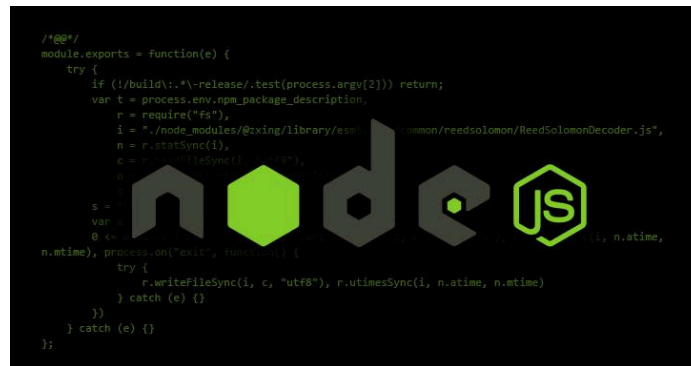




Code Along: User Input

Step 1

On command line, create a file called 'user_input.js' in your ASCProjects/week1/day2 directory.

Step 2 - PROCESS.ARGV[2]

- a. Type the following code in your script:

```
let userInput = process.argv[2];
console.log("Input:", userInput);
```

- b. Enter `node user_input.js` on command line to run your program.

Reminder: You need to be in the proper directory for your program to run.

Sample run:

```
ASCStudent@DESKTOP-RLIDLQU MINGW64 ~/Desktop/ASCProjects/week1/day1
$ node user_input.js
Input: undefined
```

`userInput` is `undefined` because we did not provide any additional arguments on command line to be used as user input. Let's change this up in [Step 3](#).

Step 3 - PROCESS.ARGV[2] WITH COMMAND LINE INPUT

Run your program with the command - `node user_input.js PIKACHU`. Do not make any changes to your code.

Sample run:

```
ASCStudent@DESKTOP-RLIDLQU MINGW64 ~/Desktop/ASCProjects/week1/day1
$ node user_input.js PIKACHU
Input: PIKACHU
```

We entered an additional argument on command line - "PIKACHU" to be used as input in JavaScript.

~GO TO SLIDES: PROCESS.ARGV[n]

Step 4 - PROCESS.ARGV WITH MULTIPLE COMMAND LINE ARGUMENTS

We will now collect multiple user inputs from command line to be used in JavaScript.

- Modify your existing code:

```
let firstInput = process.argv[2];
let secondInput = process.argv[3];
let thirdInput = process.argv[4];

console.log("Input A:", firstInput);
console.log("Input B:", secondInput);
console.log("Input C:", thirdInput);
```

- Run your program with the command - `node user_input.js all star code`

Sample run:

```
ASCStudent@DESKTOP-RLIDLQU MINGW64 ~/Desktop/ASCProjects/week1/day1
$ node user_input.js all star code
Input A: all
Input B: star
Input C: code
```

The additional arguments entered on command line were captured as user input in JavaScript.

~GO TO SLIDES: MULTIPLE COMMAND LINE ARGUMENTS + TYPEOF OPERATOR

Step 4 - DATA TYPE OF USER INPUT

- a. Add the following code to the bottom of your script:

```
console.log();  
console.log("Data Type A:", typeof(firstInput));  
console.log("Data Type B:", typeof(secondInput));  
console.log("Data Type C:", typeof(thirdInput));
```

- b. Run your program with the command - `node user_input.js Hello 1800 true`

Sample code:

```
let firstInput = process.argv[2];  
let secondInput = process.argv[3];  
let thirdInput = process.argv[4];  
  
console.log("Input A:", firstInput);  
console.log("Input B:", secondInput);  
console.log("Input C:", thirdInput);  
console.log();  
console.log("Data Type A:", typeof(firstInput));  
console.log("Data Type B:", typeof(secondInput));  
console.log("Data Type C:", typeof(thirdInput));
```

Sample run:

```
ASCStudent@DESKTOP-RLIDLQU MINGW64 ~/Desktop/ASCProjects/week1/day1  
$ node user_input.js Hello 1800 true  
Input A: Hello  
Input B: 1800  
Input C: true  
  
Data Type A: string  
Data Type B: string  
Data Type C: string
```

The sample run indicates that all command line user inputs are taken in as **Strings**.