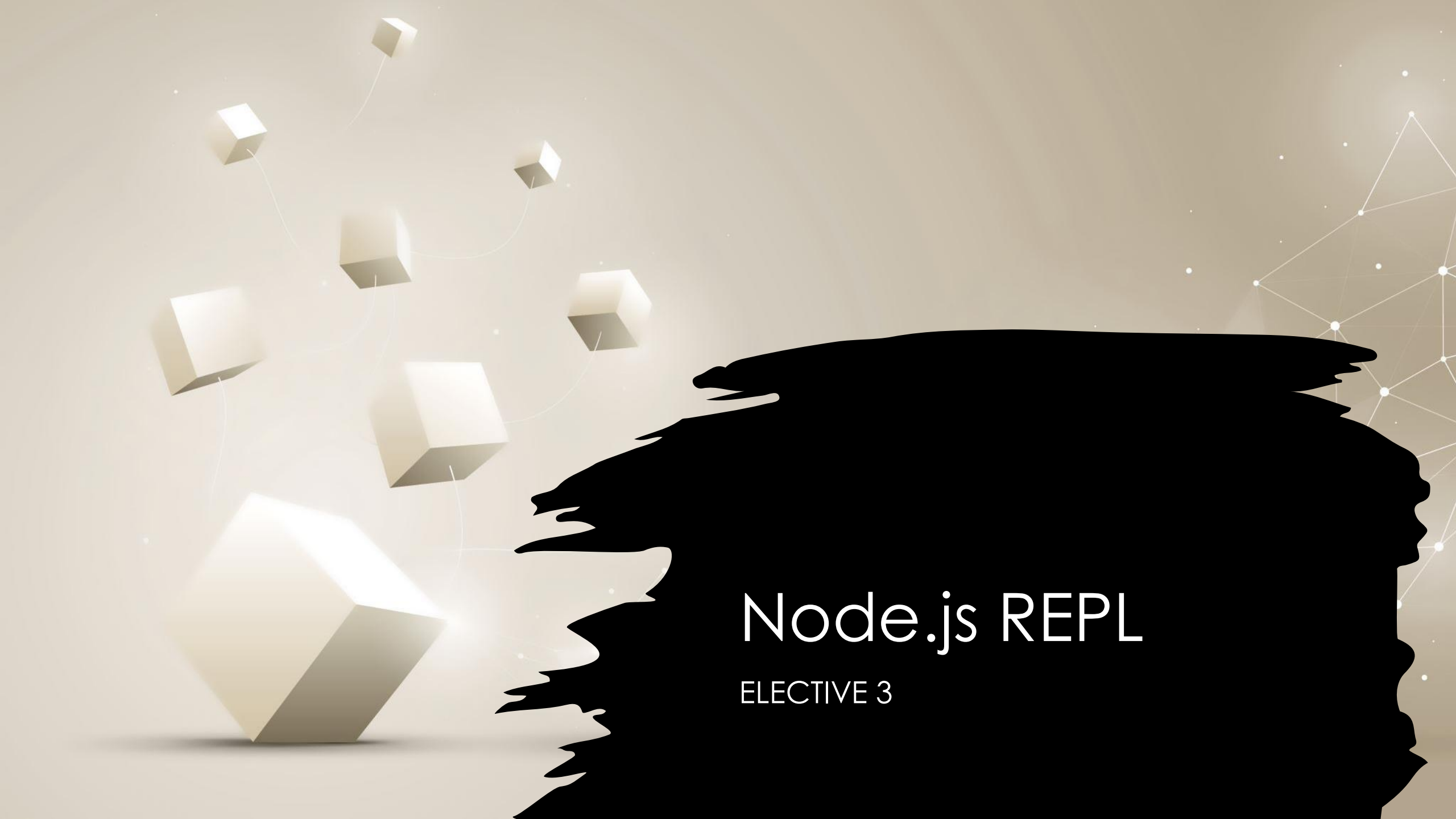

Node.js REPL

ELECTIVE 3

Understanding the Concept of REPL

- **REPL** stands for **Read-Eval-Print Loop**. It’s an interactive environment that reads your input, evaluates it, prints the result, and then loops back to read the next input. This makes it a great tool for quickly testing and debugging JavaScript code.

Familiarizing with the Programming Syntax in Node.js REPL

- When you start the Node.js REPL by typing **node** in your terminal, you'll see a prompt **(>)** where you can enter JavaScript code. Here are some basic examples:

- **Basic Operations**:

```
> 3 + 4
7
```

Defining Variables and Functions:

```
> let x = 10
undefined
> function multiply(a, b) { return a * b; }
undefined
> multiply(5, x)
50
```

Multi-line Mode: If you start typing a function or a block of code, the REPL will wait for you to complete it:

```
> function greet(name) {
... return `Hello, ‘ + name;
... }
undefined
> greet('Node.js User')
'Hello, Node.js User!'
```

Common REPL Shortcuts and Commands

Here are some useful shortcuts and commands in the Node.js REPL:

.**help**: Displays help for REPL commands.

.**editor**: Enters editor mode for multi-line input. Use Ctrl + D to finish and run the code.

.**break**: Aborts the current multi-line expression.

.**clear**: Resets the REPL context.

.**exit**: Exits the REPL.

.**save**: Saves the current REPL session to a file.

.**load**: Loads a file into the current REPL session.

Special Keys

- **Ctrl + C**: Press once to abort the current expression, press twice to exit the REPL.
- **Ctrl + D**: Exits the REPL.
- **Up/Down Arrow Keys**: Navigate through command history.
- **Tab**: Autocompletes commands and shows available options.

Here’s a quick example of a REPL session:

```
> let message = 'Hello, World!'
undefined
> console.log(message)
Hello, World!
undefined
> message.toUpperCase()
'HELLO, WORLD!'
```

Common Method

`toUpperCase()`

- Converts all characters in a string to uppercase.

`toLowerCase()`

- Converts all characters in a string to lowercase.

```
let message = 'Hello, World!’;
let lowerMessage = message.toLowerCase();
console.log(lowerMessage); // Output: 'hello, world!'
```

Common Method

`charAt()`

- Returns the character at a specified index.

```
let char = message.charAt(0);
console.log(char); // Output: ‘H’
```

`slice()`

Extracts a section of a string and returns it as a new string.

```
let slicedMessage = message.slice(0, 5);
console.log(slicedMessage); // Output: 'Hello'
```

Common Method

`replace()`

Replaces a specified value with another value in a string.

```
let newMessage = message.replace('World', 'JavaScript');
console.log(newMessage); // Output: 'Hello, JavaScript!’
```

`split()`

Splits a string into an array of substrings.

```
let words = message.split(' ');
console.log(words); // Output: ['Hello,', 'World!']
```

Common Method

`concat()`

Joins two or more strings.

```
let greeting = 'Hello';
let name = 'Alice';
let fullGreeting = greeting.concat(', ', name, '!');
console.log(fullGreeting); // Output: 'Hello, Alice!’
```

`trim()`

Removes whitespace from both ends of a string.

```
let spacedMessage = '   Hello, World!   ';
let trimmedMessage = spacedMessage.trim();
console.log(trimmedMessage); // Output: 'Hello, World!'
```