

### Regular Functions

```
function f() {
    //r eturns undefined
}
```

Function regular sin retorno

### Regular function with return keyword.

```
function f(){
    return true;
}
```

### Single parameter and return value

```
function f(name){
    return name.t oUp per Case();
}
```

### Multiple parameters and default values.

```
function f(isAlive=true, department=42){
    return true;
}
```

### IIFE that accepts parameters

```
(function f(name, message){
    con s${name} says
    ${message});
})( 'Sh eldon', 'Bazin ga');
```

### function returning an object

```
function f(){
    return {
        id: 123,
        name: 'Steve'
    }
}

function f(){
    return (
        {
            id: 123,
            name: 'Steve'
        }
    )
}
```

### Destructuring an object as a parameter

```
function f({id, name}){
    //id and name are now local
    variables inside this function
}

let myObj = {
    id: 123,
    name: 'Steve'
}

f(m yObj);
```



### Arrow functions

```
const f = () =>{  
    //r eturns undefined  
}
```

### Arrow sin retorno

### Arrow function with return keyword

```
const f = () => {  
    return true;  
}  
const f = () => true;
```

### Single parameter and return value

```
const f = name => {  
    return name.t oUp per Case();  
}  
const f = name => name.t oUp per Case();
```

### Multiple parameters and default values.

```
const f = (isAlive=true, department=42) => true;
```

### Arrow IIFE that accepts parameters

```
((name, message)=>{  
    con s${name} says  
    ${message});  
})( 'Sh eldon', 'Bazin ga');
```

### Arrow function returning an object.

```
const f = () =>{  
    return {  
        id: 123,  
        name: 'Steve'  
    }  
}  
const f = () =>({ id: 123, name: 'Steve' });
```

### Arrow destructuring an object as a parameter

```
const f = ({id, name}) =>{  
    //id and name are now local  
    variables inside this function  
}  
let myObj = {  
    id: 123,  
    name: 'Steve'  
}  
f(m yObj);
```

