ENSF 381
Full Stack Web Development

Lecture 12: JavaScript

Slides: Ahmad Abdellatif, PhD

Instructor: Novarun Deb, PhD

UNIVERSITY OF CALGARY

Outline

- Introduction to JavaScript.
- JavaScript features.
- Variables.
- Conditional and looping in JavaScript.
- String.

What is JavaScript?

- High-level and dynamic programming language.
- One of the core technologies that enable interactive and dynamic content on websites.
- Used to enhance the user experience by providing features such as client-side validation, animations, and real-time updates without requiring a page reload.
- To include JavaScript in an HTML document, use the <script> tag.

Key features

- **Integration with HTML and CSS**: allowing developers to create dynamic and interactive web pages.

- **Client-Side Scripting**: mainly employed as a client-side scripting language, which means it runs in the user's web browser. However, it can also be used for server-side development (Node.js).

- **Cross-Browser Compatibility**: JavaScript is supported by most web browsers.

- **Asynchronous Programming**: designed to **handle asynchronous tasks**, such as fetching data from a server.

Variables

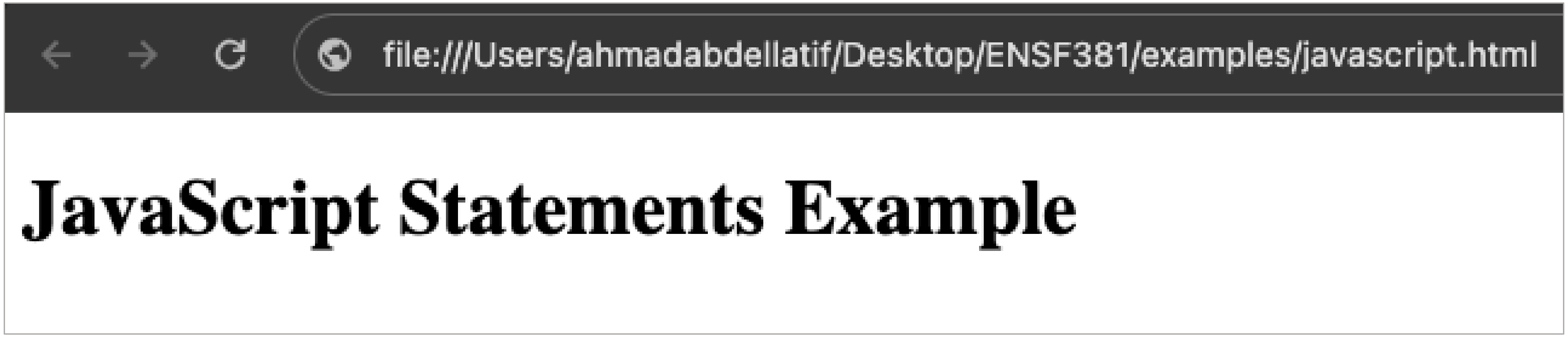
- Variables are used to store data values.
- JavaScript has several data types, including:
  - Primitive Types: String, Number, Boolean, Null, Undefined.
  - Complex Types: Object, Array, Function.
- Declare variables using:
  - var: function-scoped, meaning their scope is limited to the function in which they are declared. If declared outside any function, they **become global**.
  - let: block-scoped, meaning their scope is **limited to the block** (enclosed by curly braces) in which they are declared. Variables declared with let cannot be Redeclared in the same scope.
  - const: variables must be initialized at the time of declaration, and their **values cannot be re-assigned after initialization**.

First JavaScript example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Statements Example</title>
</head>
<body>
    <h1>JavaScript Statements Example</h1>
    <script>
        // JavaScript statements go here

        // Declaring a variable and printing it
        let message = "Hello, World!";
        console.log(message);
        // Performing a simple calculation
        let x = 5;
        let y = 10;
        let sum = x + y;
        console.log("Sum:", sum);
    </script>
</body></html>
```

First JavaScript example



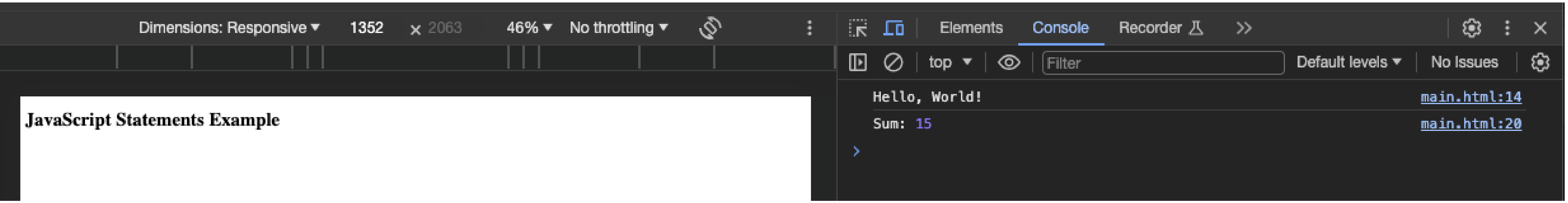
file:///Users/ahmadabdellatif/Desktop/ENSF381/examples/javascript.html

JavaScript Statements Example

Viewing the console output

- To see the example output, we need to open the Console.
- To open the Developer Console in Google Chrome, you can follow these steps:
  - **Menu Navigation**:
    - Click on the three vertical dots (ellipsis) in the top-right corner of the Chrome window.
    - Select "More tools" and then choose "Developer tools."
    - Go to the "Console" tab in the opened Developer Tools.
  - **Using Keyboard Shortcuts**:
    - For Windows/Linux: Press Ctrl + Shift + J.
    - For macOS: Press Cmd + Option + J (for Chrome) / …+ C (for Safari).

First JavaScript example



JavaScript Statements Example

```
Hello, World!                                        main.html:14
Sum: 15                                              main.html:20
>
```

Conditional and looping statements

JavaScript's `if` and `for` statements are similar to those in many other programming languages.

Condition (if) Statement:

```
if (condition) {
    // Code to execute if the condition is true
} else {
    // Code to execute if the condition is false
}
```

Examples

What is the output of the following code snippets:

```
var x = 10;
if (x<100) {
    var x = 20;
    console.log(x);
}
console.log(x);
```

```
let y = 10;
if (y==10) {
    let y = 20;
    console.log(y);
}
console.log(y);
```

```
const z = 10;
if (true) {
    const z = 20;
    console.log(z);
}
const z = 50;
console.log(z);
```

Output:

```
20
20
```

Output:

```
20
10
```

Output:

```
Uncaught SyntaxError: Identifier 'z' has already been declared
 (at main.html:15:7)
```

Conditional and looping statements

JavaScript's `if` and `for` statements are similar to those in many other programming languages.

Condition (if) Statement:

```
if (condition) {
    // Code to execute if the condition is true
} else {
    // Code to execute if the condition is false
}
```

Loop (for) Statement:

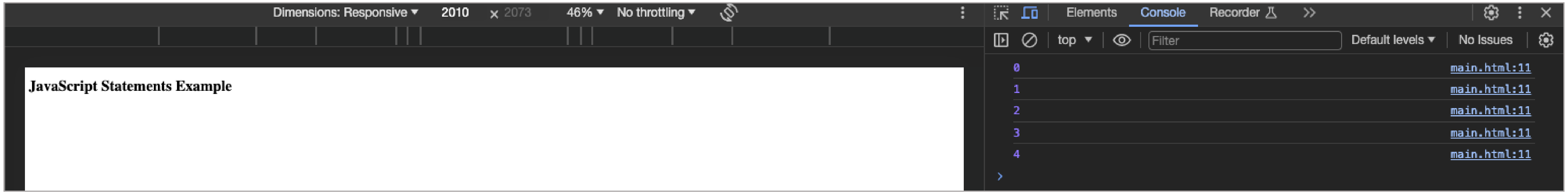
```
for (initialization; condition; increment/decrement) {
    // Code to be executed in each iteration
}
```

Loops - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Statements Example</title>
</head>
<body>
    <h1>JavaScript Statements Example</h1>

    <script>
    for (let i = 0; i < 5; i++) {
      console.log(i);
    }
    </script>
</body>
</html>
```

Loops - example



JavaScript Statements Example

```
0    main.html:11
1    main.html:11
2    main.html:11
3    main.html:11
4    main.html:11
```

Questions…

Is a semicolon mandatory in JavaScript?

No, a semicolon is not strictly mandatory in JavaScript. However, it is considered a good practice to include them to avoid potential issues and make the code more readable.

Is JavaScript a case-sensitive language?

Yes, JavaScript is a case-sensitive language. For example, variables named "myVariable" and "MyVariable" would be treated as distinct entities in JavaScript.

Strings

- A string is a sequence of characters enclosed in single (') or double (") quotes.

```
let singleQuotes = 'This is a string with single quotes.';
let doubleQuotes = "This is a string with double quotes.";
```

- Quotes within a string are permissible as long as they do not match the quotes enclosing the string.

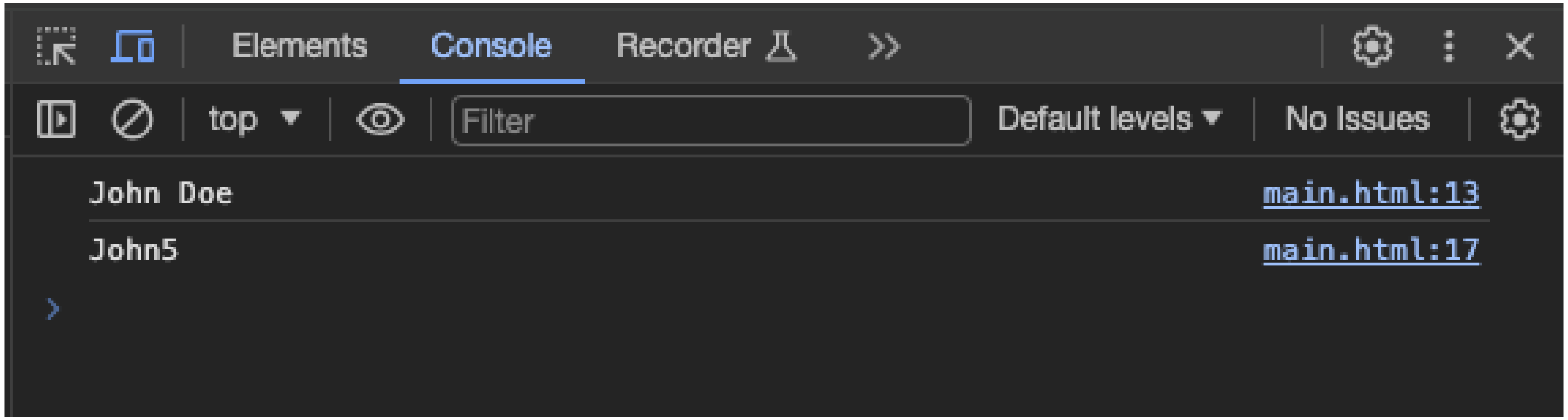
- Strings can be concatenated using the + operator.

String - example

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript String Example</title>
</head>
<body>
    <h1>JavaScript String Example</h1>
    <script>
      let firstName = 'John';
      let lastName = 'Doe';
      let fullName = firstName + ' ' + lastName;
      console.log(fullName)

      let x = 5
      let concatWithNumbers = firstName + x;
      console.log(concatWithNumbers)
    </script>
</body></html>
```

String - example



```
John Doe                                        main.html:13
John5                                           main.html:17
>
```

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.

String useful method - examples

Code snippet

```
let str = "Hello, World!";
let len = str.length;
```

$\longrightarrow$

Result

len will be 13

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.

- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.

String useful method - examples

Code snippet

Result

```
let str = "Hello, World!";
let len = str.length;
```

→ *len will be 13*

```
let str = "JavaScript";
let sub = str.substring(0, 4);
```

→ *sub will be "Java"*

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.

- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.

- `substr(start, length)`: returns a portion of a string starting from the 'start' index and continuing for a specified 'length' of characters.

String useful method - examples

Code snippet

```
let str = "Hello, World!";
let len = str.length;
```

```
let str = "JavaScript";
let sub = str.substring(0, 4);
```

```
let str = "JavaScript";
let sub = str.substr(4, 6);
```

Result

→ *len will be 13*

→ *sub will be "Java"*

→ *sub will be "Script"*

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.

- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.

- `substr(start, length)`: returns a portion of a string starting from the 'start' index and continuing for a specified 'length' of characters.

- `replace`: replace a specified substring or pattern in a string with another substring or value.

String useful method - examples

Code snippet		**Result**

`let str = "Hello, World!";`
`let len = str.length;`
	⟶	*len will be 13*

`let str = "JavaScript";`
`let sub = str.substring(0, 4);`
	⟶	*sub will be "Java"*

`let str = "JavaScript";`
`let sub = str.substr(4, 6);`
	⟶	*sub will be "Script"*

`let str = "Hello, World!";`
`let newStr = str.replace("Hello", "Hi");`
	⟶	*newStr will be "Hi, World!"*

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.
- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.
- `substr(start, length)`: returns a portion of a string starting from the 'start' index and continuing for a specified 'length' of characters.
- `replace`: replace a specified substring or pattern in a string with another substring or value.
- `toUpperCase, toLowerCase`: changing the case of characters in a string.

String useful method - examples

Code snippet	Result

`let str = "Hello, World!";`
`let len = str.length;`
	len will be 13

`let str = "JavaScript";`
`let sub = str.substring(0, 4);`
	sub will be "Java"

`let str = "JavaScript";`
`let sub = str.substr(4, 6);`
	sub will be "Script"

`let str = "Hello, World!";`
`let newStr = str.replace("Hello", "Hi");`
	newStr will be "Hi, World!"

`let str = "JavaScript";`
`let upperCase = str.toUpperCase();`
`let lowerCase = str.toLowerCase();`
	upperCase will be "JAVASCRIPT"
	lowerCase will be "javascript"

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.

- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.

- `substr(start, length)`: returns a portion of a string starting from the 'start' index and continuing for a specified 'length' of characters.

- `replace`: replace a specified substring or pattern in a string with another substring or value.

- `toUpperCase, toLowerCase`: changing the case of characters in a string.

- `Trim`: removing any leading or trailing white spaces.

String useful method - examples

Code snippet

Result

```
let str = "Hello, World!";
let len = str.length;
```

len will be 13

```
let str = "JavaScript";
let sub = str.substring(0, 4);
```

sub will be "Java"

```
let str = "JavaScript";
let sub = str.substr(4, 6);
```

sub will be "Script"

```
let str = "Hello, World!";
let newStr = str.replace("Hello", "Hi");
```

newStr will be "Hi, World!"

```
let str = "JavaScript";
let upperCase = str.toUpperCase();
let lowerCase = str.toLowerCase();
```

upperCase will be "JAVASCRIPT”
lowerCase will be "javascript"

```
let str = "   Hello, World!   ";
let trimmedStr = str.trim();
```

trimmedStr will be "Hello, World!"

String useful methods

- `length`: returns the length of a string refers to the number of characters it contains.
- `substring(start, end)`: returns a portion of a string that starts at the specified 'start' index and extends to the 'end' index.
- `substr(start, length)`: returns a portion of a string starting from the 'start' index and continuing for a specified 'length' of characters.
- `replace`: replace a specified substring or pattern in a string with another substring or value.
- `toUpperCase, toLowerCase`: changing the case of characters in a string.
- `Trim`: removing any leading or trailing white spaces.
- `IndexOf(occurance)`: find the index of the first occurrence of a specified value or substring within a string. If not found, it returns -1.

String useful method - examples

Code snippet | **Result**

```
let str = "Hello, World!";
let len = str.length;
```

→ *len will be 13*

```
let str = "JavaScript";
let sub = str.substring(0, 4);
```

→ *sub will be "Java"*

```
let str = "JavaScript";
let sub = str.substr(4, 6);
```

→ *sub will be "Script"*

```
let str = "Hello, World!";
let newStr = str.replace("Hello", "Hi");
```

→ *newStr will be "Hi, World!"*

```
let str = "JavaScript";
let upperCase = str.toUpperCase();
let lowerCase = str.toLowerCase();
```

→ *upperCase will be "JAVASCRIPT”*
lowerCase will be "javascript"

```
let str = "   Hello, World!   ";
let trimmedStr = str.trim();
```

→ *trimmedStr will be "Hello, World!"*

```
let sentence = "Welcome to ENSF381 course.";
let indexOfENSF = sentence.indexOf("ENSF381");
let indexOfCourse = sentence.indexOf("course");
```

→ *indexOfENSF will be 11*
indexOfCourse will be 19

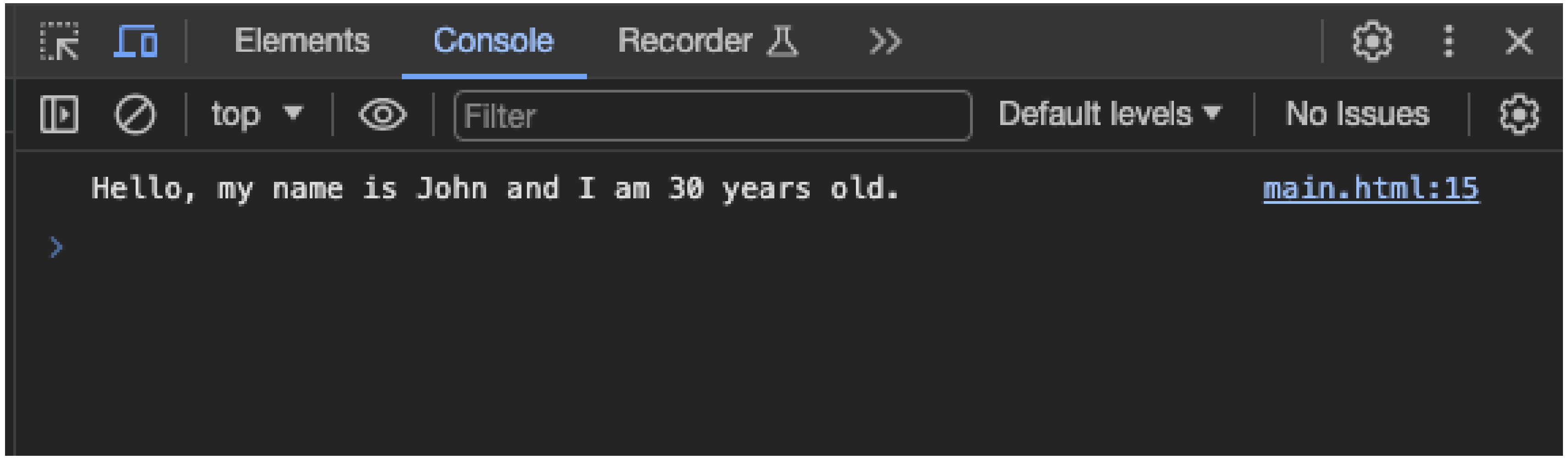
String template - example

- Allow for easy embedding of expressions within the string.
- Also known as template literals.
- The syntax for a template literal is ${expression}, where expression is any valid JavaScript expression.

```
let name = "John";
let age = 30;

let greeting = `Hello, my name is ${name} and I am ${age} years old.`;
console.log(greeting);
```

String template - example



```
Hello, my name is John and I am 30 years old.                main.html:15
```

Questions