ENSF 381
Full Stack Web Development

Lecture 14: JavaScript Functions

Slides: Ahmad Abdellatif, PhD

Instructor: Novarun Deb, PhD

UNIVERSITY OF CALGARY

Outline

- Functions in JavaScript.
- Default Parameters.
- Function Expression.
- Arrow Functions.
- Callback Functions.

What are functions in JavaScript?

- Functions in JavaScript allow developers to structure their code in a more organized and modular manner.
- Enable abstraction by allowing developers to encapsulate a set of operations behind a single function.
- Can be reused across different parts of a program.
- Facilitate collaborative development by breaking down a large project into smaller, more manageable tasks.

Function - syntax

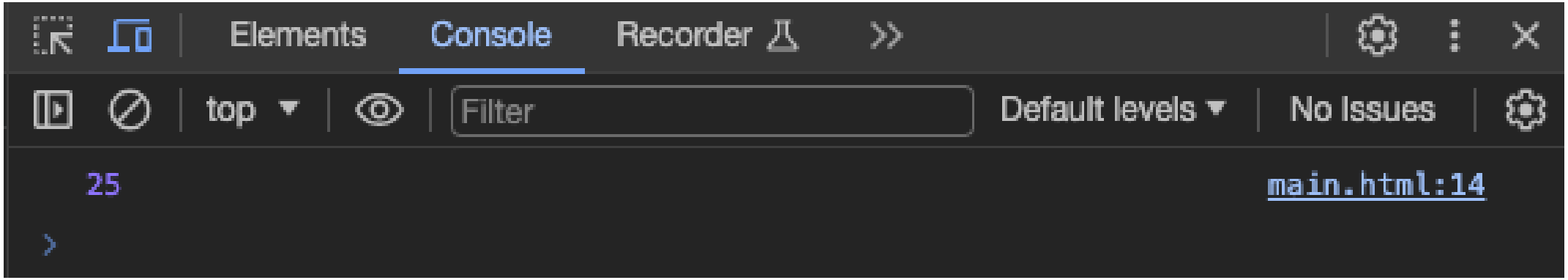
```
function functionName(parameters) {
    // Function body
}

//Call the function
functionName(parameters)
```

- Functions in JavaScript are declared using the keyword `function`.
- The code inside a function is **not executed** when the function is **defined**.
- When a function is **called**, the code within it gets **executed**.

Function - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body>
   <script>
      function calculateSquare(number) {
         return number * number;
      }

      console.log(calculateSquare(5));

   </script>
</body></html>
```

Function - example



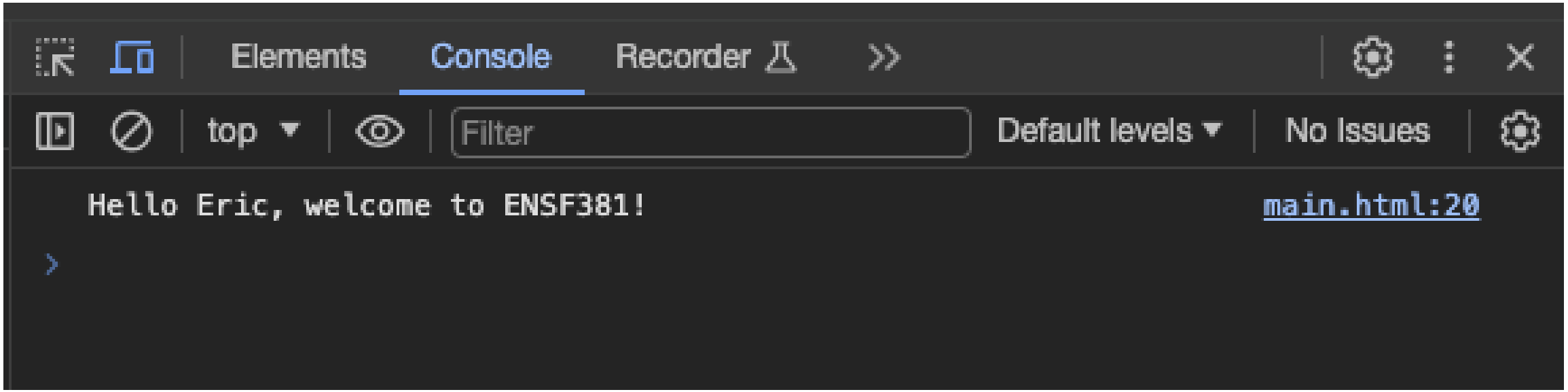
Function declarations are hoisted

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body>
     <script>

      console.log(calculateSquare(5)); //You can invoke the function before declaring it.

       function calculateSquare(number) {
         return number * number;
       }

    </script>
</body>
</html>
```

Function – example 2

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body>
    <script>

        function greetPerson(name) {
         // Using string template to create a greeting message
         const greeting = `Hello ${name}, welcome to ENSF381!`;

         // Returning the greeting
         return greeting;
        }

        let message = greetPerson('Eric');
        console.log(message);
    </script>
</body></html>
```

Function – example 2



Can we specify default values for function parameters?

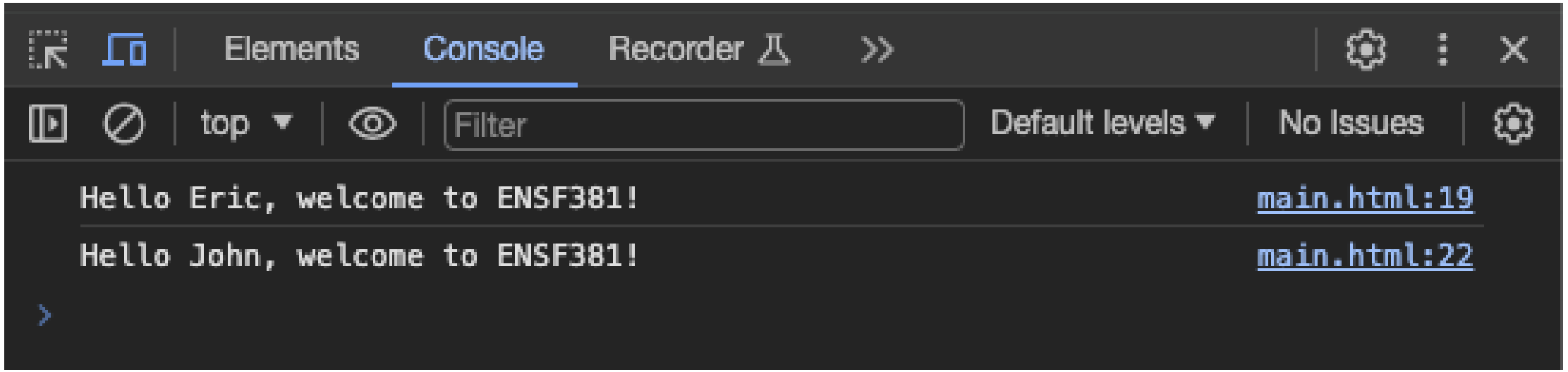
- Default parameters: A convenient way to make functions **more flexible and robust** by providing sensible default values when certain parameters are not explicitly provided during the function call.

- When a function is invoked **without certain arguments**, the default values are used instead.

Default parameters - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body> <script>

function greetPerson(name='John') {

    // Using string template to create a greeting message
    const greeting = `Hello ${name}, welcome to ENSF381!`;

    // Returning the greeting
    return greeting;
}

let message = greetPerson('Eric');
console.log(message);

message = greetPerson();
console.log(message);
    </script>
</body></html>
```

Default parameters - example



Function expression

- A way to define a function by assigning it to a variable.
- Function expression must be defined before invoking it.
- Provide flexibility in how functions are defined and used in your code:
  - Pass a function as an argument to another function.
  - Implement other advanced patterns in JavaScript (e.g., callback functions).

Function expression - syntax

```
<script>
    // Function expression
    var greet = function(name) {
      return 'Hello, ' + name + '!';
    };

    // Example usage of the function expression
    var message = greet('Eric');
    console.log(message); // Output: Hello, Eric!
</script>
```

Arrow functions

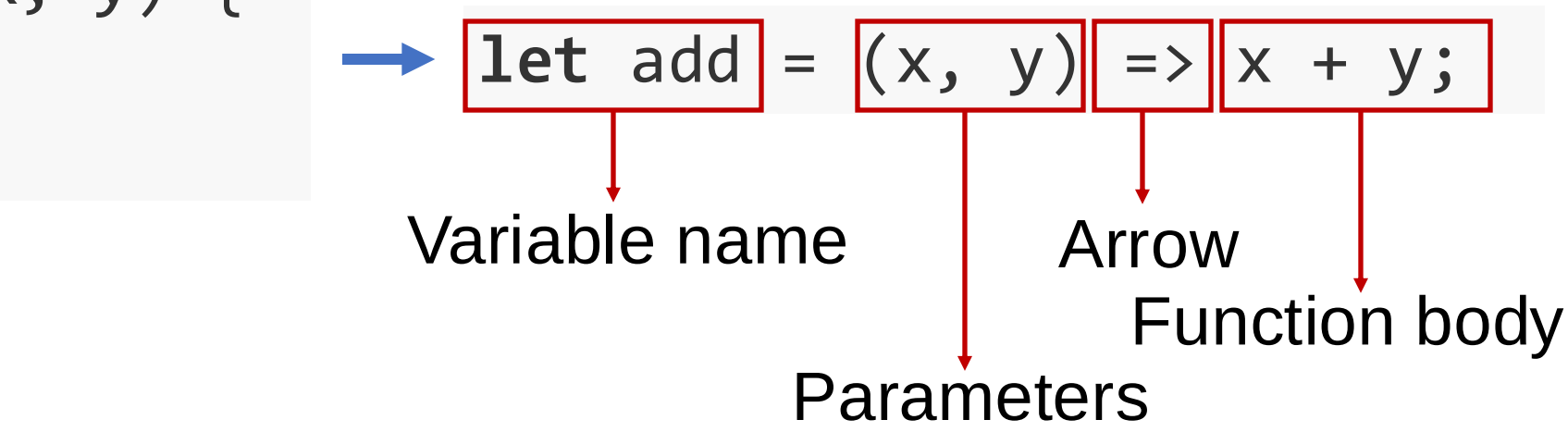
- Arrow functions are a concise and more expressive way to write anonymous function expressions in JavaScript.
- Arrow functions were introduced in ES6.
- No need to use the keyword `function` to create a function.
- If the function body is a single expression, the braces {} and the `return` keyword can be omitted.

Arrow functions - syntax

Function expression

```
let add = function(x, y) {
    return x + y;
};
```

Arrow function



Arrow functions - syntax

Traditional functions

```
function greetPerson(name='John') {
    // Using string template to create a
greeting message
    const greeting = 'Hello ${name},
welcome to ENSF381!';

    // Returning the greeting
    return greeting;
}
```

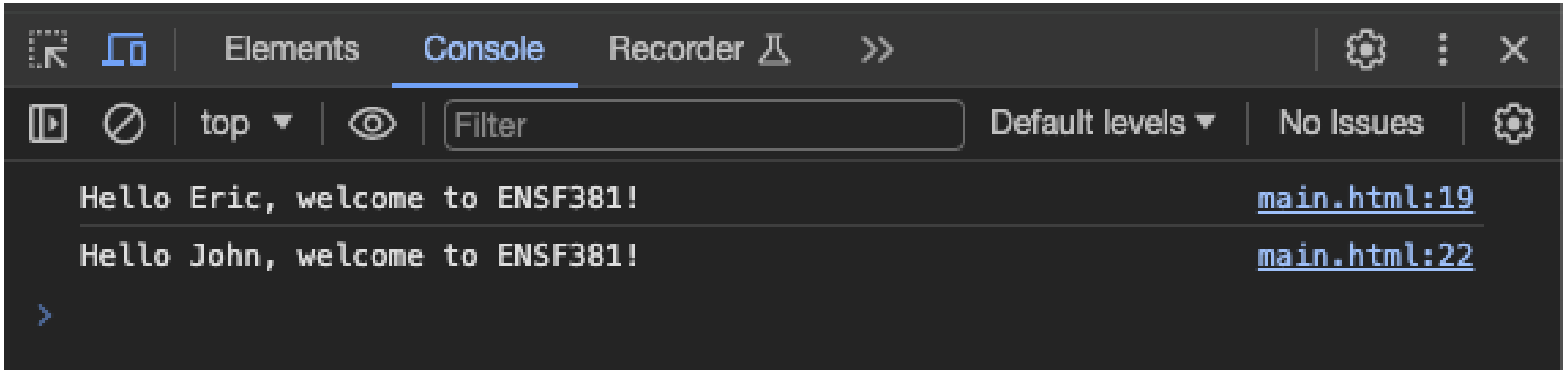
Arrow functions

```
let greetPerson = (name='John') =>
`Hello ${name}, welcome to
ENSF381!`;
```

Arrow functions - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body>

  <script>

    let greetPerson = (name='John') => `Hello ${name}, welcome to ENSF381!`;

    let message = greetPerson('Eric');
    console.log(message);

    message = greetPerson();
    console.log(message);

    </script>
</body></html>
```

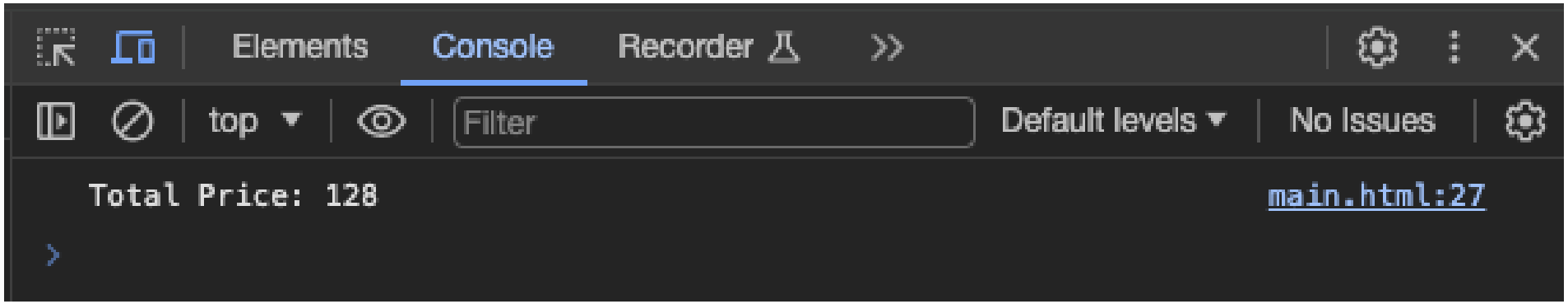
Arrow functions - example



Multiline arrow function - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body><script>

// Multiline arrow function
let calculateTotal = (price, tax) => {
    let subTotal = price + (price * tax);
    let discount = 0;

    // Apply discount if the price is above a certain threshold
    if (price > 100) {
        discount = 10;
    }

    // Calculate the total after applying the discount
    let total = subTotal - discount;
    return total;
};

let totalPrice = calculateTotal(120, 0.15);
console.log(`Total Price: ${totalPrice}`);
    </script>
</body></html>
```

Multiline arrow function - example



Callback functions

- A function passed as an argument to another function, which is then invoked or executed inside the outer function.

- Typically used to perform some action after the completion of an asynchronous operation or in response to an event.

- Executed in the order of their invocation, not in the order of their definition.

What is the output…

```
function greet(name, callback) {
  const message = "Hello, " + name + "!";
  callback(message);
}

function displayMessage(message) {
  console.log(message);
}

greet("Alice", displayMessage);
```

Output:
Hello, Alice!

Callback functions - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body> <script>
let dataSuccessfullyLoaded = function(callback) {
        console.log("Data fetched successfully.");
        callback({ data: 'Some data' });
    }
function processData(data) {
    console.log("Processing data:", data);
}

function fetchData(callback) {
    console.log("Fetching data...");
    setTimeout(dataSuccessfullyLoaded(callback), 2000);
}

fetchData(processData);
    </script>
</body>
</html>
```

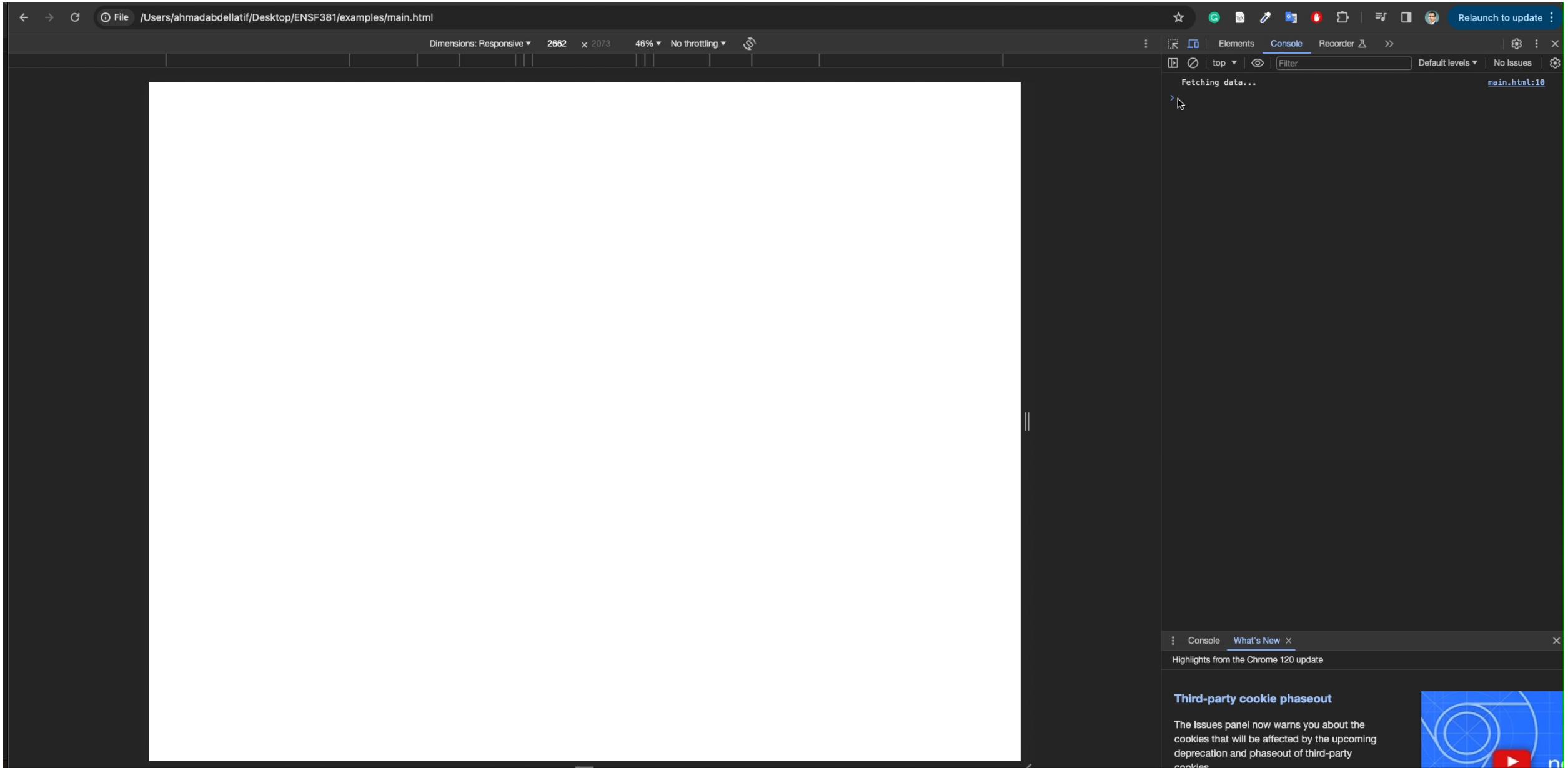
A JavaScript function that schedules the execution of a function or the evaluation of a code snippet after a specified delay (in milliseconds). The syntax is as follows:

```
setTimeout(function, delay);
```

A concise way of creating callback functions - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Function Example</title>
</head>
<body> <script>

function fetchData(callback) {
    console.log("Fetching data...");
    setTimeout(function() {
        console.log("Data fetched successfully.");
        callback({ data: 'Some data' });
    }, 2000);
}

function processData(data) {
    console.log("Processing data:", data);
}

fetchData(processData);

    </script>
</body></html>
```

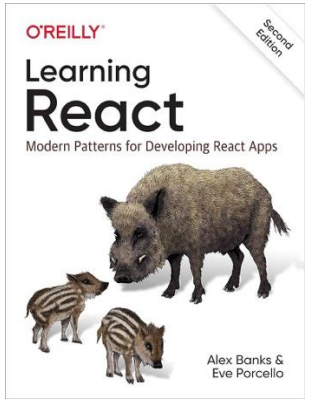
A concise way of creating callback functions - example



Questions

References

Learning React: Modern Patterns for Developing React Apps



[W3School – JavaScript Functions]

[JavaScript Info – Arrow Functions]