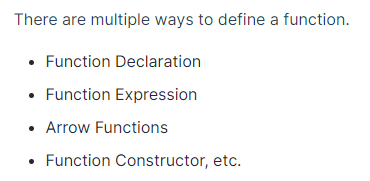
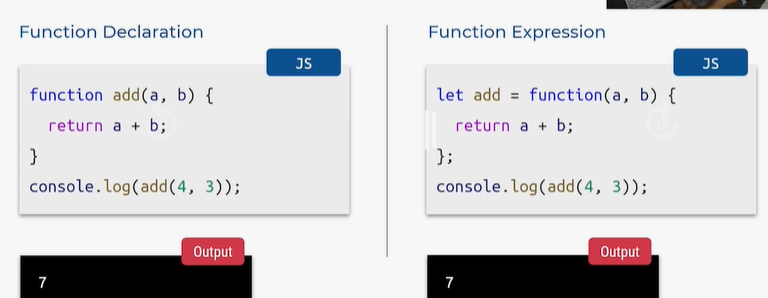
**FUNCTIONS CONCEPT IN JS**



Arrow functions types:

* Simple expression
* One parameter
* No parameter



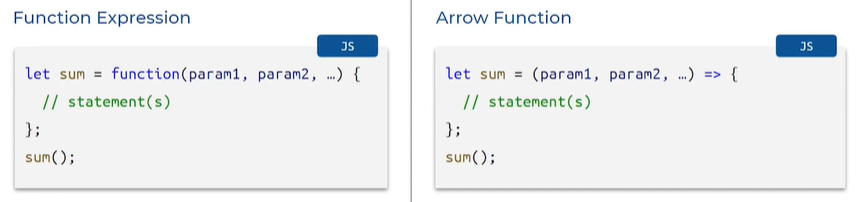
**FUNCTION DECLARATION:**

* Function = keyword
* Add = function name
* Inside brackets = passed arguments
* Inside curly brackets = block of code pettam with return keyword
* Add of ane method ni call chesam with values passed for above arguments inside console.log

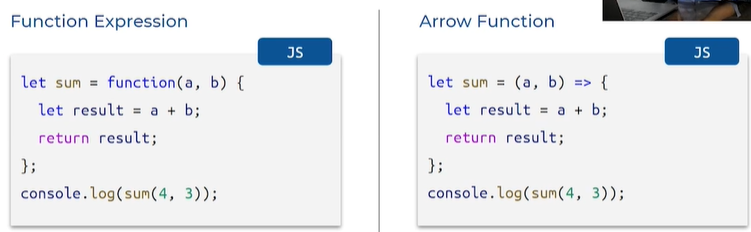
**FUNCTION EXPRESSION:**

* Let add = variable create chesam
* Function = keyword ki direct ga arguments pass chesam

**ARROW FUNCTION:**

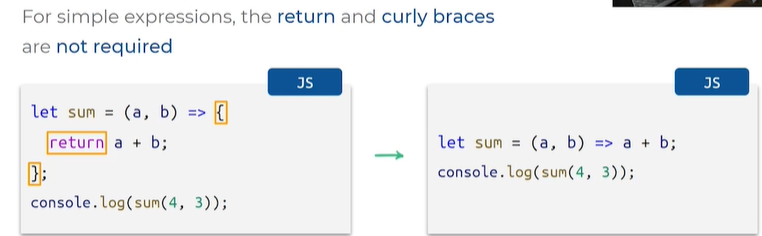
****

* Alternative to function expression
* Function keyword ledu straight away arguents
* Fat arrow => Arrow symbol
* petti block of code
* return ki variable create chesam

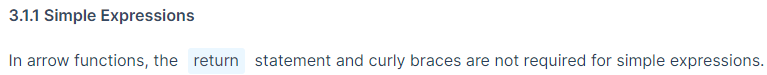


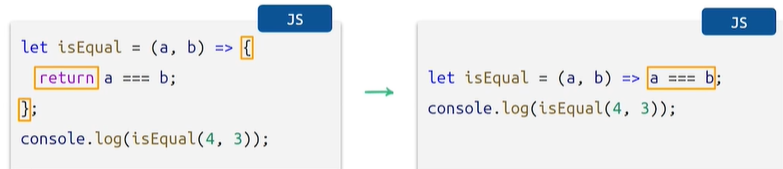
* Let Sum = variable
* a,b = arguments
* arrow function
* curly braces lo block of code
* result ki variable create chesam

**ARROW FUNCTION SIMPLE EXPRESSIONS:**

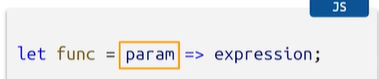
****

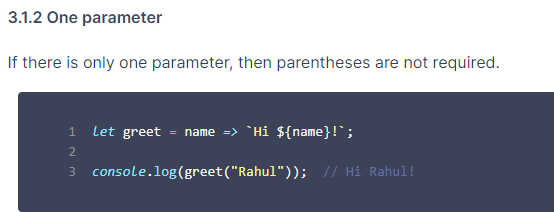
* Return thisesi st away a+b
* Simple expression we return only value
* Remove flower braces , return





**ARROW FUNCTION WITH ONE PARAMETER:**

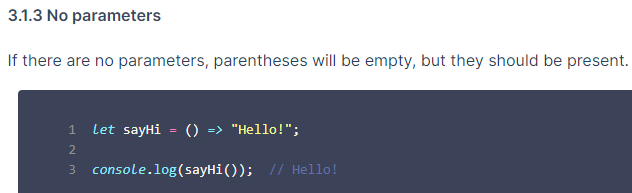




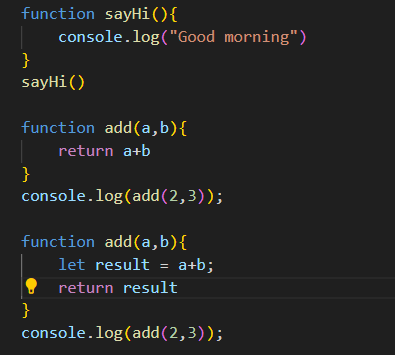


**ARROW FUNCTION WITH NO PARAMETER:**

* Brackets undali

****

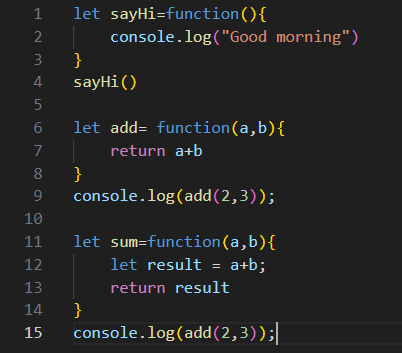
**FUNCTION DECLARATION:**



Defining and calling function

Return keyword usage

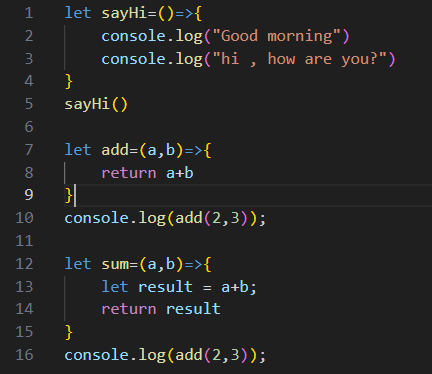
**FUNCTION EXPRESSION:**

****

Created variable with let

Return keyword usage

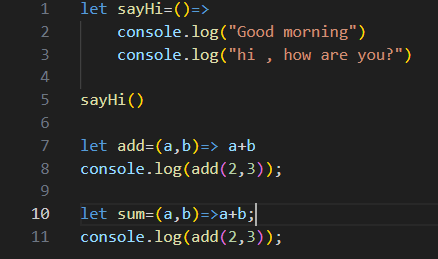
**ARROW FUNCTION:**

****

Same like function expression

Removed function keyword

**ARROW FUNCTION SIMPLE EXPRESSION:**

****

Same like arrow functions

Return and curly braces are not required

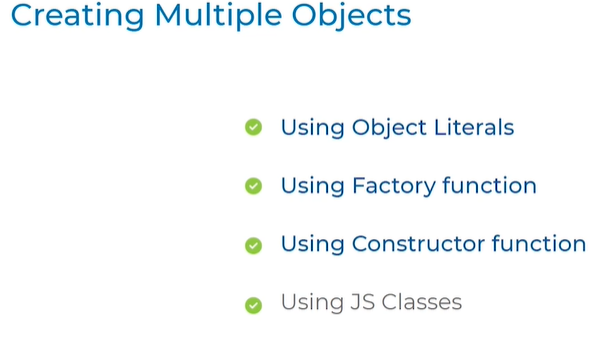
**ARROW FUNCTION ONE PARAMETER:**

* Okka value ni return cheyali ante use cheyali for eg:square

**ARROW FUNCTION NO PARAMETER:**

* Brackets undali kani danilo parameters(arguments/values)undav kani brackets undali

**FACTORY AND CONSTRUCTOR FUNCTIONS**

****

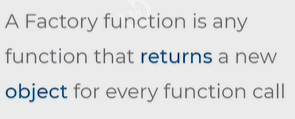
* Multiple objects can be created by using the above methods

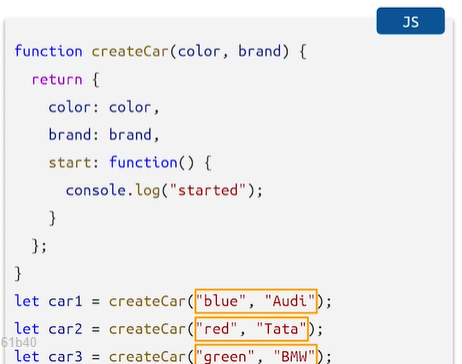
**1.Object literals:**

* Object is car
* With diff values and attributes

****

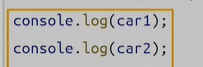
**2.using factory functions:**

****

****

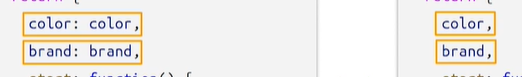
* Create car = function name
* Braket lo mana properties pass chesamu
* Value manam arguments laga kinda pas chesatham factory functions lo
* Return color property and colon arguments lo pass chese value
* To create object below car1 car2 car3 arguments passed

**Print objects in this way:**

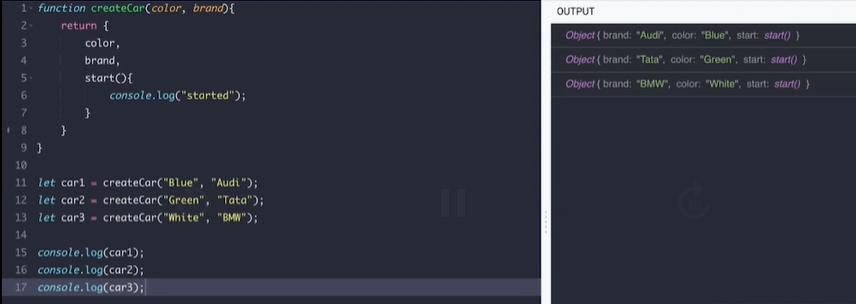
****

**SIMPLIED FACTORY FUNCTIONS:**

* Remove value and colon function
* Follow right side simplied syntax

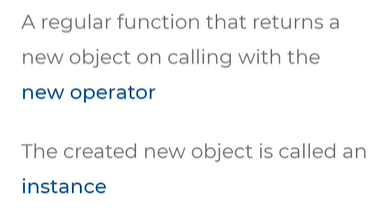
****

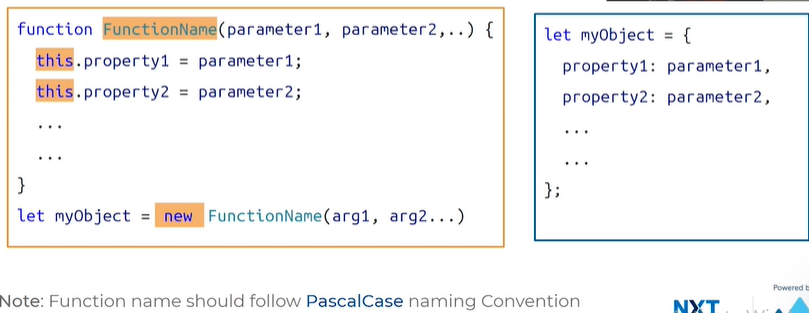
****

****

**CONSTRUCTOR FUNCTIONS:**

New operator is used to call function with objects which return new objects [ also called instance]

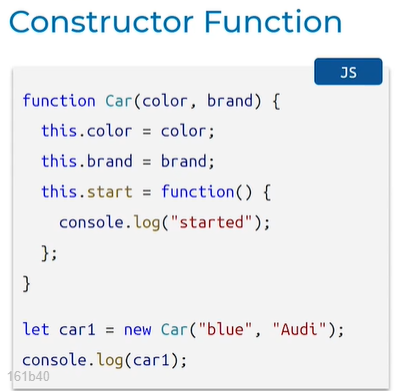
****

****

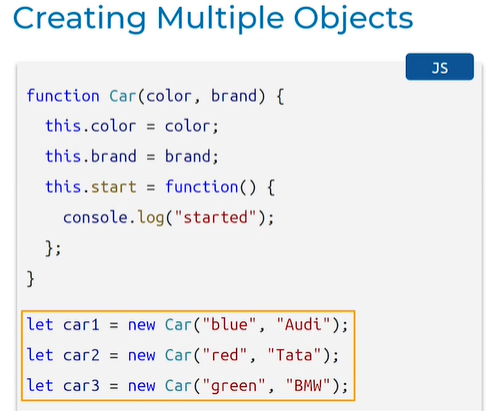
Function follows = pascal case 1st letter is upperletter every time

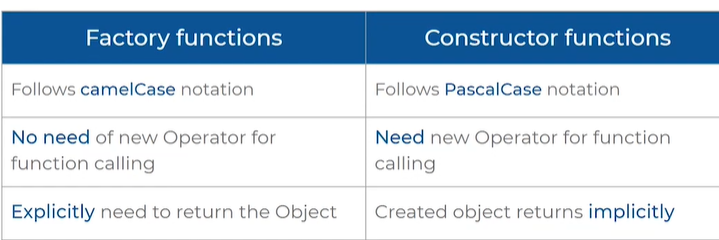
Arguments = property values

Proeprty ni set cheyadanki this.porprty 1 and dani value



* Car = function
* New car = function call passed parameters
* Values given
* New = this variable created = assign empty
* This.property = this object with dot notation assigndffi vales for properties
* New ante return avthni







.length = returns property

.typeOf = function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **declaration** | **expression** | **Arrow** | **Simple.exp** | **One parameter** | **No parameter** |
| **Return keyword** | **Factory** | **Constructor**  **This.**  **New** |  |  |  |



Area = length \* width

Convert degree Fahrenheit to degree Celsius:

return (5 / 9) \* (fahrenheit - 32);

function toCelsius(Fahrenheit){

    result =  (5/9) \* (Fahrenheit -32);

}

// console.log(toCelsius(10)); defined and called function

//declaring variable

let Fahrenheit = 10;

console.log(`${Fahrenheit} is the value of`,toCelsius(Fahrenheit) ,'degree celsius');

//tocelsius ane function fahrenheit ane variable declare chesina value tho call chesam

Numeric number convert to days : formula

**Math.floor(seconds / (3600 \* 24)).**