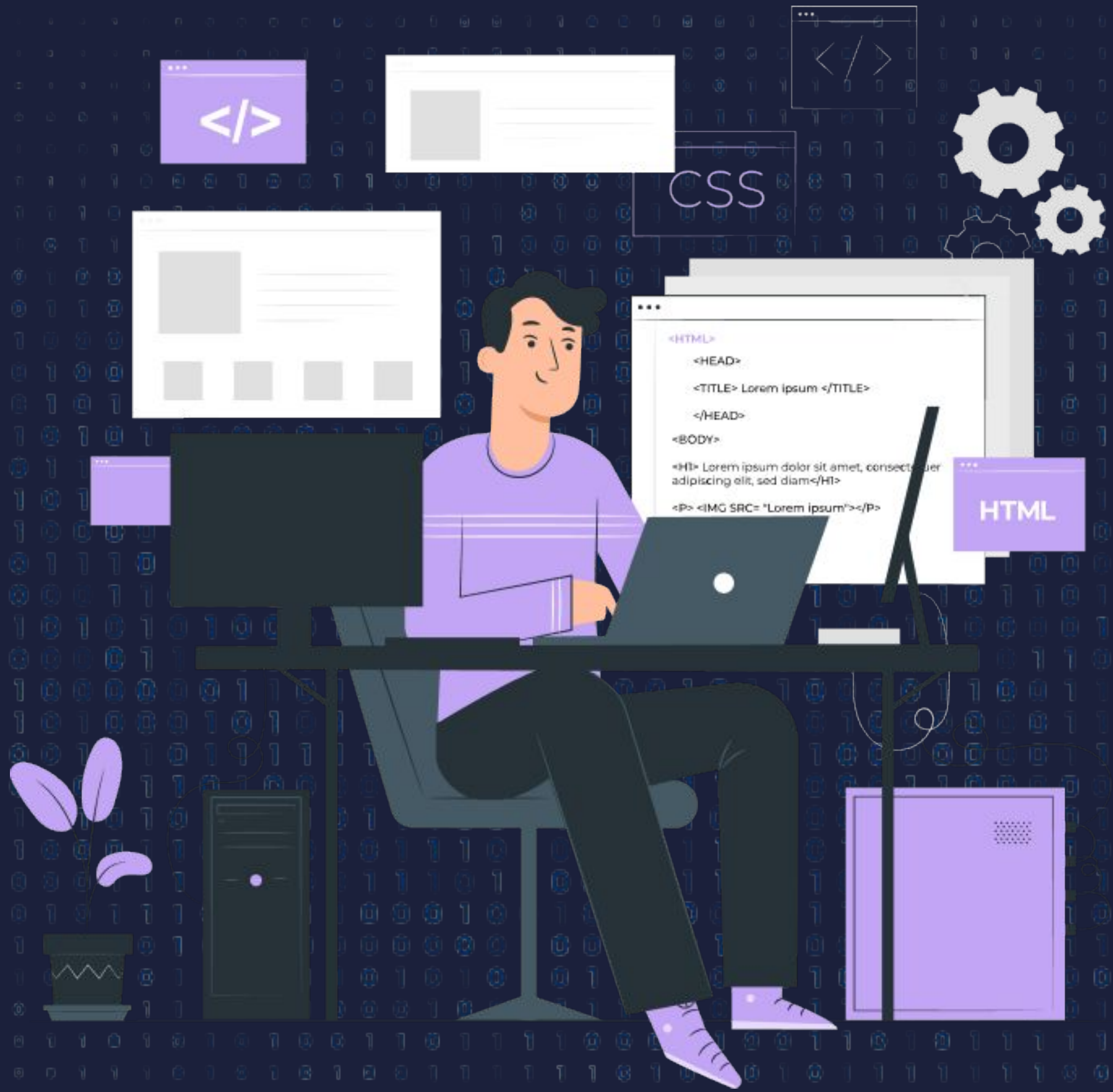
PW SKILLS

Destructuring, Aliasing, Destructuring nested objects



</>
CSS
<HTML>
<HEAD>
<TITLE> Lorem ipsum </TITLE>
</HEAD>
<BODY>
<H1> Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam</H1>
<P> <IMG SRC= "Lorem ipsum"></P>
HTML

Topics:

- Destructuring.
- Aliasing.
- Destructuring nested objects.

Destructuring

Destructuring in JavaScript is a convenient way to extract values from arrays or objects and assign them to variables in a concise manner. It allows you to unpack values from data structures, such as arrays or objects, into distinct variables. Destructuring can be done on arrays and objects using different syntaxes.
Here are examples of destructuring in JavaScript

Destructuring in arrays -

```
/*** Destructuring in array ***/
const numb = [1, 2, 3];
// destructuring an array -
const [one, two, three] = numb;
console.log(one); // 1
console.log(two); // 2
console.log(three); // 3
// Destructuring string into variable by converting
them to array
let [firstName, surName] = "llya Kantor".split(" ");
console.log(firstName); // llya
console.log(surName); // Kantor
// throw unwanted elements of the array via extra
comma.
let [firstName1, , title] = [
  "Julius",
  "Caesar",
  "Consul",
  "of the Roman Republic",
];
console.log(title); // Consul
```

```
// We can use it with any iterable.
let [a, b, c] = "abc";
console.log(a); // a
console.log(b); // b
console.log(c); // c
let [one1, two1, three1] = new Set([1, 2, 3]);
console.log(one1); // 1
console.log(two1); // 2
console.log(three1); // 3
// We can use any “assignables” at the left side. For
instance, an object property.
let user = {};
[user.name, user.surname] = "Alpha Beta".split(" ");
console.log(user.name); // Alpha
// Swapping values of two variables.
let firstName2 = "Alpha";
let surname1 = "Beta";
[firstName2, surname1] = [surname1, firstName2]; //
Swap values
console.log(`${firstName2} ${surname1}`); // Beta Alpha
```

Destructuring in Object -

```javascript
const user = {
  email: "john@gmail.com",
  userName: "john",};
const { email, userName } = user;
console.log(email); // email
console.log(userName); // john
/***** Destructuring in object ***/
const user = {
  email: "john@gmail.com",
  userName: "john",};
const { email, userName } = user;
console.log(email); // email
console.log(userName); // john
const employee = {
  empId: 1,
  userName1: "johndoe",};
// destructuring with default value
const { empId, userName1, depart = "None" } = employee;
console.log(empId); // 1
console.log(userName1); // johndoe
console.log(depart); // None
// Destructuring with default values and renaming
const car = {
```

```javascript
  name: "Toyoto",
  color: "Red",};
const { name: CarName, color: CarColor, owner = "John wick" } = car;
console.log(CarName); // Tayoto
// extracting specific properties using destructuring
const user1 = {
  name: "John",
  age: 33,
  city: "Bangalore",
  occupation: "Engineer",
  hobbies: ["coding", "traveling", "reading"],};
// extracting
const { name, age, occupation } = user1;
console.log(name); // john
console.log(age); // 33
console.log(occupation); // engineer
// Smart function parameters
// There are times when a function has many parameters, most of which are optional.
// In real-life, the problem is how to remember the order of arguments. Destructuring
// comes to the rescue!. We can pass parameters as an object, and the function immediately
```

Continue ..

```
// destructurizes them into variables.
function specialFun({ name, age, city, isAdmin =
false, ...otherProps }) {
  // Function logic using destructured variables
  console.log(`Name: ${name}`);
  console.log(`Age: ${age}`);
  console.log(`City: ${city}`);
  console.log(`Is Admin: ${isAdmin}`);
  console.log(`Other Properties:`, otherProps);}
// Call the function with an object argument
const user2 = {
  name: "John Doe",
  age: 30,
  city: "New York",
  isAdmin: true,
  profession: "Engineer",
  hobbies: ["reading", "coding"],};
specialFun(user2)
```

Aliasing

In javascript, aliasing with destructuring allows you to extract values from objects or arrays and assign them to variables, which allows you to assign the extracted values to variables with different names than the original property or array element names. This can be useful when you want to provide more meaningful or concise names for the variables.

Syntax -

```
// for object
const {OriPropName: aliasName} = Object

// for array
const {OriEleName: aliasName} = Array
```

Example of Aliasing in javaScript

```
// aliasing object
const data = {
  name: "Johnny",
  id: "1231412",
  email: "johny@gmail.com",
};
const { name: username } = data; // aliasing
console.log(username); // output - Johnny

// aliasing with arry

const user = ["johny", "john@gmail.com", "112211"];

const [userName, userEmail, userId] = user;

console.log({
  userName,
  userEmail,
  userId,
});
// output - { userName: 'johny', userEmail:
//'john@gmail.com', userId: '112211' }
```

Destructuring nested objects.

In javascript, destructuring of nested objects allows us to extract values from nested object structures and assign them to variables. We can used nested object destructuring to access and extract values from nested properties in a concise and readable manner,

Example of desturing neste

```
const user = {
  name: "Rama",
  email: "rama@gmail.com",
  id: "13343254",
  comments: {
    id: "2441413",
    date: "21 January 2023",
    post: "This is a demo example of post",
  },
};

const {
  name,
  email,
  comments: { post, date },
} = user;

console.log(name); // Rama
console.log(email); // rama@gmail.com
console.log(post); //  This is a demo example of post
console.log(date); // 21 January 2023
```

Aliasing can also be done with nested object destructuring

Example -

```
const user = {
  name: "Rama",
  email: "rama@gmail.com",
  id: "13343254",
  comments: {
    id: "2441413",
    date: "21 January 2023",
    post: "This is a demo example of post",
  },};
const {
  name,
  email,
  comments: { id: postId, post: postValue },
} = user;

console.log(name); // Rama
console.log(email); // rama@gmail.com
console.log(postValue); //This is a demo example of post
console.log(postId); // 2441413
```

SKILLS

THANK YOU