Serialization

CMSC 240 Software Systems Development

Today – Serialization

- What is serialization?
- Demo
- In-Class Exercise

RICHMOND

What is Serialization?

- The process of converting an object or a data structure into a format that can be stored (in a file or memory) or transmitted (over a network)
- Serialized data should encapsulate the object's state so that it can be recreated later

What is Deserialization?

- The reverse process of serialization
- It involves converting the serialized data back into the object or data structure it represents, effectively "rebuilding" the object from its serialized state

Serialization in C++

- Saving and loading C++ objects and data structures:
  - Classes
  - Structs
  - Vectors

- Serialization is used for:
  - Persistence
  - Storing data in a file
  - Sending messages over a network
  - Reading configuration files

Serialization Formats

- Some common serialization formats:

CSV

```
Color,Engine/Horsepower,Engine/Type,Make,Model,Price,Year
Red,301,V8,Ford,Mustang,38999.42,2004
```

YAML

```yaml
Color: Red
Engine:
    Horsepower: 301
    Type: V8
Make: Ford
Model: Mustang
Price: 38999.42
Year: 2004
```

XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<car>
  <Color>Red</Color>
  <Engine>
    <Horsepower>301</Horsepower>
    <Type>V8</Type>
  </Engine>
  <Make>Ford</Make>
  <Model>Mustang</Model>
  <Price>38999.42</Price>
  <Year>2004</Year>
</car>
```

JSON (**J**ava **S**cript **O**bject **N**otation)

- JSON Syntax Rules
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays

```
{
    "Color": "Red",
    "Engine": {
        "Horsepower": 301,
        "Type": "V8"
    },
    "Make": "Ford",
    "Model": "Mustang",
    "Price": 38999.42,
    "Year": 2004
}
```

JSON (**J**ava **S**cript **O**bject **N**otation)

- JSON Syntax Rules
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays

```
{
  "array": [
    1,
    2,
    3
  ],
  "boolean": true,
  "string": "Hello World",
  "null": null,
  "number": 123,
  "object": {
    "a": "b",
    "c": "d"
  }
}
```

External Library: JSON for Modern C++

- https://json.nlohmann.me/

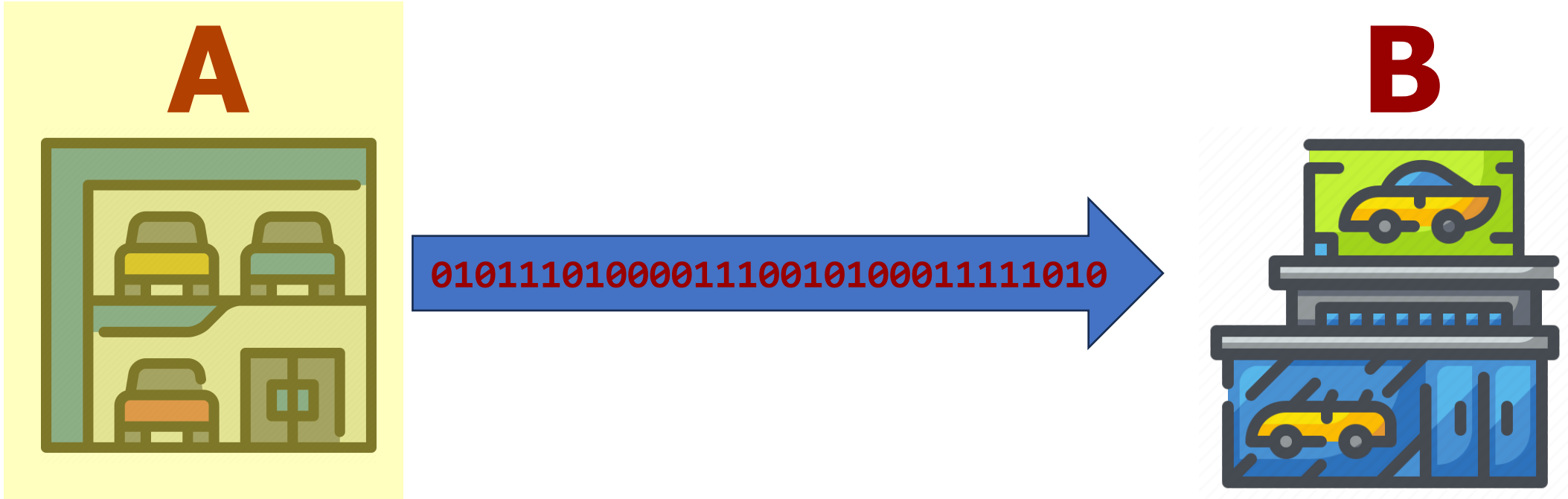
External Library: JSON for Modern C++

- https://json.nlohmann.me/

```
#include <json.hpp>
using namespace nlohmann;
```

Example Use Case: Car Dealership

Car dealership **A** needs to send information about a specific vehicle to car dealership **B**



Convert a Car object to JSON

- You want to save the state of an instance of the Car class

```
class Car
{
public:
    Car() { };
    Car(std::string make, std::string model, int year, std::string color,
        double price, std::string engineType, int horsepower);
    Car(json jsonDoc);
    void drive();
    json toJson();
    void fromJson(json jsonDoc);
private:
    std::string make;
    std::string model;
    int year;
    std::string color;
    double price;
    Engine engine;
};
```

Convert a Car object to JSON

- Creating a new instance of Car

```
Car mustang{"Ford", "Mustang", 2004, "Red", 38999.42, "V8", 301};
```

Convert a Car object to JSON

- The **`toJson()`** method

```
json Car::toJson()
{
    return json{{"Make", make}, {"Model", model}, {"Year", year},
                {"Color", color}, {"Price", price},
                {"Engine", engine.toJson()}};
}
```

Convert a Car object to JSON

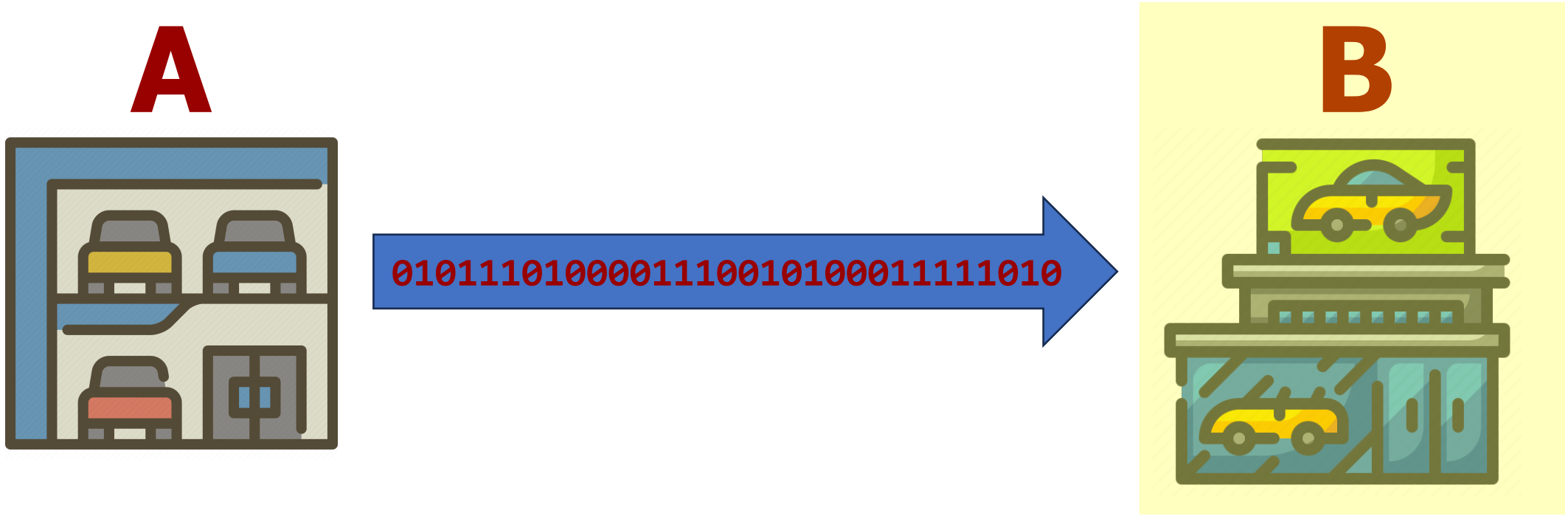
- JSON text file


```
{
    "Color": "Red",
    "Engine": {
        "Horsepower": 301,
        "Type": "V8"
    },
    "Make": "Ford",
    "Model": "Mustang",
    "Price": 38999.42,
    "Year": 2004
}
```

Example Use Case: Car Dealership

Example Use Case: Car Dealership

Car dealership **A** needs to send information about a specific vehicle to car dealership **B**



Convert a JSON file to a Car object

- The **`fromJson()`** method

```
void Car::fromJson(json jsonDoc)
{
  jsonDoc.at("Make").get_to(make);
  jsonDoc.at("Model").get_to(model);
  jsonDoc.at("Year").get_to(year);
  jsonDoc.at("Color").get_to(color);
  jsonDoc.at("Price").get_to(price);
  engine = Engine{jsonDoc.at("Engine")};
}
```

Demo