Unitary Evolution Operator

In quantum mechanics, a unitary evolution operator is a mathematical operator that describes the time evolution of a quantum system. It is a unitary matrix that acts on the state of the system, transforming it from an initial state to a final state at a later time. The unitary evolution operator is obtained by solving the Schrödinger equation, which describes how the state of a quantum system changes over time.

The unitary evolution operator has several important properties:

1. It is unitary, meaning that it preserves the inner product of two quantum states.
2. It is linear, meaning that it satisfies the superposition principle.
3. It is time-reversible, meaning that it can be inverted to find the evolution operator that describes the system going backwards in time.
4. It is deterministic, meaning that given the initial state of the system, the evolution operator completely determines the final state at any later time.

The unitary evolution operator is a fundamental concept in quantum mechanics, and plays a central role in many applications, including quantum computing, quantum information theory, and quantum field theory.

Implementation of Unitary Evolution Operator

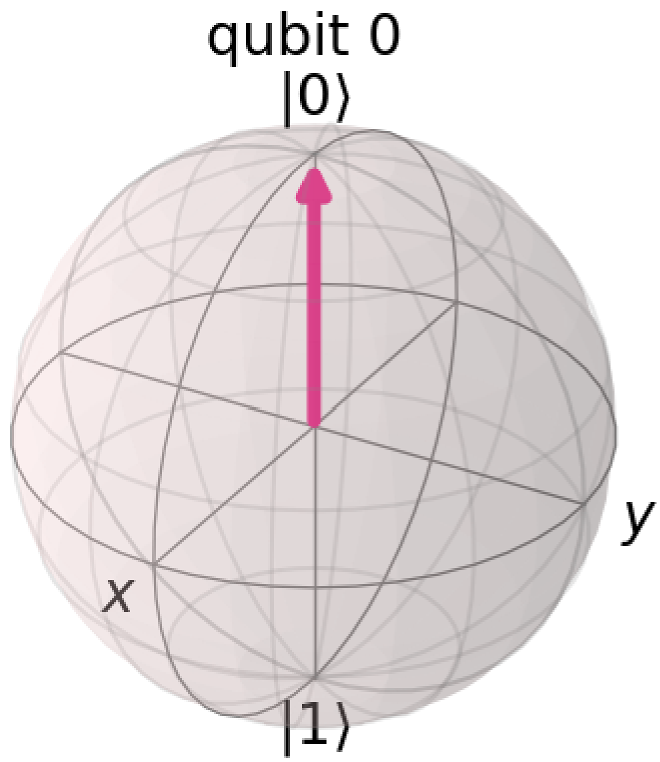
In [47]:

```
from qiskit import QuantumCircuit, Aer, execute
from qiskit.visualization import plot_bloch_multivector
# Define the initial state vector
initial_statevector = [1, 0]
# Apply the Hadamard gate to the qubit
qc.h(0)

# Simulate the circuit and get the final statevector
backend = Aer.get_backend('statevector_simulator')
final_statevector = execute(qc, backend=backend).result().get_statevector()

# Apply the Hadamard gate to the final statevector to obtain the initial state
qc2 = QuantumCircuit(1)
qc2.initialize(final_statevector, 0)
qc2.h(0)
recovered_statevector = execute(qc2, backend=backend).result().get_statevector
```

Plot the initial state vector of a qubits using the bloch sphere

In [42]:
```
plot_bloch_multivector(initial_statevector)
```

Out[42]:



In [43]:
```
initial_statevector
```

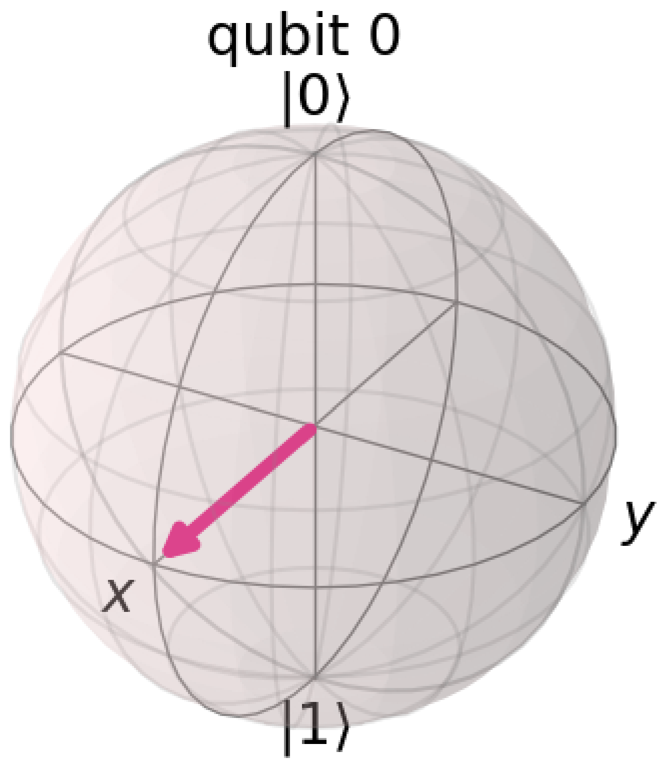
Out[43]:
```
[1, 0]
```

Plot the final state vector of a qubits using the bloch sphere

In [44]:

```
# Create a quantum circuit with a single qubit
qc = QuantumCircuit(1)
# Initialize the qubit to the initial state vector
qc.initialize(initial_statevector, 0)

# Plot the  final state vector, and recovered state vector using the bloch sph
plot_bloch_multivector(final_statevector)
```

Out[44]:



In [39]:

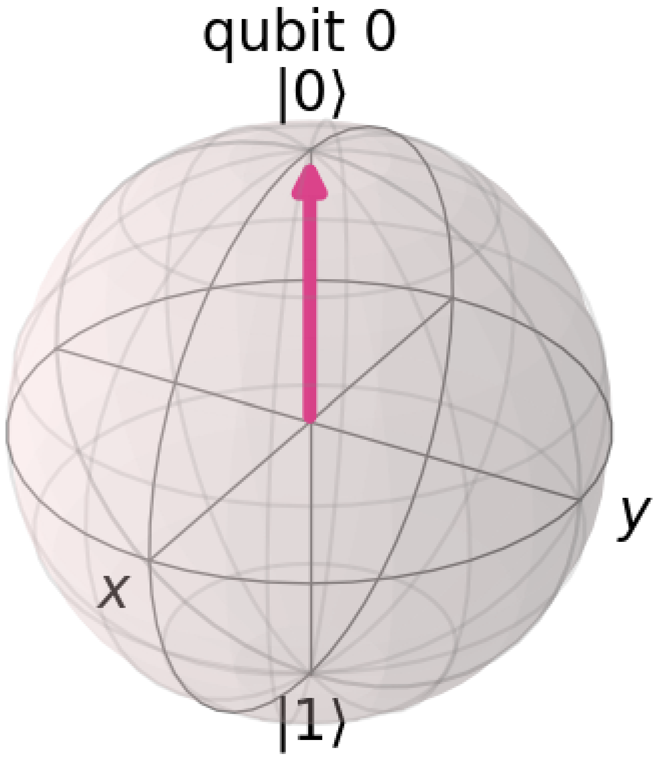
```
final_statevector
```

```
Statevector([0.70710678+0.j, 0.70710678+0.j],
            dims=(2,))
```

Plot the recover state vector of a qubits using the bloch sphere

In [45]:
```
# Plot the recovered state vector using the bloch sphere
plot_bloch_multivector( recovered_statevector)
```

Out[45]:



In [40]:
```
recovered_statevector
```

```
Statevector([1.-6.123234e-17j, 0.+6.123234e-17j],
            dims=(2,))
```