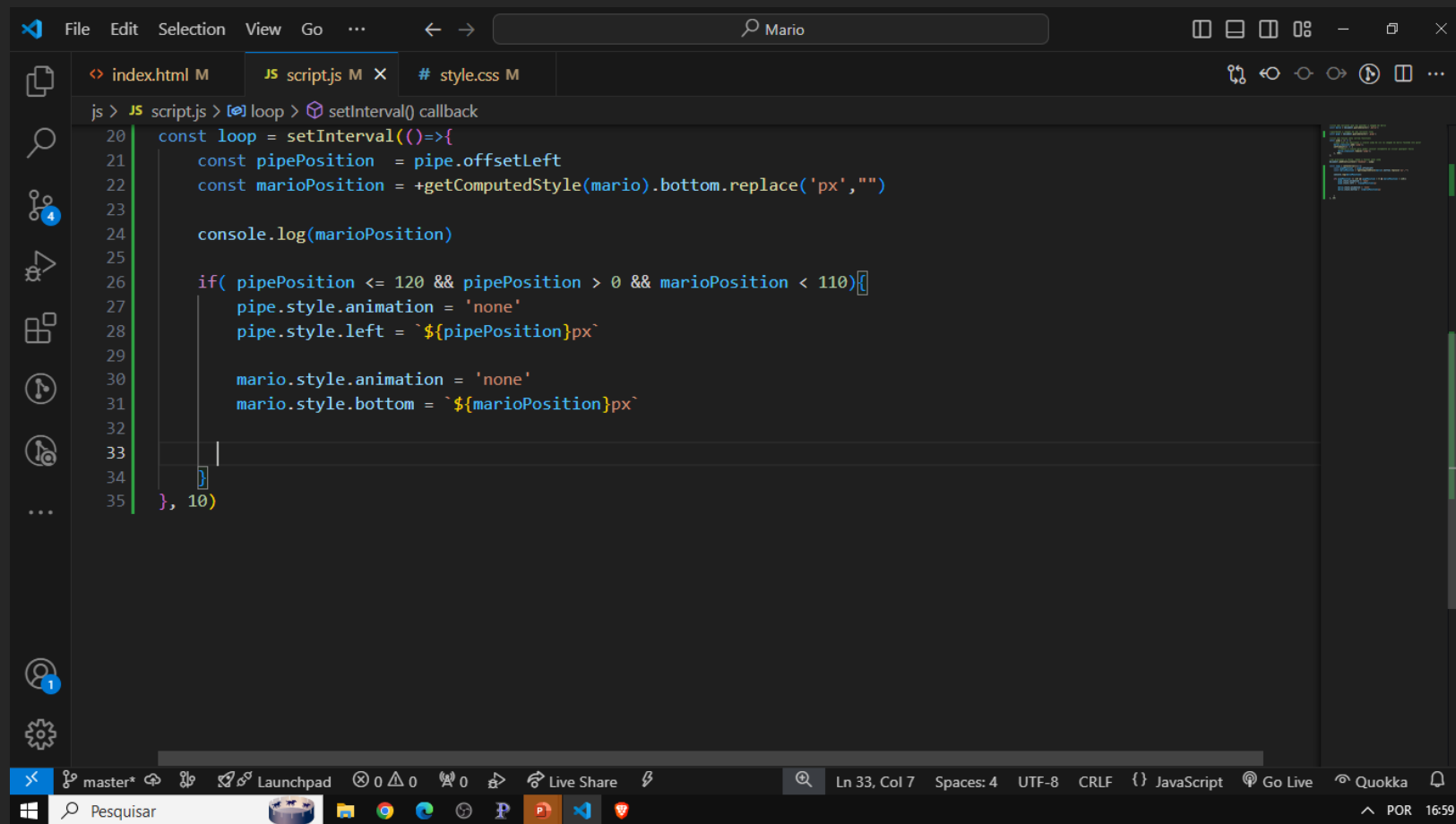
Mas podemos fazer uma coisa mais legal aqui.

Quando o mario bate no tubo, mesmo assim ele continua caindo. E o legal eh fazer ele ficar parado onde bateu



Da mesma forma que fizemos para o pipe, vamos parar a animação do mario e definir que a propriedade bottom do mario eh a posição do momento que ele bateu no tubo.

Vamos copiar as duas linhas e colar em baixo mudando parandoe a animação e pegando a posição bottom dele Agora ele para onde bateu.

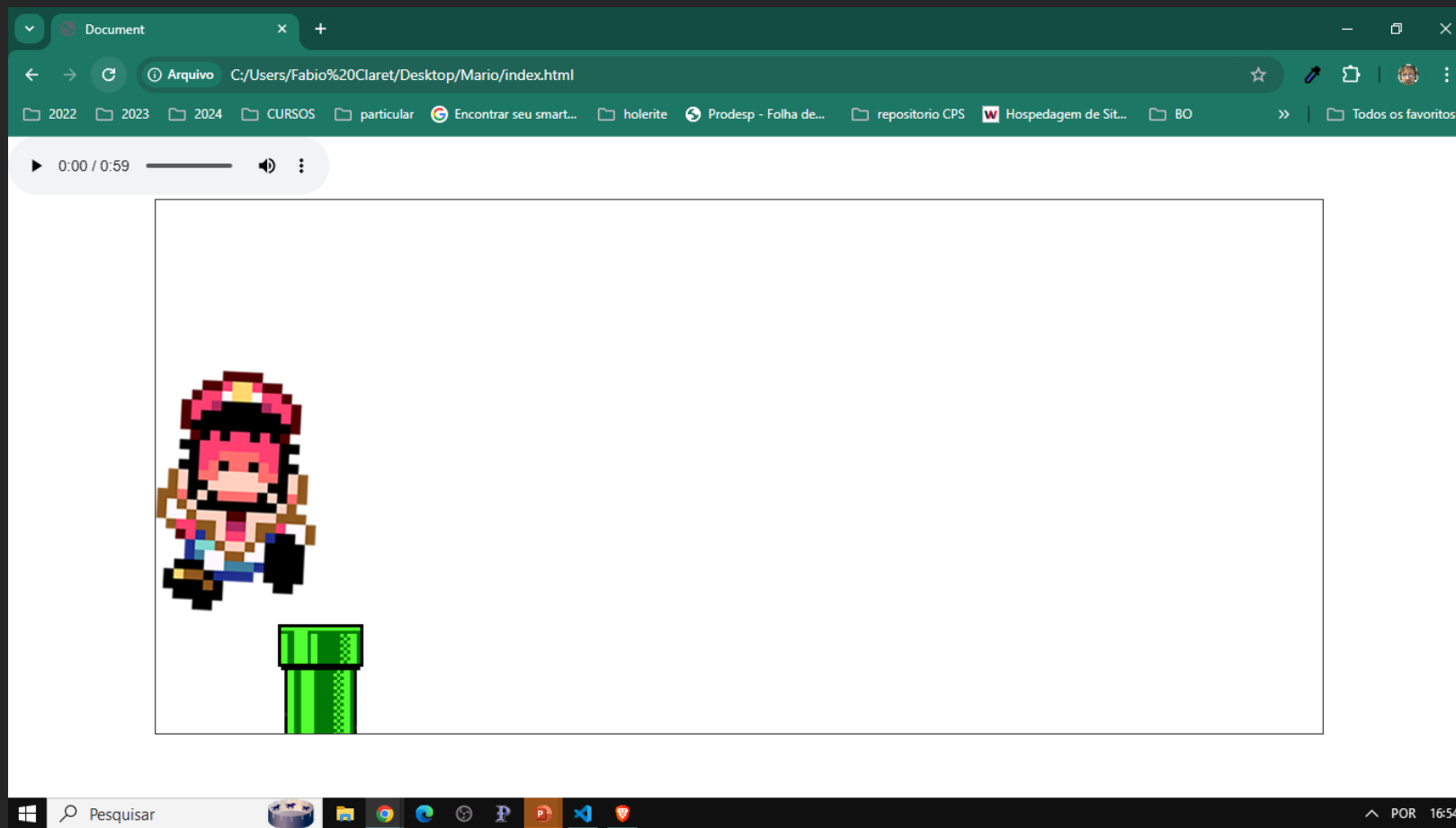


```
js > JS script.js > [1] loop > setInterval() callback
20 const loop = setInterval(()=>{
21   const pipePosition = pipe.offsetLeft
22   const marioPosition = +getComputedStyle(mario).bottom.replace('px','')
23
24   console.log(marioPosition)
25
26   if( pipePosition <= 120 && pipePosition > 0 && marioPosition < 110){
27     pipe.style.animation = 'none'
28     pipe.style.left = `${pipePosition}px`
29
30     mario.style.animation = 'none'
31     mario.style.bottom = `${marioPosition}px`
32
33   }
34 }, 10)
```

The screenshot shows a VS Code editor with three tabs: index.html, script.js, and style.css. The script.js tab is active, displaying a JavaScript loop that updates the position of a pipe and Mario. The code includes comments for the loop and the setInterval callback. The loop checks if the pipe is within a certain range and if Mario is not at the bottom, then it updates their positions and resets their animations. The status bar at the bottom shows the current line and column (Ln 33, Col 7) and the file encoding (UTF-8).

Outra coisa que podemos fazer eh mudar a imagem para a imagem do gameover. Quando o jogo acaba, eh so mudar a imagem dele.

Entao se entrar no if significa que o jogo acabou e a gente muda a imagem do mario para o game-over



Mudou a imagem. So que ele ta meio grande, então vamos mudar o style para colocar 75px

```
js > JS script.js > [0] loop > setInterval() callback
20 const loop = setInterval(()=>{
21   const pipePosition = pipe.offsetLeft
22   const marioPosition = +getComputedStyle(mario).bottom.replace('px','')
23
24   console.log(marioPosition)
25
26   if( pipePosition <= 120 && pipePosition > 0 ){
27     pipe.style.animation = 'none'
28     pipe.style.left = `${pipePosition}px`
29
30     mario.style.animation = 'none'
31     mario.style.bottom = `${marioPosition}px`
32
33     mario.src = './images/game-over.png'
34     mario.style.width = '75px'
35     mario.style.marginLeft = '50px'
36   }
37 }, 10)
```

You 7 seconds ago (October 28th, 2024 5:01 PM)

Uncommitted changes

Working Tree < >

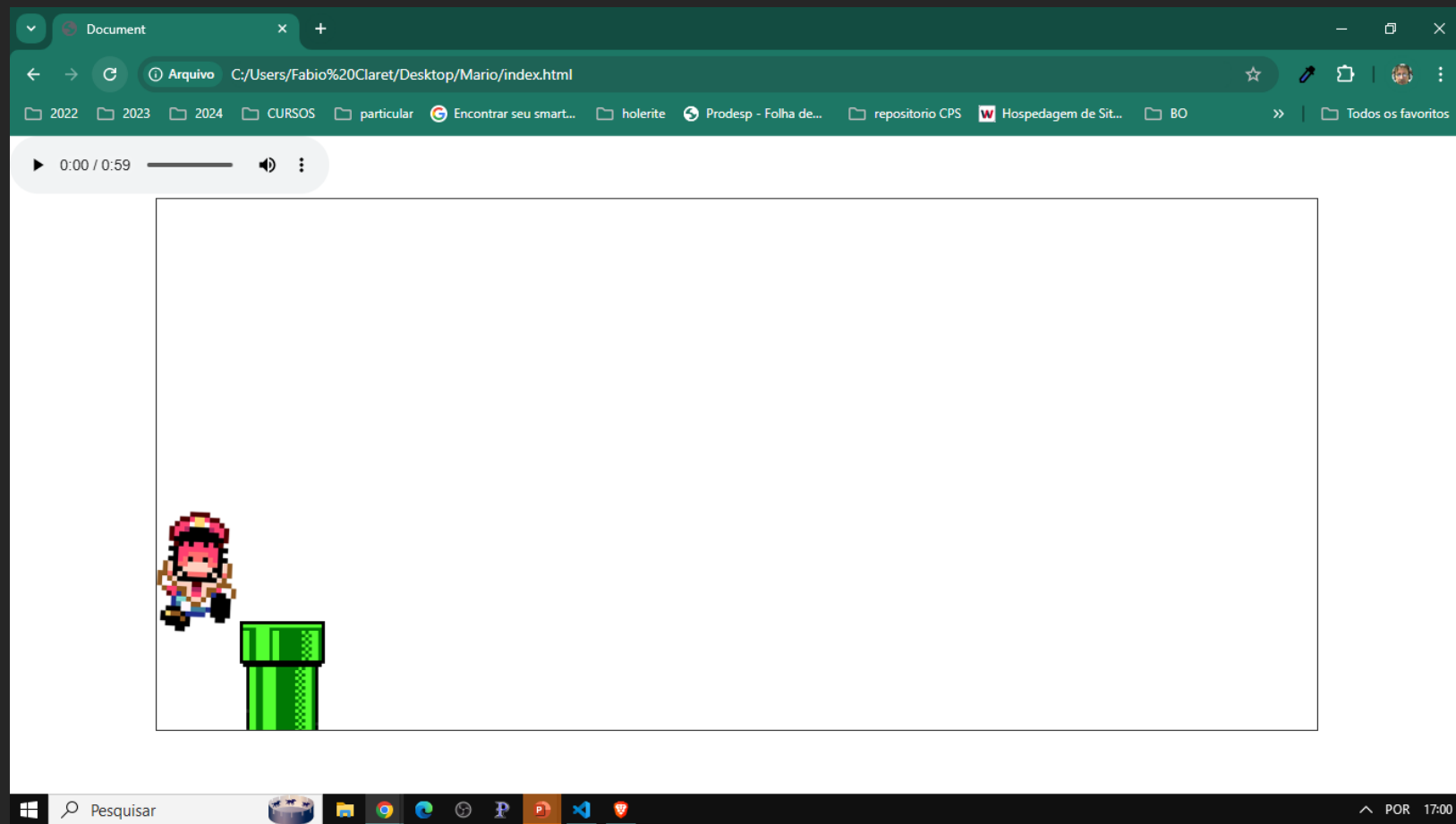
+ mario.style.marginLeft = '50px'

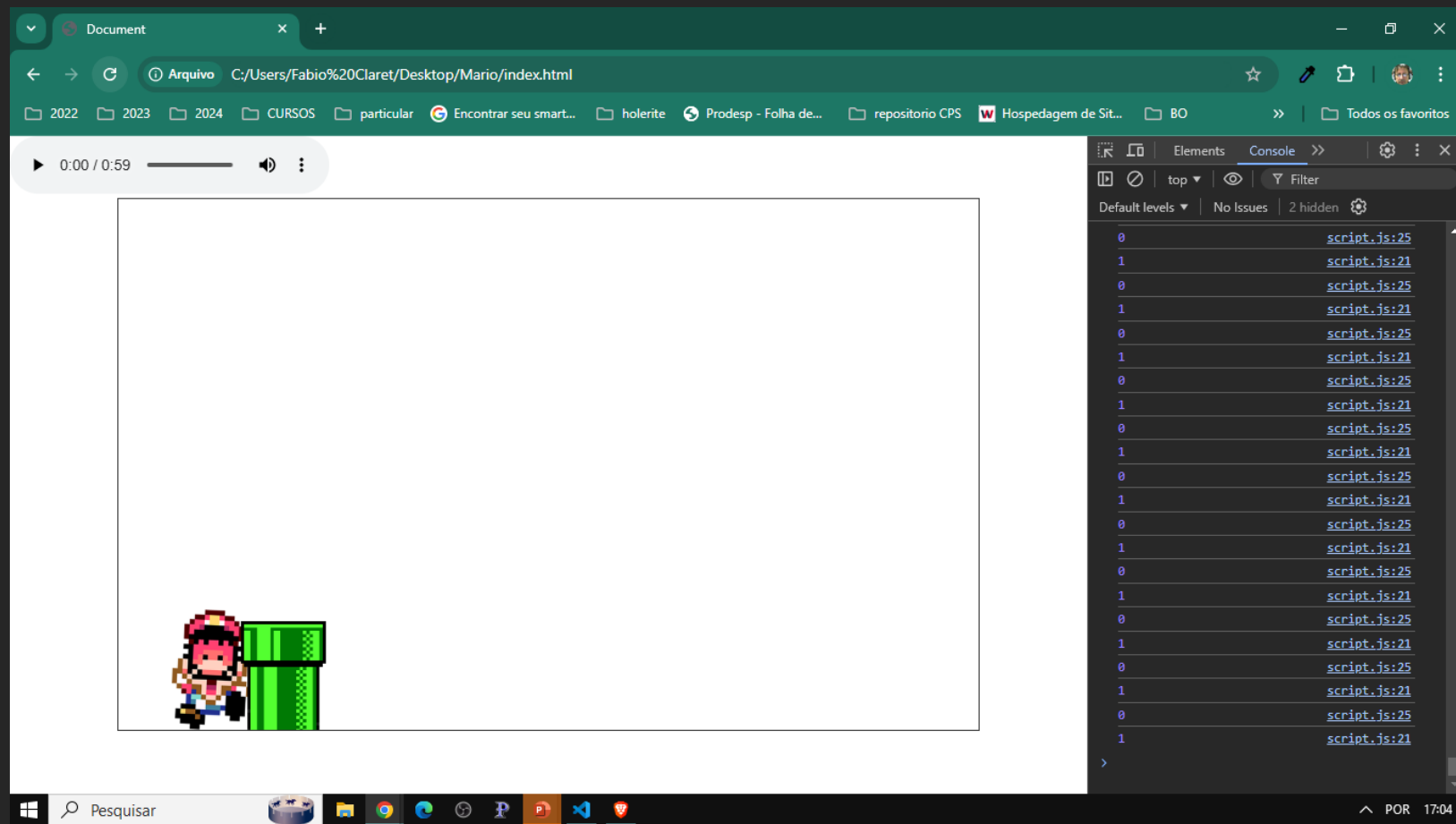
Changes 32892b3 ↔ Working Tree |

You, 6 seconds ago • Uncommitted changes

Ln 35, Col 40 (114 selected) Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Quokka

Vamos alterar a imagem para o gameover, diminuir para 75px e tirar a margem esquerda para 50px

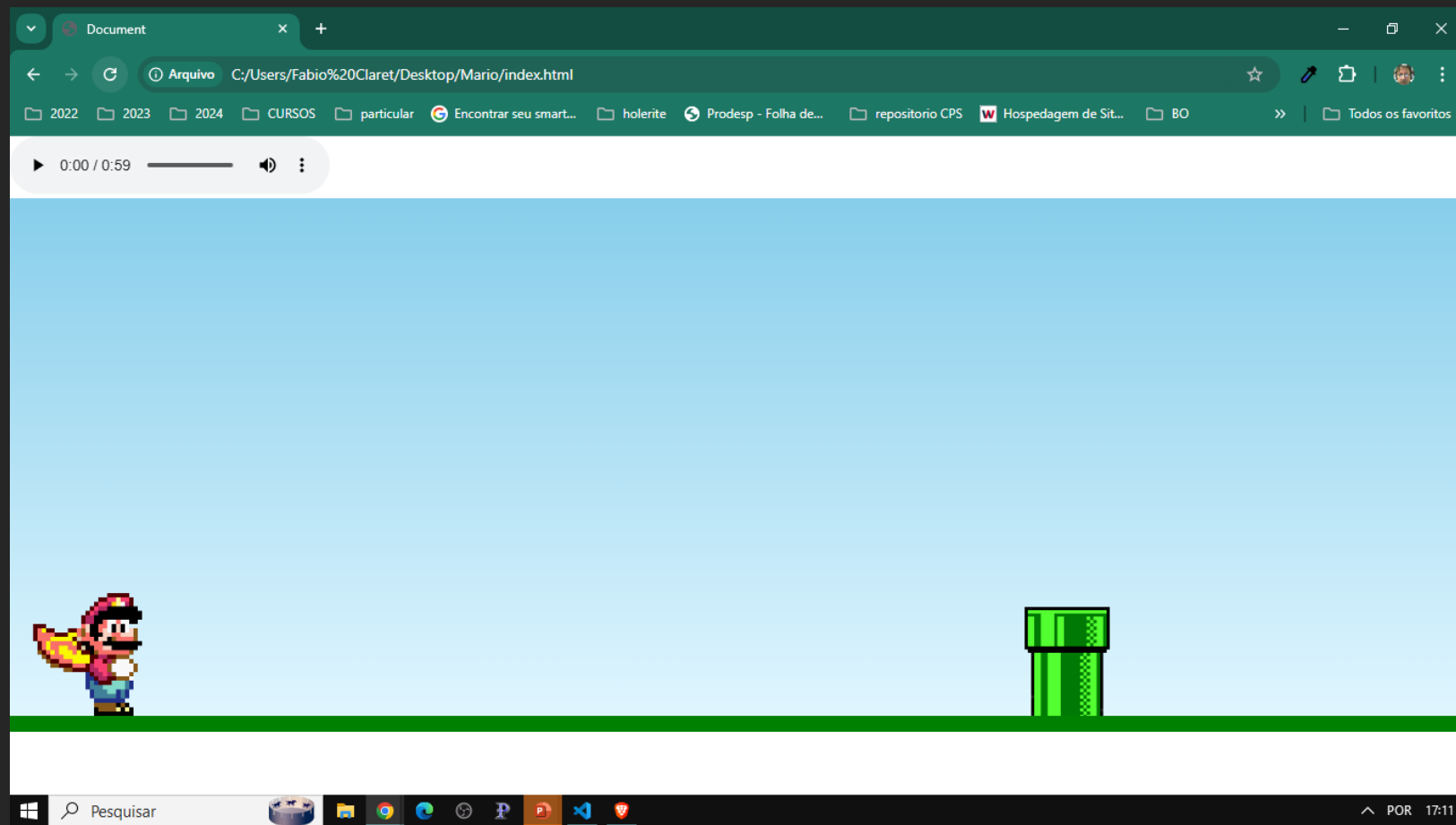




Uma coisa importante, é a gente parar o loop, pois mesmo com jogo terminado, ele continua rodando o loop, como podemos ver se colocarmos um `console.log(loop)`

```
19  
20 const loop = setInterval(()=>{  
21   console.log(loop)  
22   const pipePosition = pipe.offsetLeft  
23   const marioPosition = +getComputedStyle(mario).bottom.replace('px','')  
24  
25   console.log(marioPosition)  
26  
27  
28   if( pipePosition <= 120 && pipePosition > 0 && marioPosition < 110){  
29     pipe.style.animation = 'none'  
30     pipe.style.left = `${pipePosition}px`  
31  
32     mario.style.animation = 'none'  
33     mario.style.bottom = `${marioPosition}px`  
34  
35     mario.src = './images/game-over.png'  
36     mario.style.width = '75px'  
37     mario.style.marginLeft = '50px'  
38  
39     clearInterval(loop)  
40   }  
41 }, 10)
```

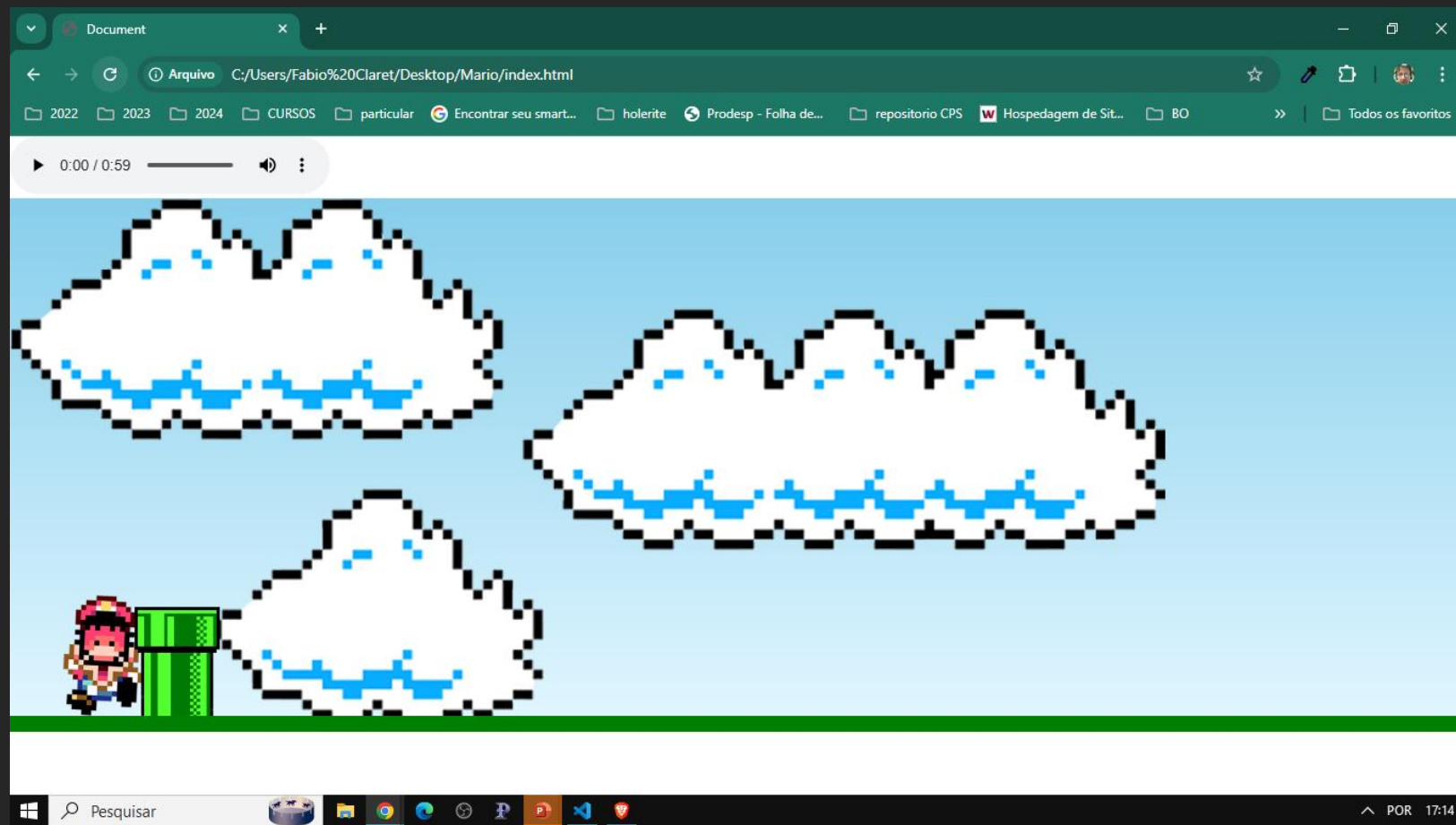
Para parar o o loop
usamos o clearInterval
passando o loop para
ele



Agora vamos deixar o tamanho da tela grande, e colocar uma borda como se fosse uma grama, e colocar um background para ficar como um céu.

```
1 <!DOCTYPE html>
2 <head>
3   <meta charset="UTF-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5   <script defer src="js/script.js"></script>
6   <link rel="stylesheet" href="css/style.css">
7   <title>Document</title>
8 </head>
9 <body>
10
11 </body>
12 <audio src="sound/mario.mp3" controls="controls"></audio>
13   <div class="game-board">
14     
15     
16     
17   </div>
18 </html>
```

Por fim, vamos colocar
umas nuvens

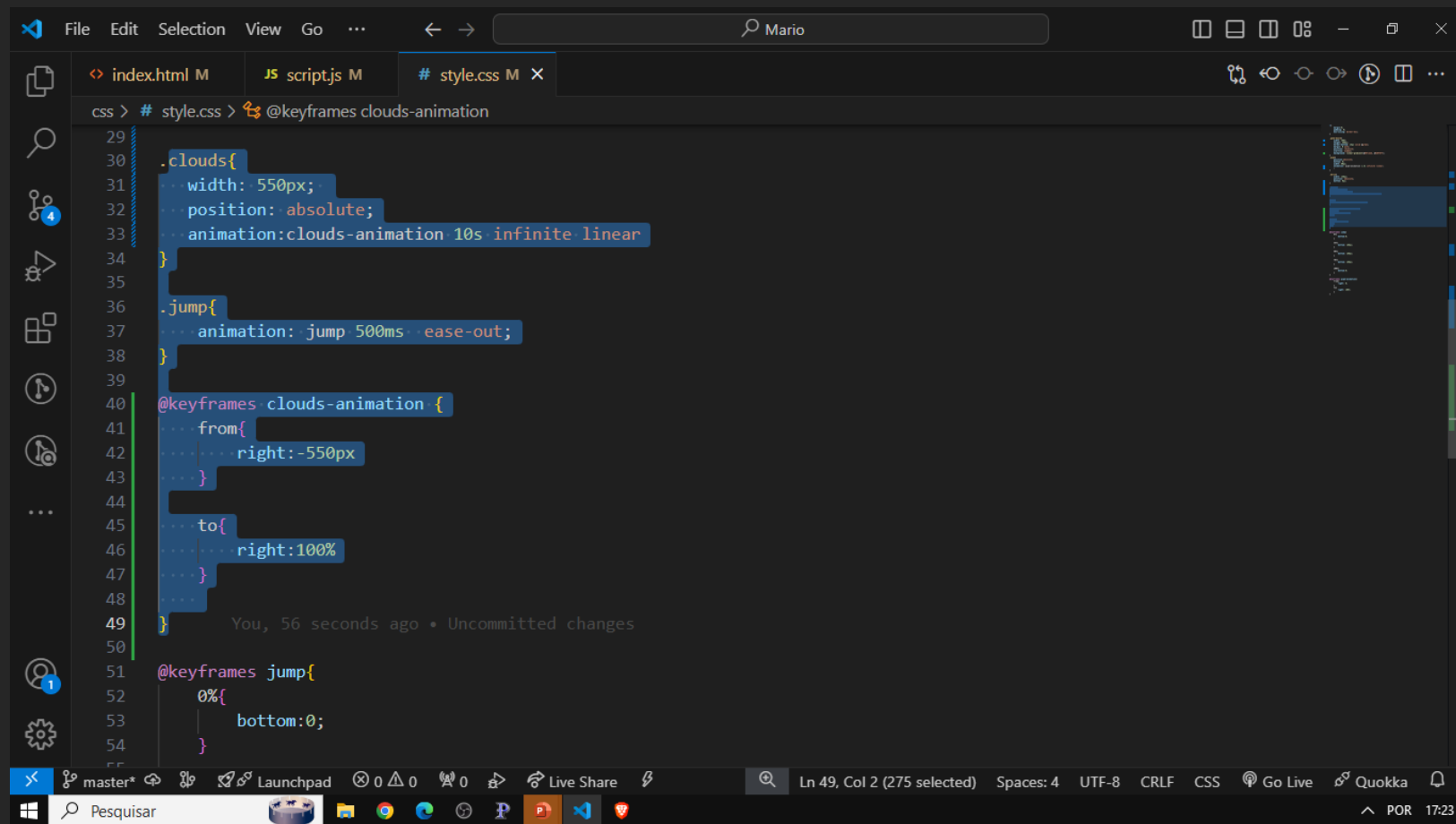


As nuvens ficaram meio grandes, vamos diminuir o tamanho delas lá no CSS

```
css > # style.css > .clouds
7  .game-board{
14 |     background: linear-gradient(■ #87ceeb, ■ #e0f6ff);
15 | }
16 | .pipe{
17 |     position: absolute;
18 |     bottom: 0;
19 |     width: 80px;
20 |     animation: pipe-animation 1.5s infinite linear;
21 |     ;
22 | }
23 |
24 | .mario{
25 |     width: 150px;
26 |     position: absolute;
27 |     bottom: 0px;
28 | }
29 |
30 | .clouds{
31 |     width: 550px;
32 |     position: absolute;
33 | }
34 |
35 | .jump{
36 |     animation: jump 500ms ease-out;
37 | }
38 |
```

Ln 32, Col 23 Spaces: 4 UTF-8 CRLF CSS Go Live Quokka

Agora so falta fazer uma animação aqui nas nuvens também.



The image shows a screenshot of the Visual Studio Code editor interface. The editor is open to a file named `style.css`. The code defines a CSS animation for clouds. The `.clouds` class is styled with `width: 550px;`, `position: absolute;`, and `animation: clouds-animation 10s infinite linear`. The `.jump` class is styled with `animation: jump 500ms ease-out;`. The `@keyframes clouds-animation` block defines the animation from `right: -550px` to `right: 100%`. The `@keyframes jump` block defines the animation from `bottom: 0` to `bottom: 0`. The status bar at the bottom indicates the current position is `Ln 49, Col 2 (275 selected)` and the file encoding is `UTF-8`.

```
css > # style.css > @keyframes clouds-animation
29
30 .clouds{
31   width: 550px;
32   position: absolute;
33   animation: clouds-animation 10s infinite linear
34 }
35
36 .jump{
37   animation: jump 500ms ease-out;
38 }
39
40 @keyframes clouds-animation {
41   from{
42     right: -550px
43   }
44
45   to{
46     right: 100%
47   }
48 }
49 }
50
51 @keyframes jump{
52   0%{
53     bottom: 0;
54   }
55 }
```

Aqui a animação das nuvens

Animation - Os `@keyframes` são como o roteiro de uma animação em CSS. Eles permitem criar sequências de animação, especificando como um elemento deve se comportar em diferentes estágios da animação. Para utilizá-los, você define os estágios-chave da animação e descreve as mudanças de estilo que devem ocorrer em cada estágio.

```
const mario = document.querySelector('.mario') - crio uma variavel que vai guardar a imagem do mario
```

```
crio uma variavel que vai guardar a imagem do mario
```

Essa função é chamada arrow-function

Em termos simples, uma arrow function é uma forma concisa de escrever uma função em JavaScript. Ela otimiza a escrita do seu código, deixando-o mais limpo, enxuto e aumentando a legibilidade.

```
const jump = () => {  
    //aqui ele vai adicionar a classe jump do css na imagem do mario fazendo ele  
    pular  
    mario.classList.add('jump');  
    setTimeout(() => {  
        //removo a classe para poder colocar novamente ao clicar qualquer tecla  
        mario.classList.remove('jump');  
    }, 500);  
};
```

```
document.addEventListener('keydown', jump)
```

Aqui adicionamos um evento que checa se uma tecla foi pressionada, e se foi chama a função jump

`setInterval()` - permite executar uma função repetidamente em um espaço de tempo definido.

Sintaxe: `window.setTimeout('funcao()', intervalo_em_milissegundos)` vai repetir uma função a cada espaço definido em milissegundos.

```
document.addEventListener('keydown', jump)
```

Aqui adicionamos um evento que checa se uma tecla foi pressionada, e se foi chama a função jump

`setInterval()` - permite executar uma função repetidamente em um espaço de tempo definido.

Sintaxe: `window.setTimeout('funcao()', intervalo_em_milissegundos)` vai repetir uma função a cada espaço definido em milissegundos.

Desafio:

Ajuste o jogo para que quando o jogo acabar, as nuvens também parem de se mover e permaneçam na posição em que estavam.

