INFO 2302 Web Technologies

JavaScript Function & Object

Dr. Najhan M.Ibrahim
Department of Information Systems
Kulliyyah of Information and Communication Technology
International Islamic University Malaysia

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
Garden of Knowledge and Virtue

LEADING THE WAY
KHALĪFAH • AMĀNAH • IQRA' • RAḤMATAN LIL-ĀLAMĪN

Function

- A JavaScript function is a block of code designed to perform a particular task.

- A JavaScript function is executed when "something" invokes it (calls it).

There are two main types of function

1- Predefine function ex. Max(), Min(), ramdom()

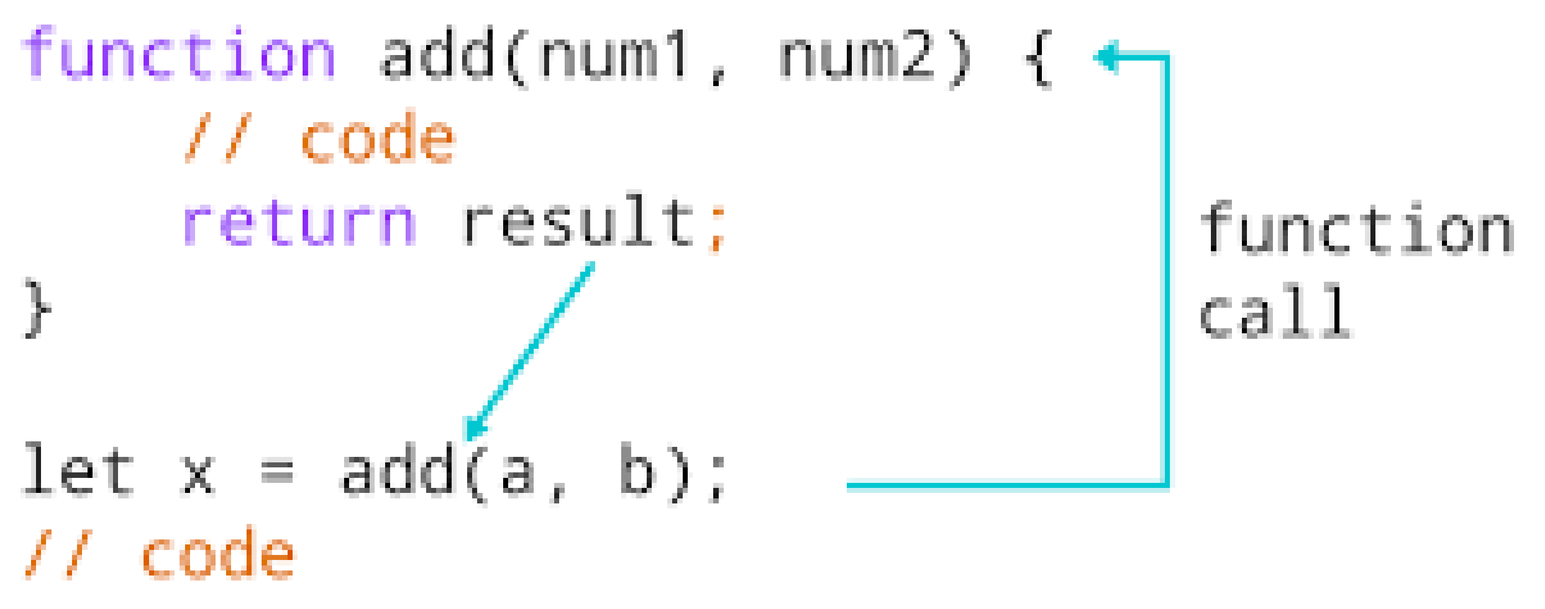
2- User define function

JavaScript Function Syntax

- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

```
function name (parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Example



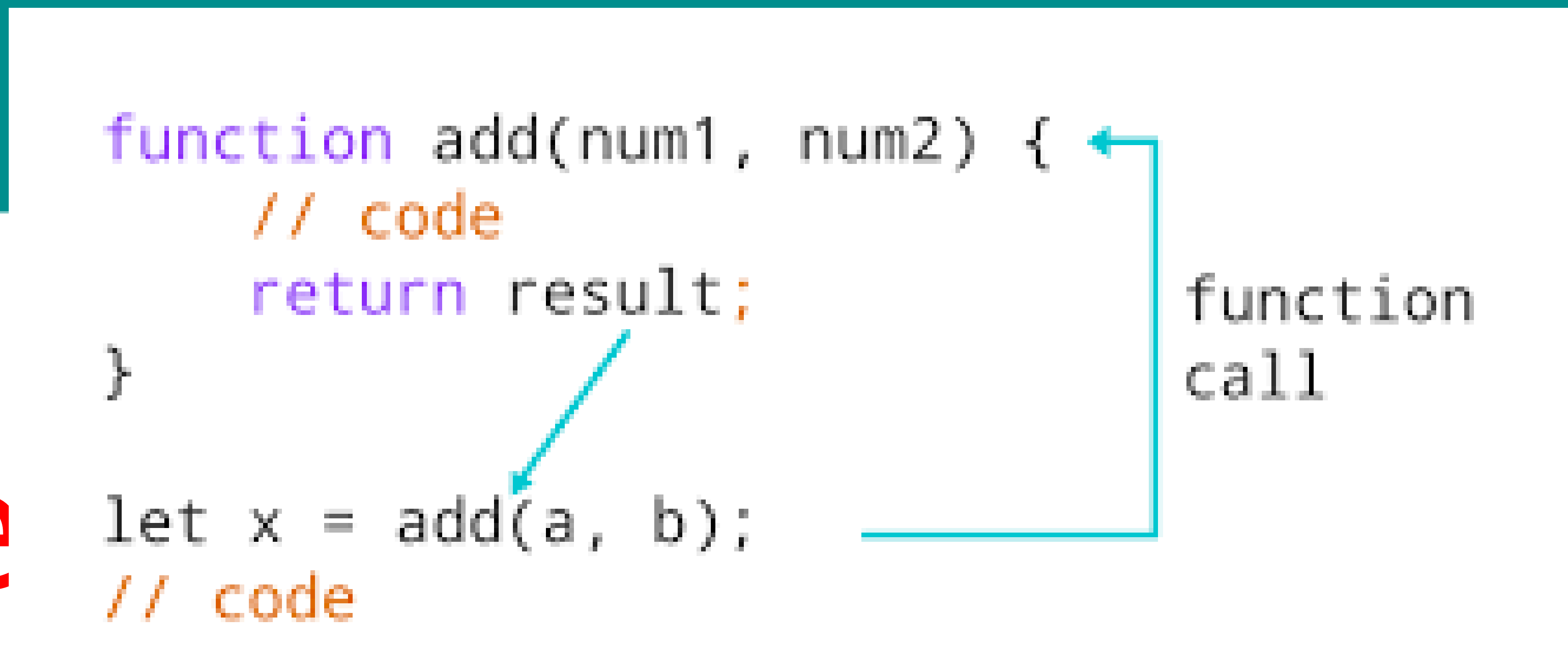
```
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

```
function multiple(p1, p2)
{
  return p1 * p2;
// The function returns the product of p1 and p2
}
```

Multiple(2,3) = 6

Multiple(5,6)

Function Invocation (Call)



```
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

- The code inside the function will e
  "something" invokes (calls) the function

1- When an event occurs (when a user clicks a button)

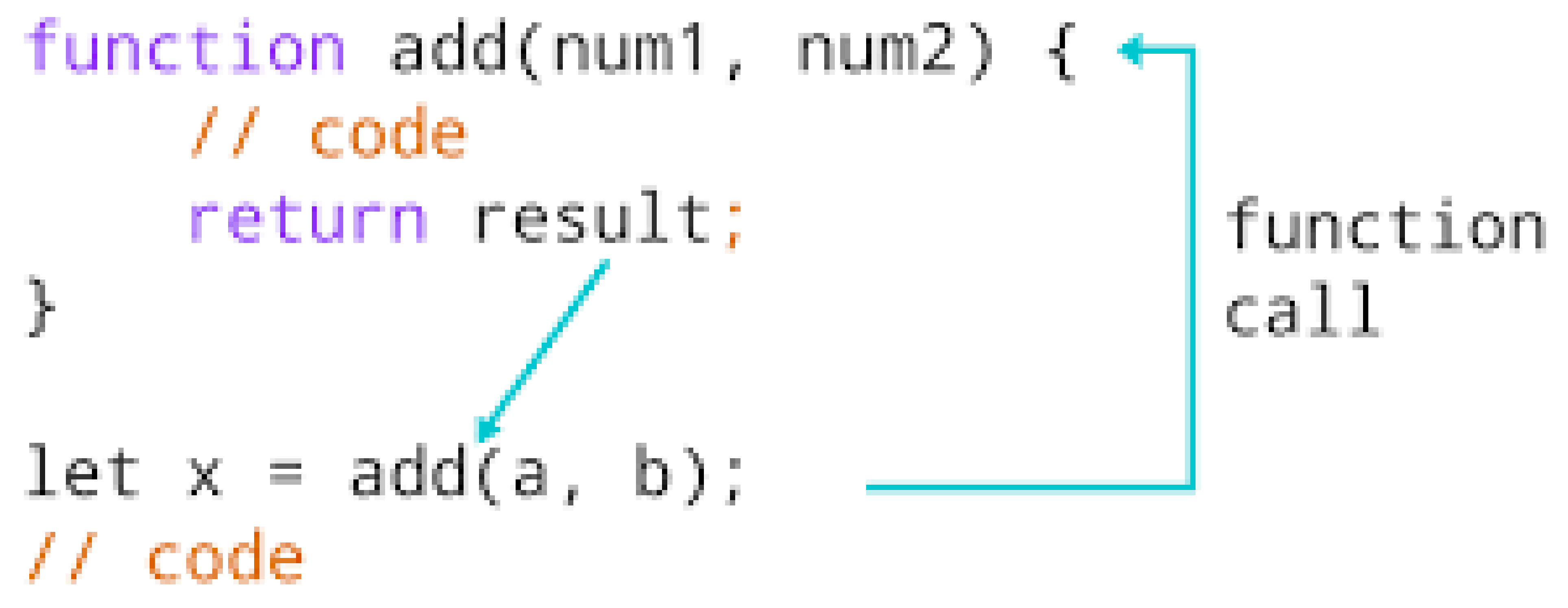
2- When it is invoked (called) from JavaScript code

3- Automatically (self invoked)

Functions – reusable block of codes

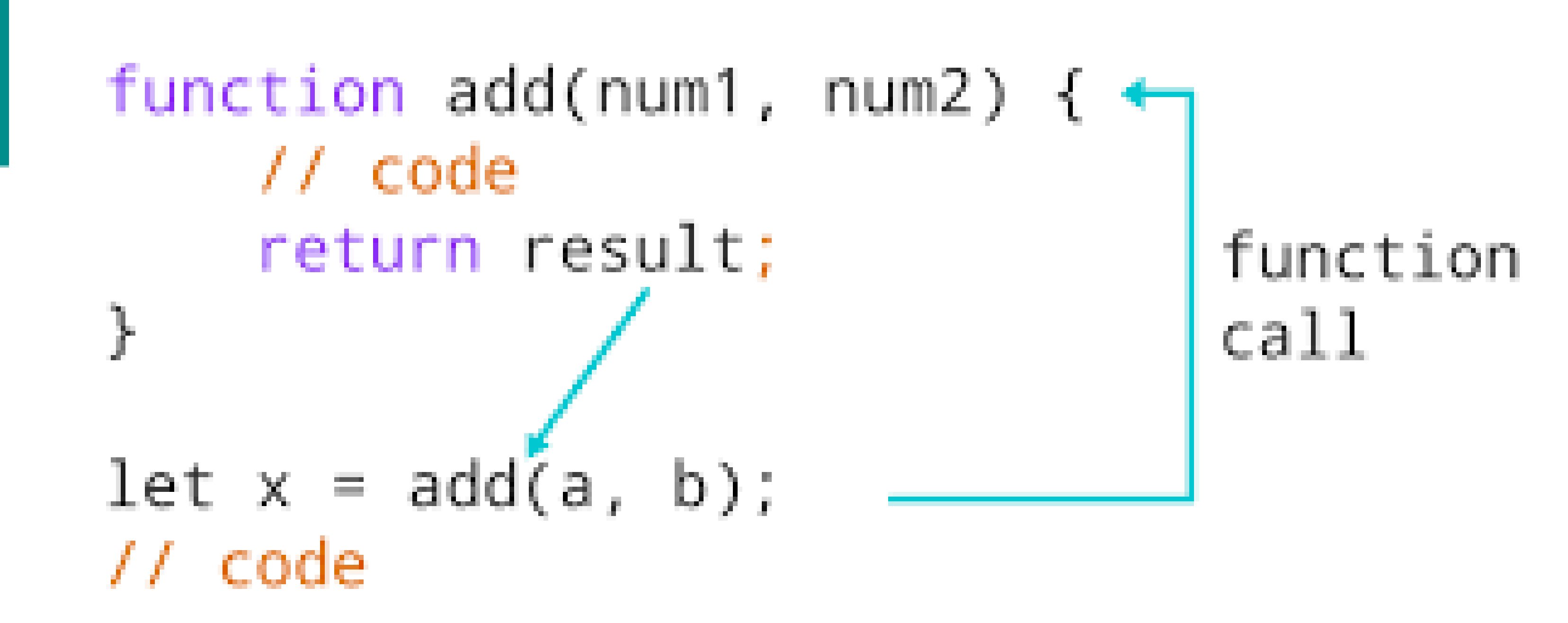
- Functions allow you to store a piece of code that does a single task inside a defined block.

- You can call that code whenever you need (as many as you need) it using a single short command — rather than ha
the same code multiple tim



```
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

Function parameters



```
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

- Parameters - values that need to be i
  the function.

- Sometimes known as arguments, properties or attributes.

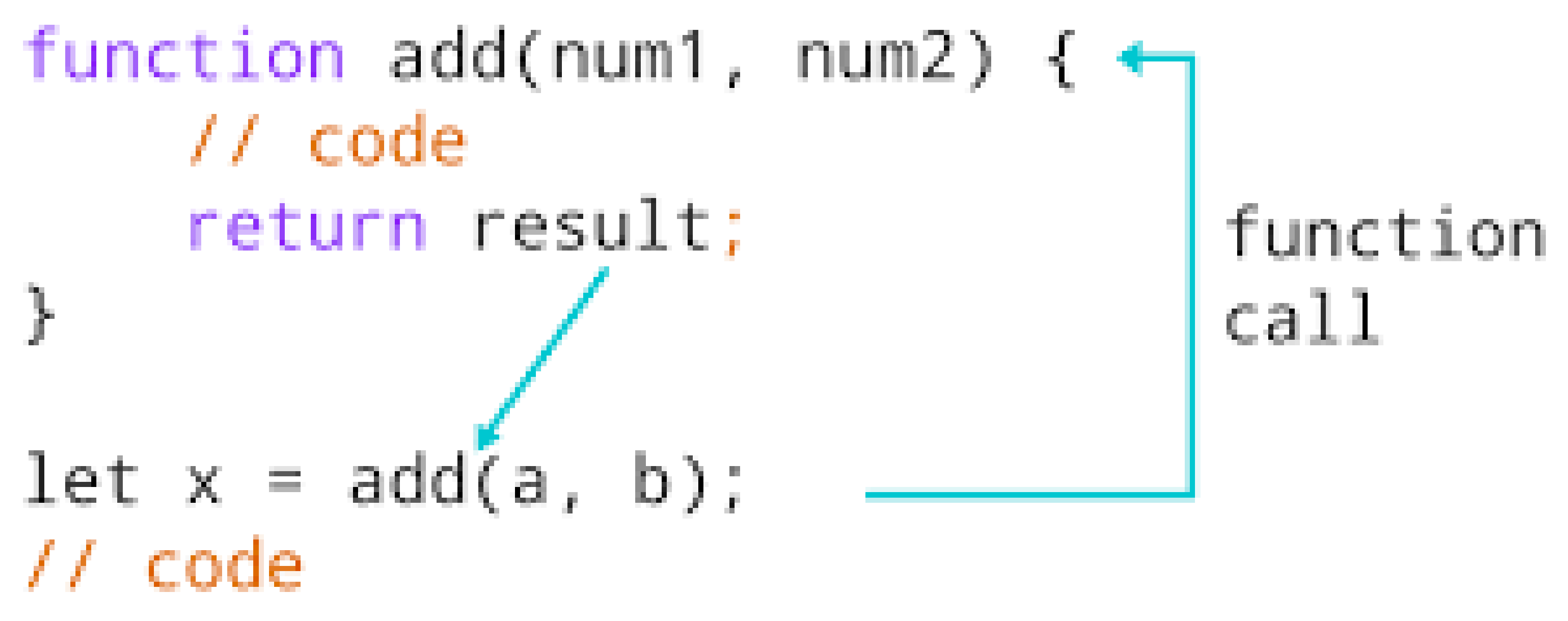
- Ex No parameter:- var myNumber = Math.random();

- Ex Two parameters:-
- var newString = myText.replace('string', 'sausage');

Function scope and conflicts

- **Local scope**: When you create a function, the variables and other things defined inside the function are inside their own separate scope

- **Global scope**: The top level outside all your functions is called the global scope. Values defined in the global scope are everywhere in the code.



```
function add(num1, num2) {
    // code
    return result;
}

let x = add(a, b);
// code
```

Example: What will be display?

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p>This example calls a function which performs a calculation and returns the result:</p>

<p id="demo"></p>

<script>
var x = myFunction(4, 3);
document.getElementById("demo").innerHTML = x;

function myFunction(a, b) {
  return a * b;
}

function myFunction(a, b) {
  return a + b;
}
</script>

</body>
</html>
```

Function Return

- Return values - Values returned by the function when it completes .

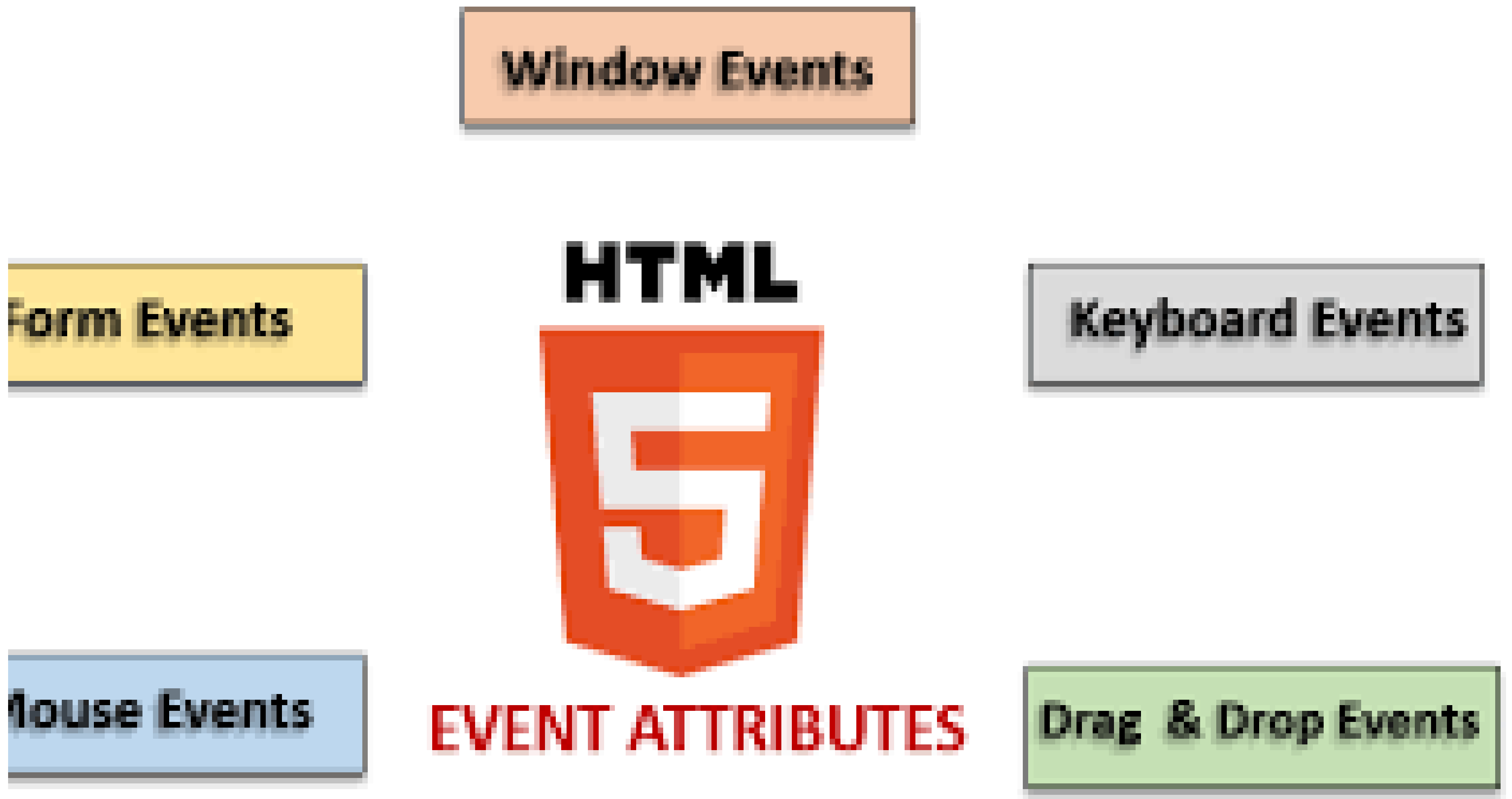
Example

```
var x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

LEADING THE WAY
KHALĪFAH • AMĀNAH • IQRA' • RAḤMATAN LIL-ĀLAMĪN

- HTML events are "**things**" that happen to HTML elements.

- When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.



- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

- *<element event='**some JavaScript**'>*
- *<element event="**some JavaScript**">*

```
<button

onclick="document.getElementById('demo').innerHTML = Date()">
The time is?

</button>
```

Common HTML Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JavaScript Objects

Real Life Objects, Properties, and Methods

In real life, a car is an object

A car has properties like weight and colour, and methods like start and stop:

Object	Properties	Methods

	car.name = Fiat
	car.model = 500
	car.weight = 850kg
	car.color = white
		car.start()
		car.drive()
		car.brake()
		car.stop()

All cars have the same properties, but the property values differ from car to car.

All cars have the same methods, but the methods are performed differently.

Objects

- You have already learned that JavaScript variables are containers for data values.
- Ex. This code assigns a **simple value** (Fiat) to a **variable** named car:
- var car = "Fiat";

```
function Book () [illegible]
{
  this.property = "value";
  this.method = function () { }; [illegible]
  this.method = method; [illegible]
}

Book.prototype.method() {} [illegible]

let Book = new Book(); [illegible]
```

JavaScript Objects cont..

- Objects are also variables. But objects can contain variety of values.

- This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

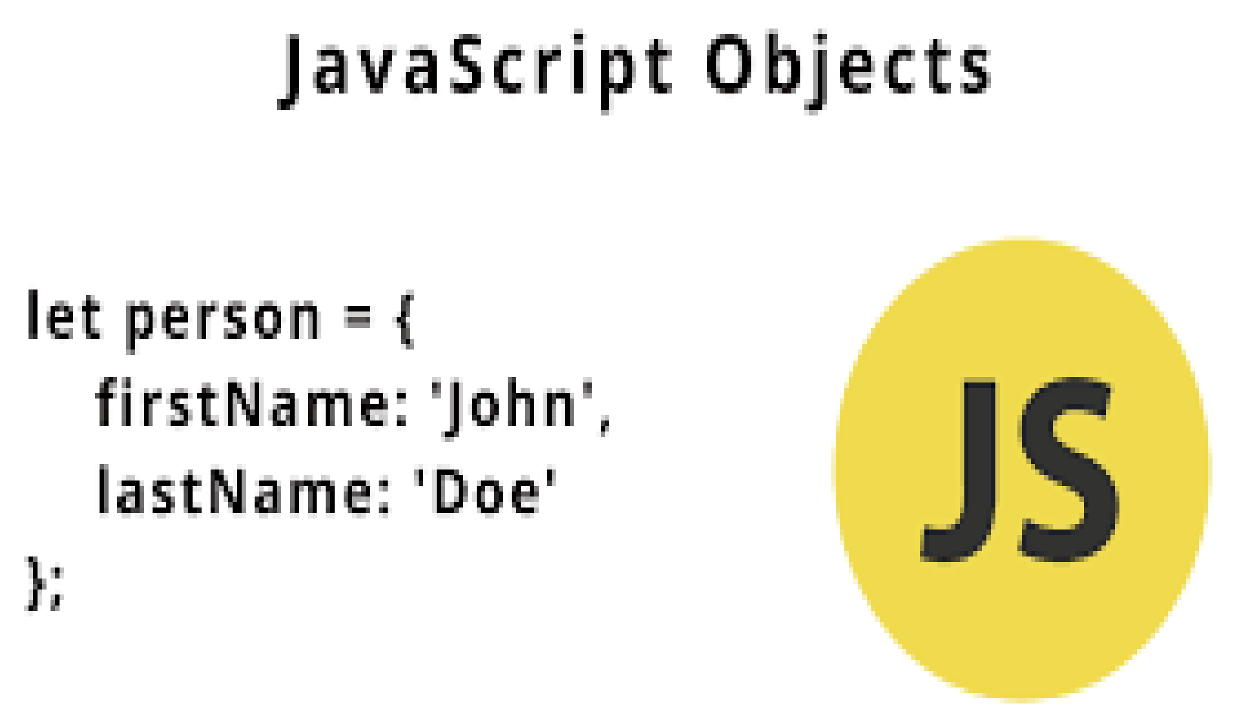
- var car = { type:"Fiat", model:"500", color:"white” };

- You define (and create) a JavaScript object with an object literal:

- Ex, var person = { firstName:"John", lastName:”Ali", age:50, eyeColor:"blue"};

- Spaces and line breaks are not important. An object definition can span multiple lines:

- var person = {
  firstName: "John",
  lastName: ”Ali",
  age: 50,
  eyeColor: "blue"
  };


JavaScript Objects

```
let person = {
   firstName: 'John',
   lastName: 'Doe'
};
```

Object Properties

- The **name:values** pairs in JavaScript objects are called **properties**:

Property	Property Value
firstName	John
lastName	Ali
age	50
eyeColor	blue

- You can access object properties in two ways:

- *objectName.propertyName,*
- *Ex,* person.lastName;

- *Or*

- *objectName["propertyName"],*
- *Ex* person["lastName"];

Object Methods

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- ion

c

Property	Property Value
firstName	John
lastName	Ali
age	50
eyeColor	blue
fullName	**function**() { return this.firstName + " " + this.lastName;}

Example

```
•var person = {
   firstName: "John",
   lastName : ”Ali",
   id       : 5566,
   fullName : function() {
     return this.firstName + " " + this.lastName;
   }
 };
```

The this Keyword

- In a function definition, this refers to the "owner" of the function.

- In the example above, this is the **person object** that "owns" the fullName function.

- In other words, this.firstName means the firstName property of **this object**.

Accessing Object Methods

- You access an object method with the following syntax:

- *objectName.methodName()*

- *Ex,* name = person.fullName();

Example, accessing object

```
•<!DOCTYPE html>
•<html>
•<body>

•<h2>JavaScript Objects</h2>

•<p id="demo"></p>

•<script>
•// Create an object:
•const car = {type:"Fiat", model:"500", color:"white"};

•// Display some data from the object:
•document.getElementById("demo").innerHTML = "The car type is " + car.model;
•</script>

•</body>
•</html>
```

Example, this keyword

- <h2>JavaScript Objects</h2>
- <p>An object method is a function definition, stored as a property value.</p>

- <p id="demo"></p>

- <script>
- // Create an object:
- const person = {
-   firstName: ”Mohd",
-   lastName: ”Ali",
-   id: 5566,
-   fullName: function() {
-     return this.firstName + " " + this.lastName;
-   }
- };

- // Display data from the object:
- document.getElementById("demo").innerHTML = person.fullName();
- </script>

Thank You
Merci
Asante
Arigato
Spasibo
Obrigado
Multumesc
Salamat
Kia Ora
Matondo
Terima Kasih
Dank Je
Kiitos