# JS BASICS 2

# FUNCTIONS

- Function definition
- Function declaration
- Return type functions

# FUNCTIONS SCOPES AND BLOCK SCOPES

- Difference between var, let and const
- Let has block scope
- Var has function scope
- Const also has block scope

# ARRAYS AND STRINGS

- Basic functionality
- Inbuilt array methods – push, pop, shift, unshift, splice, sort
- Inbuilt string methods – length, indexOf, includes, slice, substring, substr, toLowerCase, toUpperCase
- String concatenation
- For-of loops

# SLICE METHOD

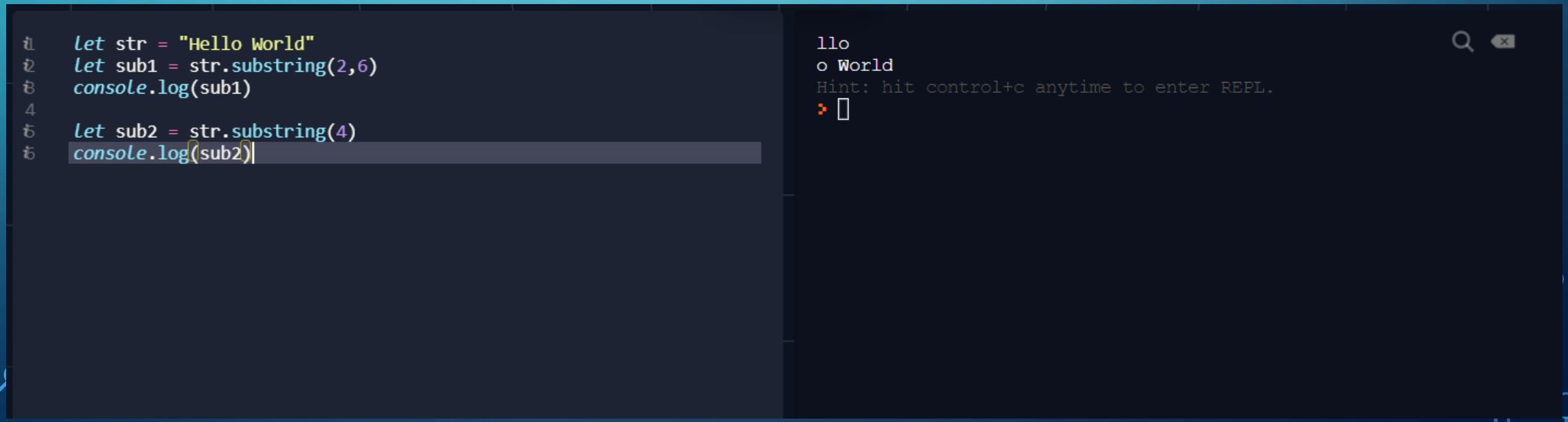
- Slice and return elements from given starting index till the ending index
- Does not modify the original string

```
let p = "Apple"
let q = p.slice(2,5)
console.log(q)


let a = "abcdef"
let b = a.slice(1,-3)

console.log(b)
```

```
ple
bc
Hint: hit control+c anytime to enter REPL.
>
```

# SUBSTRING METHOD

- Same as slice but does not accept negative arguments
- If you omit the second parameter, substring() will slice out the rest of the string.



# SUBSTR METHOD

- Slices the string using given starting index and length

```
let str = "Hello World";
let sub1 = str.substr(2,6);
console.log(sub1);

let sub2 = str.substr(6,3);
console.log(sub2);
```

```
llo Wo
Wor
Hint: hit control+c anytime to enter REPL.
>
```

# OBJECTS

Objects in JavaScript, just as in many other programming languages, can be compared to objects in real life. The concept of objects in JavaScript can be understood with real life, tangible objects.

```
11  let person = {
12      name: 'Spiderman',
13      age: 20,
14      villains : ['Green Goblin', 'Doc Ock', 'Venom', 'Vulture']
15  }
```

Since there is no fixed type for the data, objects can further contain objects and so on.

```
11  let person = {
12      name: {
13          first: 'Peter',
14          last: 'Parker'
15      },
16      age: 20,
17      villains : ['Green Goblin', 'Doc Ock', 'Venom', 'Vulture']
18  }
```

- The properties of an object can be accessed using . (dot operator) or using [] (square brackets) operators.

```
let person = {
    name: {
        first: 'Peter',
        last: 'Parker'
    },
    age: 20,
    villains : ['Green Goblin', 'Doc Ock', 'Venom', 'Vulture']
}

console.log(person.name.first)
console.log(person.name.last)

console.log(person['age'])
console.log(person['villains'])
```

- Property values can be changed in the same manner as they are accessed.
- We can also create new properties at runtime.
- If a property already exists, its value is changed. If it doesn't, a new entry is created for it.

```
let person = {
    name: {
        first: 'Peter',
        last: 'Parker'
    },
    age: 20,
    villains : ['Green Goblin', 'Doc Ock', 'Venom', 'Vulture']
}

person.name = 'Peter Parker'
person.girlfriend = 'Mary Jane Watson'

console.log(person)
```

```
{
  name: 'Peter Parker',
  age: 20,
  villains: [ 'Green Goblin', 'Doc Ock', 'Venom', 'Vulture' ],
  girlfriend: 'Mary Jane Watson'
}
```

# BONUS CONTENT – HOW TO INSTALL NODE ?

1. Go to https://nodejs.org/en/
2. Download the latest version according to your OS.
3. Install it.
4. Install Code Runner extension for VS Code
5. Restart VS Code
6. Enjoy running JS code without the browser.