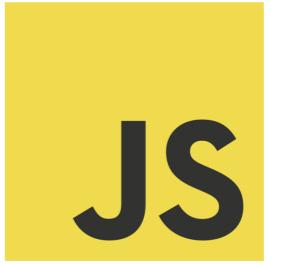


DOM



JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

- On va coder un script pour afficher une image à l'endroit où l'on click.

DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

The screenshot shows a code editor interface with three tabs: `index.html`, `main.js`, and `style.css`. The `index.html` tab displays the following HTML code:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
</body>
<script type="text/javascript" src="main.js"></script>
</html>
```

The `main.js` file is currently empty. The `style.css` file contains some placeholder styles:

```
body {
    background-color: yellow;
}
```

To the right, a browser window is open with the URL `Tuto`. The developer tools are open, showing the `Elements` tab selected. The console is empty.

JS

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT

The screenshot shows a code editor and a browser developer tools interface.

Code Editor: The file `main.js` contains the following code:

```
// Code JavaScript ici
addEventListener("click", function() {
    console.log("1 Clic");
});
```

Browser Developer Tools: The `Console` tab is active, showing the output of the `console.log` statement:

```
1 Clic
```

The browser's address bar shows `Tuto`.

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT

JS

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor: The file `main.js` contains the following code:

```
// Code JavaScript ici
addEventListener("click", function(unEvent) {
    console.log(unEvent);
});
```

Console Output: The browser's developer tools show the output of the `console.log` statement. A red oval highlights the event object properties.

```
MouseEvent {isTrusted: true, screenX: 1241, screenY: 370, clientX: 103, clientY: 91, ...}
  isTrusted: true
  screenX: 1241
  screenY: 370
  clientX: 103
  clientY: 91
  ctrlKey: false
  shiftKey: false
  altKey: false
  metaKey: false
  button: 0
```

DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
// Code JavaScript ici
addEventListener("click", function(unEvent) {
    console.log(unEvent.x, unEvent.y);
});
```

Browser Developer Tools - Console:

Line	Log	File
6	6 6	main.js:4
27	27 15	main.js:4
46	46 22	main.js:4
78	78 26	main.js:4
48	48 76	main.js:4
73	73 76	main.js:4
113	113 55	main.js:4
94	94 41	main.js:4
76	76 43	main.js:4
106	106 84	main.js:4
61	61 47	main.js:4
53	53 60	main.js:4

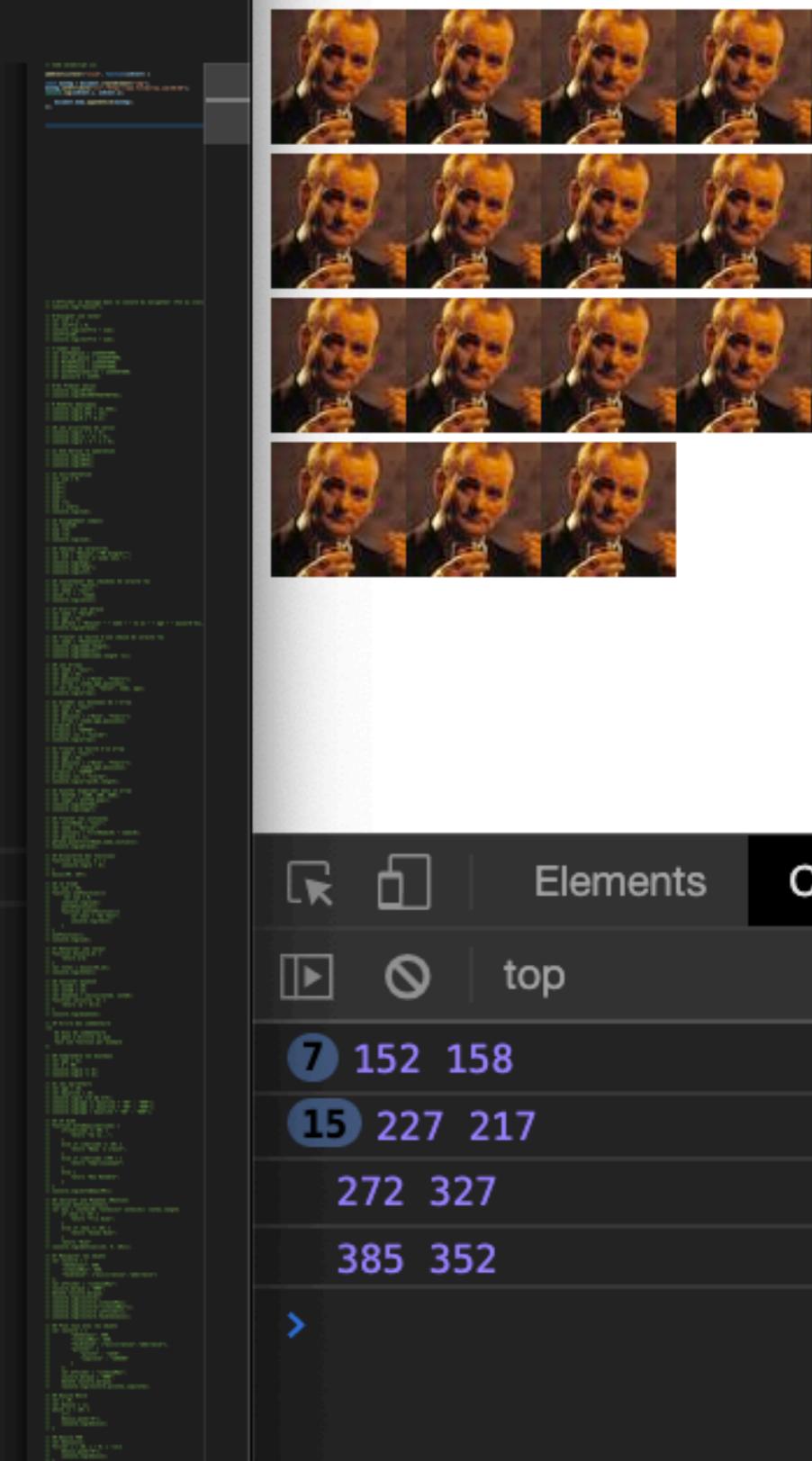
DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

main.js > ...

```
1 // Code JavaScript ici  
2  
3 addEventListener("click", function(unEvent) {  
4  
5 const monImg = document.createElement("img");  
6 monImg.setAttribute("src", "https://www.fillmurray.com/58/58");  
7 console.log(unEvent.x, unEvent.y);  
8  
9 document.body.appendChild(monImg);  
0});  
1  
2  
3  
4  
5  
6  
7  
8  
9  
0  
1  
2  
3
```



DOM

JS

EXO : AFFICHER UNE IMAGE EN CLICQUANT

JS main.js > addEventListener("click") callback

```
1 // Code JavaScript ici
2
3 addEventListener("click", function(unEvent) {
4
5 const monImg = document.createElement("img");
6 monImg.setAttribute("src","https://www.fillmurray.com/58/58");
7 // console.log(unEvent.x, unEvent.y);
8
9 monImg.style.position = "absolute";
10 monImg.style.left = event.x + "px";
11 monImg.style.top = event.y + "px";
12
13
14
15 document.body.appendChild(monImg);
16 });
17
18
19
20
21
```



DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT



The screenshot shows a browser window with a yellow status bar at the top containing the text "Tuto" and "wors". Below the status bar, the browser interface includes tabs for "index.html", "main.js", and "# style.css". The "main.js" tab is active, displaying the following JavaScript code:

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > addEventListener("click") callback
1 // Code JavaScript ici
2
3 addEventListener("click", function(unEvent) {
4
5 const monImg = document.createElement("img");
6 const taille = 44;
7 monImg.setAttribute
8 ("src", `https://www.fillmurray.com/${taille}/${taille}/`);
9
10 monImg.style.position = "absolute";
11 monImg.style.left = event.x - taille /2 + "px";
12 monImg.style.top = event.y - taille /2 + "px";
13
14
15 document.body.appendChild(monImg);
16
17});
```

The browser's developer tools are open at the bottom, showing the "Console" tab which is currently selected. The "Elements" tab is also visible. The page content area displays a large number of small, overlapping images of a man's face, indicating that the script has been triggered multiple times.

JS

DOM

EXO: INDIQUER LE % DU SCROLL

DOM

JS

EXO: INDIQUER LE % DU SCROLL

The image shows a development environment with a code editor and a browser preview.

Code Editor (index.html — base):

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <div class="boxBar">
        <div class="bar"></div>
    </div>
    <h1>Super Scroll Ultra</h1>
<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Browser Preview:

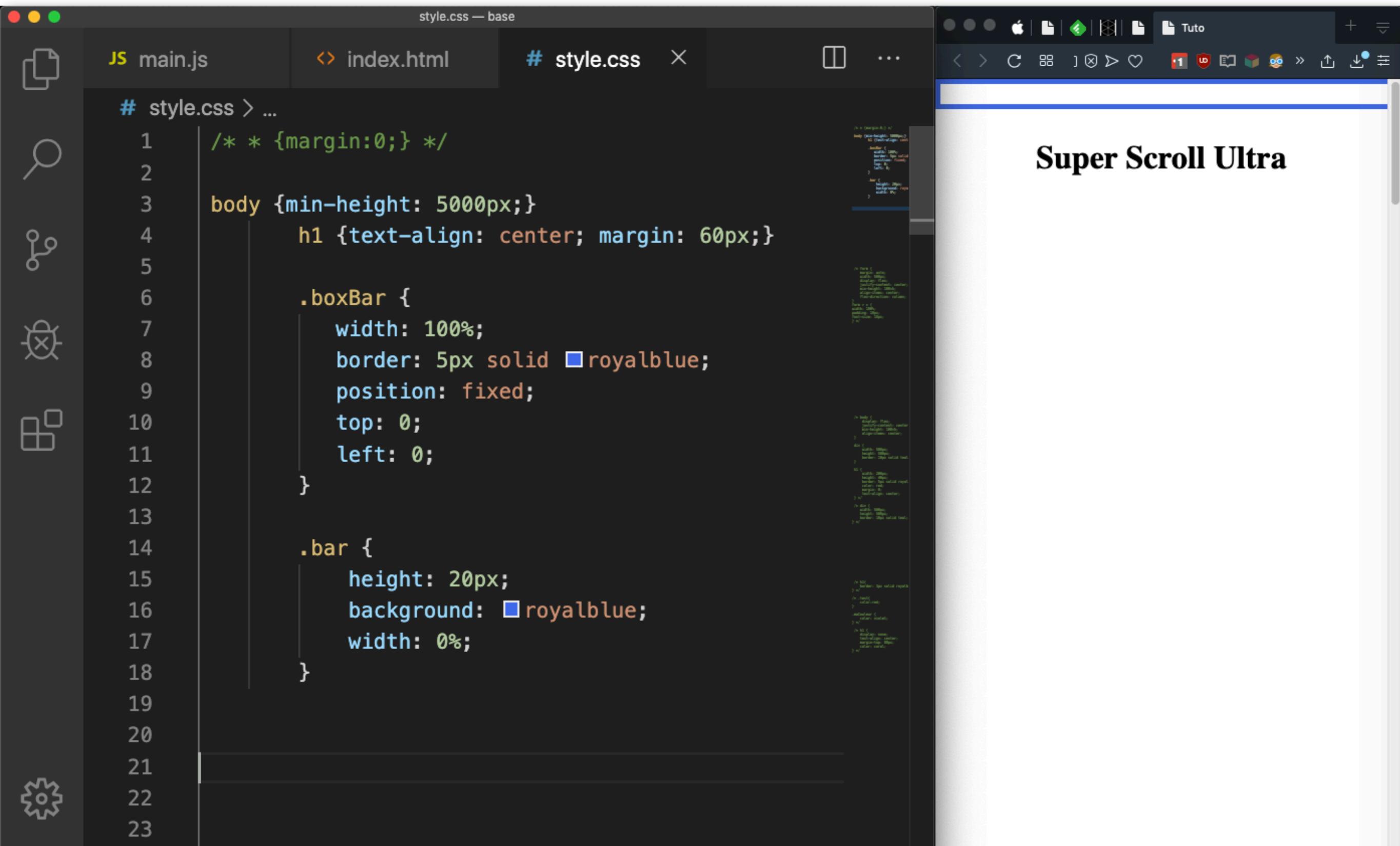
The browser window displays the following content:

Super Scroll Ultra

DOM

EXO: INDIQUER LE % DU SCROLL

JS



The screenshot shows a code editor interface with several tabs: 'main.js', 'index.html', and '# style.css'. The '# style.css' tab is active, displaying the following CSS code:

```
# style.css > ...
1  /* * {margin:0;} */
2
3  body {min-height: 5000px;}
4      h1 {text-align: center; margin: 60px;}
5
6  .boxBar {
7      width: 100%;
8      border: 5px solid royalblue;
9      position: fixed;
10     top: 0;
11     left: 0;
12 }
13
14 .bar {
15     height: 20px;
16     background: royalblue;
17     width: 0%;
18 }
```

On the right side of the editor, there is a browser window showing a white page with a blue header bar at the top. The main content area contains the text "Super Scroll Ultra". The browser's address bar shows the URL "Tuto".

DOM

JS

EXO: INDIQUER LE % DU SCROLL

The screenshot shows a browser window with developer tools open. On the left, the code editor displays `main.js` with the following script:

```
// Code JavaScript ici
const bar = document.querySelector(".bar");
addEventListner("scroll", function() {
    console.log(`Hauteur page : ${document.body.scrollHeight}
    Hauteur affichage : ${innerHeight}
    Scroll Position : ${pageYOffset}`);
})
```

The browser's console tab shows the output of the script as the page is scrolled:

- Initial scroll (top):
Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 748
- Intermediate scroll:
Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 749
- Final scroll (bottom):
Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 750
- Final scroll (bottom):
Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 751

The browser's address bar shows the URL `Tuto`.

DOM

JS

EXO: INDIQUER LE % DU SCROLL

```
main.js > ⚡ addEventListener("scroll") callback
```

```
// Code JavaScript ici
```

```
const bar = document.querySelector(".bar");
```

```
addEventListener("scroll", function() {
```

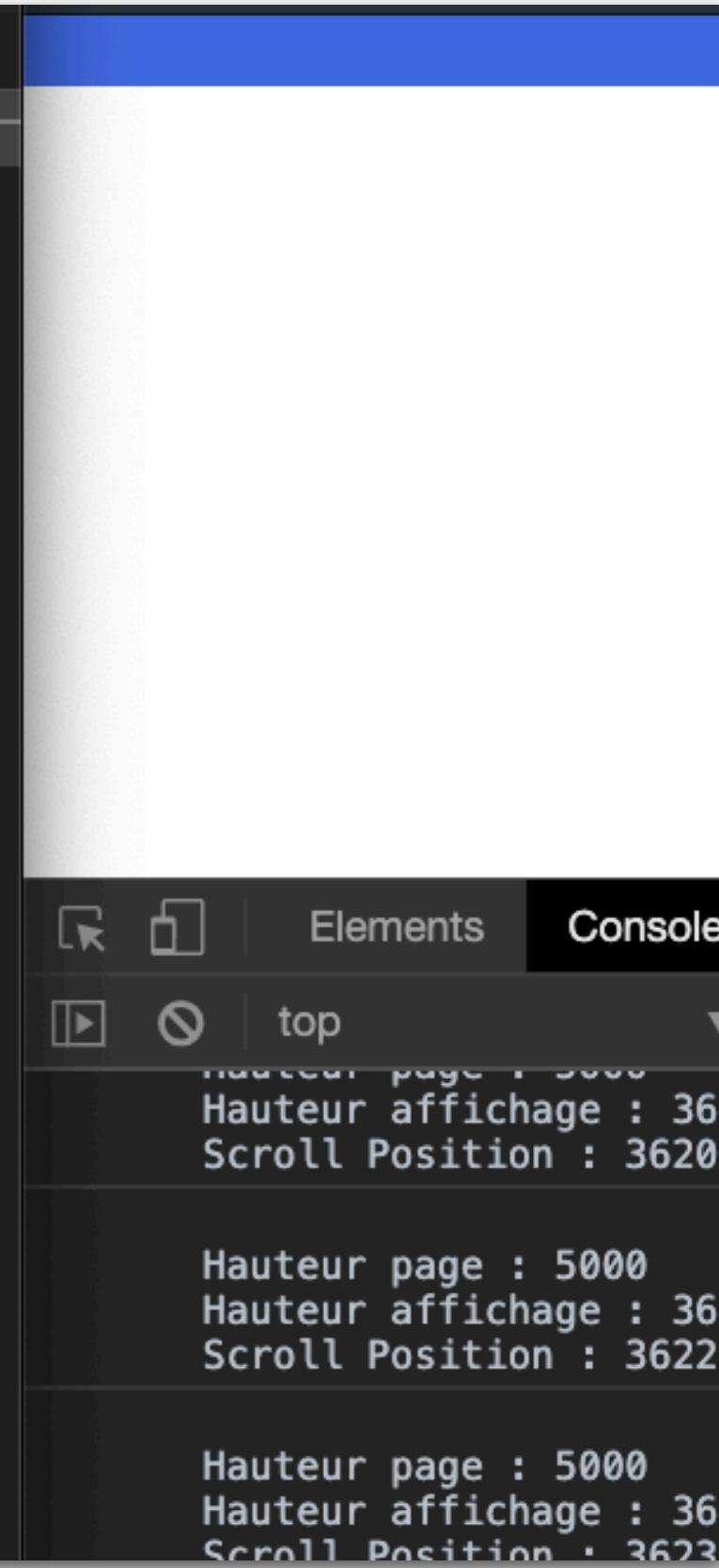
```
    const scrollMax = document.body.scrollHeight - innerHeight;
```

```
    const onEstOu = pageYOffset / scrollMax * 100;
```

```
    bar.style.width = onEstOu + "%";
```

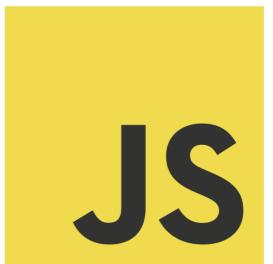
```
    console.log(`  
        Hauteur page : ${document.body.scrollHeight}  
        Hauteur affichage : ${innerHeight}  
        Scroll Position : ${pageYOffset}`);
```

```
}
```



DOM

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT



- Un nouvel event
- load
- Permet de faire une action uniquement après qu'un élément est chargé.
 - On met le load sur un élément en particulier
 - ex : images trop lourdes sur un site on veut pouvoir utiliser la page avant que les images soit totalement chargées.
 - On met le load sur l'ensemble de la page pour exécuter notre script une fois que toute la page soit chargée

DOM

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT

```
<meta charset="UTF-8">
<title>Tuto</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<!-- Code HTML -->












<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

DOM

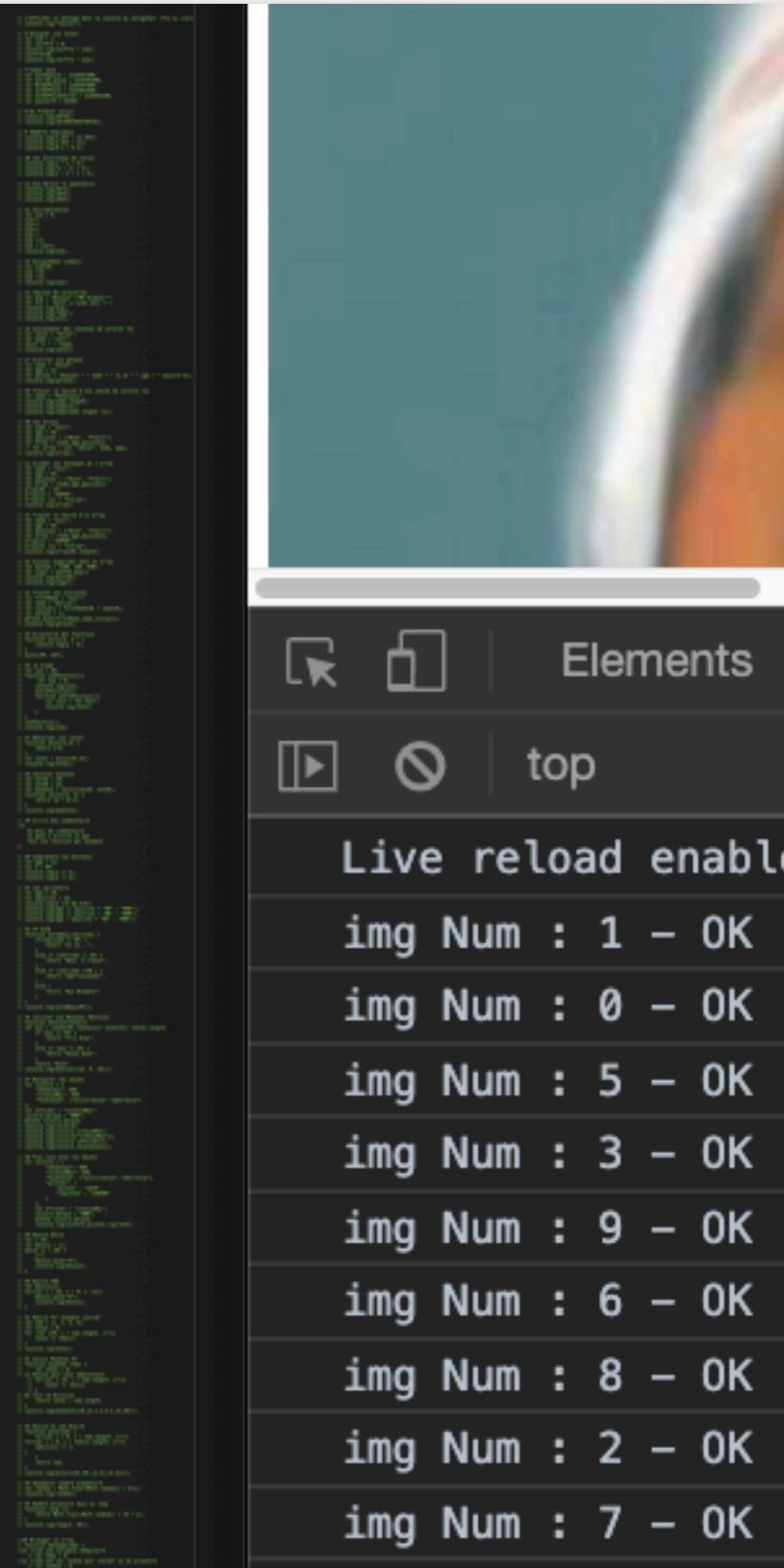
JS

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT

```
const images = document.querySelectorAll("img");

const tabImg = Array.from(images);

tabImg.map((image,i) => image.addEventListener("load",
function(){
    console.log(`img Num : ${i} - OK`);
}))
```



DOM



JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

- L'utilisateur tape du texte
- On le récupère
- Et on le console.log sans les voyelles.
- Array includes
- Array join

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a development environment with the following components:

- Code Editor:** The left pane displays the `index.html` file with the following content:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
<textarea cols="50" rows="10"></textarea>
<script type="text/javascript" src="main.js"></script>
</body>
</html>
```
- Browser Preview:** The right pane shows a blank white page, indicating the current state of the application.
- Toolbar:** On the far left, there is a vertical toolbar with icons for file operations (New, Open, Save, Find, etc.), search, and settings.
- Bottom Bar:** At the bottom of the editor, there is a navigation bar with tabs like Elements, Console, and others, along with various icons for file operations.

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a code editor interface with several tabs:

- JS main.js
- index.html
- # style.css (selected)

The style.css file contains the following CSS rules:

```
# style.css > ...
1
2
3  textarea {
4      display: block;
5      margin: 60px auto;
6  }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

The browser window shows a simple text area centered on the page.

The browser's developer tools are open, showing the Elements and Console tabs. The Console tab is active and displays a single arrow symbol (>).

DOM

JS

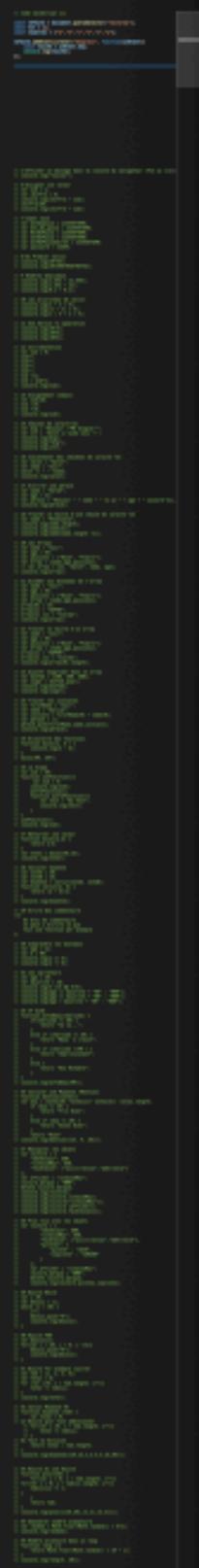
EXO: SUPPRIMER LES VOYELLES DU CLAVIER

main.js > ...

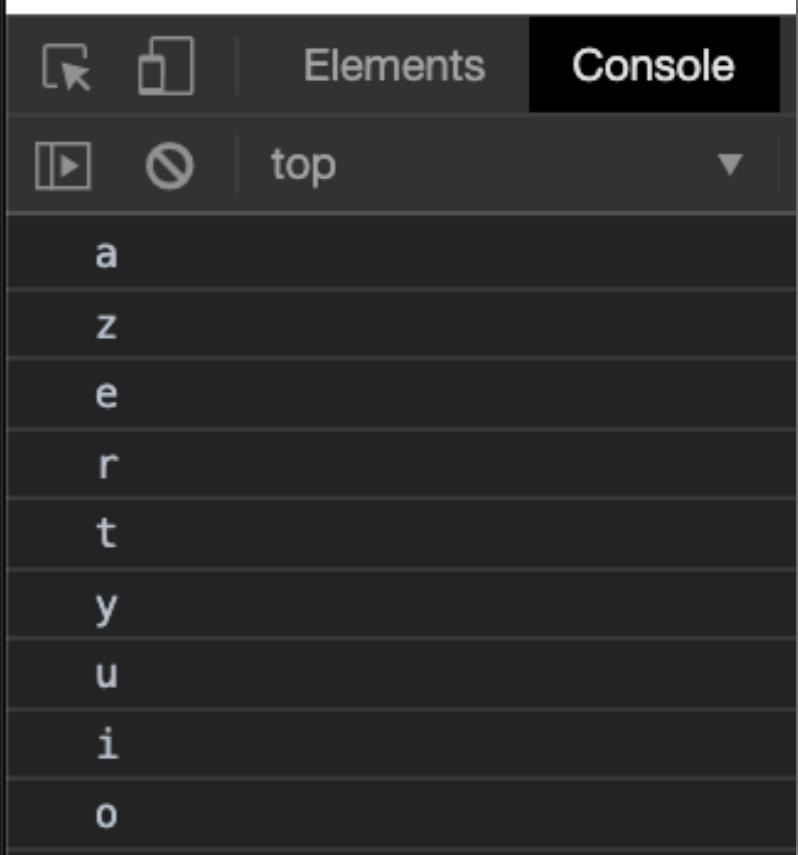
```
// Code JavaScript ici

const leTexte = document.querySelector("textarea");
const txt = [];
const voyelles = ["a","e","i","o","u","y"];

leTexte.addEventListener("keypress", function(unEvent){
    const touche = unEvent.key;
    console.log(touche);
});
```



azertyuiopl



DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
main.js — base
JS main.js    X   index.html   # style.css
JS main.js > leTexte.addEventListener("keypress") callback
1 // Code JavaScript ici
2
3 const leTexte = document.querySelector("textarea");
4 const txt = [];
5 const voyelles = ["a","e","i","o","u","y"];
6
7 leTexte.addEventListener("keypress", function(unEvent){
8     const touche = unEvent.key;
9
10    if(voyelles.includes(touche)) {
11        txt.push(touche);
12    }
13    console.log(txt);
14});
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Browser Developer Tools - Console:

- Elements tab is active.
- Console tab is active.
- Output:
 - ▶ ["a"] main.js:13
 - ▶ ["a"] main.js:13
 - ▶ (2) ["a", "e"] main.js:13
 - ▶ (2) ["a", "e"] main.js:13
 - ▶ (2) ["a", "e"] main.js:13
 - ▶ (3) ["a", "e", "y"] main.js:13
 - ▶ (4) ["a", "e", "y", "u"] main.js:13
 - ▶ (5) ["a", "e", "y", "u", "i"] main.js:13
 - ▶ (6) ["a", "e", "y", "u", "i", "o"] main.js:13

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The image shows a development environment with a code editor and a browser. The code editor on the left has tabs for main.js, index.html, and style.css. The main.js file contains the following code:

```
main.js — base
JS main.js    X   ↗ index.html   # style.css
JS main.js > leTexte.addEventListener("keypress") callback
1 // Code JavaScript ici
2
3 const leTexte = document.querySelector("textarea");
4 const txt = [];
5 const voyelles = ["a","e","i","o","u","y"];
6
7 leTexte.addEventListener("keypress", function(unEvent){
8     const touche = unEvent.key;
9
10    if(!voyelles.includes(touche)) {
11        txt.push(touche);
12    }
13    console.log(txt);
14});
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

The browser window on the right shows a text area containing "azertyuiopl". The developer tools' Console tab is open, displaying the following log entries:

Log	File	Line
▶ []		main.js:13
▶ ["z"]		main.js:13
▶ ["z"]		main.js:13
▶ (2) ["z", "r"]		main.js:13
▶ (3) ["z", "r", "t"]		main.js:13
▶ (3) ["z", "r", "t"]		main.js:13
▶ (3) ["z", "r", "t"]		main.js:13
▶ (3) ["z", "r", "t"]		main.js:13
▶ (4) ["z", "r", "t", "p"]		main.js:13

DOM

JS

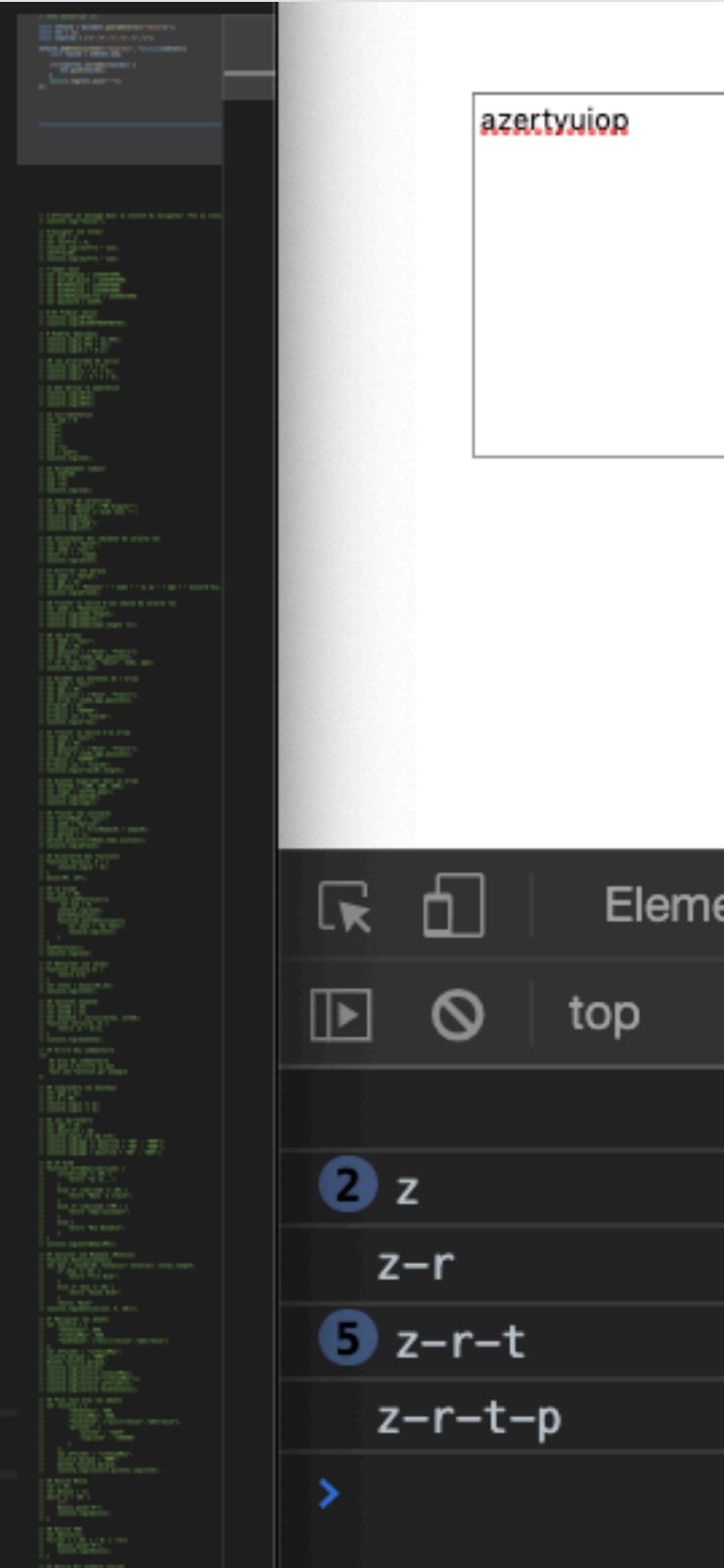
EXO: SUPPRIMER LES VOYELLES DU CLAVIER

// Code JavaScript ici

```
const leTexte = document.querySelector("textarea");
const txt = [];
const voyelles = ["a","e","i","o","u","y"];

leTexte.addEventListener("keypress", function(unEvent){
    const touche = unEvent.key;

    if(!voyelles.includes(touche)) {
        txt.push(touche);
    }
    console.log(txt.join("-"));
});
```



DOM PREVENTDEFAULT

JS

JS

DOM PREVENTDEFAULT

The image shows a development environment with a code editor and a browser. On the left, a code editor displays two files: `index.html` and `main.js`. The `index.html` file contains an HTML form with a text input placeholder "Votre nom" and a submit button labeled "Connexion". A script tag includes a `src="main.js"` attribute. The `main.js` file is currently empty. On the right, a browser window shows the rendered HTML form. Below the browser is a developer tools interface with tabs for "Elements" and "Console".

```
index.html — base
JS main.js    < index.html > ...    # style.css
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Tuto</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<!-- Code HTML -->
<form>
<input type="text" placeholder=" Votre nom"/>
<button type="submit">Connexion</button>
</form>
<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Votre nom

Connexion

Elements Console

top

DOM

PREVENTDEFAULT

JS

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The left pane displays a file named `style.css` with the following CSS code:

```
1
2  form {
3      margin: auto;
4      width: 500px;
5      display: flex;
6      justify-content: center;
7      min-height: 100vh;
8      align-items: center;
9      flex-direction: column;
10 }
11 form > * {
12     width: 100%;
13     padding: 10px;
14     font-size: 18px;
15 }
```

Browser: The right pane shows a simple login form with two fields: "Votre nom" and "Connexion". The "Connexion" field has a red border, indicating it is the currently selected or focused input.

Developer Tools: A sidebar on the right contains the following elements:

- Elements tab (selected)
- Console tab
- File icon
- Search icon
- Settings icon

The console area at the bottom of the tools shows the command `>`.

JS

DOM PREVENTDEFAULT

The screenshot shows a developer's workspace with a code editor and a browser window.

Code Editor: The file `main.js` contains the following code:

```
// Code JavaScript ici
const monForm = document.querySelector("form");
monForm.addEventListener("submit", function(){
    console.log("OK");
})
```

Browser Output: The browser window displays a simple form with a text input field labeled "Votre nom" and a button labeled "Connexion". The browser's developer tools are open, showing the "Console" tab which contains numerous log entries from the executed JavaScript code.

DOM PREVENTDEFAULT

JS

The image shows a developer setup with a code editor, a browser, and developer tools.

Code Editor: The file `main.js` contains the following code:

```
// Code JavaScript ici
const monForm = document.querySelector("form");

monForm.addEventListener("submit", function(event){
    event.preventDefault();
    console.log("OK");
})
```

Browser: A simple form with fields for "Nom" (azertyuiop) and "Prenom" (Connexion) is displayed. The "Nom" field has a blue border, indicating it is focused or selected.

Developer Tools: The browser's developer tools are open, showing the following log output in the Console tab:

```
azertyuiop
Connexion
OK
main.js:8
```

The developer tools also show the Element tab with the form element selected, and the Network tab showing various requests and responses.

DOM

PREVENTDEFAULT

JS

The image shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
main.js — base
JS main.js    X   index.html  # style.css
JS main.js > ⚡ monForm.addEventListener("submit") callback
1 // Code JavaScript ici
2
3
4 const monForm = document.querySelector("form");
5
6 monForm.addEventListener("submit", function(event){
7     event.preventDefault();
8     console.log("OK");
9     monForm.reset();
10})
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Browser Developer Tools:

- Elements:** Shows a simple form with an input field labeled "Votre nom" and a button labeled "Connexion".
- Console:** Shows the output of the JavaScript code:

```
Votre nom
Connexion
OK
main.js:8
```