

Universidad Galileo
Gerardo Francisco Gutiérrez Valenzuela
Introducción a la programación



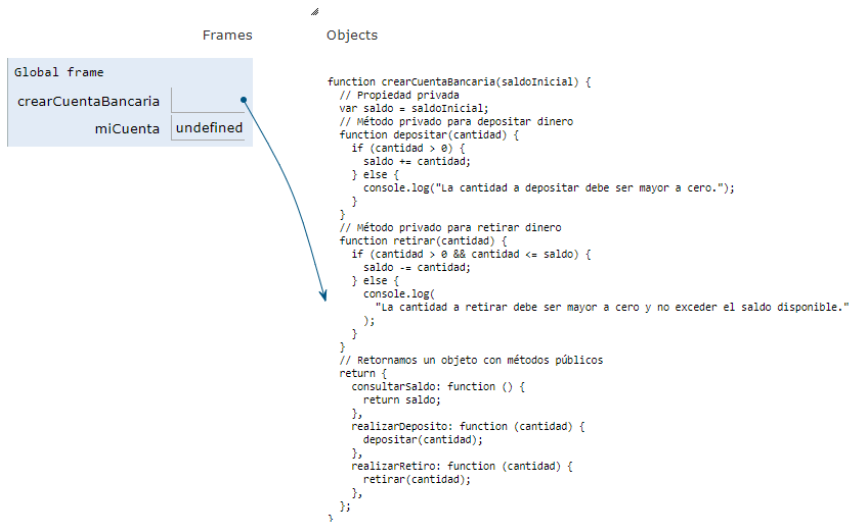
Tarea 8 – Análisis de código



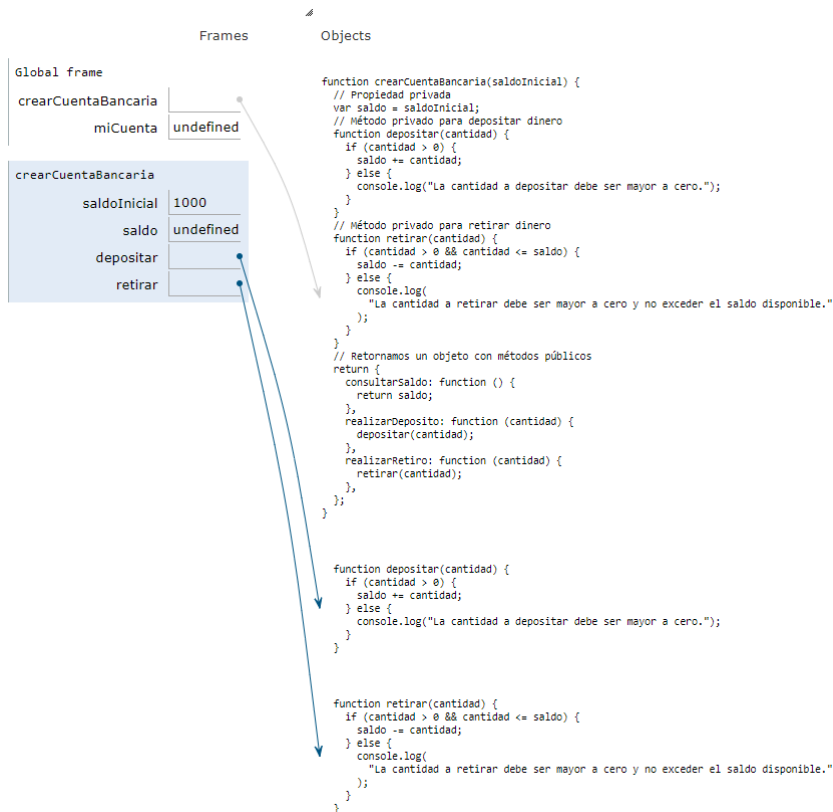
Byron Fernando Cardona Sánchez
Carné: 24011342
25 de Junio del 2024

Análisis de ejecución de fabrica de Cuenta Bancaria

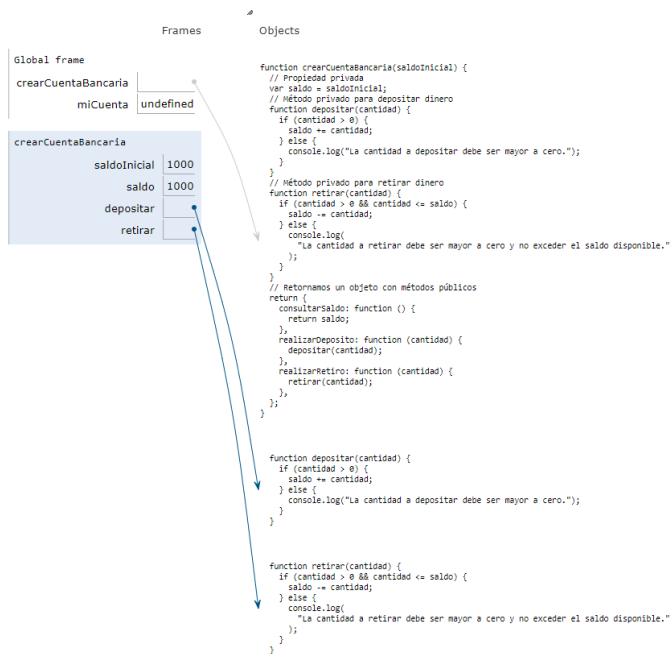
1. La primer ejecución que se realiza es a la función **Cuenta Bancaria** por medio de la variable **miCuenta**



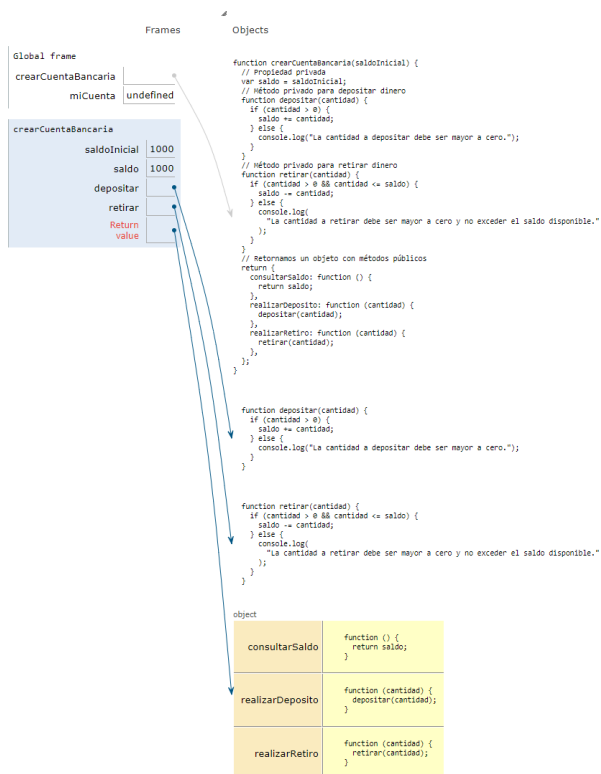
2. Lo siguiente es iniciar el valor de la variable **saldo** con 1000



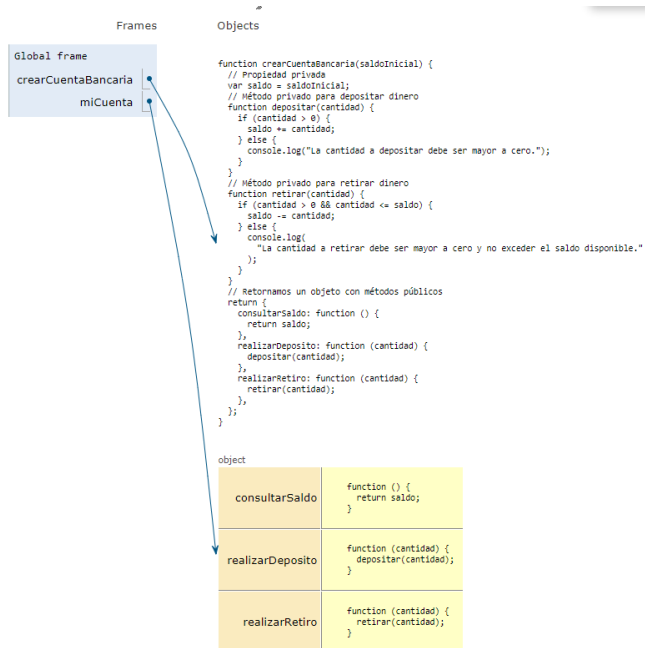
3. Luego la función **crearCuentaBancaria** retorna un objeto de **3** métodos (return): **consultarSaldo**, **realizarDeposito**, y **realizarRetiro**



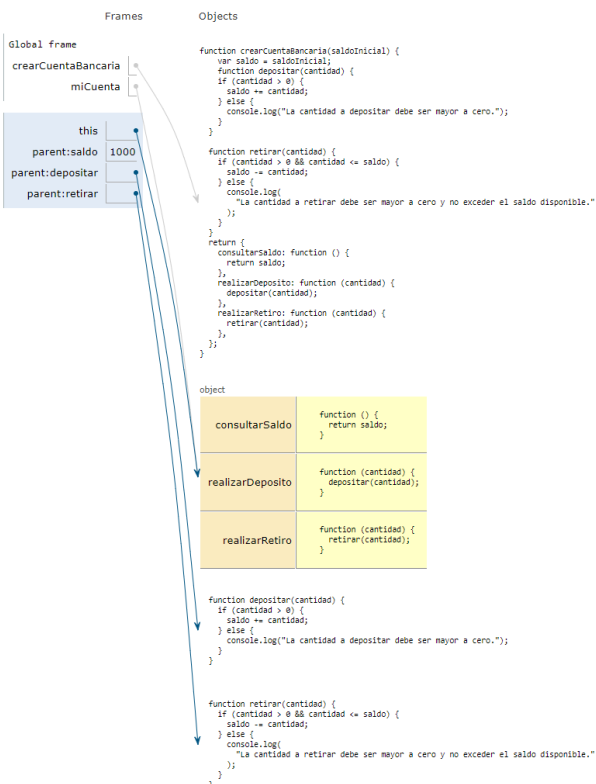
4. Return comienza con un objeto con métodos públicos **consultarSaldo** que regresa **1000** como valor, **realizarDeposito** y **realizarRetiro** aún no tienen valores asignados.



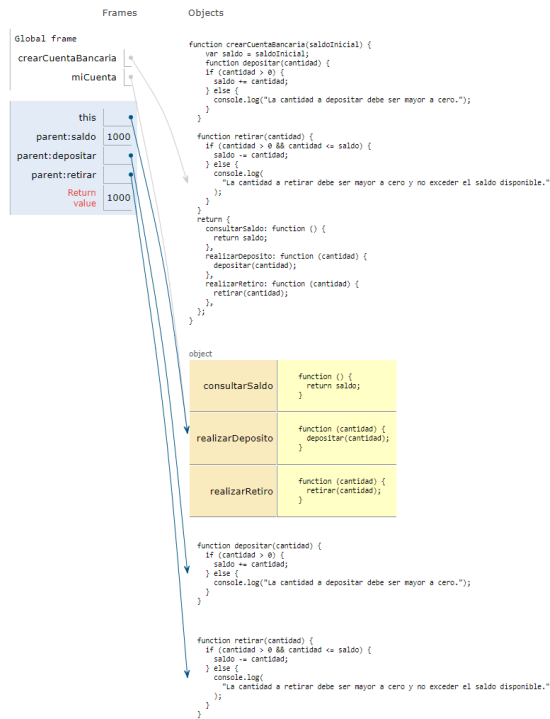
5. Regresa a ejecutar la línea **console.log Saldo inicia** y se hace el llamado a la variable **miCuenta** para obtener el saldo **funcion crearCuentaBancaria**



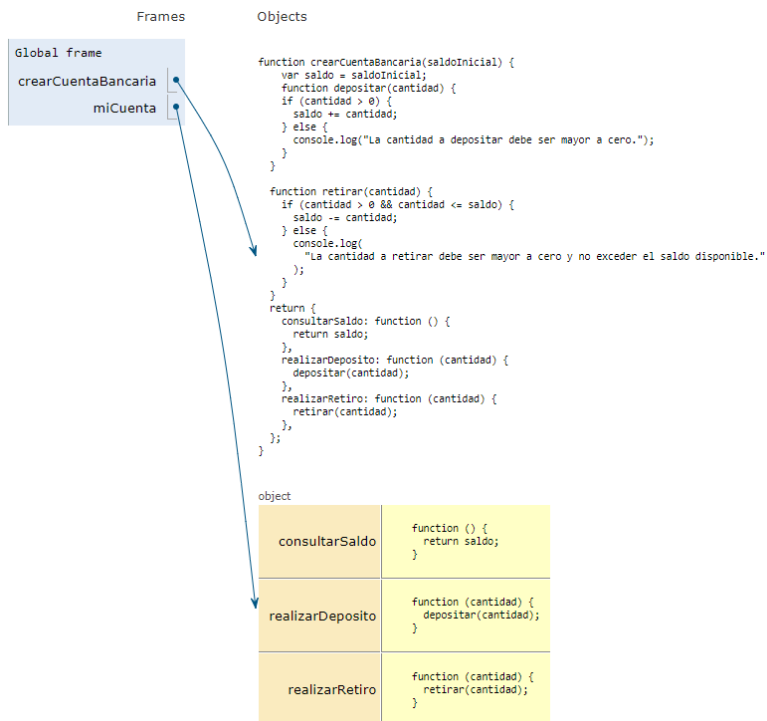
6. Retorna el valor en el objeto de método público **consultarSaldo** el cual trae la variable **saldo**



7. El cual retorna un valor de **1000**



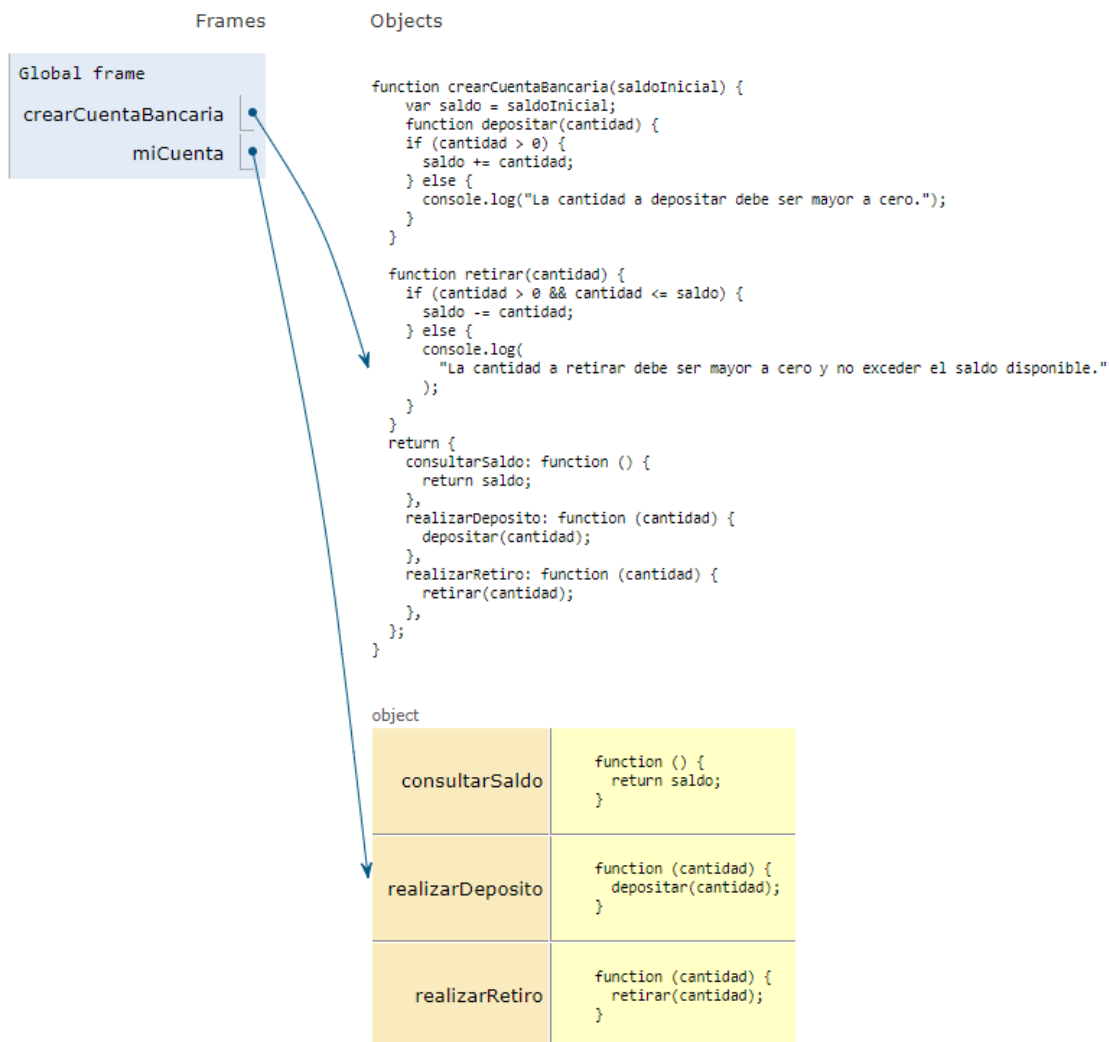
8. Regresa a la línea de **console.log Saldo inicial** y en este punto imprime **Saldo Inicial**: con el valor que llama de **miCuenta.consultarSaldo()**



9. Se llama al metodo **realizarDeposito** del objeto **realizarDeposito**

Print output (drag lower right corner to resize)

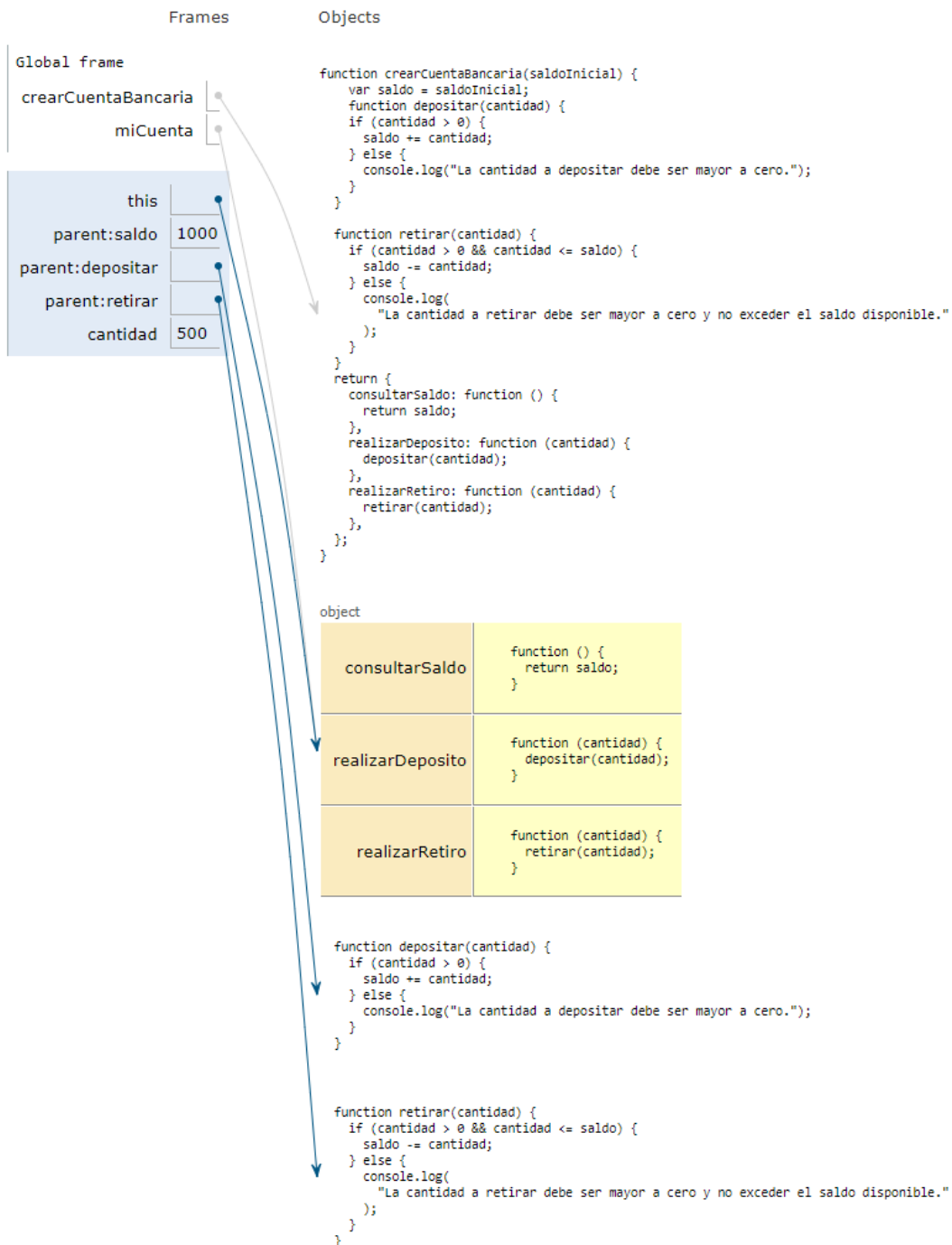
Saldo inicial: 1000



10. Dentro del metodo **realizarDeposito** se ejecuta la función depositar con valor **500** como argumento.

Print output (drag lower right corner to resize)

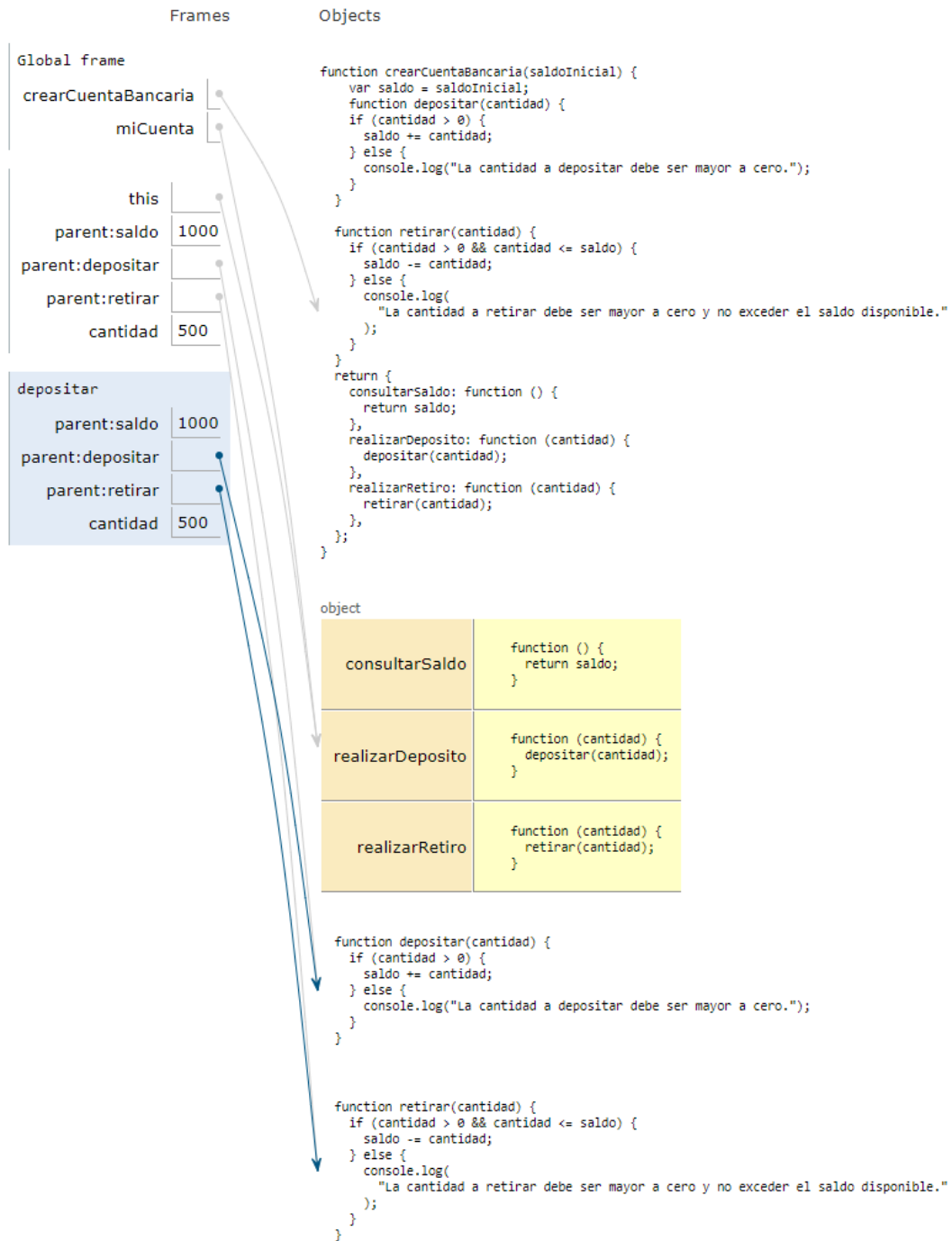
Saldo inicial: 1000



11. La **función** depositar verifica si cantidad es mayor a **0**.

Print output (drag lower right corner to resize)

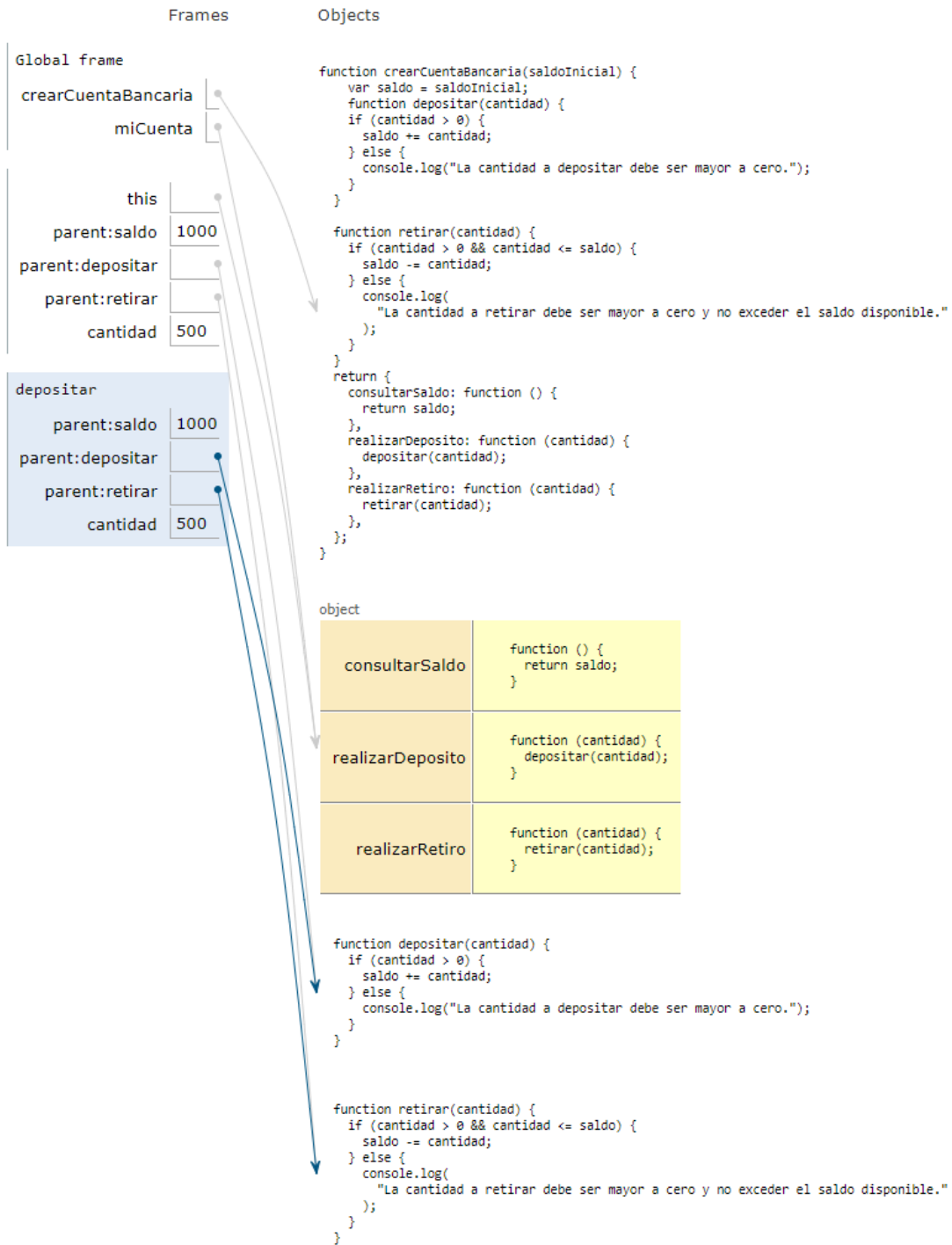
Saldo inicial: 1000



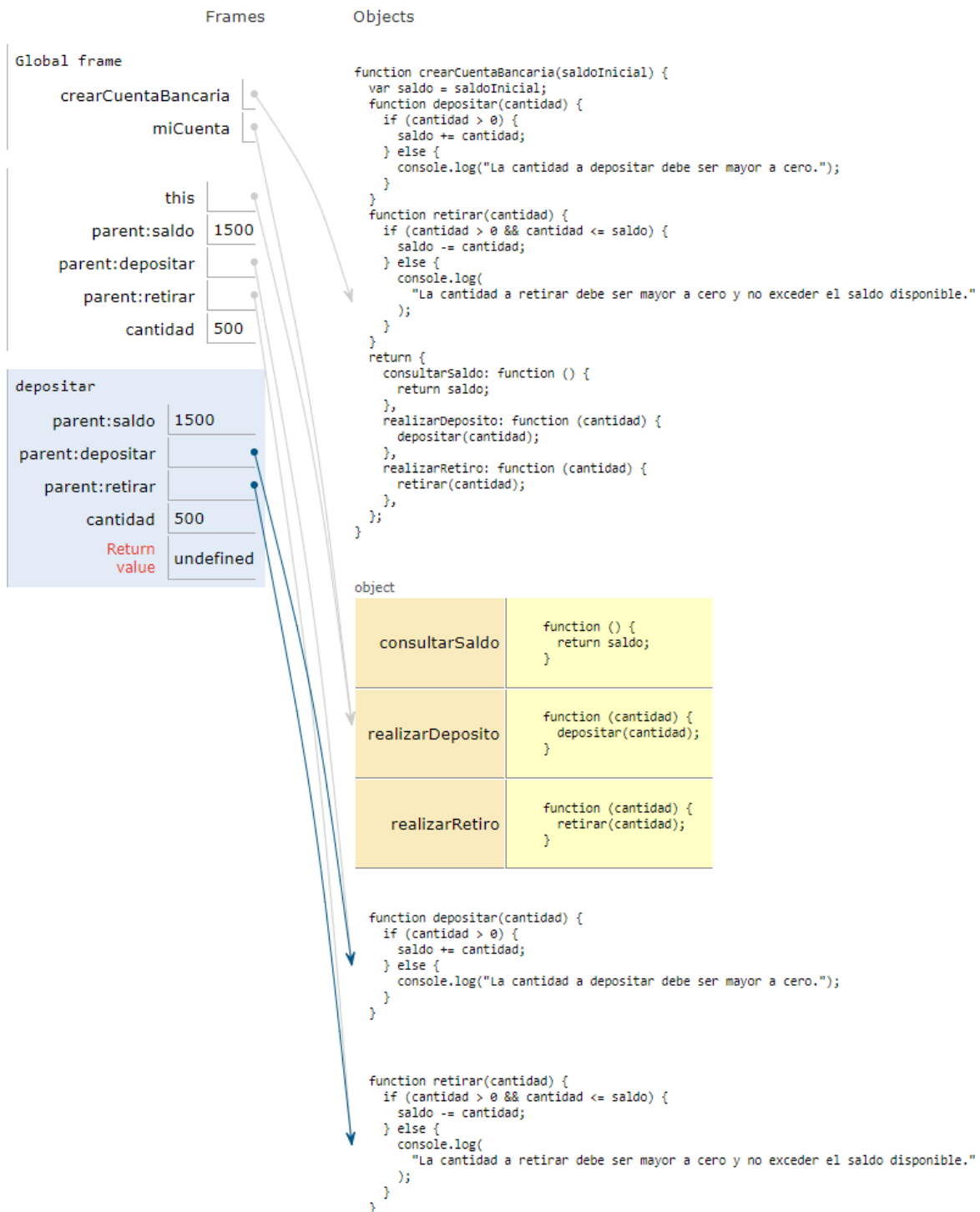
12. cantidad es **500**, por lo que la condición es **verdadera**.

Print output (drag lower right corner to resize)

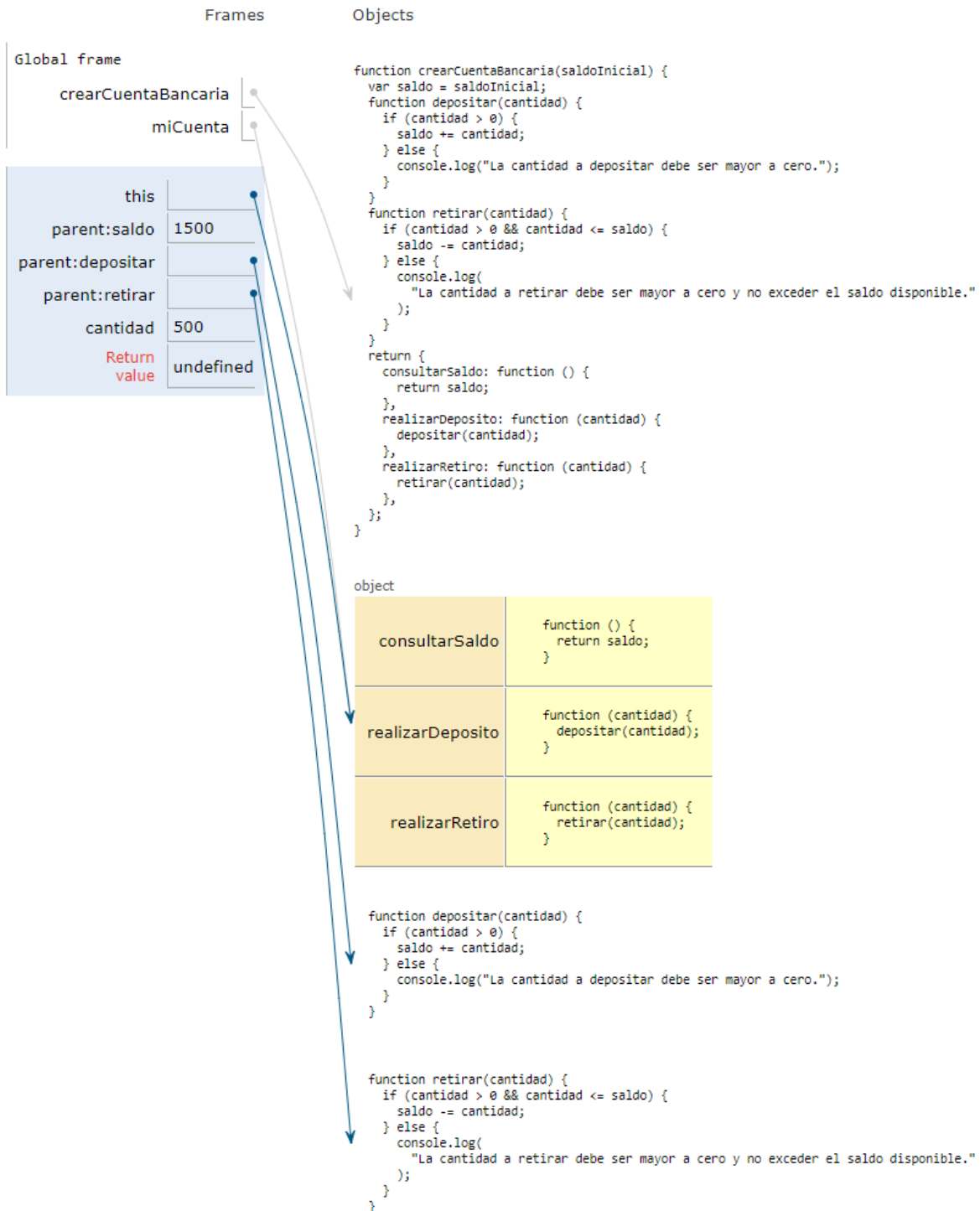
Saldo inicial: 1000



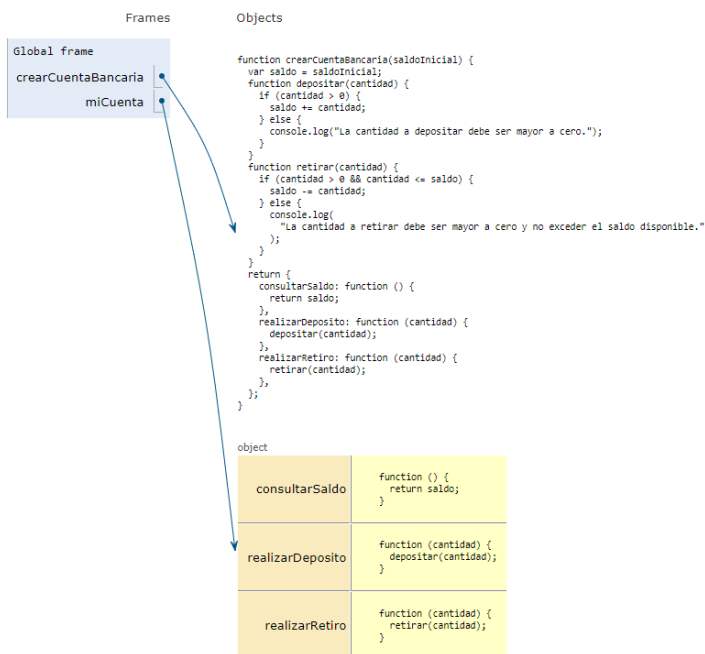
13. Se suma cantidad **500** a saldo **1000**.



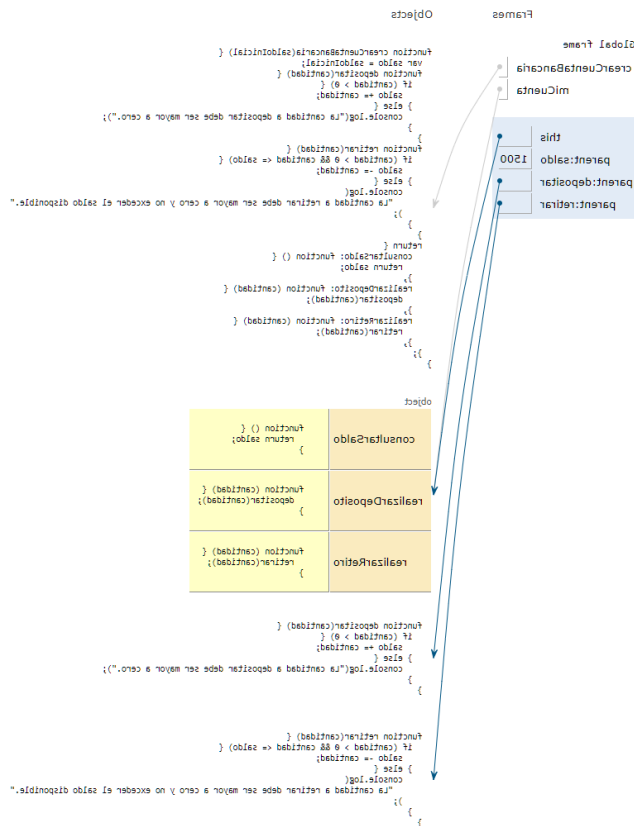
14. **saldo** ahora es **1500**.



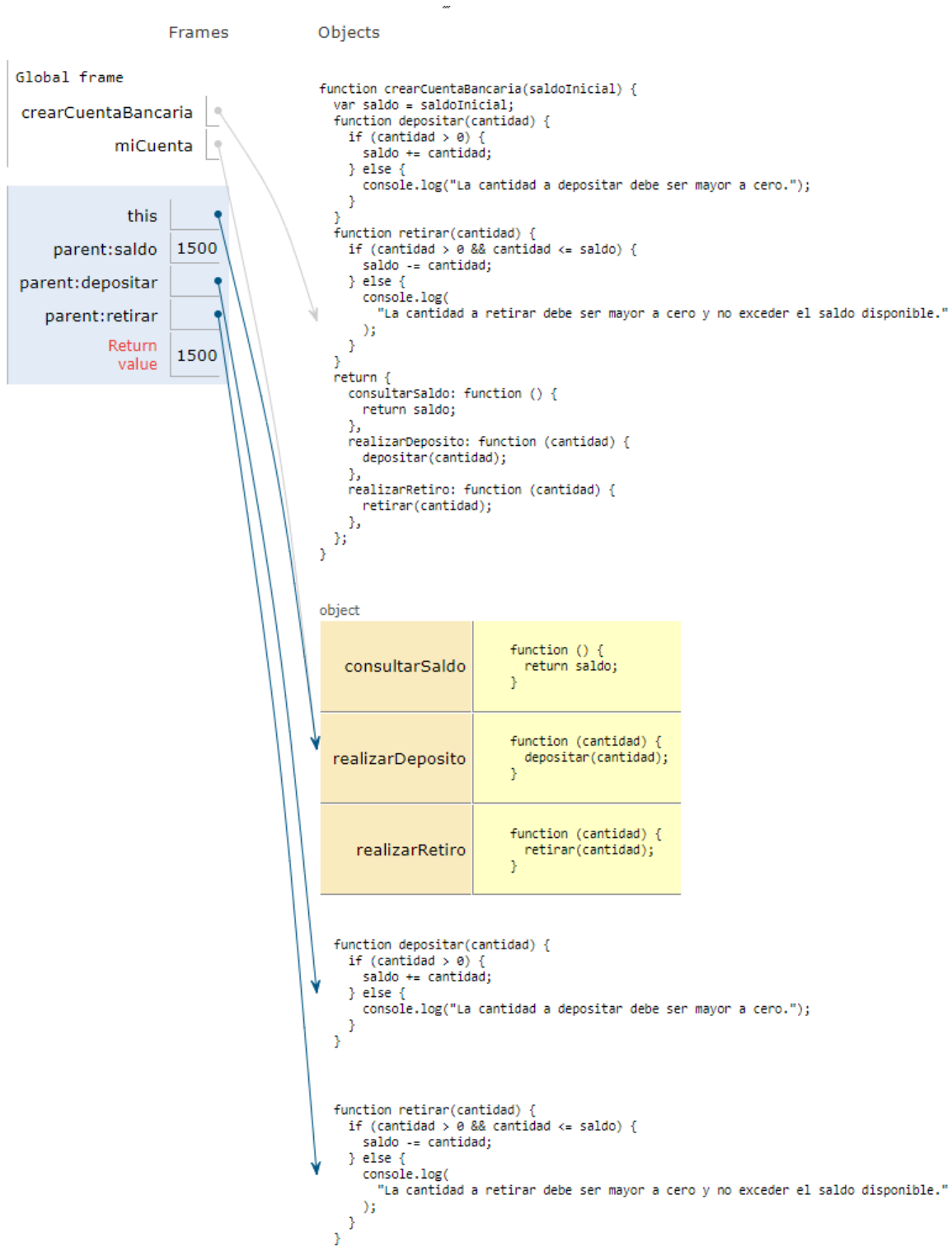
15. Vamos al siguiente **console.logSaldo** despues del deposito Se llama al método **consultarSaldo** del objeto **miCuenta** para obtener el saldo



16. El método **consultarSaldo** retorna el valor de saldo.



17. El cual pasa un valor de **1500**



18. Retornamos a la línea de **console.logSaldo** después del depósito con valor **1500** que ahora imprimirá en consola **Saldo después del depósito: 1500**

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  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.");
    }
  }
  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); } |

19. El método **realizarRetiro** llama a la función retirar con cantidad igual a **200**.

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  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.");
    }
  }
  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); } |

20. Dentro del objeto **realizarRetiro** se pasa el valor de **200**

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo después del depósito: 1500
```

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

| | |
|------------------|------|
| this | |
| parent:saldo | 1500 |
| parent:depositar | |
| parent:retirar | |
| cantidad | 200 |

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  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."
      );
    }
  }
  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); } |

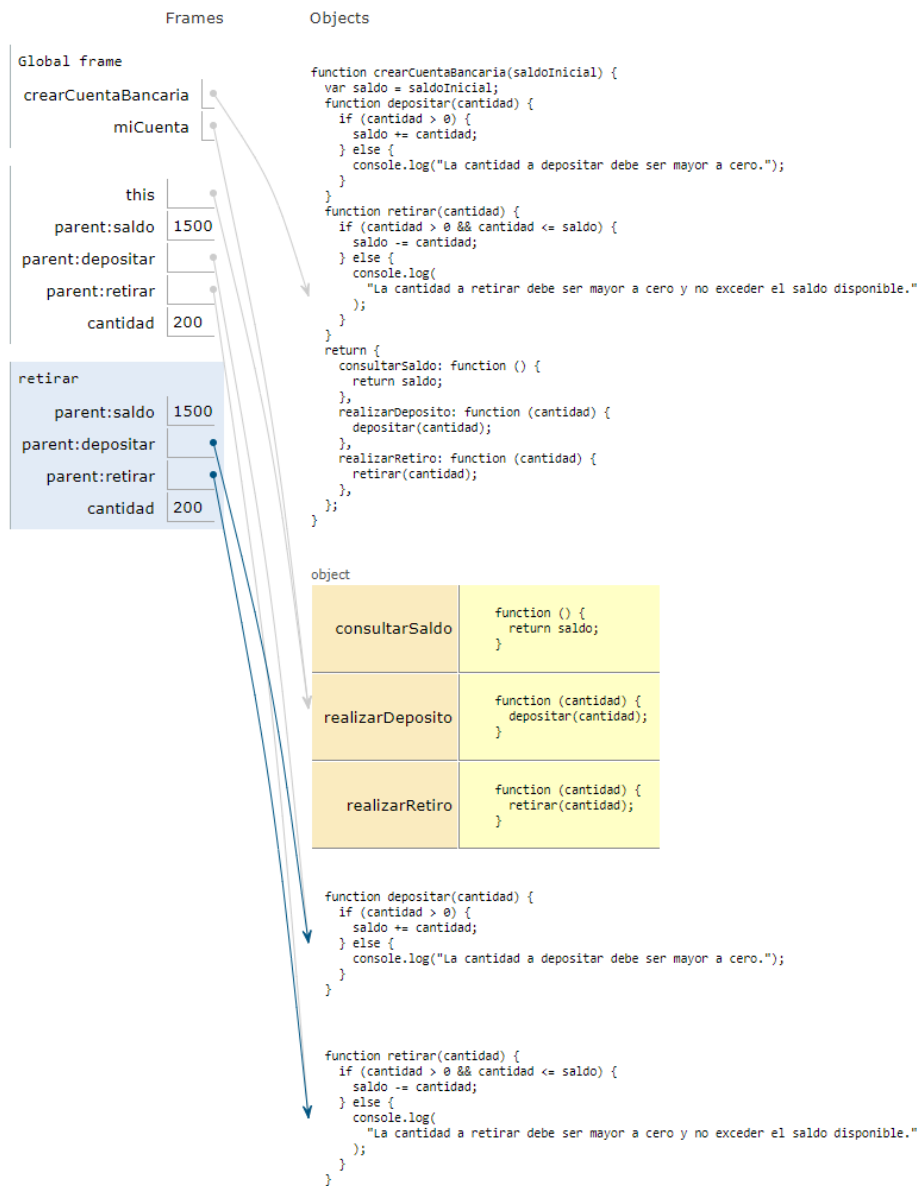
```
function depositar(cantidad) {
  if (cantidad > 0) {
    saldo += cantidad;
  } else {
    console.log("La cantidad a depositar debe ser mayor a cero.");
  }
}
```

```
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."
    );
  }
}
```

21. La función **retirar** verifica si cantidad es mayor a 0 y menor o igual a **saldo**

Print output (drag lower right corner to resize)

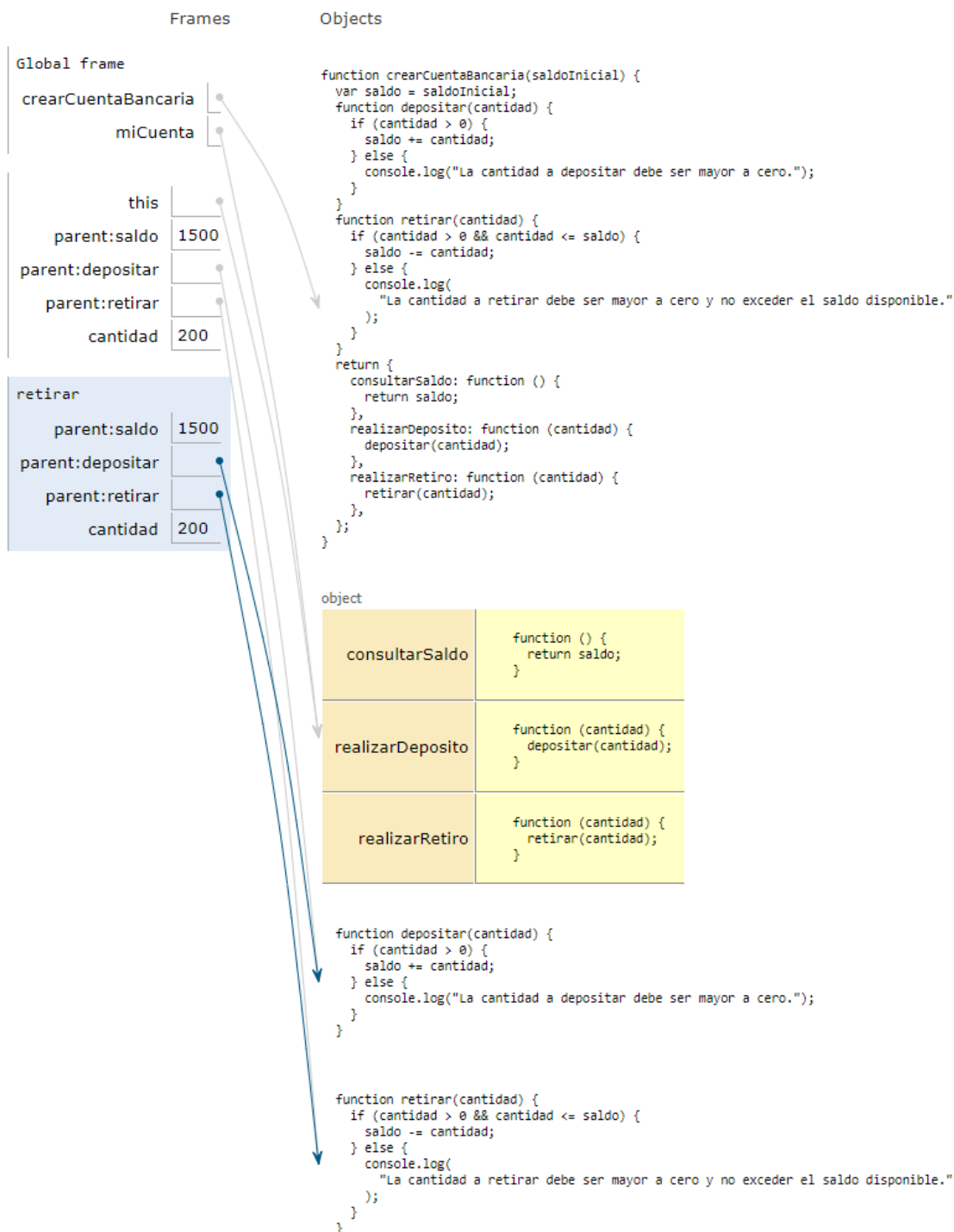
Saldo inicial: 1000
Saldo después del depósito: 1500



22. Se resta cantidad **200** de saldo **1500**.

Print output (drag lower right corner to resize)

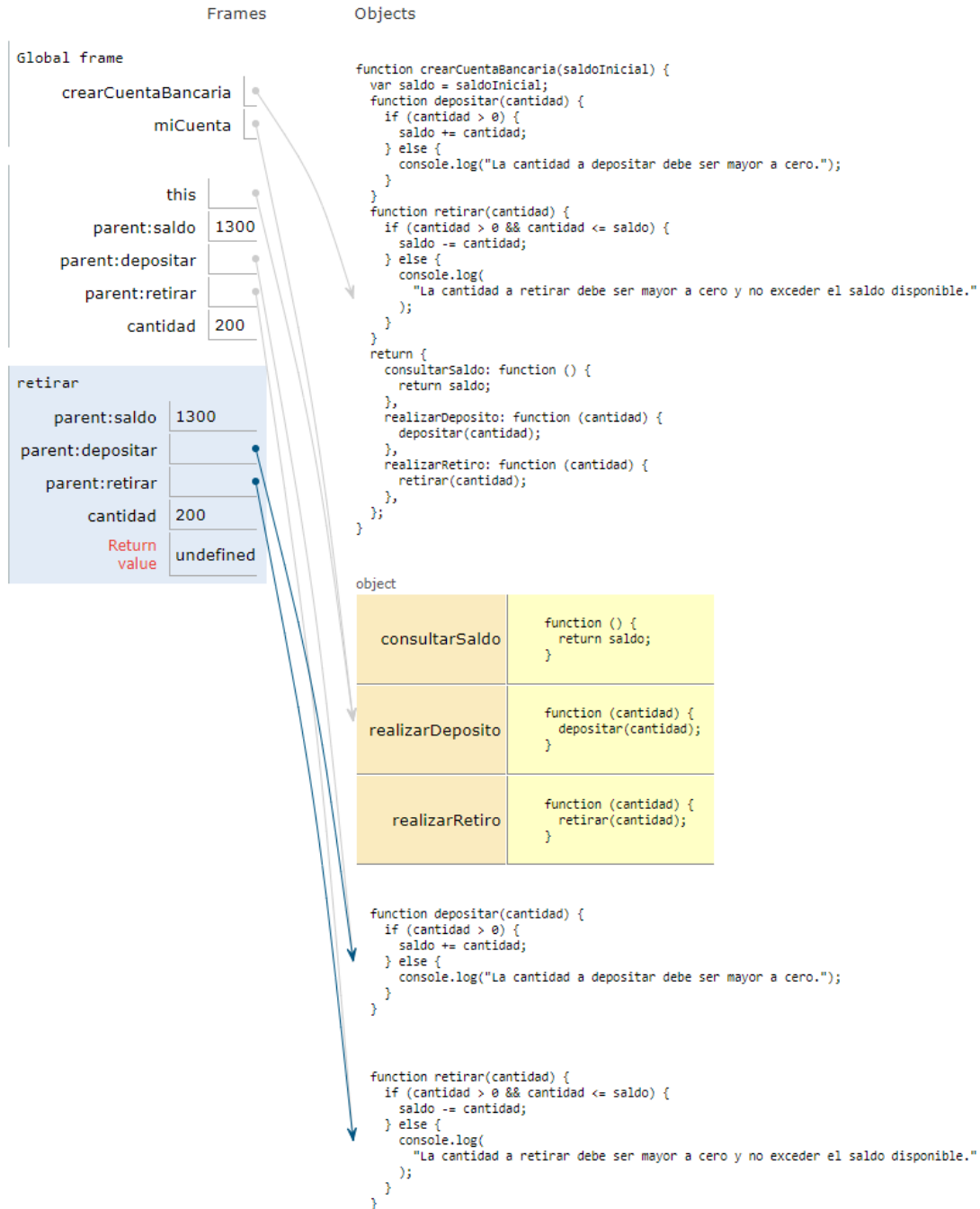
Saldo inicial: 1000
Saldo después del depósito: 1500



23. **saldo** ahora es **1300**.

Print output (drag lower right corner to resize)

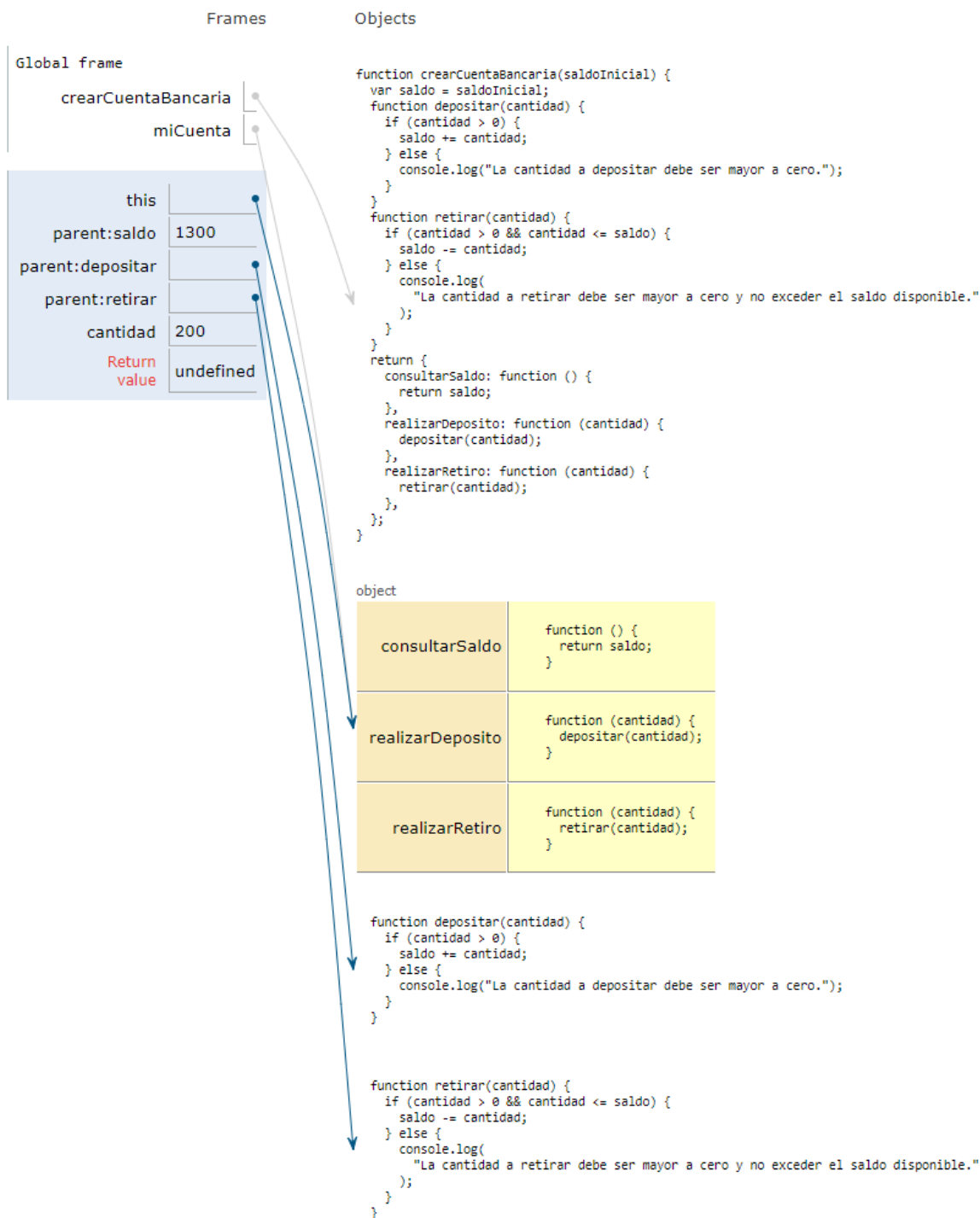
Saldo inicial: 1000
Saldo después del depósito: 1500



24. Finaliza de ejecutarse la función **retirar**

Print output (drag lower right corner to resize)

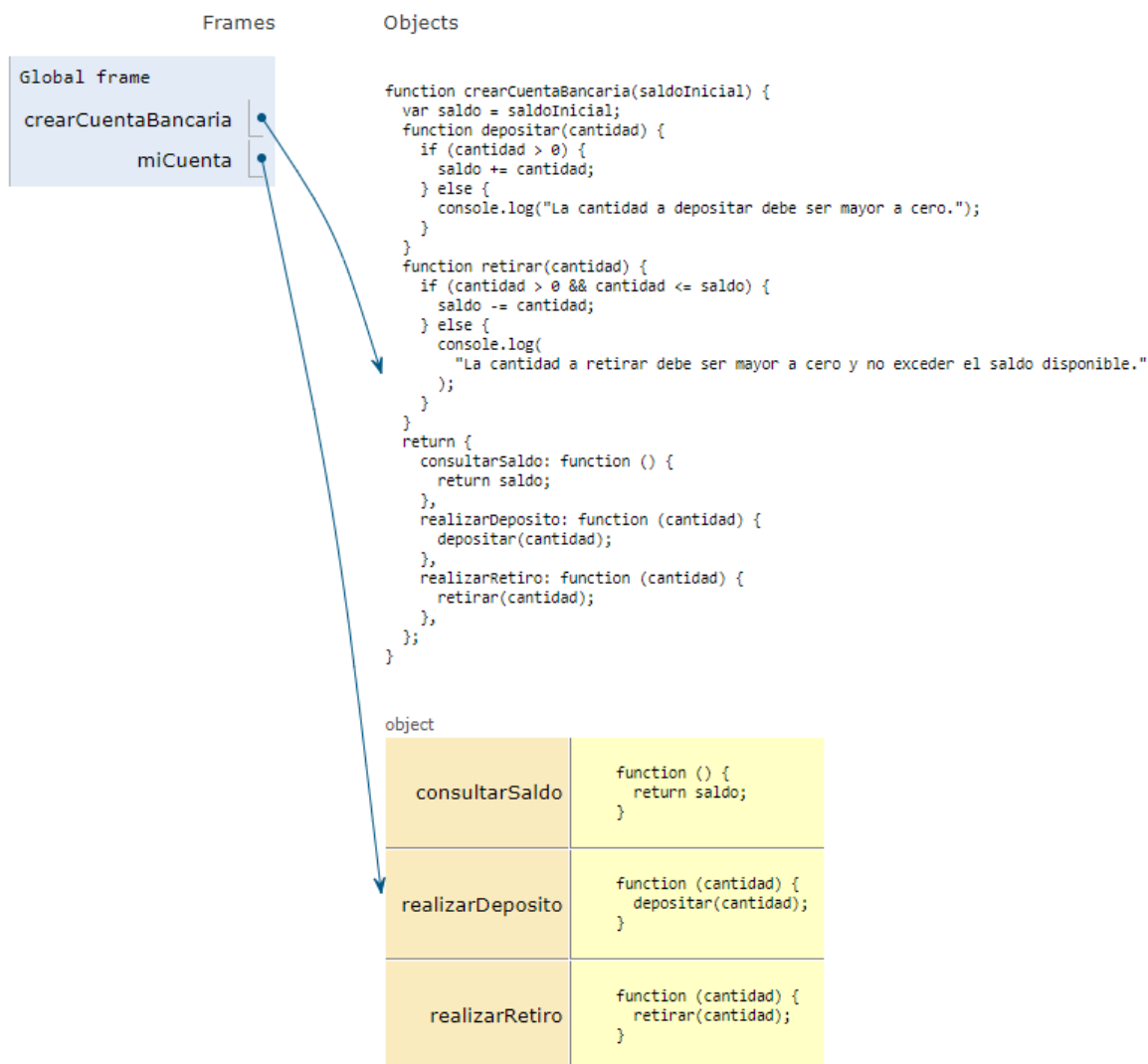
Saldo inicial: 1000
Saldo después del depósito: 1500



25. En este proceso se mueve al siguiente **console.logSaldo después del retiro** Se llama al método **consultarSaldo** del objeto **miCuenta** para obtener el saldo después del **retiro**

Print output (drag lower right corner to resize)

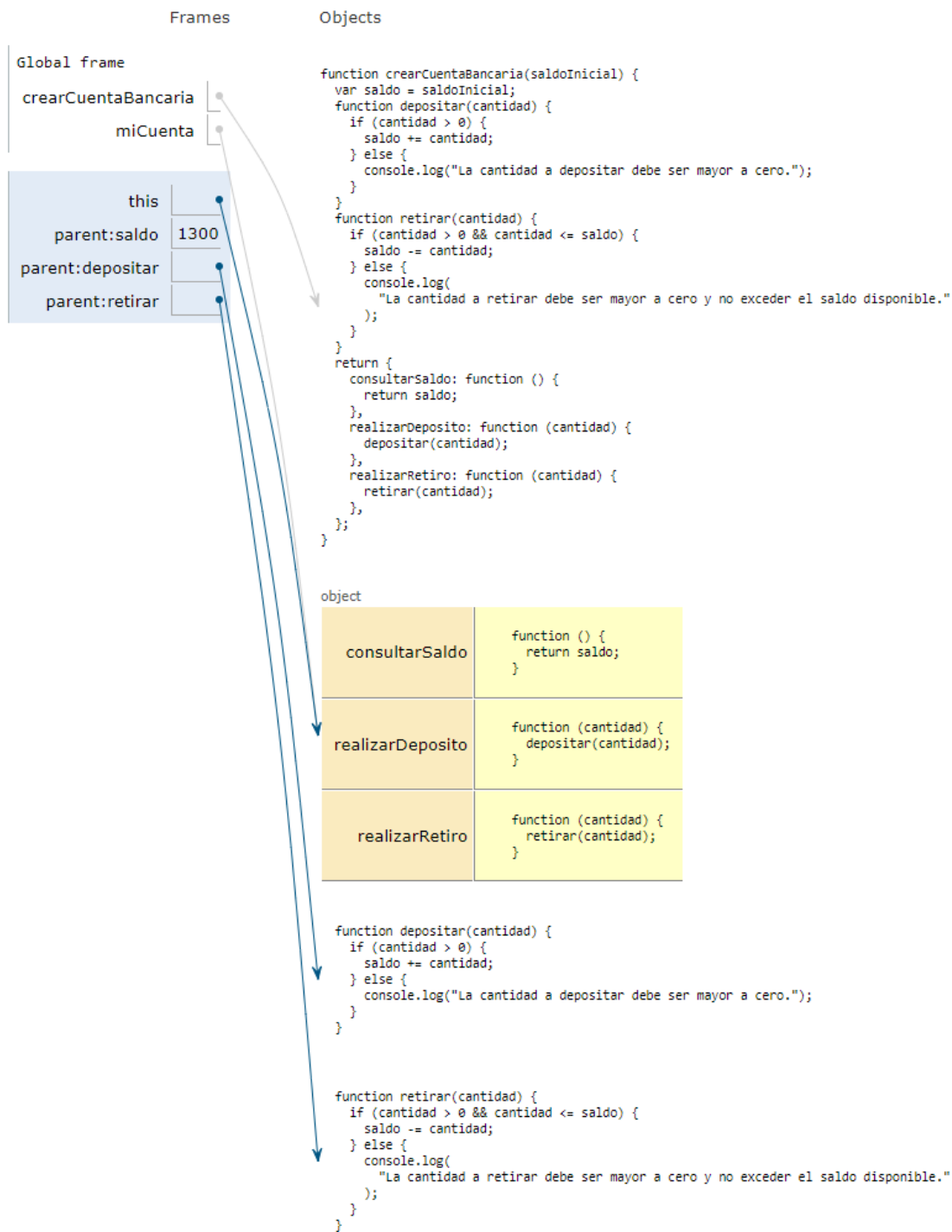
Saldo inicial: 1000
Saldo después del depósito: 1500



26. Se llama a **miCuenta.consultarSaldo**

Print output (drag lower right corner to resize)

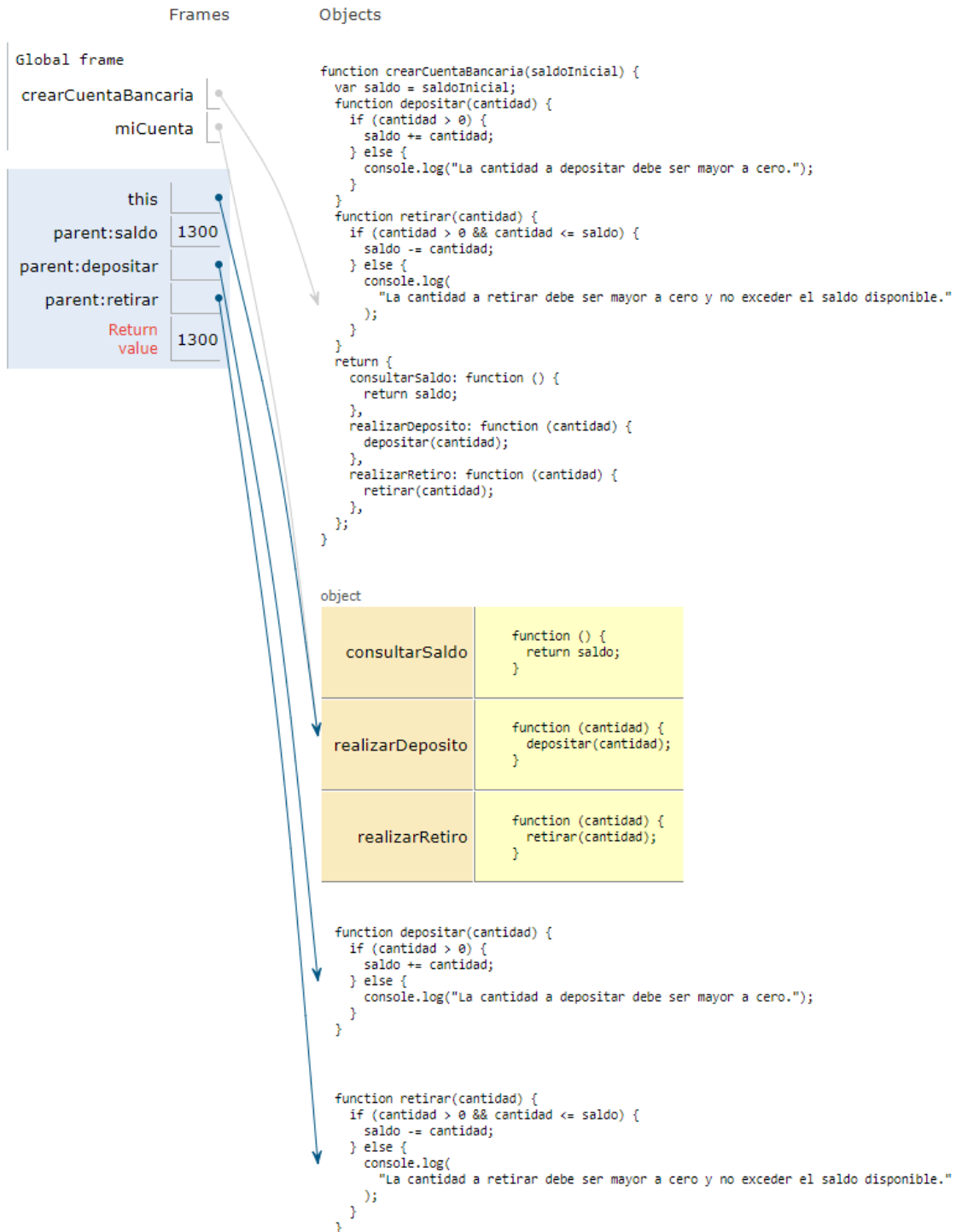
```
Saldo inicial: 1000
Saldo después del depósito: 1500
```



27. El cual a este punto tiene un valor de **1300**

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo después del depósito: 1500
```



28. Luego se continúa en el **console.logSaldo después del retiro** con el valor de **saldo** y se imprimirá **Saldo después del retiro: 1300**

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo después del depósito: 1500
```

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  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."
      );
    }
  }
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function (cantidad) {
      retirar(cantidad);
    },
  };
};
```

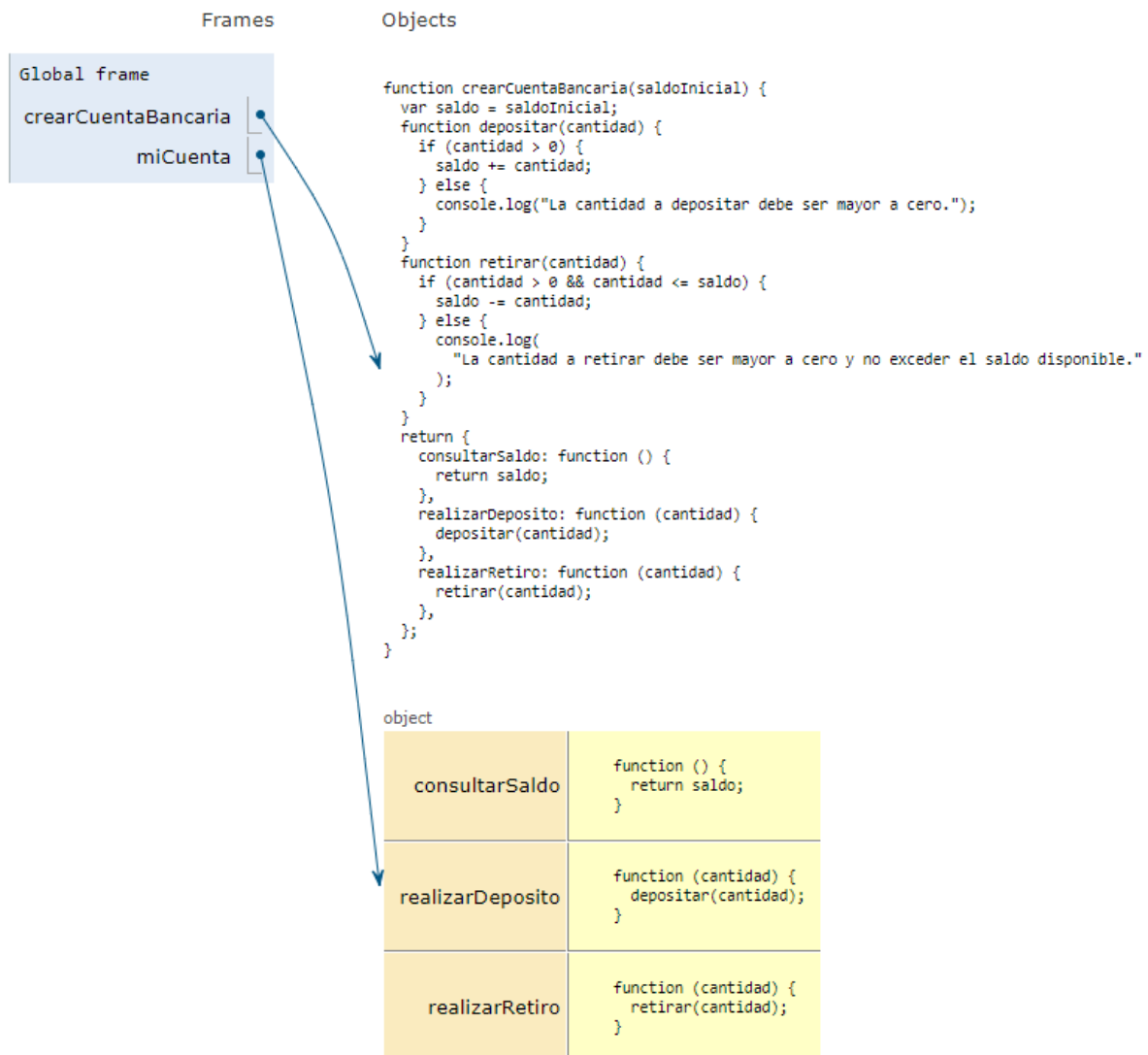
object

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

29. Se intenta llamar a **miCuenta.depositar(100)**, pero este método no existe en el objeto **miCuenta**.

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000  
Saldo después del depósito: 1500  
Saldo después del retiro: 1300
```



30. Ocurre un error porque **miCuenta.depositar** no es una **función**.

```
41 try {
→ 42   miCuenta.depositar(100);
43 } catch (e) {
44
45   console.log(e.message);
46 }
47 try {
```

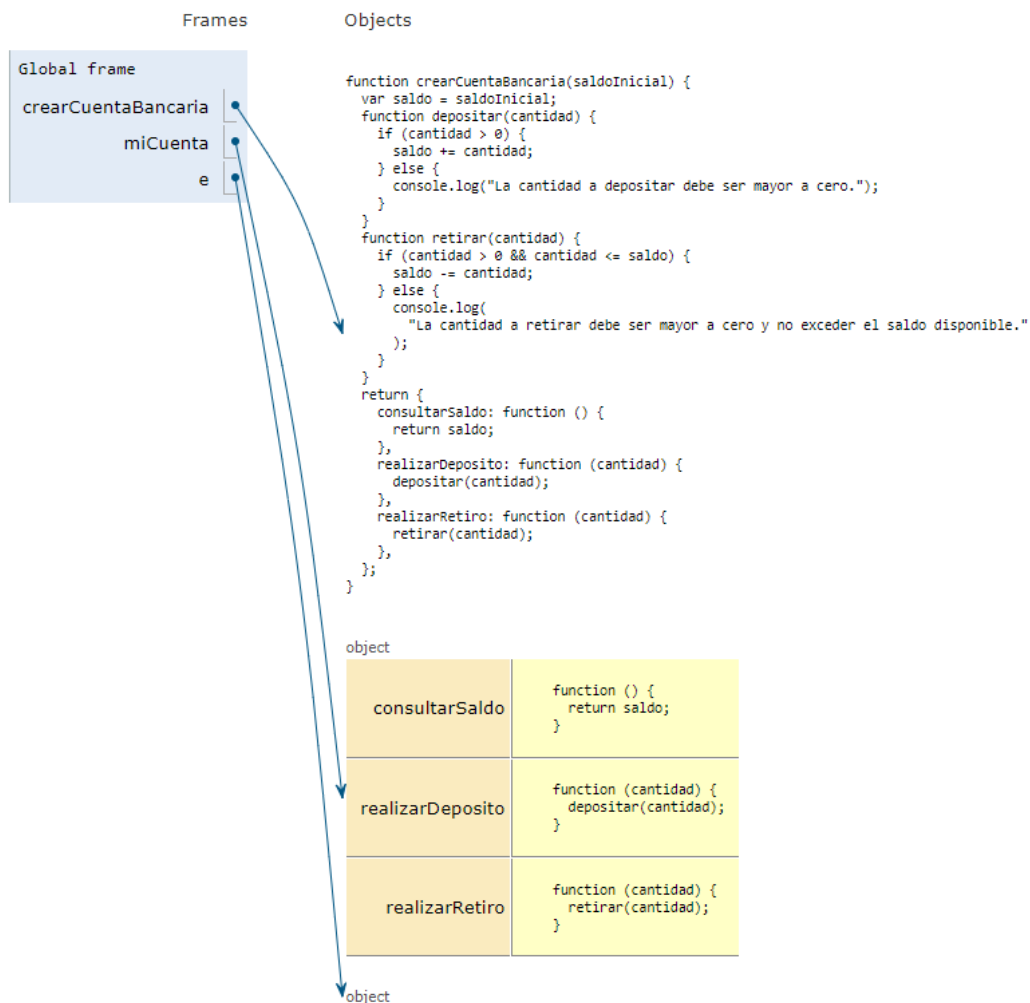
[Edit this code](#)

→ line that just executed
→ next line to execute

Step 30 of 34

TypeError: miCuenta.depositar is not a function

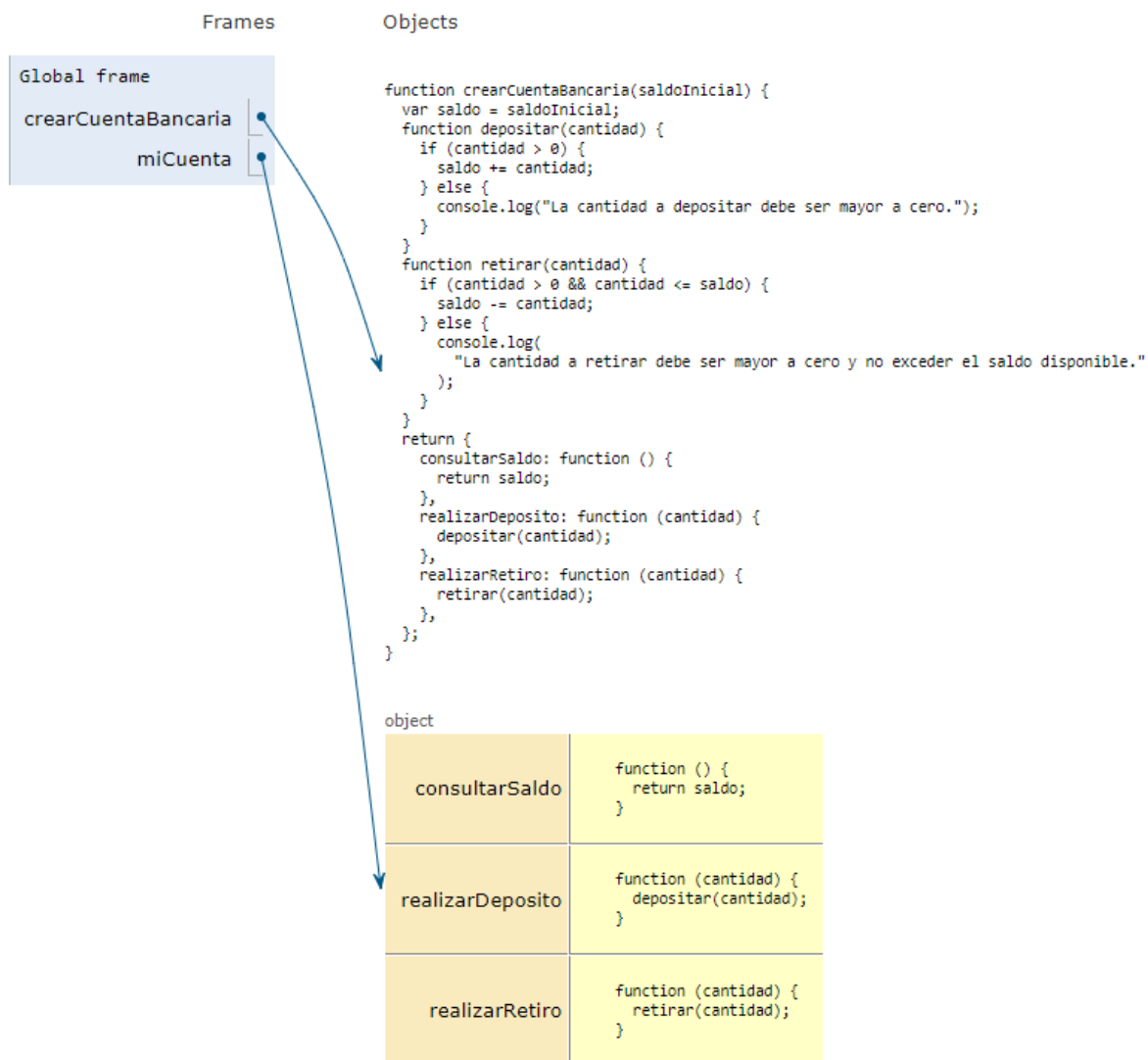
31. Se captura el **error** y se imprime el mensaje de **error** en la **consola**.



32. Se intenta llamar a **miCuenta.retirar(100)**, pero este método no existe en el objeto **miCuenta**.

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
miCuenta.depositar is not a function
```



33. Ocurre un error porque **miCuenta.retirar** no es una **función**.

```
→ 48  miCuenta.retirar(100);
    49  } catch (e) {
    50    console.log(e.message);
    51  }
```

[Edit this code](#)

→ line that just executed

→ next line to execute



<< First

< Prev

Next >

Last >>

Step 33 of 34

TypeError: miCuenta.retirar is not a function

34. Se captura el error y se **imprime** el **mensaje de error** en la **consola**.

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
miCuenta.depositar is not a function
miCuenta.retirar is not a function
```

Frames

Objects

Global frame
crearCuentaBancaria
miCuenta

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  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."
      );
    }
  }
  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); } |