JavaScript Functions — Complete Guide for Beginners

Welcome! This document explains **JavaScript functions** in a clear, structured way—similar to your Conditional & Looping PDF filecite turn0file0 . Functions are the quiet engines of JavaScript; they let your code reuse logic, stay organized, and behave like a well-trained troupe of performers.

---

1. What Are Functions?

A **function** is a reusable block of code designed to perform a specific task.

In simple terms:

```
"Give me input, I'll give you output. Sometimes I just do a task quietly."
```

Functions help: - avoid repeating code - make programs modular and readable - handle input and return values

---

2. Function Syntax

Function Declaration

```
function greet() {
  console.log("Hello!");
}
```

- Can be called *before* they're defined (hoisted)

Function Expression

```
const greet = function () {
  console.log("Hello!");
};
```

- Not hoisted

Arrow Function (ES6)

```
const greet = () => {
  console.log("Hello!");
};
```

- Shorter syntax; great for callbacks

Function with Parameters and Return Value

```
function add(a, b) {
  return a + b;
}
```

3. When to Use Functions

Use functions when: - Code needs to be reused - Logic belongs together - You want modular, clean structure - A certain task needs abstraction (e.g., authentication logic)

---

4. Examples

Example 1: Basic Function

```
function sayHello() {
  console.log("Hello World");
}
```

Used anywhere you want repeated messages.

Example 2: Function with Inputs

```
function welcome(name) {
  console.log(`Welcome, ${name}!`);
}
```

Used for greeting users.

Example 3: Function Returning Output

```
function square(num) {
  return num * num;
}
```

Used in math apps.

Example 4: Arrow Function

```
const multiply = (x, y) => x * y;
```

Used in modern JS codebases.

Example 5: Default Parameters

```
function power(base, exp = 2) {
  return base ** exp;
}
```

Useful for optional input.

---

5. Types of Functions

1. Pure Functions

- No side effects
- Same input → Same output

2. Impure Functions

- Modify external data or depend on it

3. Callback Functions

Functions passed as arguments.

```
setTimeout(() => console.log("Done"), 1000);
```

4. Higher-Order Functions

Functions that *return* or *receive* other functions.

5. Recursive Functions

Functions that call themselves.

```
function countdown(n) {
  if (n === 0) return;
  console.log(n);
  countdown(n - 1);
}
```

---

6. Project-Level Example

User Authentication Simulation

```
function isValidUser(username, password) {
  return username === "admin" && password === "1234";
}

function login(username, password) {
  if (isValidUser(username, password)) {
    console.log("Login successful");
  } else {
    console.log("Invalid credentials");
  }
}

login("admin", "1234");
```

Used in backend systems.

---

7. Interview-Level Questions

1. Difference between function declaration and function expression.
2. What is hoisting in functions?
3. What is a pure function?
4. Explain callback functions.
5. What is a closure?
6. What is the difference between `call`, `apply`, and `bind`?

---

8. Real-Time Example

Cart Price Calculator

```
function calculateTotal(cart) {
  let total = 0;
  for (let item of cart) {
    total += item.price * item.qty;
  }
  return total;
}

const cart = [
  { price: 100, qty: 2 },
  { price: 50, qty: 1 }
];

console.log("Total:", calculateTotal(cart));
```

Used in e-commerce applications.

---

9. Key Takeaways

- Functions organize logic
- They help reuse code
- Arrow functions are concise but behave differently with `this`
- Pure functions make code easier to test
- Practice writing functions daily

---

10. Practice Tasks

1. Write a function to check if a number is even.
2. Write a function that returns the largest of three numbers.
3. Create a function that greets a list of students.
4. Write a recursive function to calculate factorial.
5. Build a small calculator using functions.

If you'd like, I can convert this into **PDF**, **add visuals**, or **match the exact styling** of your original document.