Closures

1. **Simple Closure**:

```
function createCounter() {
  let count = 0;
  return function() {
    count++;
    return count;
  };
}
const counter = createCounter();
console.log(counter()); // Expected output: 1
console.log(counter()); // Expected output: 2
console.log(counter()); // Expected output: 3
```

2. **Closure for Private Variables**:

```
function person() {
  let name = 'John';
  return {
    getName: function() {
      return name;
    },
    setName: function(newName) {
      name = newName;
    }
  };
}
const john = person();
console.log(john.getName()); // Expected output: John
john.setName('Doe');
console.log(john.getName()); // Expected output: Doe
```

3. **Closure with Asynchronous Code**:

```
function delayedGreeting(name) {
  return function() {
    setTimeout(() => {
      console.log(`Hello, ${name}!`);
    }, 1000);
```

```
  };
}
const greet = delayedGreeting('Alice');
greet(); // Expected output after 1 second: Hello, Alice!
```

Event Bubbling and Capturing

1. **Event Bubbling Example**:

```
<div id="outerDiv">
  <button id="innerButton">Click me!</button>
</div>
<script>
  document.getElementById('innerButton').addEventListener('click', function() {
    alert('Button clicked!');
  });
  document.getElementById('outerDiv').addEventListener('click', function() {
    alert('Div clicked!');
  });
</script>
```

2. **Stop Event Bubbling**:

```
<div id="outerDiv">
  <button id="innerButton">Click me!</button>
</div>
<script>
  document.getElementById('innerButton').addEventListener('click', function(event) {
    alert('Button clicked!');
    event.stopPropagation();
  });
  document.getElementById('outerDiv').addEventListener('click', function() {
    alert('Div clicked!');
  });
</script>
```

3. **Event Capturing Example**:

```
<div id="outerDiv">
  <button id="innerButton">Click me!</button>
</div>
<script>
  document.getElementById('outerDiv').addEventListener('click', function() {
    alert('Div clicked!');
  }, true); // Use capturing
  document.getElementById('innerButton').addEventListener('click', function() {
    alert('Button clicked!');
  });
</script>
```

this Keyword

1. **Global Scope Example**:

```
console.log(this); // Expected output: Window object (in browsers)
```

2. **Function Context Example**:

```
function showThis() {
  console.log(this);
}
showThis(); // Expected output: Window object (in non-strict mode)
```

3. **Method Context Example**:

```
const obj = {
  value: 42,
  showValue: function() {
    console.log(this.value);
  }
};
obj.showValue(); // Expected output: 42
```

4. **Arrow Function Context Example**:

```
const obj = {
  value: 42,
  showValue: () => {
    console.log(this.value);
  }
};
obj.showValue(); // Expected output: undefined (arrow function does not have its own this)
```

5. **Event Handler Context Example**:

```
<button id="btn">Click me!</button>
<script>
  document.getElementById('btn').addEventListener('click', function() {
    console.log(this); // Expected output: The button element
  });
</script>
```

Call, Apply, and Bind Functions

1. **Using call()**:

```
const obj = { num: 10 };
function add(a, b) {
  return this.num + a + b;
}
const result = add.call(obj, 20, 30);
console.log(result); // Expected output: 60
```

2. **Using apply()**:

```
const obj = { num: 10 };
function add(a, b) {
  return this.num + a + b;
```

```
}
const result = add.apply(obj, [20, 30]);
console.log(result); // Expected output: 60
```

3. **Using bind()**:

```
const obj = { num: 10 };
function add(a, b) {
  return this.num + a + b;
}
const boundAdd = add.bind(obj);
console.log(boundAdd(20, 30)); // Expected output: 60
```

4. **Method Borrowing with call()**:

```
const obj1 = { num: 10 };
const obj2 = { num: 20 };
function showNum() {
  console.log(this.num);
}
showNum.call(obj1); // Expected output: 10
showNum.call(obj2); // Expected output: 20
```