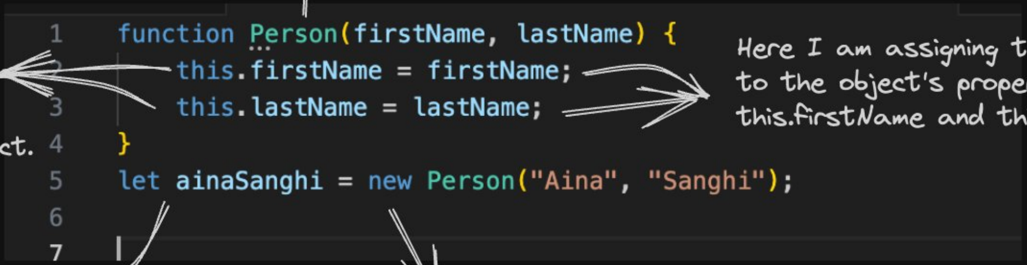
This function is like a blueprint for creating person objects.
It takes two parameters:
"firstName" and "lastName".

this keyword to refer to the object
I'm creating.
It's a way to set properties for the object.



```javascript
function Person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}
let ainaSanghi = new Person("Aina", "Sanghi");


```

Here I am assigning the values of firstName and lastName
to the object's properties
this.firstName and this.lastName.

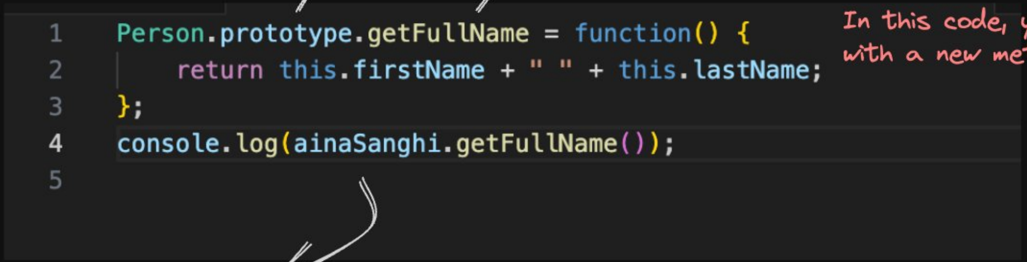
Here, I am creating a new person object
called ainaSanghi

To create a new person, I have used the "new" keyword followed
by the function Person
with the values "Aina" and "Sanghi" as arguments.

These values get passed to the firstName and lastName
parameters of the Person function.
As a result, a new person object is created with properties
firstName set to "Aina" and lastName set to "Sanghi".

"Person.prototype" is like a special place where you can add methods
(functions) that will be available to all objects
created from the Person constructor function.

The getFullName method is a function that combines the
firstName and lastName properties of a person object
and returns the full name as a single string.



```javascript
Person.prototype.getFullName = function() {
    return this.firstName + " " + this.lastName;
};
console.log(ainaSanghi.getFullName());

```

In this code, you're extending the Person object's prototype
with a new method called getFullName

After adding the getFullName method to the prototype,
you can use it with objects created from the Person constructor function.
In this case, you have a ainaSanghi object:

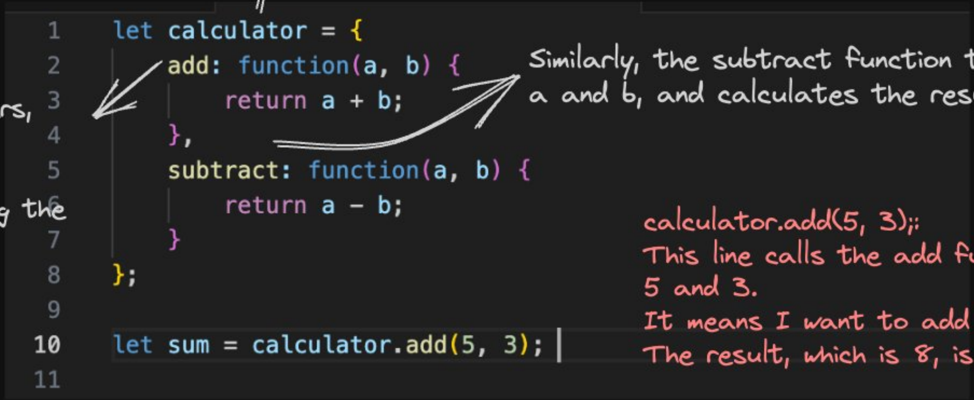
To get the full name of the ainaSanghi object,
you call the getFullName method on it using dot notation (ainaSanghi.getFullName()).

The method retrieves the firstName and lastName properties of the ainaSanghi object and
combines them to create the full name, which is "Aina Sanghi."

The console.log statement prints the result, "Aina Sanghi," to the console.

this is an object which contains two properties
"add" and "subtract". These properties hold functions.

The add function takes two parameters,
a and b.
Inside the add function,
it calculates the sum of a and b using the
+ operator and returns the result.



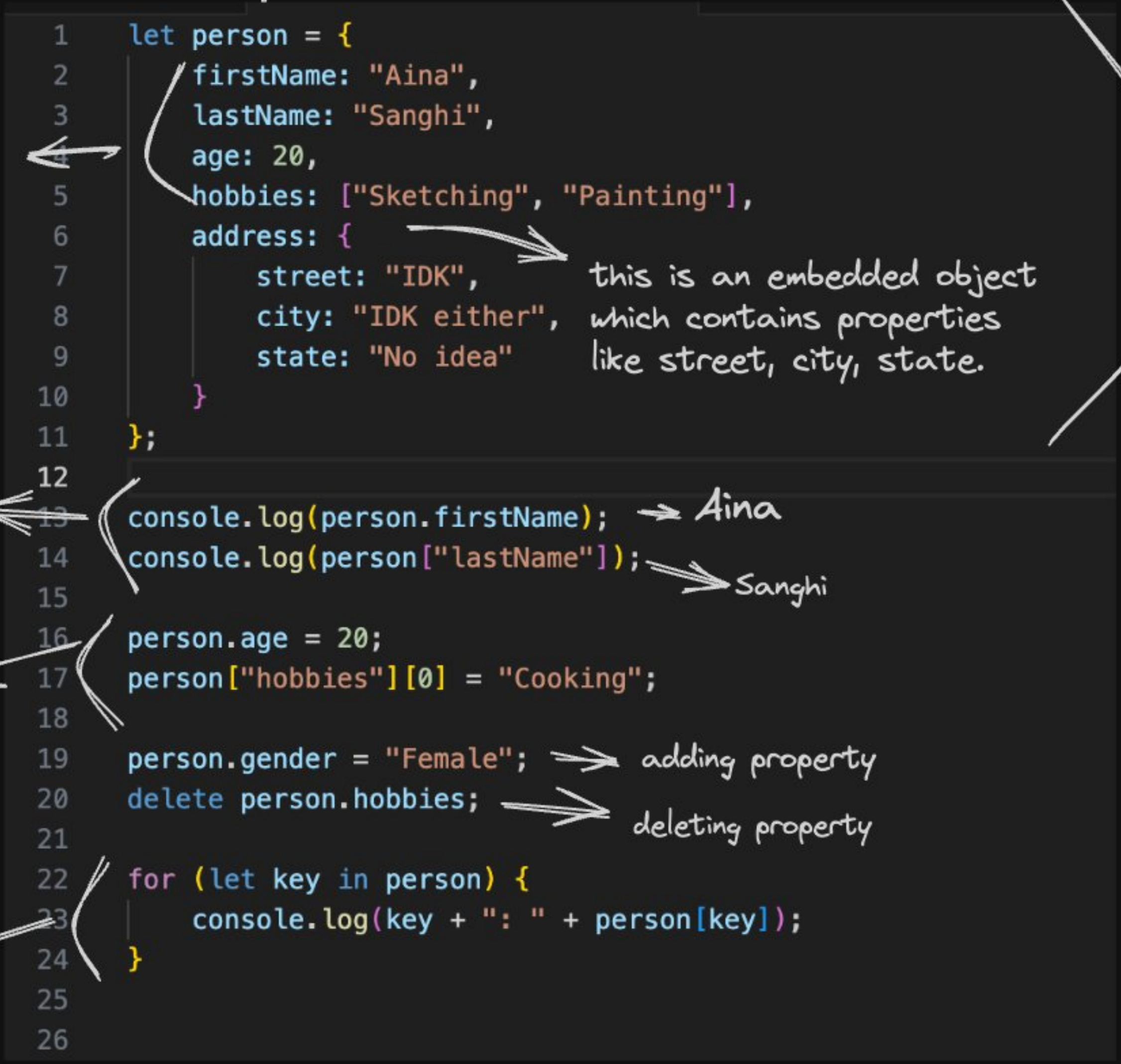
```javascript
let calculator = {
    add: function(a, b) {
        return a + b;
    },
    subtract: function(a, b) {
        return a - b;
    }
};

let sum = calculator.add(5, 3);
```

Similarly, the subtract function takes two parameters,
a and b, and calculates the result of subtracting b from a.

calculator.add(5, 3);:
This line calls the add function from the calculator object with the arguments
5 and 3.
It means I want to add 5 and 3 together.
The result, which is 8, is stored in the sum variable.

In this code, I've created a calculator object with functions for addition and subtraction.
When I use the add function with the numbers 5 and 3,
it calculates the sum, which is 8, and assigns it to the sum variable.

this is an object that contains various properties like firstName, lastName, age, hobbies



```
let person = {
  firstName: "Aina",
  lastName: "Sanghi",
  age: 20,
  hobbies: ["Sketching", "Painting"],
  address: {
      street: "IDK",
      city: "IDK either",
      state: "No idea"
  }
};

console.log(person.firstName);
console.log(person["lastName"]);

person.age = 20;
person["hobbies"][0] = "Cooking";

person.gender = "Female";
delete person.hobbies;

for (let key in person) {
    console.log(key + ": " + person[key]);
}
```

properties

this is an embedded object which contains properties like street, city, state.

creating an object

Accessing Object Properties

Aina

Sanghi

Modifying Object Properties

adding property

deleting property

Looping Through Object Properties