

TAREA 8

Paso 1 de 34: Se define la función crearCuentaBancaria que toma saldoInicial como parámetro.

```
31     realizarRetiro: function (cantidad) {
32         retirar(cantidad);
33     },
34 };
35 }
36
37 // Ejemplo de uso
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSal);
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo después del depósito: " + miCuenta);
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo después del retiro: " + miCuenta);
44 // Intento de acceder a métodos privados (no funciona)
45
46 // Como manejar excepciones en JavaScript utilizando try-catch
47 try {
```

Global frame

crearCuentaBancaria	undefined
---------------------	-----------

Objects

```
function crearCuentaBancaria(saldoInicial) {
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function () {
            return saldo;
        },
        realizarDeposito: function (cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function (cantidad) {
            retirar(cantidad);
        }
    };
}
```

Paso 2 de 34: Se declara una variable saldo que almacenará el saldo inicial de la cuenta.

```
4 var saldo = saldoInicial;
5 // Método privado para depositar dinero
6 function depositar(cantidad) {
7     if (cantidad > 0) {
8         saldo += cantidad;
9     } else {
10        console.log("La cantidad a depositar debe ser mayor a cero.");
11    }
12 }
13 // Método privado para retirar dinero
14 function retirar(cantidad) {
15     if (cantidad > 0 && cantidad <= saldo) {
16         saldo -= cantidad;
17     } else {
18         console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
19     }
20 }
21 // Retornamos un objeto con métodos públicos
22 return {
23     consultarSaldo: function () {
24         return saldo;
25     },
26     realizarDeposito: function (cantidad) {
27         depositar(cantidad);
28     },
29     realizarRetiro: function (cantidad) {
30         retirar(cantidad);
31     }
32 };
33 }
34
```

Global frame

crearCuentaBancaria	undefined
---------------------	-----------

crearCuentaBancaria

saldoInicial	1000
saldo	undefined
depositar	
retirar	

Objects

```
function crearCuentaBancaria(saldoInicial) {
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function () {
            return saldo;
        },
        realizarDeposito: function (cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function (cantidad) {
            retirar(cantidad);
        }
    };
}
```

```
function depositar(cantidad) {
    if (cantidad > 0) {
        saldo += cantidad;
    } else {
        console.log("La cantidad a depositar debe ser mayor a cero.");
    }
}

function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
        saldo -= cantidad;
    } else {
        console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
    }
}
```

Paso 3 de 34:Se retorna un objeto que contiene la funcion consultar saldo

```
17 } else {
18   console.log(
19     "La cantidad a retirar debe ser mayor a cero
20   );
21 }
22 }
23 // Retornamos un objeto con métodos públicos
24 return {
25   consultarSaldo: function () {
26     return saldo;
27   },
28   realizarDeposito: function (cantidad) {
29     depositar(cantidad);
30   },
31   realizarRetiro: function (cantidad) {
32     retirar(cantidad);
33   },
24
```

Global frame	
crearCuentaBancaria	
miCuenta	undefined
crearCuentaBancaria	
saldoInicial	1000
saldo	1000
depositar	
retirar	

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  var saldo = saldoInicial;
  // Método privado para depositar dinero
  function depositar(cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log(
        "La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible."
      );
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function (cantidad) {
      retirar(cantidad);
    },
  };
}

function depositar(cantidad) {
  if (cantidad > 0) {
    saldo += cantidad;
  } else {
    console.log("La cantidad a depositar debe ser mayor a cero.");
  }
}

function retirar(cantidad) {
  if (cantidad > 0 && cantidad <= saldo) {
    saldo -= cantidad;
  } else {
    console.log(
      "La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible."
    );
  }
}
```

Paso 4 de 34:Ejecuta la funcion consultar saldo

```
24 return {
25   consultarSaldo: function () {
26     return saldo;
27   },
28   realizarDeposito: function (cantidad) {
29     depositar(cantidad);
30   },
31   realizarRetiro: function (cantidad) {
32     retirar(cantidad);
33   },
24
```

crearCuentaBancaria	
saldoInicial	1000
saldo	1000
depositar	
retirar	
Return value	

```
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log(
        "La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible."
      );
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function (cantidad) {
      retirar(cantidad);
    },
  };
}

function depositar(cantidad) {
  if (cantidad > 0) {
    saldo += cantidad;
  } else {
    console.log("La cantidad a depositar debe ser mayor a cero.");
  }
}

function retirar(cantidad) {
  if (cantidad > 0 && cantidad <= saldo) {
    saldo -= cantidad;
  } else {
    console.log(
      "La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible."
    );
  }
}
```

object	
consultarSaldo	function () { return saldo; }
realizarDeposito	function (cantidad) { depositar(cantidad); }
realizarRetiro	function (cantidad) { retirar(cantidad); }

Paso 5 de 34:Se imprime el saldo inicial de la cuenta utilizando miCuenta.consultarSaldo().

```
33 },
34 };
35 }
36
37 // Ejemplo de uso
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSal
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo después del depósito: " + miCuenta
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo después del retiro: " + miCuenta.c
44 // Intento de acceder a métodos privados (no funciona
45
46 //Como manejar excepciones en JavaScript utilizando t
47 try {
48 //El código dentro de try se ejecuta. Si no hay err
```

[Edit this code](#)

Step 5 of 34

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

Global frame

- crearCuentaBancaria
- miCuenta

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  var saldo = saldoInicial;
  // Método privado para depositar dinero
  function depositar(cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cer
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log(
        "La cantidad a retirar debe ser mayor a cero y no excec
      );
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function (cantidad) {
      retirar(cantidad);
    },
  };
}
```

object	
consultarSaldo	function () { return saldo; }
realizarDeposito	function (cantidad) { depositar(cantidad); }
realizarRetiro	function (cantidad) { retirar(cantidad); }

Paso 6 de 34:regresa el saldo de la cuenta.

```
25   consultarSaldo: function () {
26     return saldo;
27   },
28   realizarDeposito: function (cantidad) {
29     depositar(cantidad);
30   },
31   realizarRetiro: function (cantidad) {
32     retirar(cantidad);
33   },
34 };
35 }
```

[Edit this code](#)

Step 6 of 34

Paso 7 de 34: regresa el saldo de la cuenta.

```
24  return {
25    consultarSaldo: function () {
26      return saldo;
27    },
28    realizarDeposito: function (cantidad) {
29      depositar(cantidad);
30    },
31    realizarRetiro: function (cantidad) {
32      retirar(cantidad);
33    },
34  };
35 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 7 of 34

X

Paso 8 de 34: Se imprime el saldo inicial de la cuenta utilizando `miCuenta.consultarSaldo()`.

```
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a métodos privados (no funciona)
45
46 // Como manejar excepciones en JavaScript utilizando try-catch
47 try {
48   // El código dentro de try se ejecuta. Si no hay error, se ejecuta el código dentro de catch.
49 } catch (error) {
50   console.log("Error: " + error);
51 }
52
53 // Cerrar el archivo
54 console.log("Fin del programa");
55
56 // Cerrar el archivo
57 console.log("Fin del programa");
58
59 // Cerrar el archivo
60 console.log("Fin del programa");
61
62 // Cerrar el archivo
63 console.log("Fin del programa");
64
65 // Cerrar el archivo
66 console.log("Fin del programa");
67
68 // Cerrar el archivo
69 console.log("Fin del programa");
70
71 // Cerrar el archivo
72 console.log("Fin del programa");
73
74 // Cerrar el archivo
75 console.log("Fin del programa");
76
77 // Cerrar el archivo
78 console.log("Fin del programa");
79
80 // Cerrar el archivo
81 console.log("Fin del programa");
82
83 // Cerrar el archivo
84 console.log("Fin del programa");
85
86 // Cerrar el archivo
87 console.log("Fin del programa");
88
89 // Cerrar el archivo
90 console.log("Fin del programa");
91
92 // Cerrar el archivo
93 console.log("Fin del programa");
94
95 // Cerrar el archivo
96 console.log("Fin del programa");
97
98 // Cerrar el archivo
99 console.log("Fin del programa");
100
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 8 of 34

Paso 9 de 34: Se realiza un depósito de 500 utilizando `miCuenta.realizarDeposito(500)`.

```
39 console.log("Saldo inicial: " + miCuenta.consultarSal  
40 miCuenta.realizarDeposito(500);  
41 console.log("Saldo después del depósito: " + miCuenta  
42 miCuenta.realizarRetiro(200);  
43 console.log("Saldo después del retiro: " + miCuenta.c  
44 // Intento de acceder a métodos privados (no funciona  
45  
46 //Como manejar excepciones en JavaScript utilizando t  
47 try {  
48 //El código dentro de try se ejecuta. Si no hay error
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 9 of 34

Paso 10 de 34: define la función de realizar el depósito

```
27 },  
28 realizarDeposito: function (cantidad) {  
29     depositar(cantidad);  
30 },  
31 realizarRetiro: function (cantidad) {  
32     retirar(cantidad);  
33 },  
34 };  
35 }  
36  
37 // Ejemplo de uso  
38 var miCuenta = crearCuentaBancaria(1000);  
39 console.log("Saldo inicial: " + miCuenta.consultarSal  
40 miCuenta.realizarDeposito(500);  
41 console.log("Saldo después del depósito: " + miCuenta  
42 miCuenta.realizarRetiro(200);  
43 console.log("Saldo después del retiro: " + miCuenta.c
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 10 of 34

Paso 11 de 34: evalúa la condición

```
6 function depositar(cantidad) {  
→ 7   if (cantidad > 0) {  
8     saldo += cantidad;  
9   } else {  
10    console.log("La cantidad a depositar debe ser m  
11  }  
12 }  
13 // Método privado para retirar dinero  
14 function retirar(cantidad) {  
15   if (cantidad > 0 && cantidad <= saldo) {  
16     saldo -= cantidad;  
17   } else {  
18     console.log(  
19       "La cantidad a retirar debe ser mayor a cero  
20   ):
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 11 of 34

Paso 12 de 34: evalúa la condición +=

```
7  
→ 8   saldo += cantidad;  
9   } else {  
10    console.log("La cantidad a depositar debe ser m  
11  }  
12 }  
13 // Método privado para retirar dinero  
14 function retirar(cantidad) {  
15   if (cantidad > 0 && cantidad <= saldo) {  
16     saldo -= cantidad;  
17   } else {  
18     console.log(  
19       "La cantidad a retirar debe ser mayor a cero  
20   ):
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 12 of 34

Paso 13 de 34: evalúa la condición else

JavaScript (ES6)
[known limitations](#)

```
1 // Funcion fabrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   var saldo = saldoInicial;
5   // Método privado para depositar dinero
6   function depositar(cantidad) {
7     if (cantidad > 0) {
8       saldo += cantidad;
9     } else {
10      console.log("La cantidad a depositar debe ser m
11    }
12  }
13  // Método privado para retirar dinero
14  function retirar(cantidad) {
15    if (cantidad > 0 && cantidad <= saldo) {
16      saldo -= cantidad;
17    } else {
18      console.log(
19        "La cantidad a retirar debe ser mayor a cero
20      ):

```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 13 of 34

Paso 14 de 34: se ejecuta la funcion realizarretiro

```
17 // La cantidad a retirar debe ser mayor a cero
20     });
21 }
22 }
23 // Retornamos un objeto con métodos públicos
24 return {
25     consultarSaldo: function () {
26         return saldo;
27     },
28     realizarDeposito: function (cantidad) {
29         depositar(cantidad);
30     },
31     realizarRetiro: function (cantidad) {
32         retirar(cantidad);
33     },
34 };
35 }
36
37 // Ejemplo de uso
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSal
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 14 of 34

Paso 15 de 34: se muestra el mensaje de la sumatoria

```
41 console.log("Saldo después del depósito: " + miCuenta
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo después del retiro: " + miCuenta.c
44 // Intento de acceder a métodos privados (no funciona
45
46 //Como manejar excepciones en JavaScript utilizando t
47 try {
48     //El código dentro de try se ejecuta. Si no hay err
49     miCuenta.depositar(100); // Error: miCuenta.deposit
50 } catch (e) {
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 15 of 34

Paso 16 de 34: se retorna el saldo

```
25     realizarRetiro: function (cantidad) {  
→ 26         return saldo;  
27     },  
28     realizarDeposito: function (cantidad) {  
29         depositar(cantidad);  
30     },  
31     realizarRetiro: function (cantidad) {  
32         retirar(cantidad);  
33     },  
34 };  
35 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 16 of 34

Paso 17 de 34: se retorna el saldo

```
→ 26     return saldo;  
27 },  
28 realizarDeposito: function (cantidad) {  
29     depositar(cantidad);  
30 },  
31 realizarRetiro: function (cantidad) {  
32     retirar(cantidad);  
33 },  
34 };  
35 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 17 of 34

Paso 18 de 34: se muestra el mensaje con la sumatoria tomando en cuenta el retiro

```
→ 41 console.log("Saldo después del depósito: " + miCuenta
    42 miCuenta.realizarRetiro(200);
    43 console.log("Saldo después del retiro: " + miCuenta.c
    44 // Intento de acceder a métodos privados (no funciona
    45
    46 //Como manejar excepciones en JavaScript utilizando t
    47 try {
    48     //El código dentro de try se ejecuta. Si no hay err
    49     miCuenta.depositar(100); // Error: miCuenta.deposit
    50 } catch (e) {
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 18 of 34

Paso 19 de 34: se ejecuta el retiro

```
→ 41 console.log("Saldo después del depósito: " + miCuenta
→ 42 miCuenta.realizarRetiro(200);
    43 console.log("Saldo después del retiro: " + miCuenta.c
    44 // Intento de acceder a métodos privados (no funciona
    45
    46 //Como manejar excepciones en JavaScript utilizando t
    47 try {
    48     //El código dentro de try se ejecuta. Si no hay err
    49     miCuenta.depositar(100); // Error: miCuenta.deposit
    50 } catch (e) {
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 19 of 34

Paso 20 de 34: se llama la funcion de retirar

```
→ 32     retirar(cantidad);
    33 },
    34 };
    35 }
    36
    37 // Ejemplo de uso
    38 var miCuenta = crearCuentaBancaria(1000);
    39 console.log("Saldo inicial: " + miCuenta.consultarSal
    40 miCuenta.realizarDeposito(500);
    41 console.log("Saldo después del depósito: " + miCuenta
→ 42 miCuenta.realizarRetiro(200);
    43 console.log("Saldo después del retiro: " + miCuenta.c
    44 // Intento de acceder a métodos privados (no funciona
    45
    46 //Como manejar excepciones en JavaScript utilizando t
    47 try {
    48     //El codigo dentro de try se ejecuta. Si no hay err
    49     miCuenta.depositar(100); // Error: miCuenta.deposit
    50 } catch (e) {
```

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First < Prev Next > Last >>

Step 20 of 34

Paso 21 de 34: se evalua la condicion

```
→ 15     if (cantidad > 0 && cantidad <= saldo) {
    16         saldo -= cantidad;
    17     } else {
    18         console.log(
    19             "La cantidad a retirar debe ser mayor a cero
    20         );
    21     }
    22 }
    23 // Retornamos un objeto con métodos públicos
    24 return {
```

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First < Prev Next > Last >>

Step 21 of 34

Paso 22 de 34: se evalua la condición

```
15  if (cantidad > 0 && cantidad <= saldo) {  
16    saldo -= cantidad;  
17  } else {  
18    console.log(  
19      "La cantidad a retirar debe ser mayor a cero  
20    );  
21  }  
22 }  
23 // Retornamos un objeto con métodos públicos  
24 return f
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 22 of 34

Paso 23 de 34: se evalua la condición

```
16  saldo -= cantidad;  
17  } else {  
18    console.log(  
19      "La cantidad a retirar debe ser mayor a cero  
20    );  
21  }  
22 }  
23 // Retornamos un objeto con métodos públicos  
24 return f
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 23 of 34

Paso 24 de 34:

```
30     },
31     realizarRetiro: function (cantidad) {
32         retirar(cantidad);
33     },
34 };
35 }
36
37 // Ejemplo de uso
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 24 of 34

Paso 25 de 34 Se ejecuta el mensaje en consola

```
43 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a métodos privados (no funciona)
45
46 // Como manejar excepciones en JavaScript utilizando try-catch
47 try {
48     // El código dentro de try se ejecuta. Si no hay error, se ejecuta el código dentro de catch.
49     miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
50 } catch (e) {
51     // el parámetro e es una referencia al objeto de excepción
52     console.log(e.message); // message es la propiedad de error
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 25 of 34

Paso 26 de 34 Se retorna saldo

```
→ 26     return saldo;
    27 },
    28 realizarDeposito: function (cantidad) {
    29     depositar(cantidad);
    30 },
    31 realizarRetiro: function (cantidad) {
    32     retirar(cantidad);
    33 },
    34 };
    35 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 26 of 34

Paso 27 de 34 Se retorna saldo

```
→ 26     return saldo;
    27 },
    28 realizarDeposito: function (cantidad) {
    29     depositar(cantidad);
    30 },
    31 realizarRetiro: function (cantidad) {
    32     retirar(cantidad);
    33 },
    34 };
    35 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 27 of 34

Paso 28 de 34

Se imprime el saldo después del retiro.

```
→ 43 console.log("Saldo después del retiro: " + miCuenta.c  
44 // Intento de acceder a métodos privados (no funciona  
45  
46 //Como manejar excepciones en JavaScript utilizando t  
47 try {  
48 //El código dentro de try se ejecuta. Si no hay err  
49 miCuenta.depositar(100); // Error: miCuenta.deposit  
50 } catch (e) {  
51 //el parámetro e es una referencia al objeto de exc  
52 console.log(e.message); //message es la propiedad d
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 28 of 34

Paso 29 de 34

Se ejecuta la función depositar

```
→ 49 miCuenta.depositar(100); // Error: miCuenta.deposit  
50 } catch (e) {  
51 //el parámetro e es una referencia al objeto de exc  
52 console.log(e.message); //message es la propiedad d
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 29 of 34

Paso 30 de 34

Se ejecuta la función depositar

```
40 //El código dentro de try se ejecuta. Si no hay err  
→ 49 miCuenta.depositar(100); // Error: miCuenta.deposit  
50 } catch (e) {  
51 //el parámetro e es una referencia al objeto de exc  
52 console.log(e.message); //message es la propiedad d
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 30 of 34

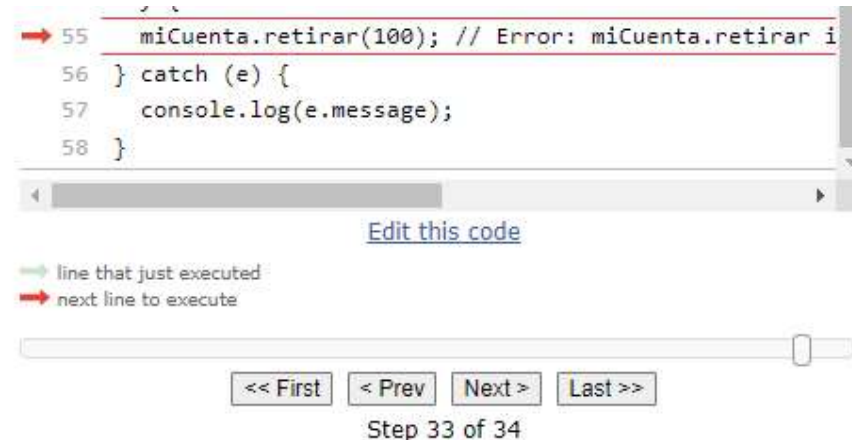
Paso 31 de 34 Se muestra el mensaje



Paso 32 de 34 Se muestra el mensaje



Paso 33 de 34 Se muestra el mensaje



Paso 34 de 34 código en consola

