CADT IDT

Next-Gen Engagement Program

JAVASCRIPT

Week 2

Functions and Introduction to ES6

Ory Chanraksa & Sao Visal

TABLE CONTENT

- **Function:**
  1. Return with parameter
  2. Return without parameter
  3. No return with Parameter
  4. No return without Parameter

- **ES6:**
  1. Arrow Function
  2. Modules
  3. Template Literals
  4. Destructuring
  5. Async/Await

FUNCTION

- **Return with Parameter:** accept parameter(s) and return a value

- **Ex:** Passing a user id to get the user information.
  Passing 2 values to return a sum of the 2 numbers.

```
function returnWithParameter(a,b){
    return a+b
}
console.log(returnWithParameter(1,2) )// returns 3
```

FUNCTION

- **Return without Parameter:** doesn’t accept any parameters and return a value

- **Ex:** Break a big code into smaller that return a value like generate Id
  Call a function to get a value

```
function returnWithoutParameter(){
    return "Hello"
}
console.log(returnWithoutParameter()) // returns "Hello"
```

FUNCTION

- **No Return with Parameter:** does not return any value but accept parameter(s)

- **Ex:** Pass in a value to store in the database
  Pass in a value to display it

```
function noReturnWithParameter(a){
    console.log(a)
}
noReturnWithParameter("Hello") // returns Hello
```

FUNCTION

- **No Return without Parameter:** Does not have a return and has no passed in parameter

- **Ex:** Break big code into smaller functions like connecting to database
  Call a function to display “Hello”

```
function noReturnWithoutParameter(){
    console.log("Hello")
}
noReturnWithoutParameter() // returns Hello
```

ES6

- ES6 is a major update to JavaScript in 2015 that brought many modern changes.
- ES6 is used because of the modern syntax which make the code cleaner and easier to manage

ARROW FUNCTION

- Alternate way of declaring a function
- Useful in array method such as .map(), .filter()
- Shorter and cleaner when using it in nested code

ARROW FUNCTION

```
function normalFunction(a, b) {
    return a + b;
}
console.log(normalFunction(1, 2)); // returns 3

const arrowFunction = (a, b) => {
    return a + b;
};
console.log(arrowFunction(1, 2)); // returns 3
💡
//! or

const arrowFunctionShort = (a, b) => a + b;
console.log(arrowFunctionShort(1, 2)); // returns 3
```

ARROW FUNCTION

```
function normalFunction(arr) {
    for (let i = 0; i < arr.length; i++) {
        console.log(arr[i]);
    }
}

const array = [1, 2, 3, 4, 5];
normalFunction(array);

array.forEach((element) => {
    console.log(element);
});
```

MODULES

- Modules allow you to split your code into smaller files.
- Each file can export pieces of code (functions, variables, etc.) and import them into other files.
- This keeps code organized, reusable, and easier to maintain.

MODULE

Export

```
const a=0
const b=1
const c=2

export default a // Default export
export { a } // Export one constant
export { b, c } // Export multiple constants
export const d = 3; // Inline export
```

Import

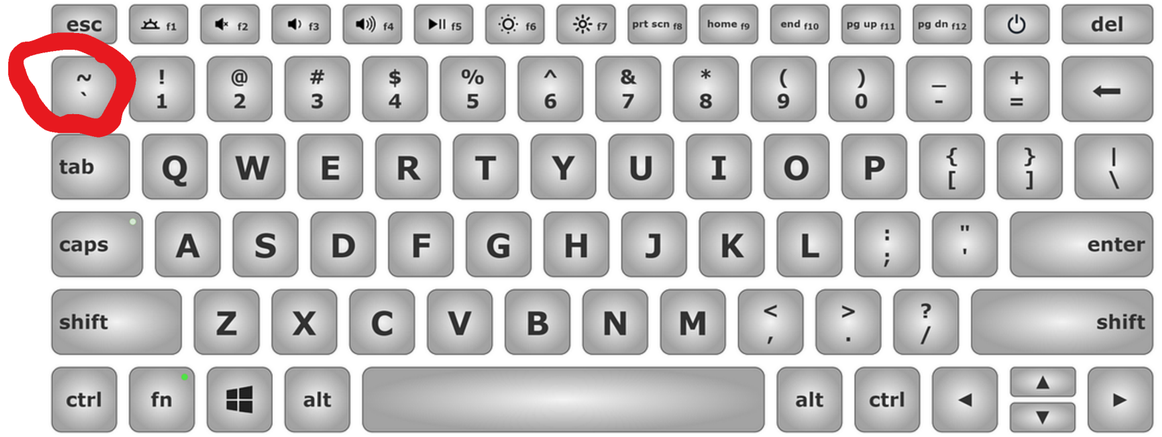
```
import {b,c} from './export.js';
import a from './export.js';
import {d} from './export.js';

console.log(a); //0
console.log(b); //1
console.log(c); //2
console.log(d); //3
```

TEMPLATE LITERALS

Template literals are strings wrapped in backticks (`) that allow for:

- Multi-line strings
- Put variable(s) inside the string through ${variable_name}
- Easier to read



TEMPLATE LITERALS

Code

```
console.log(`
    Hello
    Everyone
    `);

const a=0;
console.log(`The value is ${a}`);
```

Result

```
Hello
Everyone

The value is 0
```

DESTRUCTURING

Destructuring is a shortcut that lets you extract values from arrays or objects and assign them to variables in a clean, readable way:

- Extract multiple values at once
- Works with arrays and objects
- Makes code shorter and clearer

OBJECT

```
const student = {
    fullName: "John Doe",
    age: 20,
    course: "Introduction to JavaScript",
};

const {fullName, age, course} = student;
//! With destructuring
console.log(fullName);// Output: John Doe
console.log(age);// Output: 20
console.log(course);// Output: Introduction to JavaScript
//! Without destructuring
console.log(student.fullName);// Output: John Doe
console.log(student.age);// Output: 20
console.log(student.course);// Output: Introduction to JavaScript
```

ARRAY

```
const names= ['Alice', 'Bob', 'Charlie'];
const [first, second, third] = names;
console.log(first, second, third);
//Alice Bob Charlie
```

ASYNC/AWAIT

- Enable writing asynchronous code
- **Asynchronous code:** allow code to run asynchronously when a variable is put on await
- **await():** a method that give the code time to execute before giving error (Ex: fetching API, using methods that takes times...)

FETCHING API

```
const fetchApi=async()=>{
    const response = await fetch('https://jsonplaceholder.typicode.com/users');
    const data = await response.json();
    console.log(data);
}
fetchApi();
```

- Without await, the fetching will not happen since it takes time for it fetch and display properly

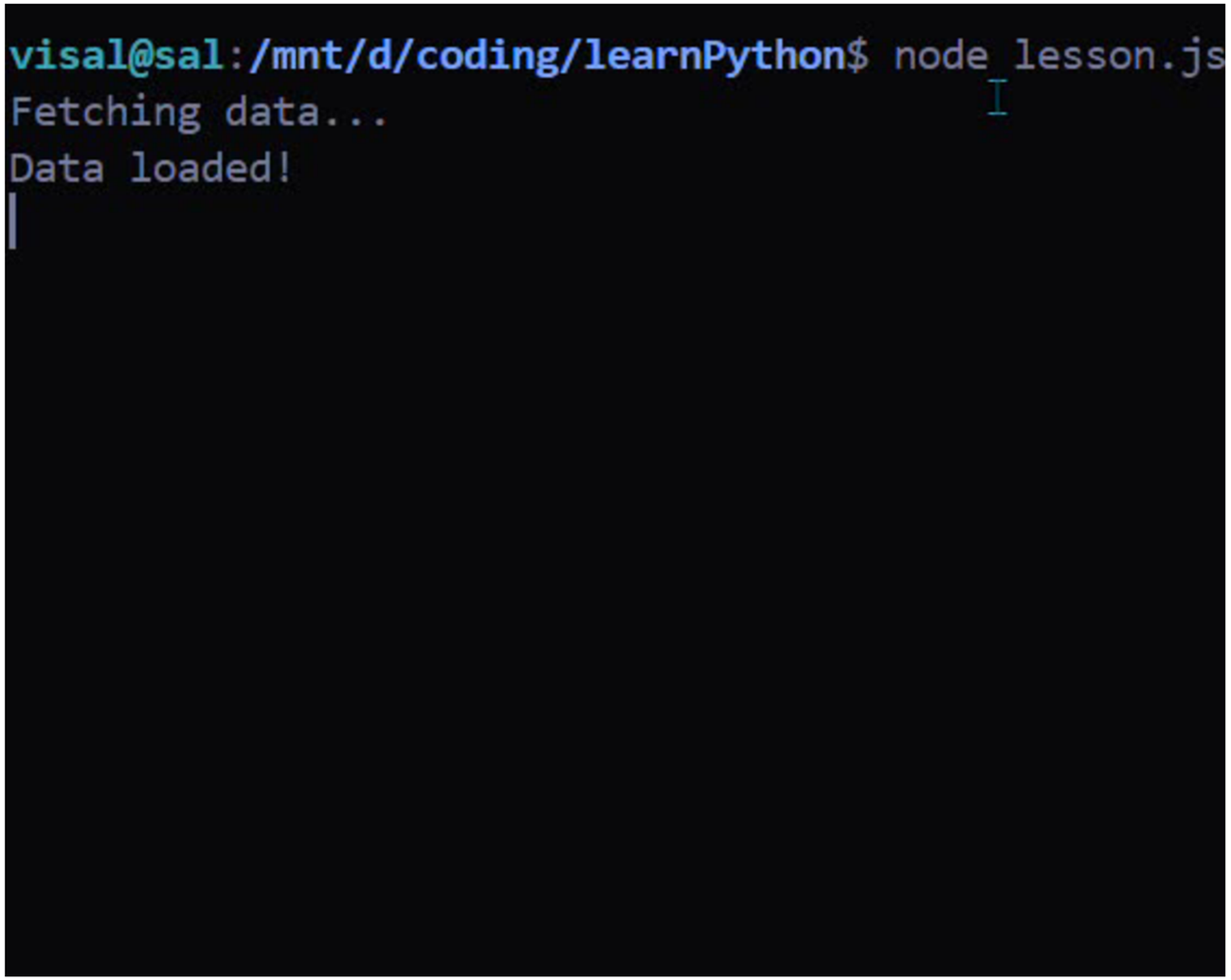
FETCHING DATA SIMULATION

```
//! This function takes 5s to complete
const fetchData = () => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve("Data loaded!");
        }, 5000);
    });
}

const demoAsyncAwait = async () => {
    console.log("Fetching data...");
    const result = await fetchData();
    console.log(result);
}

demoAsyncAwait();
```

FETCHING DATA SIMULATION



WHAT WE HAVE LEARNT

- All type of functions
- Arrow function
- Modules
- Template Literals
- Destructuring
- async/await

THANK YOU