Getting Started

→ Output
→ variables
→ Input
→ if-else
→ loops
→ Is prime
→ First function
→ First Array

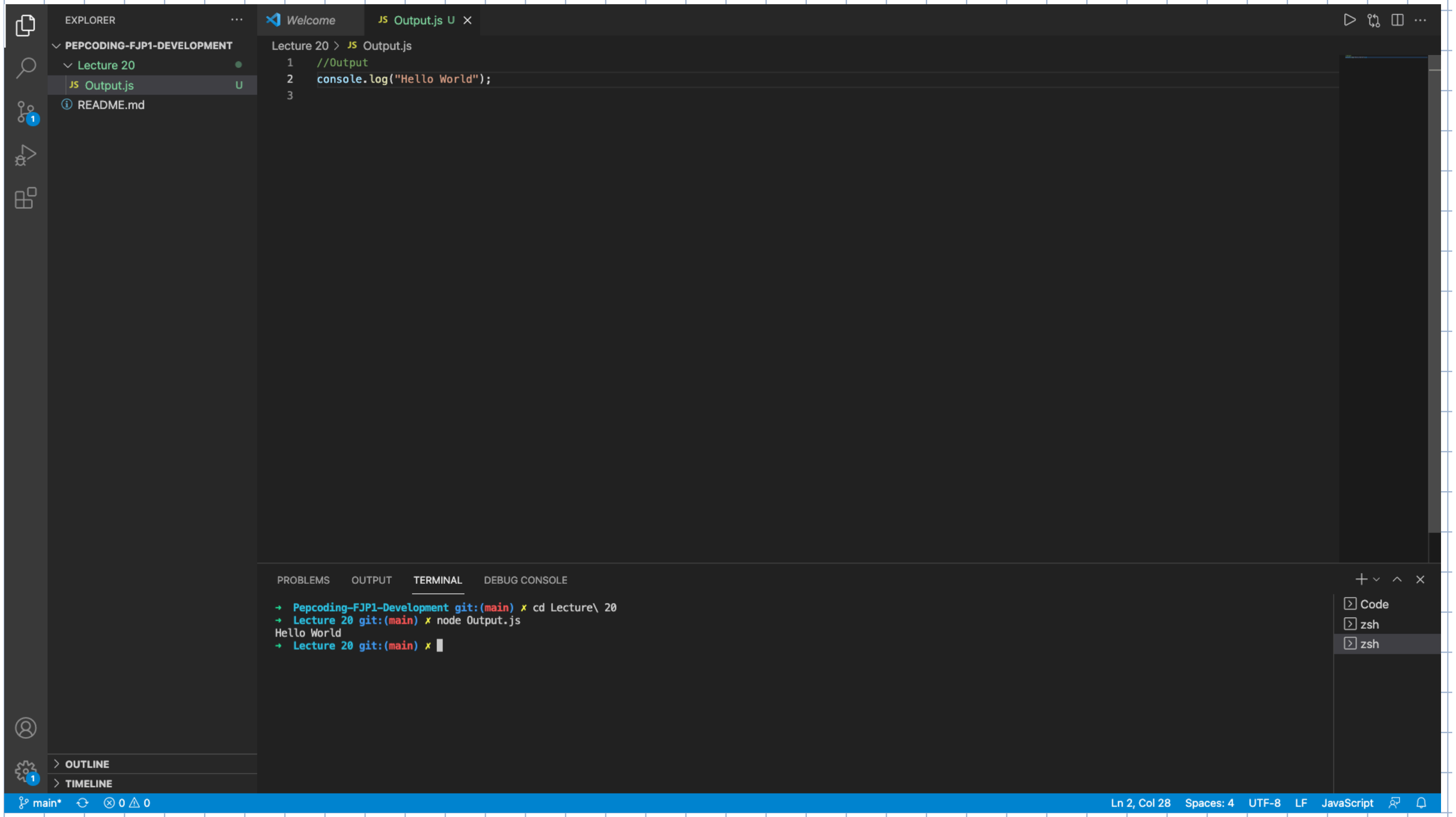
https://www.geeksforgeeks.org/javascript-console-log-with-examples/

OUTPUT

Output.js

console.log("Hello World");

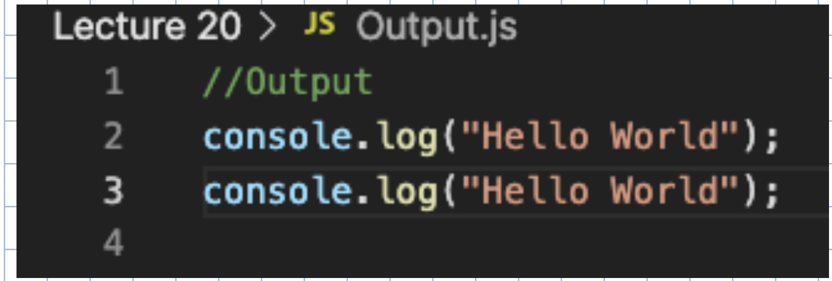
a function in JavaScript which is used to print any kind of variables defined before in it or just print any message that needs to be displayed to the user



Lecture 20 > Output.js

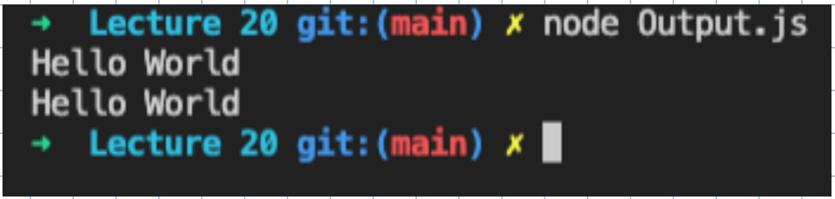
```
//Output
console.log("Hello World");
```

```
➜  Pepcoding-FJP1-Development git:(main) ✗ cd Lecture\ 20
➜  Lecture 20 git:(main) ✗ node Output.js
Hello World
➜  Lecture 20 git:(main) ✗ ▌
```



Lecture 20 > Output.js

```
//Output
console.log("Hello World");
console.log("Hello World");
```



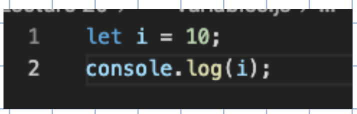
```
➜  Lecture 20 git:(main) ✗ node Output.js
Hello World
Hello World
➜  Lecture 20 git:(main) ✗ ▌
```

VARIABLE

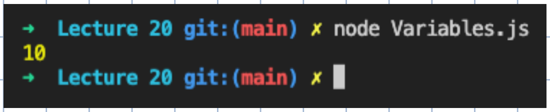
→ Javascript is a dynamically-typed language.

Java is static-typed language

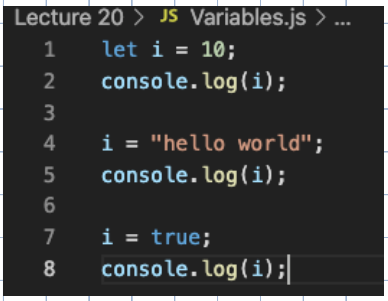
In Javascript, here when a variable is defined, the datatype of a variable can change.



```
let i = 10;
console.log(i);
```

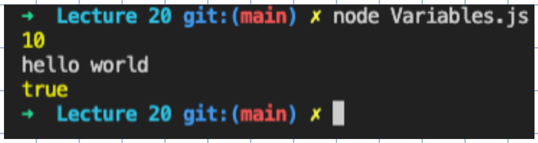


```
→ Lecture 20 git:(main) ✗ node Variables.js
10
→ Lecture 20 git:(main) ✗ █
```



Lecture 20 > JS Variables.js > ...

```
let i = 10;
console.log(i);

i = "hello world";
console.log(i);

i = true;
console.log(i);
```

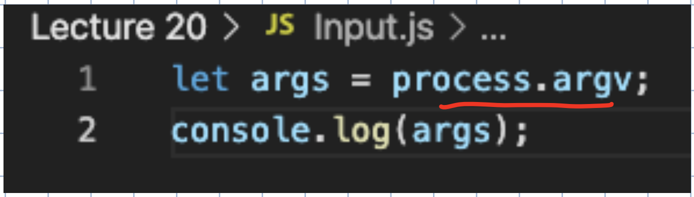


```
→ Lecture 20 git:(main) ✗ node Variables.js
10
hello world
true
→ Lecture 20 git:(main) ✗ █
```

https://www.w3schools.com/js/js_variables.asp

https://medium.com/@easyexpresssoft/dynamic-typing-coercion-and-operators-a8986be8c198

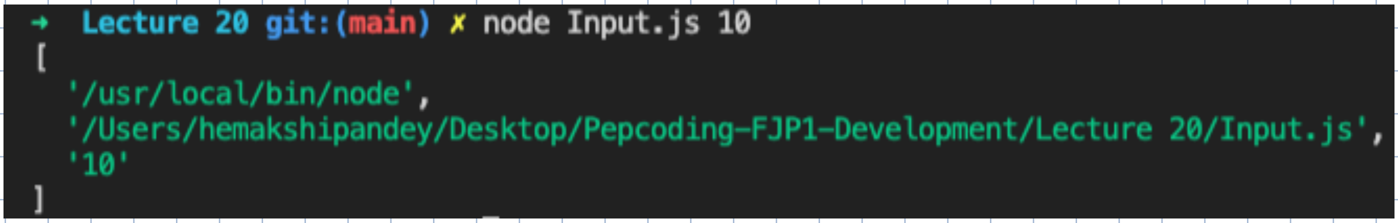
INPUT



Lecture 20 > JS Input.js > ...

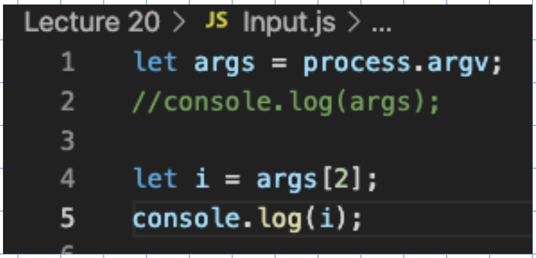
```
let args = process.argv;
console.log(args);
```

command line argument



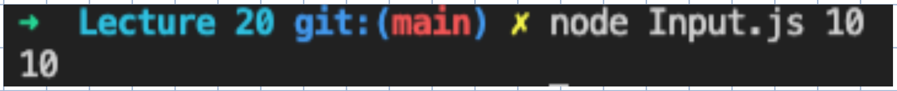
```
→ Lecture 20 git:(main) ✗ node Input.js 10
[
  '/usr/local/bin/node',
  '/Users/hemakshipandey/Desktop/Pepcoding-FJP1-Development/Lecture 20/Input.js',
  '10'
]
```

https://nodejs.org/en/knowledge/command-line/how-to-parse-command-line-arguments/

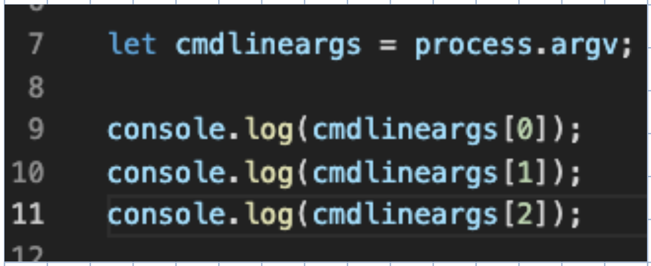


Lecture 20 > JS Input.js > ...

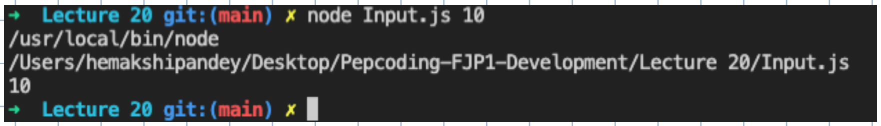
```
let args = process.argv;
//console.log(args);

let i = args[2];
console.log(i);
```



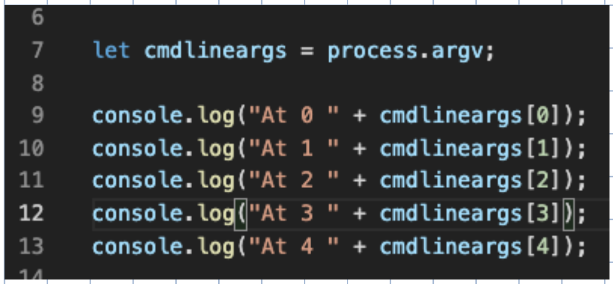
```
→ Lecture 20 git:(main) ✗ node Input.js 10
10
```



```js
let cmdlineargs = process.argv;

console.log(cmdlineargs[0]);
console.log(cmdlineargs[1]);
console.log(cmdlineargs[2]);
```

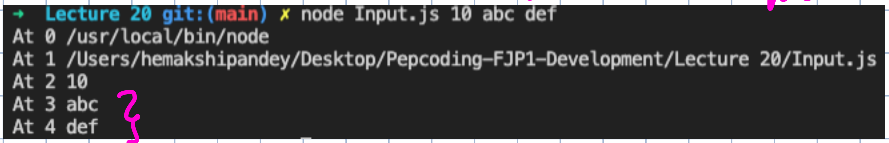


```
→  Lecture 20 git:(main) ✗ node Input.js 10
/usr/local/bin/node
/Users/hemakshipandey/Desktop/Pepcoding-FJP1-Development/Lecture 20/Input.js
10
→  Lecture 20 git:(main) ✗ 
```

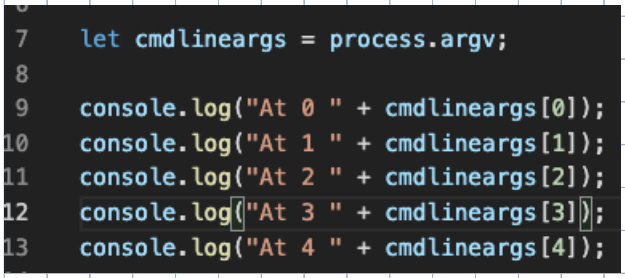


```js
let cmdlineargs = process.argv;

console.log("At 0 " + cmdlineargs[0]);
console.log("At 1 " + cmdlineargs[1]);
console.log("At 2 " + cmdlineargs[2]);
console.log("At 3 " + cmdlineargs[3]);
console.log("At 4 " + cmdlineargs[4]);
```

without double quotes

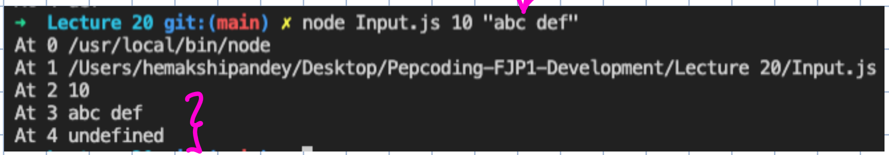


```
→  Lecture 20 git:(main) ✗ node Input.js 10 abc def
At 0 /usr/local/bin/node
At 1 /Users/hemakshipandey/Desktop/Pepcoding-FJP1-Development/Lecture 20/Input.js
At 2 10
At 3 abc
At 4 def
```

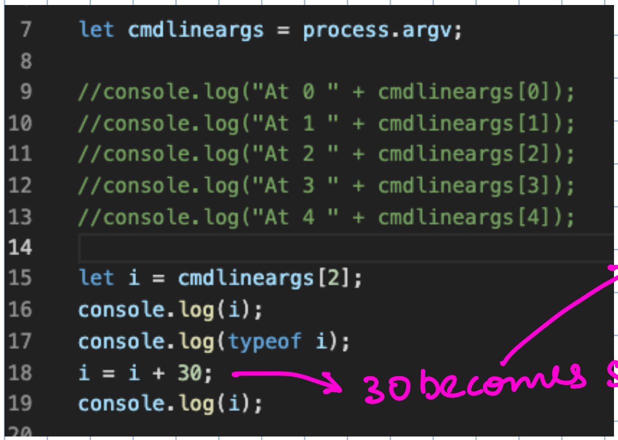


```js
let cmdlineargs = process.argv;

console.log("At 0 " + cmdlineargs[0]);
console.log("At 1 " + cmdlineargs[1]);
console.log("At 2 " + cmdlineargs[2]);
console.log("At 3 " + cmdlineargs[3]);
console.log("At 4 " + cmdlineargs[4]);
```

with double quotes

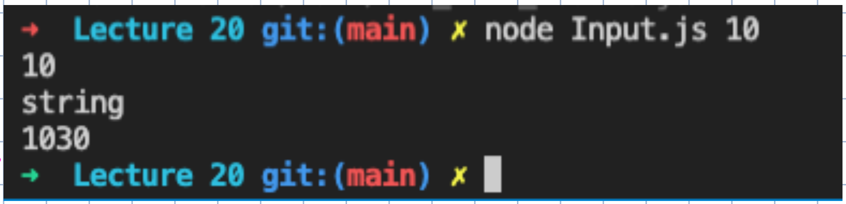


```
→  Lecture 20 git:(main) ✗ node Input.js 10 "abc def"
At 0 /usr/local/bin/node
At 1 /Users/hemakshipandey/Desktop/Pepcoding-FJP1-Development/Lecture 20/Input.js
At 2 10
At 3 abc def
At 4 undefined
```



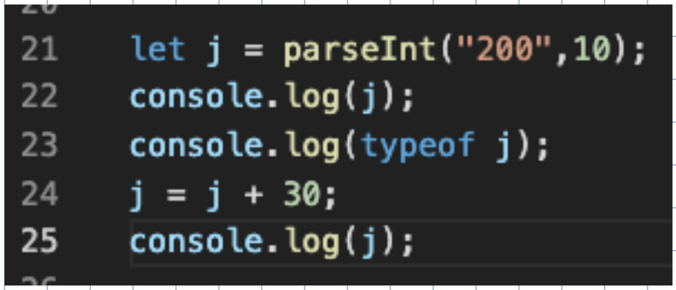
```js
let cmdlineargs = process.argv;

//console.log("At 0 " + cmdlineargs[0]);
//console.log("At 1 " + cmdlineargs[1]);
//console.log("At 2 " + cmdlineargs[2]);
//console.log("At 3 " + cmdlineargs[3]);
//console.log("At 4 " + cmdlineargs[4]);

let i = cmdlineargs[2];
console.log(i);
console.log(typeof i);
i = i + 30;
console.log(i);
```

30 becomes string

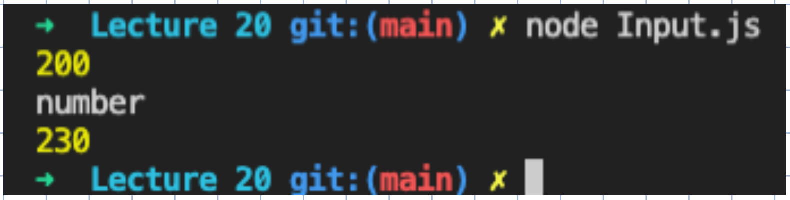


```
→  Lecture 20 git:(main) ✗ node Input.js 10
10
string
1030
→  Lecture 20 git:(main) ✗ 
```

ParseInt( ) function

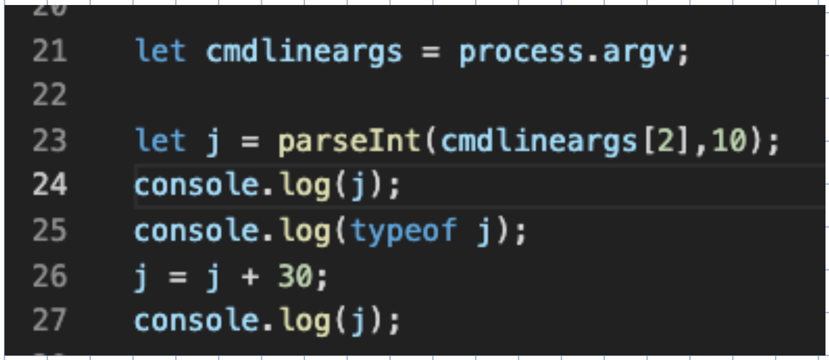


```js
let j = parseInt("200",10);
console.log(j);
console.log(typeof j);
j = j + 30;
console.log(j);
```

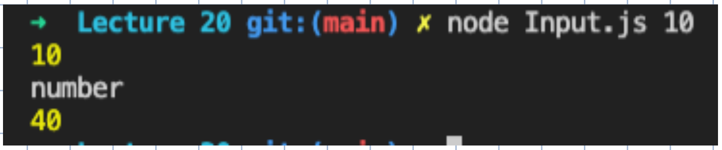


```
→  Lecture 20 git:(main) ✗ node Input.js
200
number
230
→  Lecture 20 git:(main) ✗ 
```

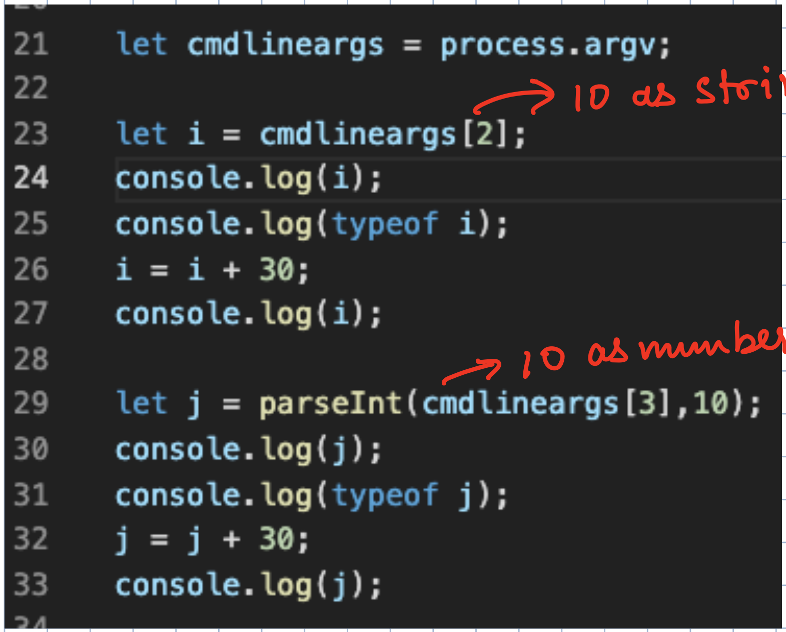
https://www.w3schools.com/jsref/jsref_parseint.asp



```javascript
let cmdlineargs = process.argv;

let j = parseInt(cmdlineargs[2],10);
console.log(j);
console.log(typeof j);
j = j + 30;
console.log(j);
```



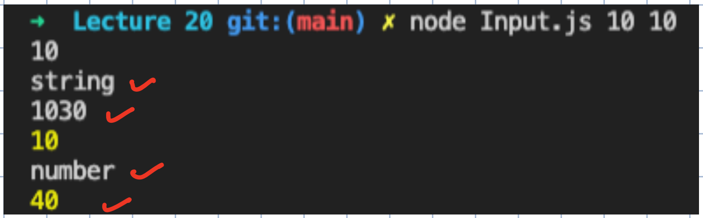
```
→ Lecture 20 git:(main) ✗ node Input.js 10
10
number
40
```



```javascript
let cmdlineargs = process.argv;

let i = cmdlineargs[2];
console.log(i);
console.log(typeof i);
i = i + 30;
console.log(i);

let j = parseInt(cmdlineargs[3],10);
console.log(j);
console.log(typeof j);
j = j + 30;
console.log(j);
```

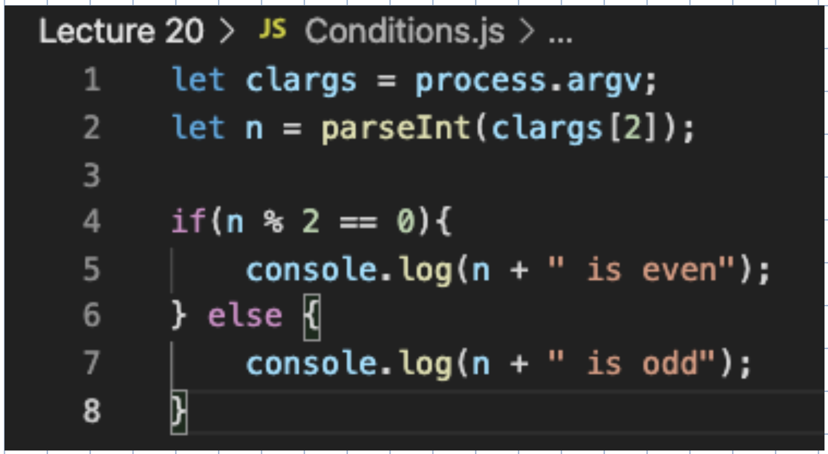
→ 10 as string

→ 10 as number



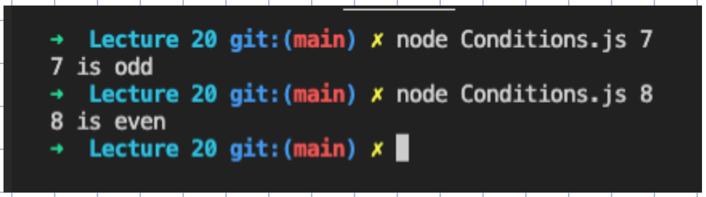
```
→ Lecture 20 git:(main) ✗ node Input.js 10 10
10
string ✓
1030 ✓
10
number ✓
40 ✓
```

CONDITIONS : if-else



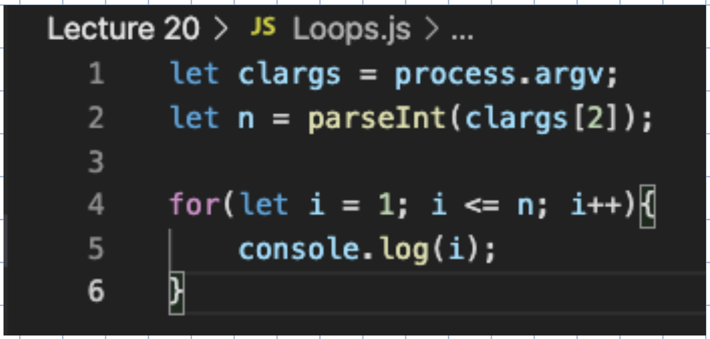
Lecture 20 > JS Conditions.js > ...

```javascript
let clargs = process.argv;
let n = parseInt(clargs[2]);

if(n % 2 == 0){
    console.log(n + " is even");
} else {
    console.log(n + " is odd");
}
```

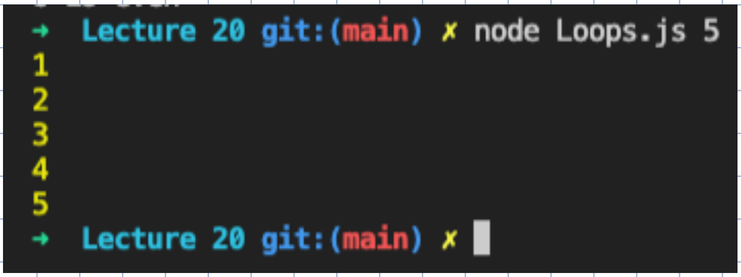


```
→ Lecture 20 git:(main) ✗ node Conditions.js 7
7 is odd
→ Lecture 20 git:(main) ✗ node Conditions.js 8
8 is even
→ Lecture 20 git:(main) ✗
```

Loops

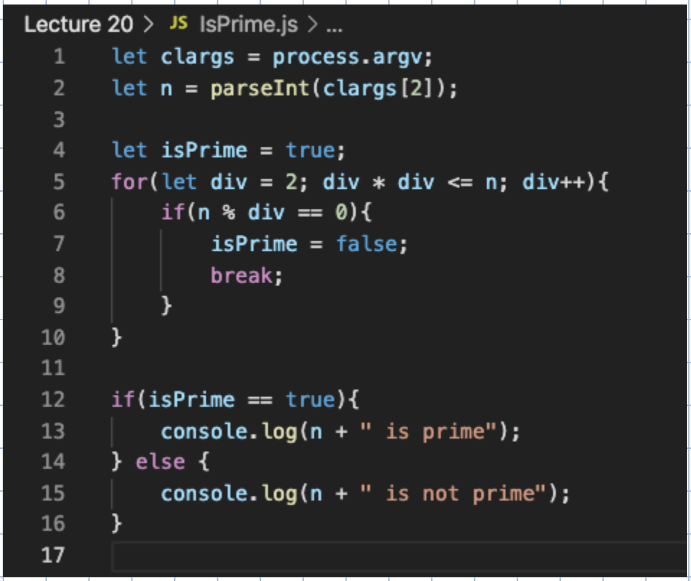


```
Lecture 20 > JS Loops.js > ...
let clargs = process.argv;
let n = parseInt(clargs[2]);

for(let i = 1; i <= n; i++){
    console.log(i);
}
```

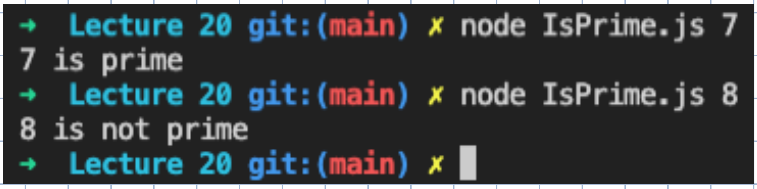


```
→  Lecture 20 git:(main) ✗ node Loops.js 5
1
2
3
4
5
→  Lecture 20 git:(main) ✗ 
```

isprime

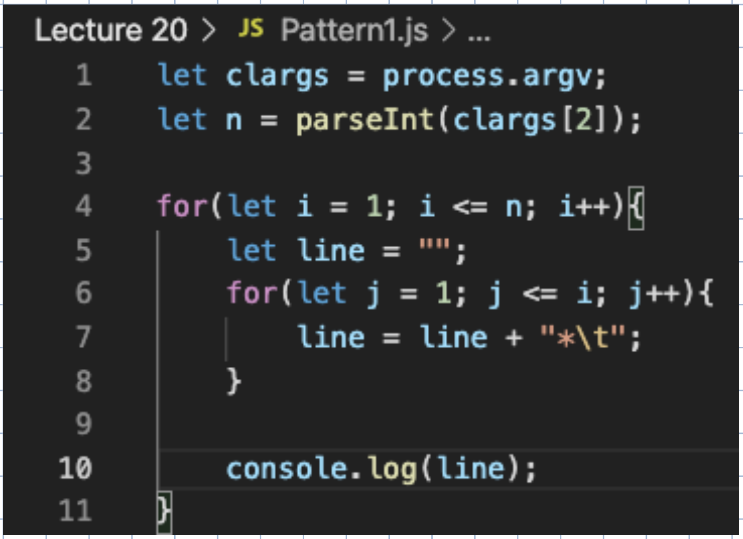


```
Lecture 20 > JS IsPrime.js > ...
let clargs = process.argv;
let n = parseInt(clargs[2]);

let isPrime = true;
for(let div = 2; div * div <= n; div++){
    if(n % div == 0){
        isPrime = false;
        break;
    }
}

if(isPrime == true){
    console.log(n + " is prime");
} else {
    console.log(n + " is not prime");
}
```

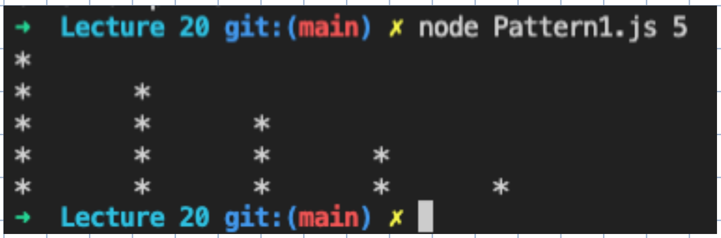


```
→  Lecture 20 git:(main) ✗ node IsPrime.js 7
7 is prime
→  Lecture 20 git:(main) ✗ node IsPrime.js 8
8 is not prime
→  Lecture 20 git:(main) ✗ 
```

Pattern



```
Lecture 20 > JS Pattern1.js > ...
let clargs = process.argv;
let n = parseInt(clargs[2]);

for(let i = 1; i <= n; i++){
    let line = "";
    for(let j = 1; j <= i; j++){
        line = line + "*\t";
    }

    console.log(line);
}
```



```
→  Lecture 20 git:(main) ✗ node Pattern1.js 5
*
*       *
*       *       *
*       *       *       *
*       *       *       *       *
→  Lecture 20 git:(main) ✗ 
```