📘 Assignment: Understanding `this` in JavaScript

🔍 Instructions:

- Carefully read each question and analyze the code.
- Predict the output and write your explanation.
- Do not run the code until all answers are written.
- Use browser or Node.js for verification after completing the assignment.

✅ 1. Global Context

```
const name = "Ram";
function say() {
  console.log(this.name);
}
say();
```

Q1: What will be the output of the above code in:

- (a) Browser?
- (b) Node.js?

✅ 2. Function Inside Object (Not a Method)

```
const user = {
  name: "Aashi",
  greet: function () {
    function inner() {
      console.log(this.name);
    }
    inner();
  }
};
user.greet();
```

Q2: What will be the output of `inner()`? Explain the behavior of `this`.

✅ 3. Arrow Function Inside a Method

```
const obj = {
  name: "Yogita",
  show: function () {
    const arrow = () => {
      console.log(this.name);
    };
    arrow();
  }
};
obj.show();
```

Q3: Predict the output and explain how `this` works inside the arrow function.

✅ 4. Arrow Function Assigned Later

```
const arrow = () => {
  console.log(this);
};

const user = {
  arrowFunc: arrow
};

user.arrowFunc();
```

Q4: Will `this` refer to `user`? Justify your answer.

✅ 5. Arrow Function Inside Constructor

```
function Person(name) {
  this.name = name;
  this.say = () => {
    console.log(this.name);
  };
}

const p = new Person("Ram");
const sayFn = p.say;
sayFn();
```

Q5: What will be the output of `sayFn()`? Explain the reference of `this`.

✅ 6. Nested Arrow and Regular Function

```
const obj = {
  name: "Guru",
  method: function () {
    const arrow1 = () => {
      function regular() {
        console.log(this.name);
      }
      regular();
    };
    arrow1();
  }
};
obj.method();
```

Q6: What will be the output and why? Focus on `this` in the regular function.

✅ 7. Returning a Regular Function from Method

```
const person = {
  name: "Rahul",
  getName: function () {
    return function () {
      console.log(this.name);
    };
  }
};

const fn = person.getName();
fn();
```

Q7: What will be logged to the console? Why?

✅ 8. Returning an Arrow Function from Method

```
const person = {
  name: "Raj",
  getName: function () {
    return () => {
      console.log(this.name);
    };
  }
};

const fn = person.getName();
fn();
```

Q8: What will be the output here? How does arrow function affect `this`?

✅ 9. Method in Nested Object

```
const outer = {
  name: "Outer",
  inner: {
    name: "Inner",
    getName: function () {
      console.log(this.name);
    }
  }
};

outer.inner.getName();
```

Q9: Which object does `this` refer to? What will be the output?

✅ 10. `this` in IIFE Inside a Method

```
const obj = {
  name: "Ram",
  method: function () {
    (function () {
      console.log(this.name);
    })();
  }
};

obj.method();
```

Q10: Analyze the behavior of `this` inside the IIFE.

✅ 11. `this` in `setTimeout` (Regular Function)

```
const obj = {
  name: "Timer",
  show: function () {
    setTimeout(function () {
      console.log(this.name);
    }, 0);
  }
};

obj.show();
```

Q11: What will be logged? What does `this` refer to inside `setTimeout`?

✅ 12. `this` in `setTimeout` (Arrow Function)

```
const obj = {
  name: "Timer",
  show: function () {
    setTimeout(() => {
      console.log(this.name);
    }, 0);
  }
};

obj.show();
```

Q12: What will be the output? Why is arrow function behavior different?