



Education
@topdev_media



JavaScript

#1/2



Functions

Types

Save it for later !

Function Declaration

```
function greet(name) {  
  return `Hello, ${name}`;  
}
```

-  Hoisted: can be called before it's defined.
-  this is dynamic (depends on how the function is called).

Use case: When defining utility functions you want available throughout your code.

Function Expression



```
const greet = function(name) {  
  return `Hello, ${name}`;  
};
```

- ✗ Not hoisted.
- Can be anonymous or named.
- Stores the function in a variable.

Use case: Useful for callbacks or assigning functions conditionally.

Arrow Function

```
const greet = (name) => `Hello, ${name}`;
```

-  Concise syntax.
-  Does not bind its own this.
- No arguments object.

Use case: Ideal for inline functions, like array operations or event handlers.

Constructor Function

```
function Person(name) {  
  this.name = name;  
}  
const john = new Person("John");
```

- Used with new to instantiate objects.
- Works like a class before class existed.

Use case: When creating multiple instances of similar objects.

Generator Function

```
function* idGenerator() {  
  let id = 0;  
  while (true) yield id++;  
}
```

- Use yield to pause/resume execution.
- Returns an iterator.

Use case: When generating an infinite or controlled sequence of values.

Async Function

```
async function fetchData() {  
  const res = await fetch("/api/data");  
  return res.json();  
}
```

- Always returns a Promise.
- Uses await for asynchronous code.

Use case: For API calls, database queries, or asynchronous workflows.