**Arrow Function:**

arrow function syntax uses the => operator to define the function. Arrow functions are often shorter and more concise than anonymous functions.

anonymous function is simply a function without a name.

Arrow functions can often be more concise and easier to read than anonymous functions.

**Example:**

// Syntax of the arrow function

// Remove the function keyword only

// Add arrow symbole

document.addEventListener('click', () => {

  alert('Click Event Happened');

});

In arrow function, if the body (starts with { and ends with })of the arrow function sits on a single line then so we do-not need the curly brackets surrounding the body.

document.addEventListener('click', () => alert('Click Event Happened'));

**Features of the arrow function:**

1)When the body of the arrow function sits on a single line, arrow function will automatically return what ever is put after =>

Example:

let my\_array = [1, 2, 3, 4];

// map method is a higher order function so it expects a function inside ()

// We are adding anonymous function to the () of map method

// map method will loop through the elements of my\_array array

// map method returns a new array

// Including anonymous function inside the () of map method

//x represnts the current element of my\_array corresponding to current call.

console.log(

  my\_array.map(function (x) {

    // Returning 2\*x inside the body of the function

    return 2 \* x;

  })

);

// Converting anonymous function into arrow function

console.log(

  my\_array.map((x) => {

    return 2 \* x;

  })

);

// Making the body of the anonymous sit on same line, so we remove reurn {} of the body

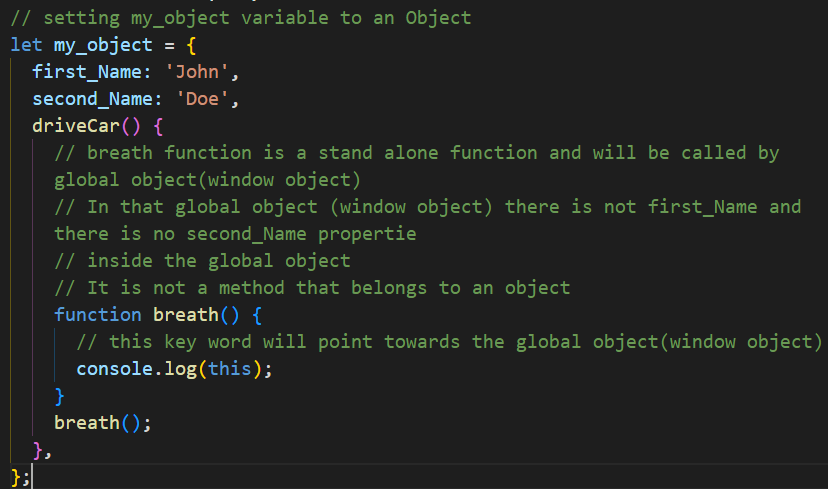
// of arrow function and return key word of the arrow function and anonymous function

// For single parameter we donot need () around x, if we have multiple parameters or zero parameters then we would need ()

console.log(my\_array.map((x) => 2 \* x));

2)Arrow functions do not have this key word; the way regular functions have.

If we use arrow function instead of a regular function inside a method, then this key word will point towards the object that is containing the method, this key word will not point towards the global object(window object)



Arrow function will look one level up as they do not have this keyword (inside body of the driveCar method, this key word exists and is pointing towards the object as driveCar method is called by object using. and ()) for this key word.

my\_object.driveCar();

**Overall code:**

// setting my\_object variable to an Object

let my\_object = {

  first\_Name: 'John',

  second\_Name: 'Doe',

  driveCar() {

    // breath function is a stand alone function and will be called by global object(window object)

    // In that global object (window object) there is not first\_Name and there is no second\_Name propertie

    // inside the global object

    // It is not a method that belongs to an object

    breath();

    function breath() {

      // this key word will point towards the global object(window object)

      console.log(this);

    }

    // Converting breath standalone regular function into an arrow function

    // setting arrow function to a variable like breating  and using name of that variable and () like breating() to call the arrow function

    let breathing = () => {

      // // If we use arrow function instead of regular function inside object's method

      // then this key word will point towards the object

      console.log(this);

    };

    breathing();

  },

};

my\_object.driveCar();

Converting function into arrow function and console logging this key word:

// Converting breath standalone regular function into an arrow function

    // setting arrow function to a variable like breating  and using name of that variable and () like breating() to call the arrow function

    let breathing = () => {

      // // If we use arrow function instead of regular function inside object's method

      // then this key word will point towards the object

      console.log(this);

    };

    breathing();