MODULE 5 – SERVER-SIDE DATA FORMATS

IT 207 – IT Programming

GEORGE MASON UNIVERSITY

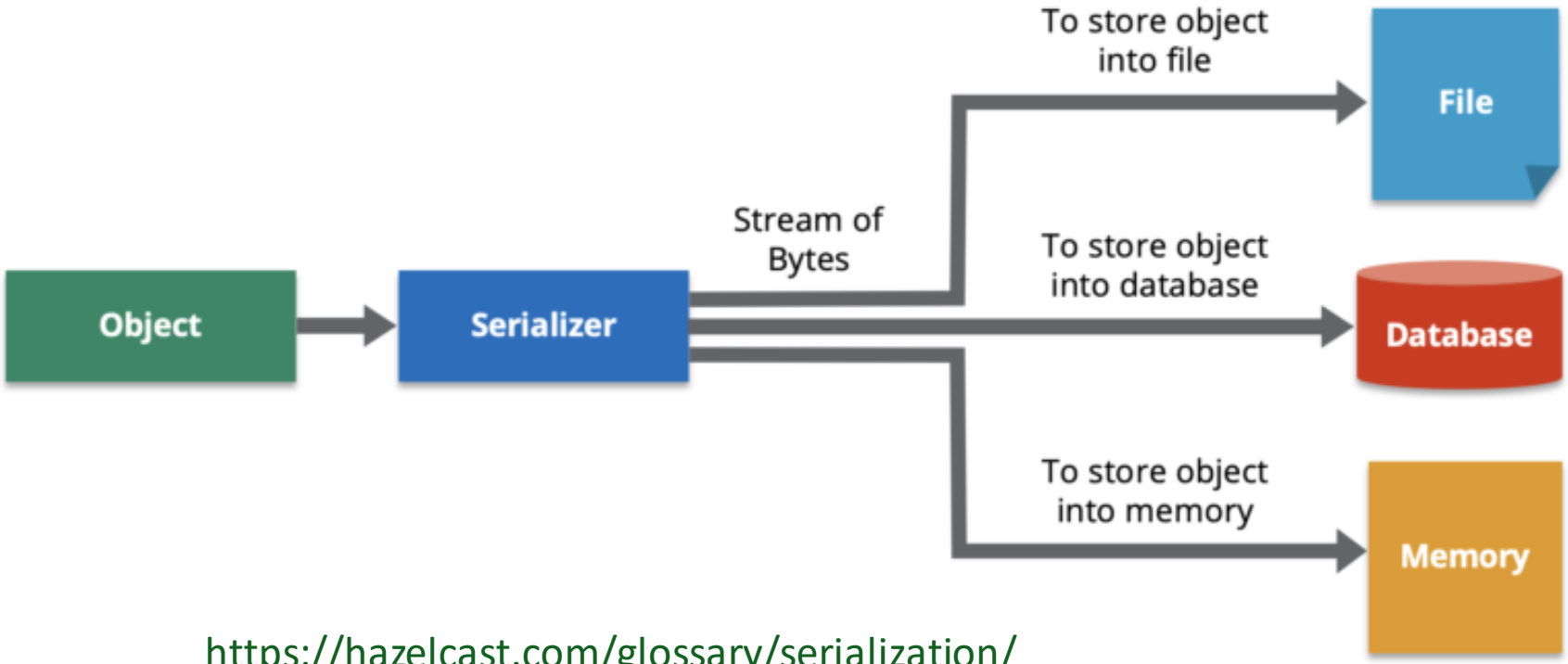
Lecture Outline

- State the motivation for Data Formats
- List Data Formats used in modern APIs.
- Compare JavaScript Objects to JavaScript Object Notations
- Use JSON in reading and writing data in Nodejs
- Experiment with the URL

Motivation for Data Formats

- In server-client communication data is exchanged between the server and the client over the network.
  - Exchanged data must be in a format that can be understood by both the server and the client applications
  - Exchanged data must be serialized to be transferred on the network

DATA SERIALIZATION



https://hazelcast.com/glossary/serialization/

- Data serialization is the process of converting an object into a stream of bytes to more easily save or transmit it.
- Serialized data is represented in a text format that can be understood by the communicating parties

Types of Serialization Formats

- All serialization formats are text formats.
- The most common text format used in Nodejs applications are:
  - JSON: JavaScript Object Notation
  - XML: eXtensible Markup Language
  - CSV: Comma Separated Values
  - Buffer: raw data

JAVASCRIPT OBJECTS

JavaScript Data Types

- JavaScript has two categories of data types: primitives and objects.
  - Primitive types are the simplest, most basic types in JavaScript.
  - **The primitive types are:**
    - string, number, boolean, null, undefined, bigint, and symbol.
  - Objects include, but aren't limited to, the following types:
    - Simple Object, Array, Date, Function

JavaScript Objects

- In JS objects store a collection of key-value pairs
  - The key-value pairs are called the object properties.
  - The values of each pair can be any type.
- Object properties can be added, updated and/or deleted

```
let person = {
    name:    'Jane',
    age:     37,
    hobbies: ['photography', 'genealogy'],
};
```

Each pair has a key, a colon, and a value

Each key-value pair ends with a comma

The comma that follows the last pair is optional

Braces delimit the list of key-value pairs contained by the object

Quotes for keys are omitted if they consist entirely of alphanumeric characters and underscores

Accessing Object Values

```
let person = {
        name: 'Jane',
        age: 37,
        hobbies: ['photography', 'genealogy'],
};
```

- The value of a key in an object can be accessed in two ways

1. Dot Notation

```
person.name   // dot notation

        ==> 'Jane'
```

2. Bracket Notation

```
person['age'] // bracket notation
        ==> 37

let key = 'name'
person[key]
        ==> 'Jane'
```

Deleting Object Properties

```
let person = {
        name: 'Jane',
        age: 37,
        hobbies: ['photography', 'genealogy'],
};
```

- The delete keyword removes the key-value pair from the object and returns true.

```
delete person.age // dot notation
= true

delete person['hobbies'] // bracket notation
= true

person
= { name: 'Jane' }
```

FUNCTIONS OVERVIEW

- Like all programming languages, a function in JS is a named unit of code.
  - Function definition

```
function funcName() {
        func_body;
}
```

- Functions can be defined with any number of parameters and may be passed any number of arguments.
  - Arguments are objects or primitive values passed to the function
  - parameters are declarations for the local variables used inside the function to access the arguments.
- Function names and parameters are both considered variable names in JavaScript.

Function Definition in JS

- There are three ways to define a function in JS
  1. Function declaration

```
greetPeople();

function greetPeople() {
        console.log("Good Morning!");
}
```

In JS a function can be called before it is declared

Function Definition in JS

- There are three ways to define a function in JS

2. Function expression:

   - JavaScript functions are first-class functions that can be assigned to a variable.

```
let greetPeople = function () {
console.log("Good Morning!");
};

greetPeople();
```

A function declared using a function expression **can not** be called before it is declared

Function Definition in JS

- There are three ways to define a function in JS
  3. Arrow function
     - Arrow functions are similar to function expressions, but they use a different syntax.

```
let greetPeople = () => console.log("Good Morning!");

greetPeople();
```

Objects and Methods

- When the value on an object is a function, the property is called a method.
  - Methods are used to describe the behavior of the object
- Object methods can be defined on an object in different ways:
  1. Function expression

```
let person = {
        firstName: 'John',
        lastName: 'Doe'
};

person.greet = function () { //adds the greet method to the person object
        console.log('Hello!');
}
person.greet();
```

Objects and Methods

- When the value on an object is a function, the property is called a method.
  - Methods are used to describe the behavior of the object
- Object methods can be defined on an object in different ways:
  2. Direct assignment

```
let person = {
        firstName: 'John',
        lastName: 'Doe'
};

function greet() {
        console.log('Hello, World!');
}

person.greet = greet;
person.greet();
```

Objects and Methods

- When the value on an object is a function, the property is called a method.
  - Methods are used to describe the behavior of the object
- Object methods can be defined on an object in different ways:
  3. Concise method syntax

```
let person = {
        firstName: 'John',
        lastName: 'Doe',
        greet() {
                console.log('Hello, World!');
        }
};

person.greet();
```

THIS IN A METHOD

- In JavaScript, the this keyword refers to an object.
- When used in an object method, this refers to the object

```
const person = {
        firstName: "John",
        lastName : "Doe",
        id : 5566,
        fullName : function() {
                return this.firstName + " " + this.lastName;
        }
};
```

John Doe

JAVASCRIPT OBJECT NOTATION

JSON Overview

- JSON stands for JavaScript Object Notation
- JSON is a text-based data format that is used to store and transfer data.
  - JSON syntax is self-describing and human readable
  - JSON is mostly used when data is sent from a server to a web page
- JSON is independent of any programming language.
- The bulk of modern programming languages contain code that can generate and parse JSON data.

JSON Syntax

- JSON was derived from JavaScript, so its syntax resembles JavaScript object literal syntax.
- JSON Object
  - Written inside curly braces { }.
  - Can contain multiple key/value pairs.
  - Both the key and values are written in double quotes and are separated by a colon :
- JSON Array
  - Written inside square brackets [ ]
  - Could contain literal, objects, or arrays

```
// JSON object
{
  "name": "John",
  "age": 22,
  "gender": "male",
}
```

```
// JSON array
[ "apple", "mango", "banana"]
// JSON array containing objects
[
  { "name": "John", "age": 22 },
  { "name": "Peter", "age": 20 }.
  { "name": "Mark", "age": 23 }
]
```

JSON data cannot contain functions as values.

ACCESSING JSON DATA

- JSON data can be accessed using the dot notation or the bracket notation

```
// JSON object
const data = {
                "name": "John",
                "age": 22,
                "hobby": {
                "reading" : true,
                "gaming" : false,
                "sport" : "football"
                },
                "class" : ["JavaScript", "HTML", "CSS"]
}
// accessing JSON object
console.log(data.name); // John

console.log(data.hobby); // { gaming: false, reading: true, sport: "football"}

console.log(data.hobby.sport); // football

console.log(data.class[1]); // HTML

console.log(data["name"]); // John
```

JavaScript Objects vs JSON

JSON

- The key in key/value pair should be in double quotes.
- JSON cannot contain functions.
- JSON can be created and used by other programming languages.

JavaScript Objects

- The key in key/value pair can be without double quotes.
- JavaScript objects can contain functions.
- JavaScript objects can only be used in JavaScript.

READING AND WRITING JSON

- JS provides two methods to handle JSON data:
  1. JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.
  2. JSON.stringify method return a JSON string corresponding to the specified value.

```
const carObj1 = {
    make: "Ford",
    model:"Fusion",
    year:2018,
    rate_dollars:3000
};
const CarStr = JSON.stringify(carObj1); //converts an object to a string
console.log(CarStr);
const carObj2 = JSON.parse(CarStr); //converts a string to an object
console.log (carObj2);
```

READING AND WRITING JSON

- JS provides two methods to handle JSON data:
  1. JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.
  2. JSON.stringify method return a JSON string corresponding to the specified value.

```
const carObj1 = {
    make: "Ford",
    model:"Fusion",
    year:2018,
    rate_dollars:3000
};
const CarStr = JSON.stringify(carObj1); //converts an object to a string
console.log(CarStr);
const carObj2 = JSON.parse(CarStr); //converts a string to an object
console.log (carObj2);
```

```
Module 5 % node JsonEx.js
{"make":"Ford","model":"Fusion","year":2018,"rate_dollars":3000}
{ make: 'Ford', model: 'Fusion', year: 2018, rate_dollars: 3000 }
```

INSPECTING THE URL OBJECT

URL Module

- The URL module is one of the core modules in Node.js.
- The URL module contains functions that help in parsing a URL.
- To have access to the URL functions, the URL module must be imported using a require statement `require('url')`

Nodejs documentation at https://nodejs.org/docs/v0.4.7/api/url.html provides details of the extracting parts of the URL.

Query String Overview

- A query string is a set of characters tacked onto the end of a URL.
- The query string begins after the question mark (?) and can include one or more parameters.
- Each parameter is represented by a unique key-value pair or a set of two linked data items.
  - An equal sign (=) separates each key and value.
  - An ampersand (&) separates multiple parameters
  - A plus (+) replaces a white space.
- The query string is the way to pass information to the web server, telling it what content to deliver or action to take

https://www.tech.com/search?query=database+tools&star_rating=4&order=alphabetical

Query String

Extracting the Query String

- url.parse() method parses the URL and return a URL object with each part of the address as properties:

```
const url = require('url’);
const adr = 'http://localhost:8080/default.htm?year=2017&month=february’;
const q = url.parse(adr, true); // true: use the query string module to parse the query


console.log(q.host); //returns 'localhost:8080'
console.log(q.pathname); //returns '/default.htm'
console.log(q.search); //returns '?year=2017&month=february’

const qdata = q.query; //returns an object: { year: 2017, month: 'february' }
console.log(qdata.month); //returns 'february'
```

The server response can be customized based on the query string parameters

Extracting the Query String using WHATWG API

- WHATWG stands for Web Hypertext Application Technology Working Group
  - A community of people interested in evolving HTML and related technologies.
- Keeping with the WHATWG URL Standard used by web browsers, Nodejs provides a new API for parsing the URL and extracting the Query String
  - URL class must be accessed via require('url').
  - URLSearchParams class provides methods to access the query sting of a URL

https://nodejs.org/dist/latest-v8.x/docs/api/url.html

THE WHATWG API

```
const {URL} = require('url');
const adr = 'http://localhost:8080/default.htm?year=2017&month=february';
const url = new URL(adr); //Creates a new URL object by parsing adr
console.log(url.host);//returns 'localhost:8080'
console.log(url.pathname);//returns '/default.htm'
console.log(url.search);//returns '?year=2017&month=february’

const urlquery = url.searchParams;
//returns an instance of URLSearchParams { 'year' => '2017', 'month' => 'february’ }

console.log(urlquery);//returns 'february'
console.log(urlquery.get('month'));//returns 'february'
console.log(urlquery.get('year'));//returns ‘2017'
```

SUMMARY

- Data serialization is a technique that facilitates the transfer of data from one machine to another over the network.
- There are several serialization formats used in web applications
- JSON is a serializable data format that is supported different programming languages.
- JSON is a text-based data format derived from JS Objects
- JS provides JSON.parse() and JSON.stringify to handle JSON data
- A query string is a set of characters tacked onto the end of a URL.
- The query string is used to customize the server response
- The query string can be extractedfrom the URL using the url.parse() method and setting the second parameter to true