nlohmann::json Cheat Sheet (C++)

1. Create JSON Objects

```
json j = { {"id", 1}, {"name", "Siva"}, {"email", "siva@example.com"} };
```

2. Parse from JSON String

```
json j = json::parse(R"({"id":1, "name":"Siva"})");
```

3. Parse with Error Handling

```
try {
    json j = json::parse(some_str);
} catch (const json::parse_error& e) {
    std::cerr << "Parse failed: " << e.what() << std::endl;
}
```

4. Serialize to String

```
std::string str = j.dump();       // Compact
std::string pretty = j.dump(4);   // Pretty-print (4 spaces)
```

5. Access Elements

```
int id = j["id"];
std::string name = j["name"];
```

6. Check Existence

```
if (j.contains("email")) { ... }
```

7. Get with Default

```
std::string name = j.value("name", "Unknown");
int age = j.value("age", 0);
```

8. Update/Add Elements

```
j["age"] = 35;
```

9. Remove Key

```
j.erase("email");
```

10. Loop Over JSON Object

```
for (auto& [key, value] : j.items()) {
    std::cout << key << ": " << value << std::endl;
}
```

11. JSON Arrays

```
json arr = json::array({1, 2, 3});
for (auto& v : arr) std::cout << v << std::endl;
```

12. Convert Struct to/from JSON

```
struct User {
    int id;
    std::string name;
};
void to_json(json& j, const User& u) {
    j = json{{"id", u.id}, {"name", u.name}};
}
void from_json(const json& j, User& u) {
    j.at("id").get_to(u.id);
    j.at("name").get_to(u.name);
}
```

13. Type Checking

```
j["id"].is_number();
j["tags"].is_array();
```

14. Merge / Update

```
j.update(json{{"email", "new@example.com"}});
```