### **1. Function Declaration**

**Syntax**:  
function functionName(parameters) {

// function body

return value; // optional

}

* **Definition**: A function declaration is a named function that can be called anywhere in the scope (even before it’s defined) due to hoisting.

**Example**:  
function greet(name) {

return "Hello, " + name;

}

### **2. Function Expression**

**Syntax**:  
const functionName = function(parameters) {

// function body

return value; // optional

};

* **Definition**: A function expression defines a function and assigns it to a variable. It can only be called after it’s defined, as it’s not hoisted.

**Example**:  
const greet = function(name) {

return "Hello, " + name;

};

### **3. Arrow Function**

**Syntax**:  
const functionName = (parameters) => {

// function body

return value; // optional, implicit if only one line

};

* **Definition**: An arrow function provides a shorter syntax and doesn’t have its own this, arguments, or super. Commonly used in callbacks and when working with anonymous functions.

**Example**:  
const greet = (name) => "Hello, " + name;

### **4. Anonymous Function**

**Syntax**:  
function(parameters) {

// function body

}

* **Definition**: A function without a name, often used as arguments for other functions or immediately invoked.

**Example**:  
setTimeout(function() {

console.log("Hello");

}, 1000);

### **5. Immediately Invoked Function Expression (IIFE)**

**Syntax**:  
(function(parameters) {

// function body

})();

* **Definition**: An IIFE is a function that runs immediately after it’s defined, often used to create a local scope.

**Example**:  
(function() {

console.log("This runs immediately!");

})();

### **6. Generator Function**

**Syntax**:  
function\* generatorFunction() {

yield value;

}

* **Definition**: A generator function is a special function that can pause and resume its execution. It returns a generator object and is identified by the function\* syntax.

**Example**:  
function\* numbers() {

yield 1;

yield 2;

yield 3;

}

### **7. Constructor Function**

**Syntax**:  
function ConstructorFunction(parameters) {

this.property = value;

}

* **Definition**: A constructor function is used to create objects. It is usually capitalized and used with the new keyword.

**Example**:  
function Person(name, age) {

this.name = name;

this.age = age;

}

const person1 = new Person("Alice", 25);

### **8. Async Function**

**Syntax**:  
async function functionName(parameters) {

await expression;

}

* **Definition**: An async function always returns a Promise and allows the use of await to handle asynchronous code in a cleaner, synchronous-looking way.

**Example**:  
async function fetchData() {

const response = await fetch('url');

const data = await response.json();

return data;

}