Modern JavaScript — Reference Guide

Strict Mode

```
"use strict";
// Catches common mistakes: undeclared variables, duplicate
        params, etc.
```

Classes

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    return `${this.name} makes a sound.`;
  }
}

class Dog extends Animal {
  speak() {
    return `${this.name} barks.`;
  }
}
```

Error Handling Patterns

```
class AppError extends Error {
  constructor(message, statusCode) {
    super(message);
    this.statusCode = statusCode;
  }
}

try {
  throw new AppError("Not found", 404);
} catch (err) {
  if (err instanceof AppError) {
    console.log(err.statusCode, err.message);
```

```
  }
}
```

Local Storage

```
localStorage.setItem("theme", "dark");
const theme = localStorage.getItem("theme");  // "dark"
localStorage.removeItem("theme");
```

Fetch API

```
const res = await fetch("/api/data", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ name: "test" }),
});
const data = await res.json();
```

Key Takeaways

1. Use classes for structured, reusable object-oriented code.
2. Custom error classes help categorize and handle errors cleanly.
3. The Fetch API is the modern replacement for XMLHttpRequest.
4. Store user preferences in localStorage, sensitive data in secure cookies.
