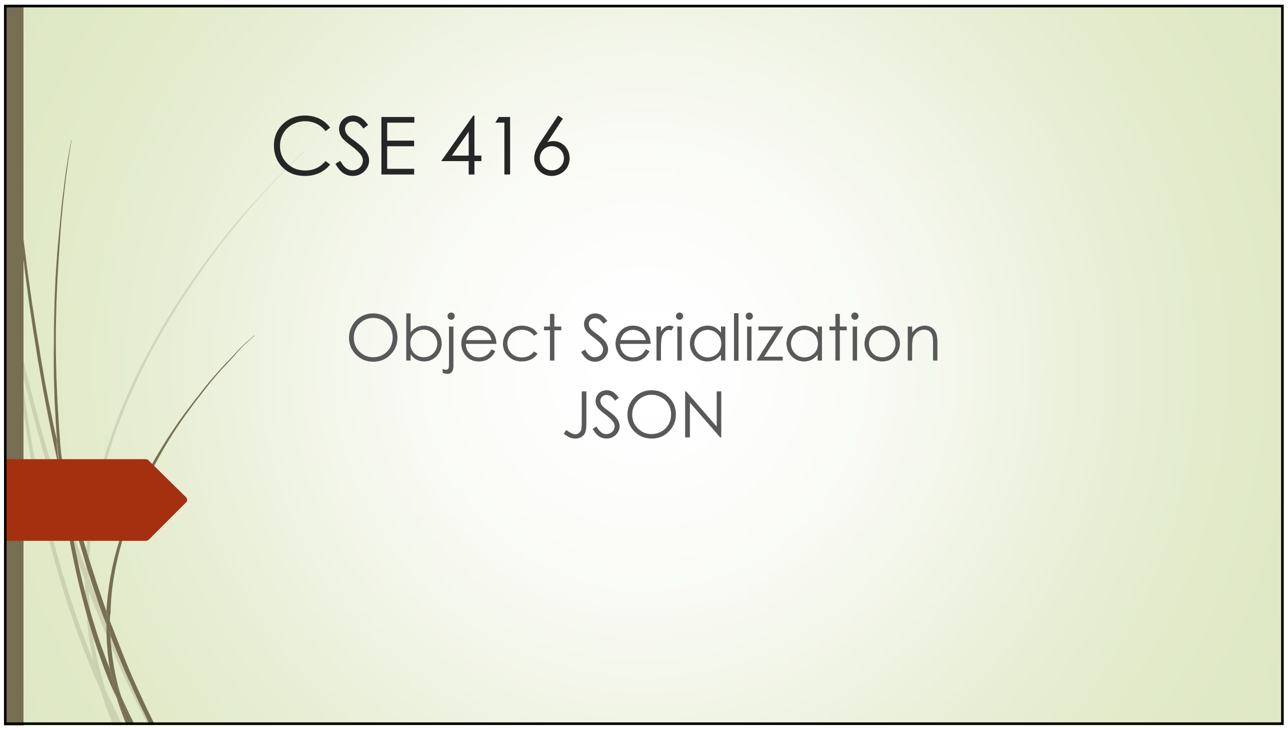
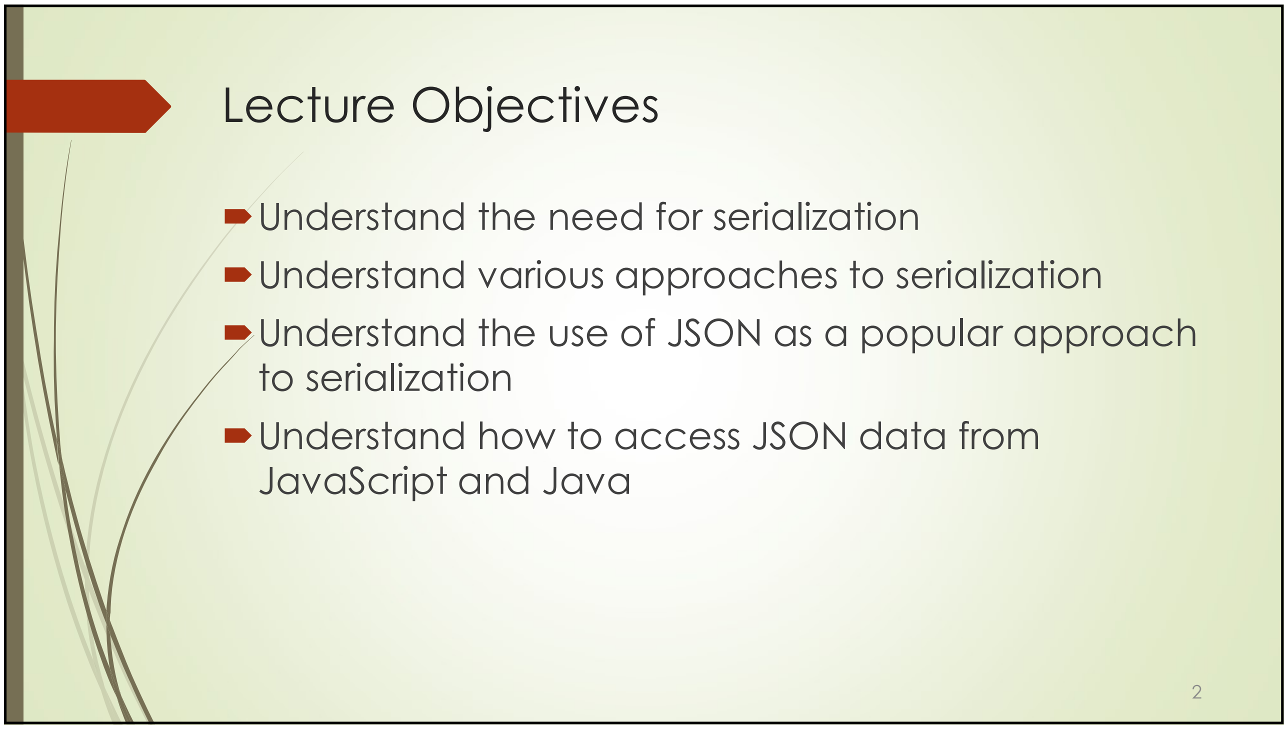


CSE 416

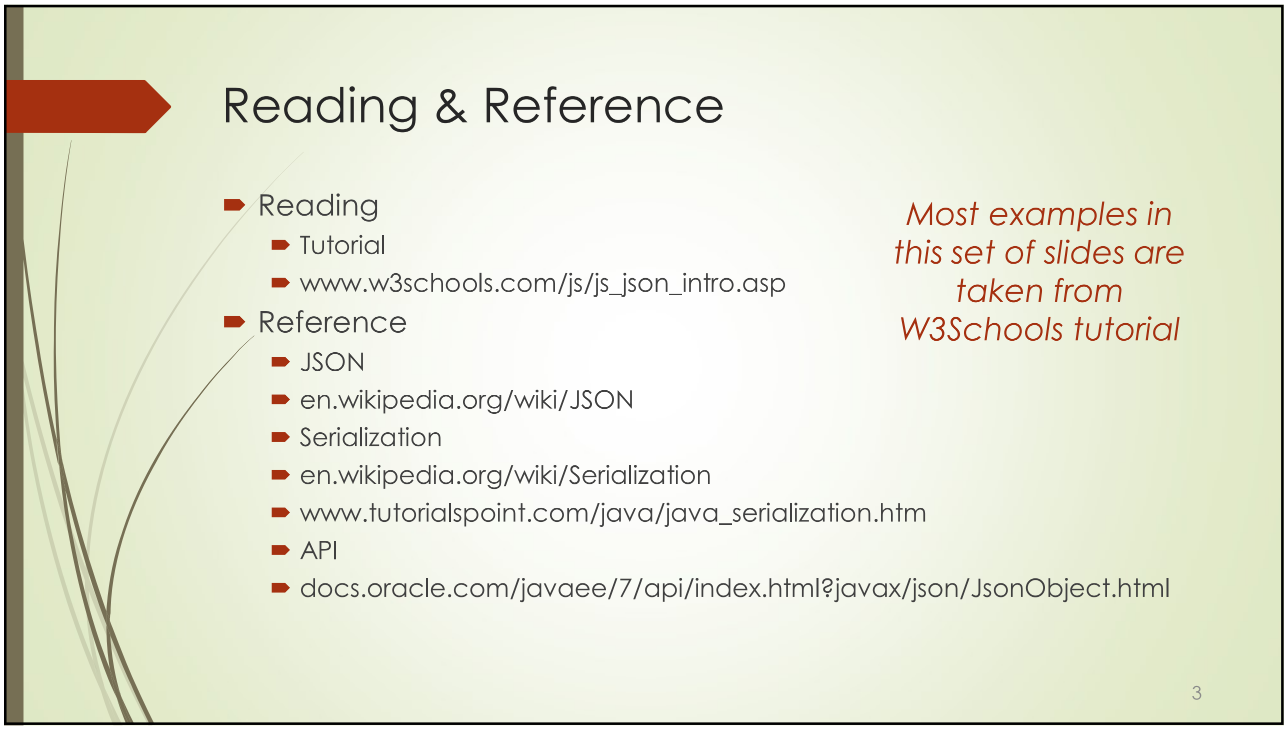
Object Serialization
JSON



Lecture Objectives

- Understand the need for serialization
- Understand various approaches to serialization
- Understand the use of JSON as a popular approach to serialization
- Understand how to access JSON data from JavaScript and Java

2

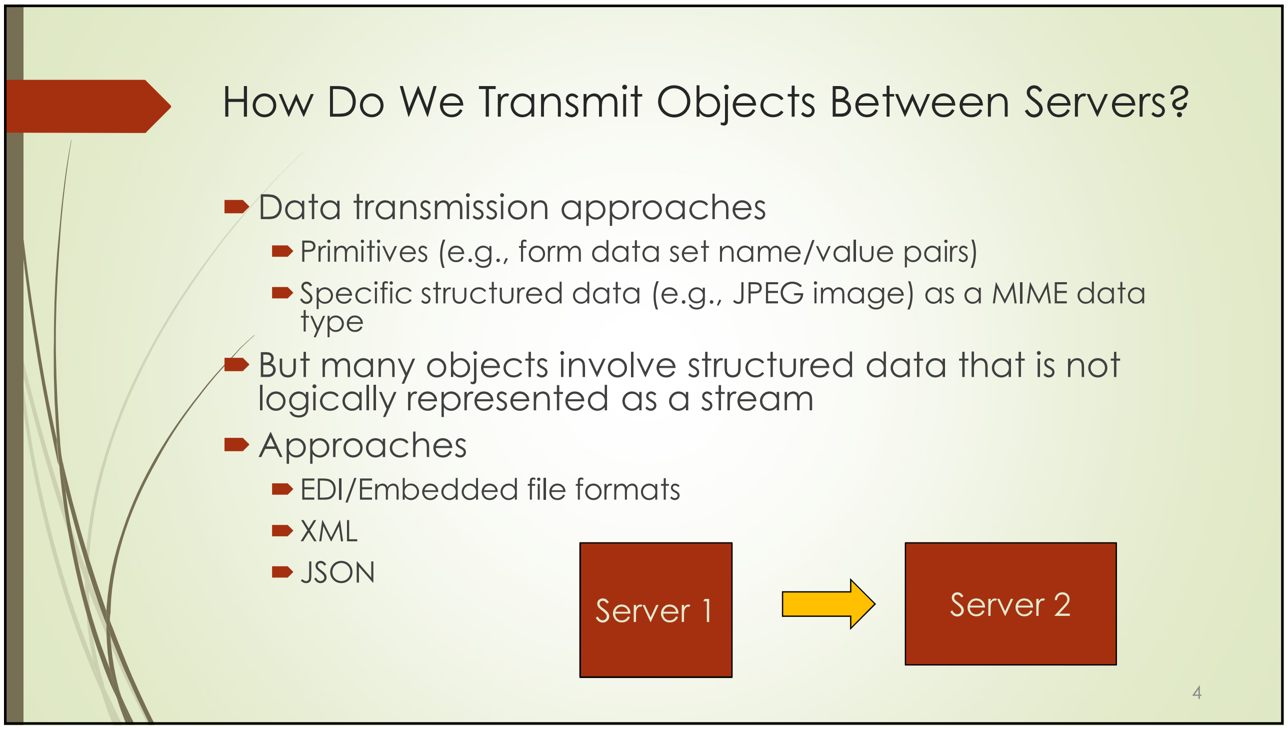


Reading & Reference

- Reading
  - Tutorial
  - www.w3schools.com/js/js_json_intro.asp
- Reference
  - JSON
  - en.wikipedia.org/wiki/JSON
  - Serialization
  - en.wikipedia.org/wiki/Serialization
  - www.tutorialspoint.com/java/java_serialization.htm
  - API
  - docs.oracle.com/javaee/7/api/index.html?javax/json/JsonObject.html

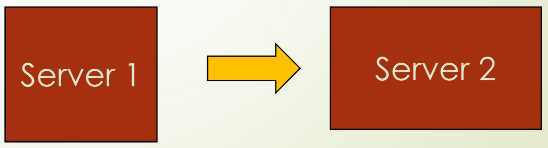
Most examples in this set of slides are taken from W3Schools tutorial

3

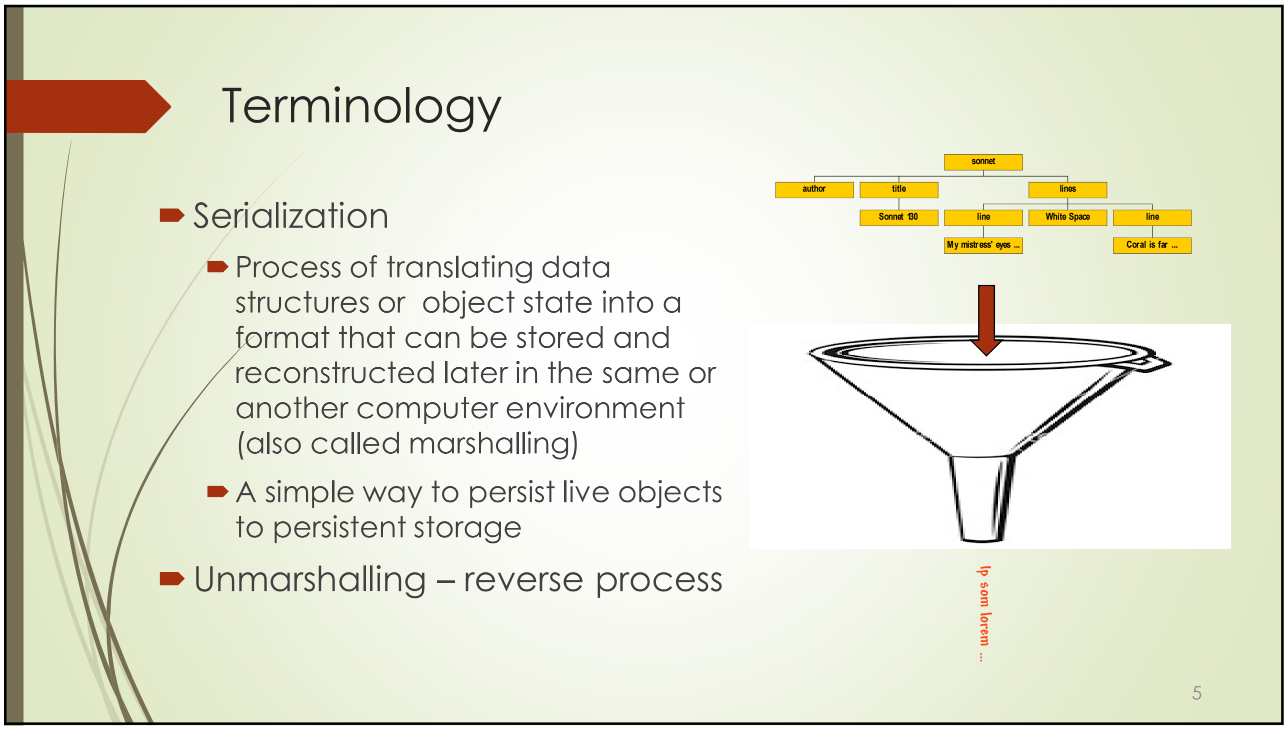


How Do We Transmit Objects Between Servers?

- Data transmission approaches
  - Primitives (e.g., form data set name/value pairs)
  - Specific structured data (e.g., JPEG image) as a MIME data type
- But many objects involve structured data that is not logically represented as a stream
- Approaches
  - EDI/Embedded file formats
  - XML
  - JSON

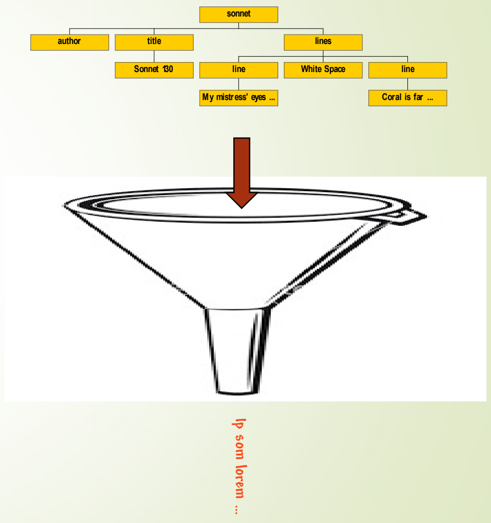


4

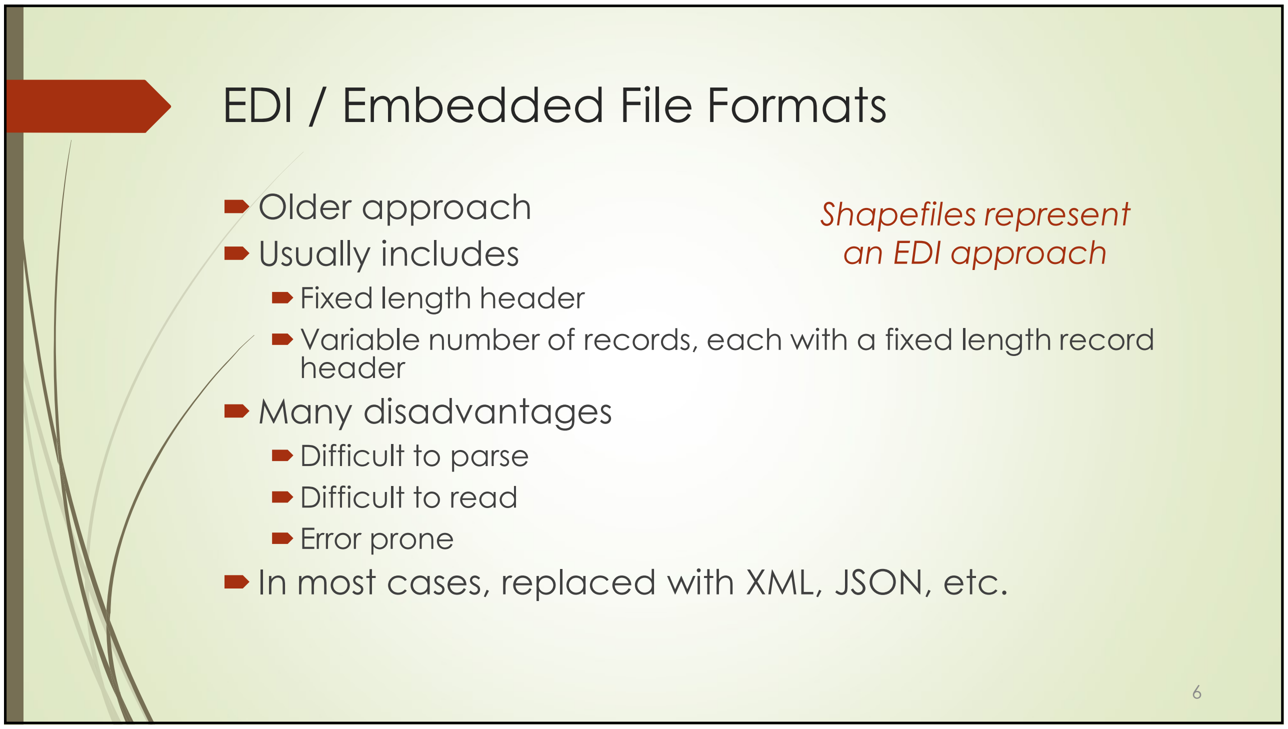


Terminology

- Serialization
  - Process of translating data structures or object state into a format that can be stored and reconstructed later in the same or another computer environment (also called marshalling)
  - A simple way to persist live objects to persistent storage
- Unmarshalling – reverse process



5

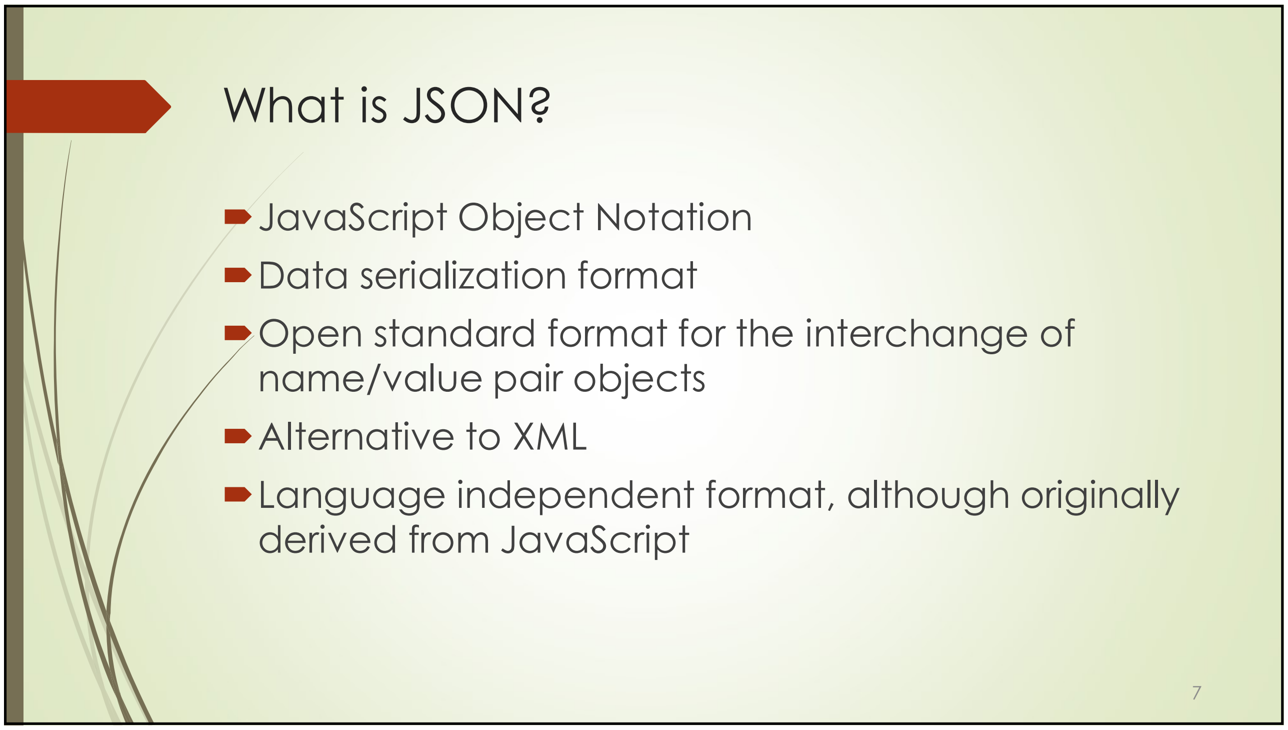


EDI / Embedded File Formats

- Older approach
- Usually includes
  - Fixed length header
  - Variable number of records, each with a fixed length record header
- Many disadvantages
  - Difficult to parse
  - Difficult to read
  - Error prone
- In most cases, replaced with XML, JSON, etc.

Shapefiles represent an EDI approach

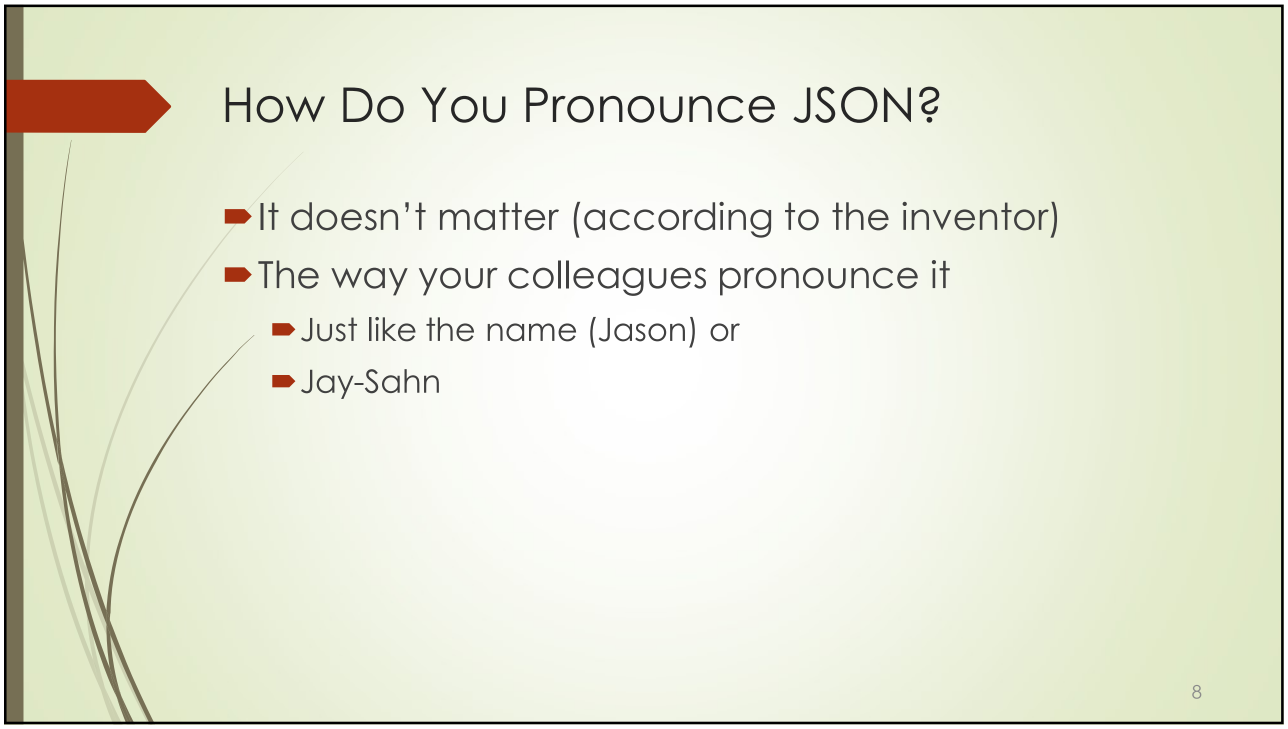
6



What is JSON?

- JavaScript Object Notation
- Data serialization format
- Open standard format for the interchange of name/value pair objects
- Alternative to XML
- Language independent format, although originally derived from JavaScript

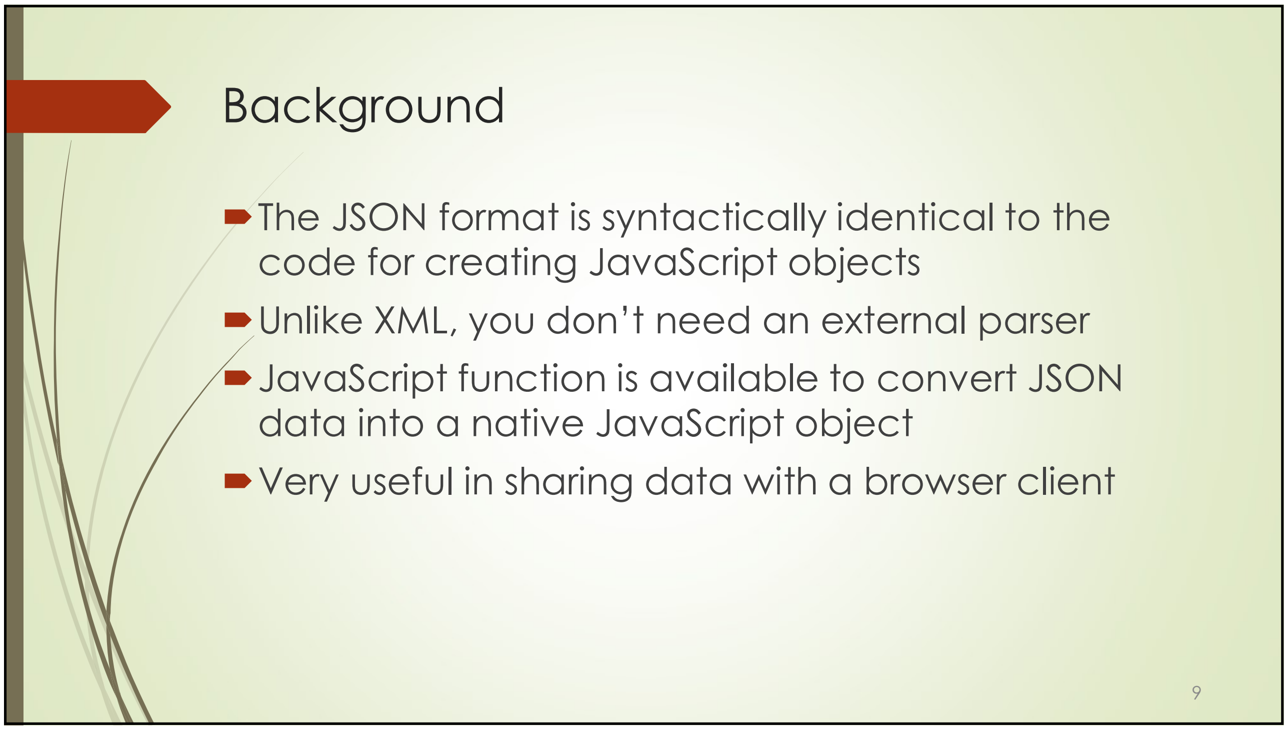
7



How Do You Pronounce JSON?

- It doesn't matter (according to the inventor)
- The way your colleagues pronounce it
  - Just like the name (Jason) or
  - Jay-Sahn

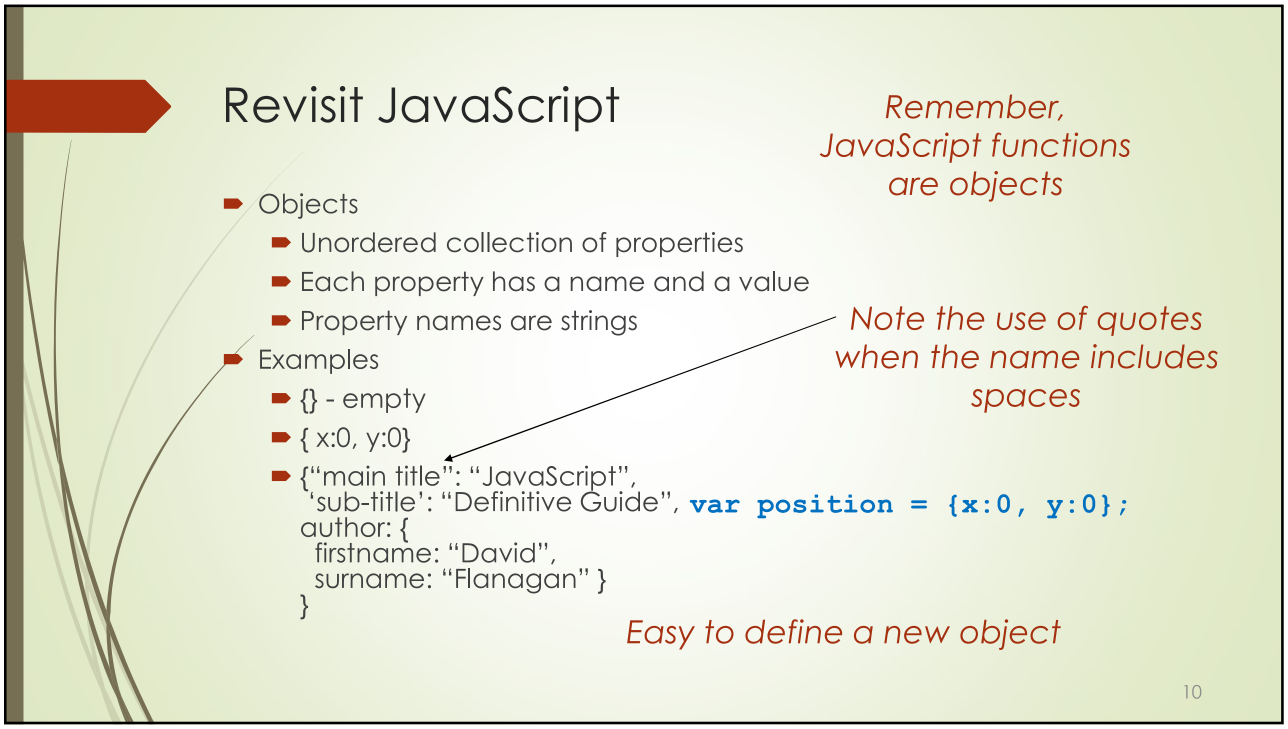
8



Background

- The JSON format is syntactically identical to the code for creating JavaScript objects
- Unlike XML, you don't need an external parser
- JavaScript function is available to convert JSON data into a native JavaScript object
- Very useful in sharing data with a browser client

9



Revisit JavaScript

Remember, JavaScript functions are objects

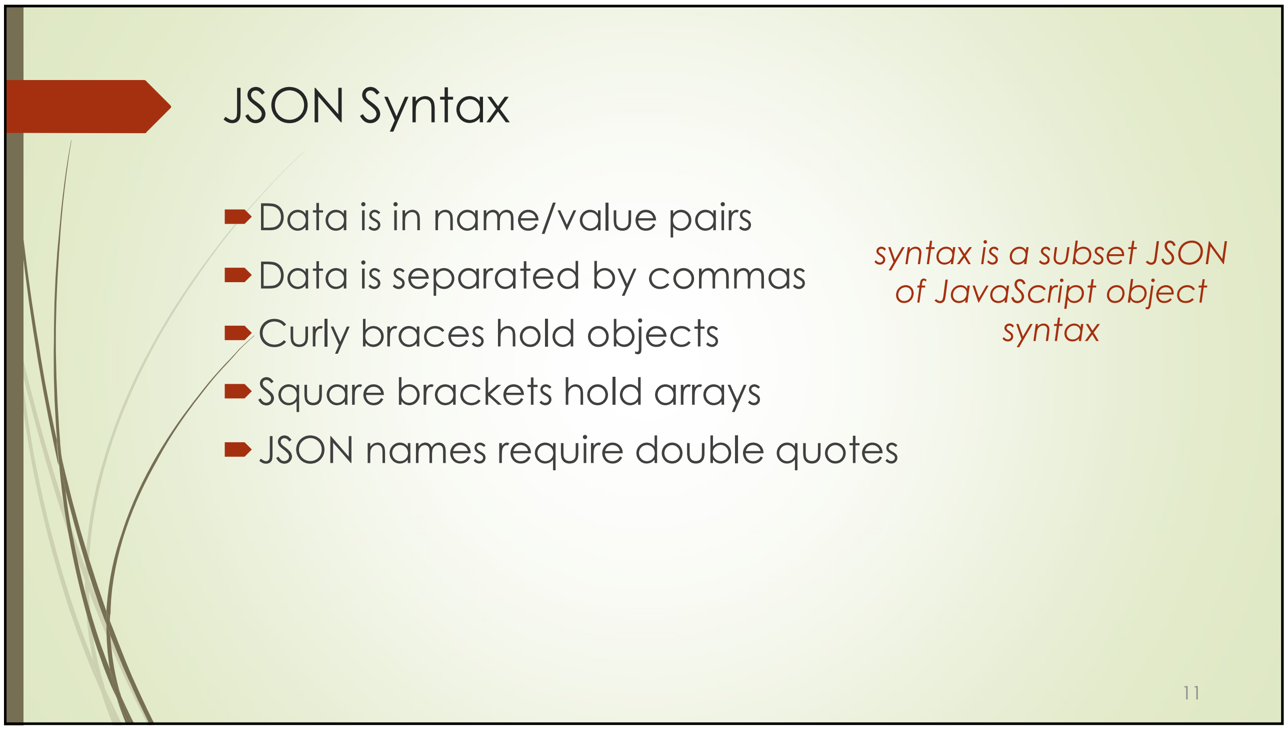
- Objects
  - Unordered collection of properties
  - Each property has a name and a value
  - Property names are strings
- Examples
  - {} - empty
  - { x:0, y:0}
  - {"main title": "JavaScript",
    'sub-title': "Definitive Guide",
    author: {
      firstname: "David",
      surname: "Flanagan" }
    }

Note the use of quotes when the name includes spaces

```
var position = {x:0, y:0};
```

Easy to define a new object

10

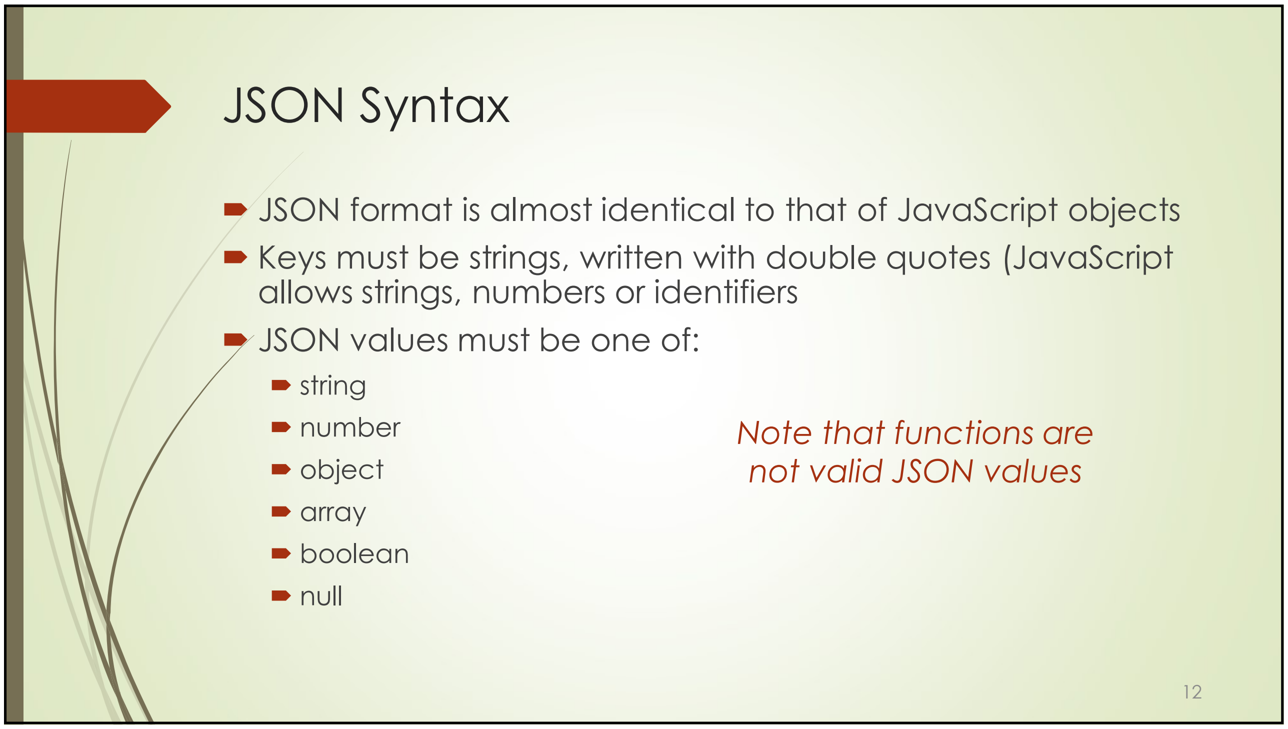


JSON Syntax

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays
- JSON names require double quotes

syntax is a subset JSON of JavaScript object syntax

11



JSON Syntax

- JSON format is almost identical to that of JavaScript objects
- Keys must be strings, written with double quotes (JavaScript allows strings, numbers or identifiers
- JSON values must be one of:
  - string
  - number
  - object
  - array
  - boolean
  - null

Note that functions are not valid JSON values

12

Arrays

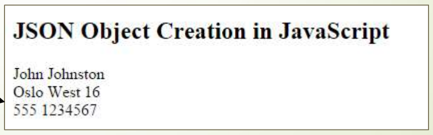
- Arrays
  - Ordered collection of values
  - Untyped
  - Array elements may be objects or other arrays

Example

- Code below shows parsing of JSON text data

```
<!DOCTYPE html>
<html>
<body>
<h2>JSON Object Creation in JavaScript</h2>
<p id="demo"> </p>
<script>
var text = '{"name":"John Johnson", "street":"Oslo West 16",
"phone":"555 1234567"}';
var obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
  obj.name + "<br>" +
  obj.street + "<br>" +
  obj.phone;
</script>
</body>
</html>
```

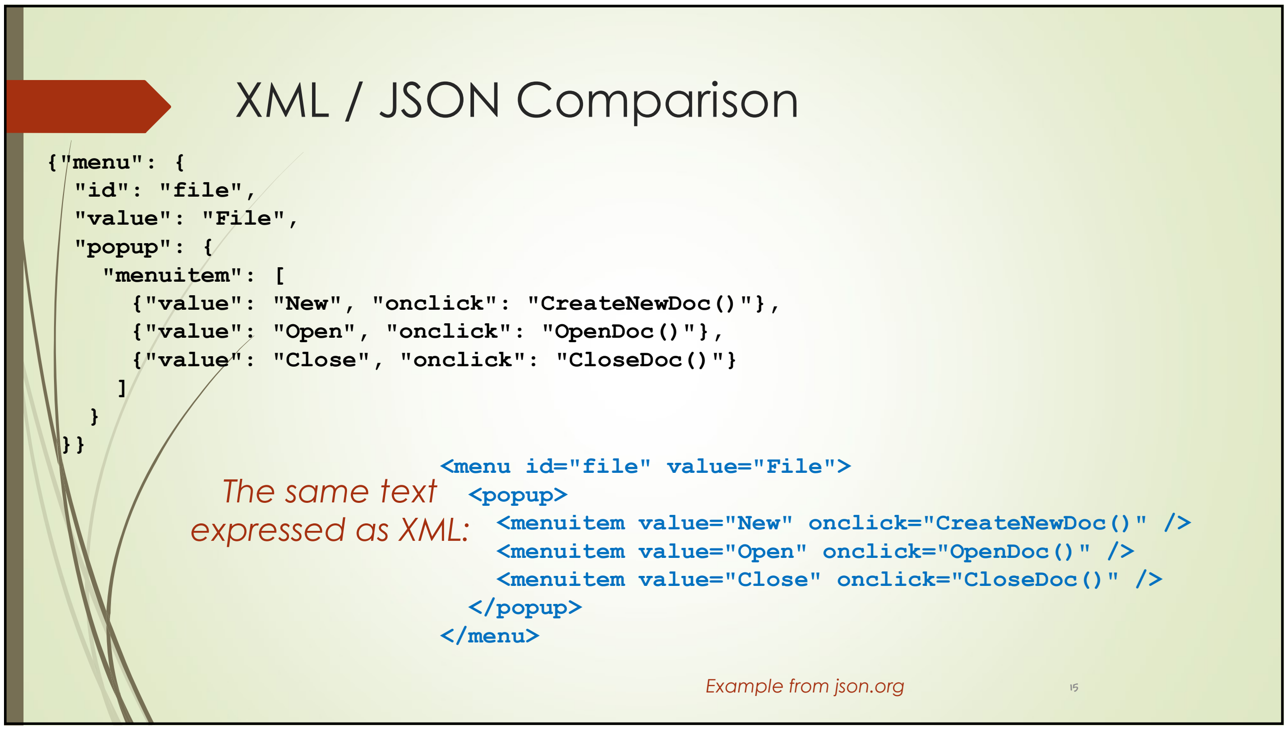
Parses a JSON formatted string



JSON Object Creation in JavaScript

John Johnston
Oslo West 16
555 1234567

Example from W3Schools



XML / JSON Comparison

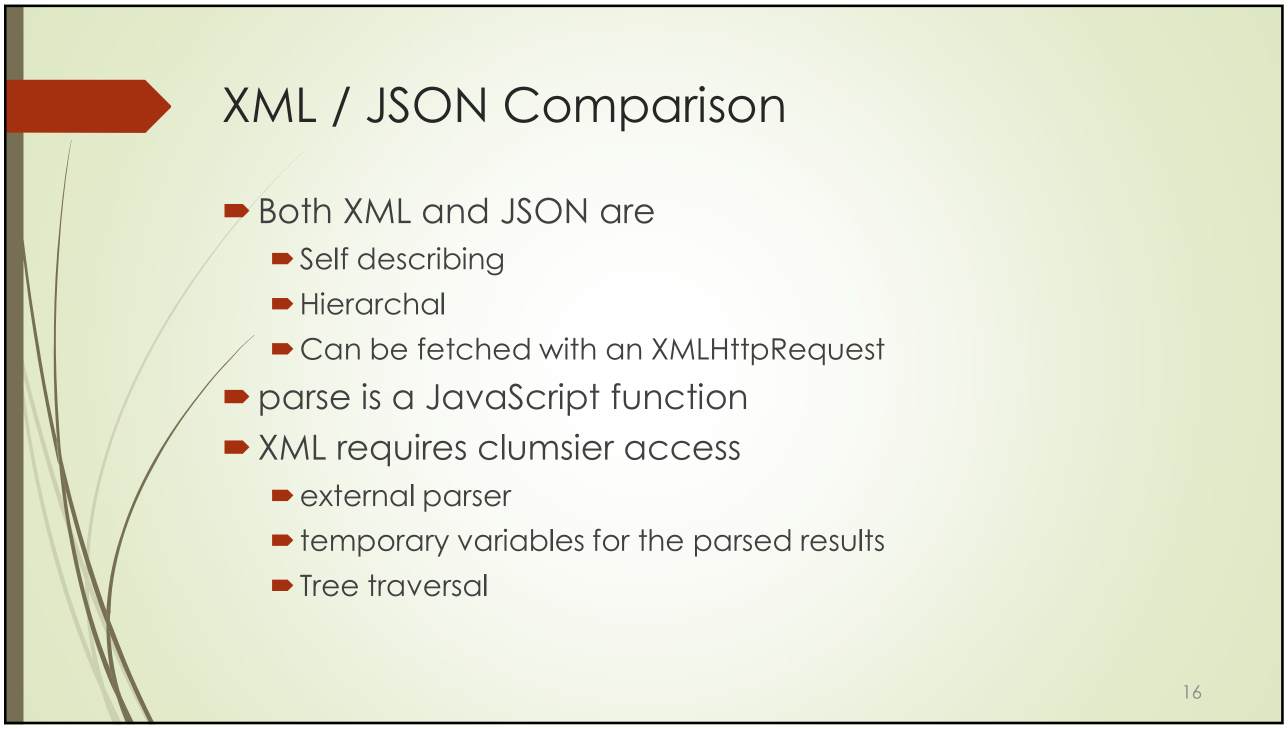
```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

The same text expressed as XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Example from json.org

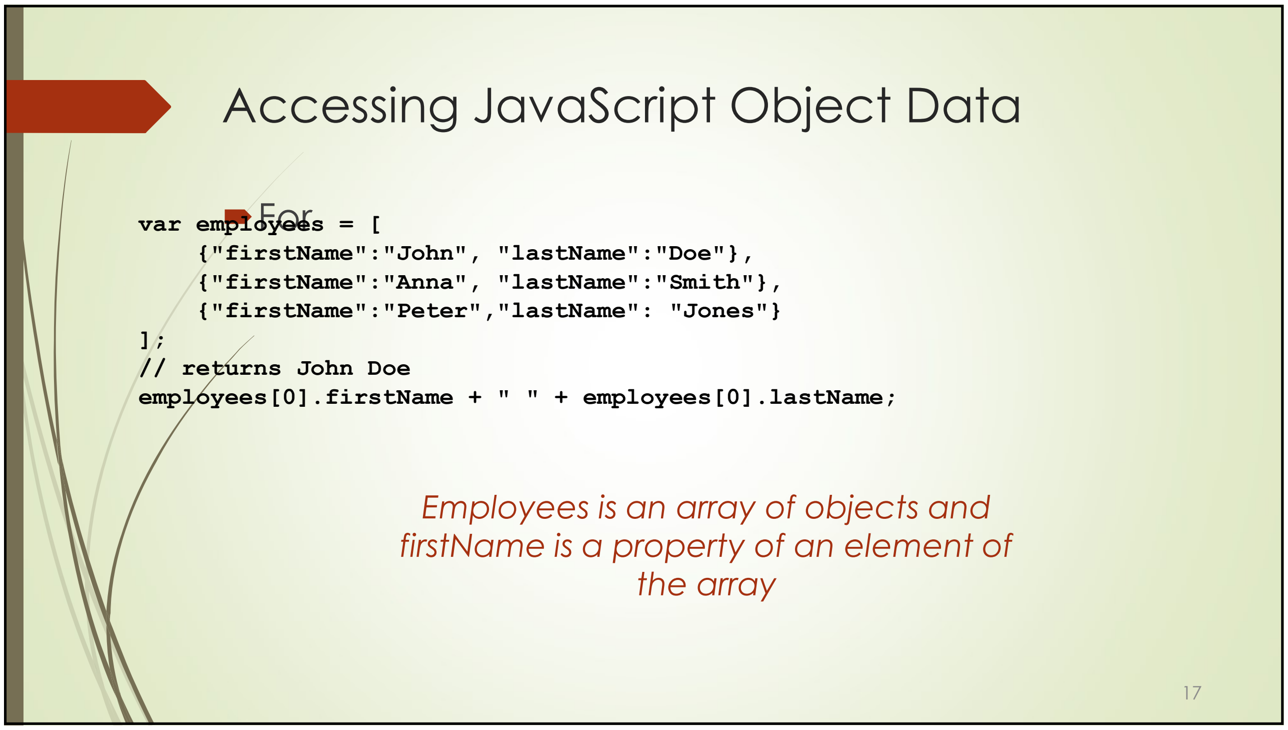
15



XML / JSON Comparison

- Both XML and JSON are
  - Self describing
  - Hierarchal
  - Can be fetched with an XMLHttpRequest
- parse is a JavaScript function
- XML requires clumsier access
  - external parser
  - temporary variables for the parsed results
  - Tree traversal

16



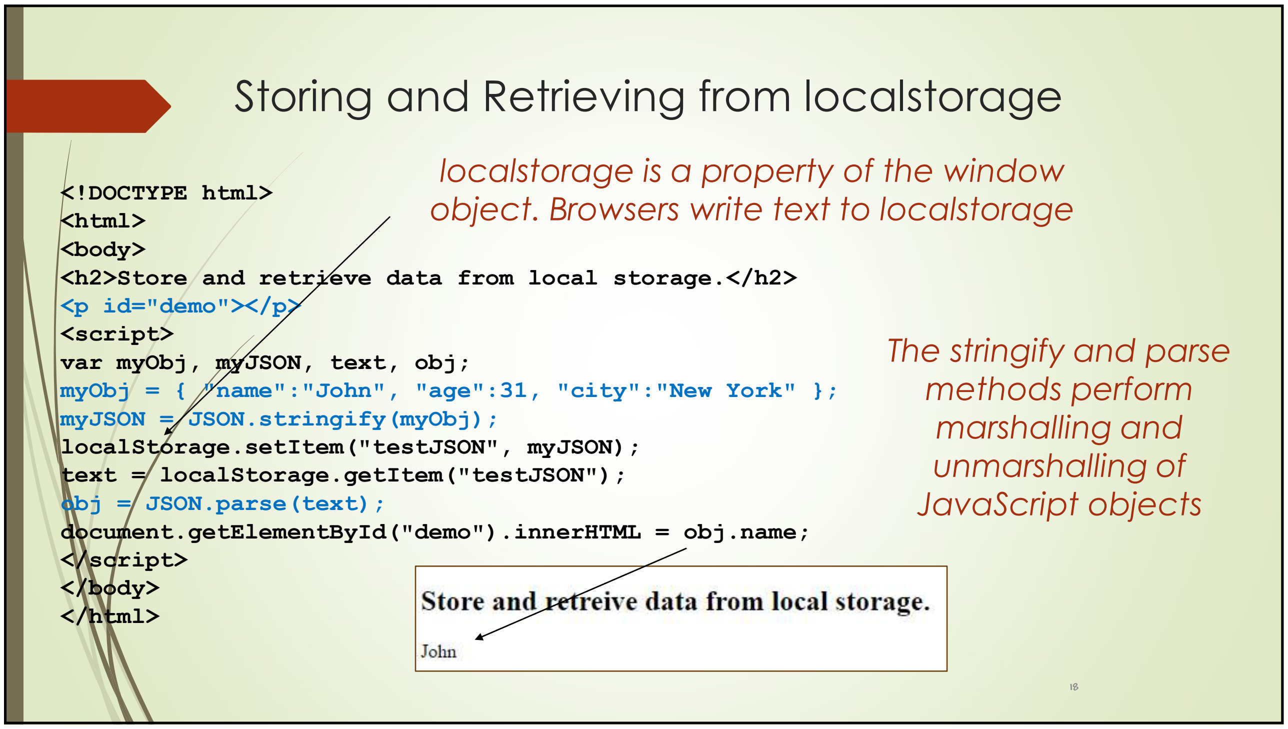
Accessing JavaScript Object Data

For

```
var employees = [
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName": "Jones"}
];
// returns John Doe
employees[0].firstName + " " + employees[0].lastName;
```

Employees is an array of objects and firstName is a property of an element of the array

17



Storing and Retrieving from localstorage

localstorage is a property of the window object. Browsers write text to localstorage

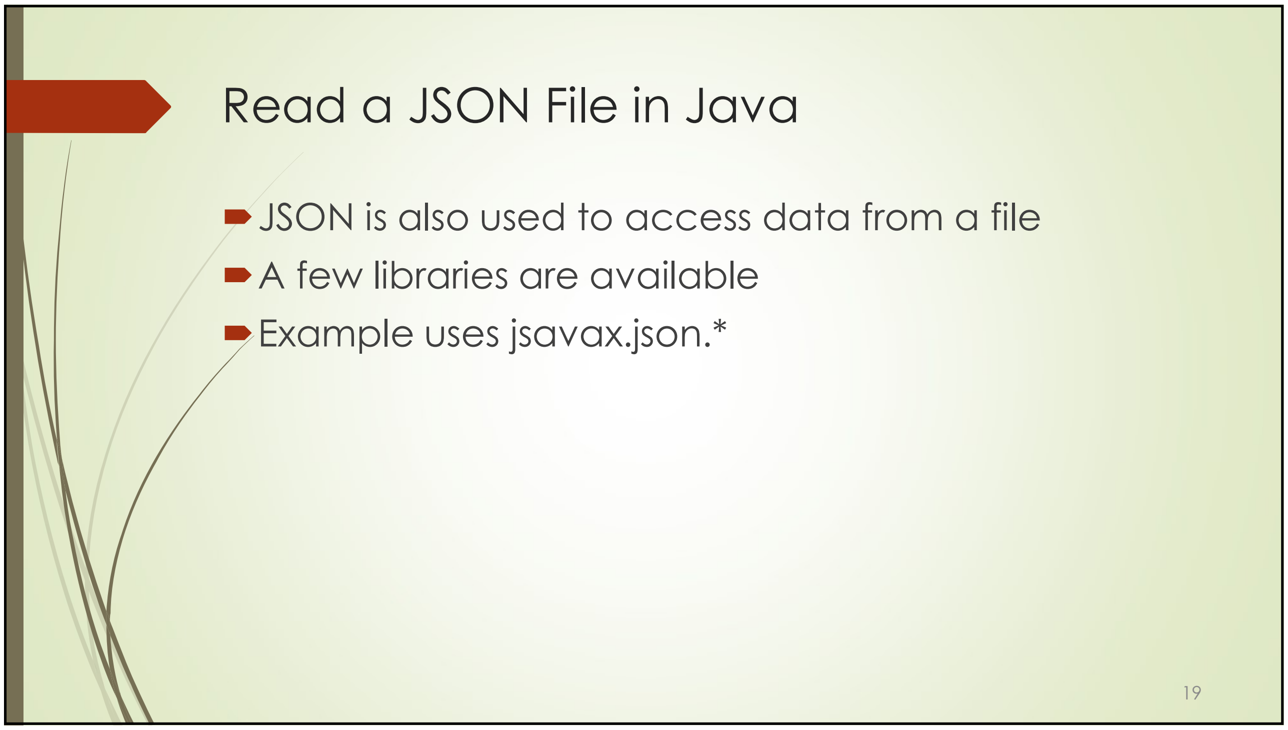
```
<!DOCTYPE html>
<html>
<body>
<h2>Store and retrieve data from local storage.</h2>
<p id="demo"></p>
<script>
var myObj, myJSON, text, obj;
myObj = { "name":"John", "age":31, "city":"New York" };
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
</script>
</body>
</html>
```

The stringify and parse methods perform marshalling and unmarshalling of JavaScript objects

```
Store and retreive data from local storage.

John
```

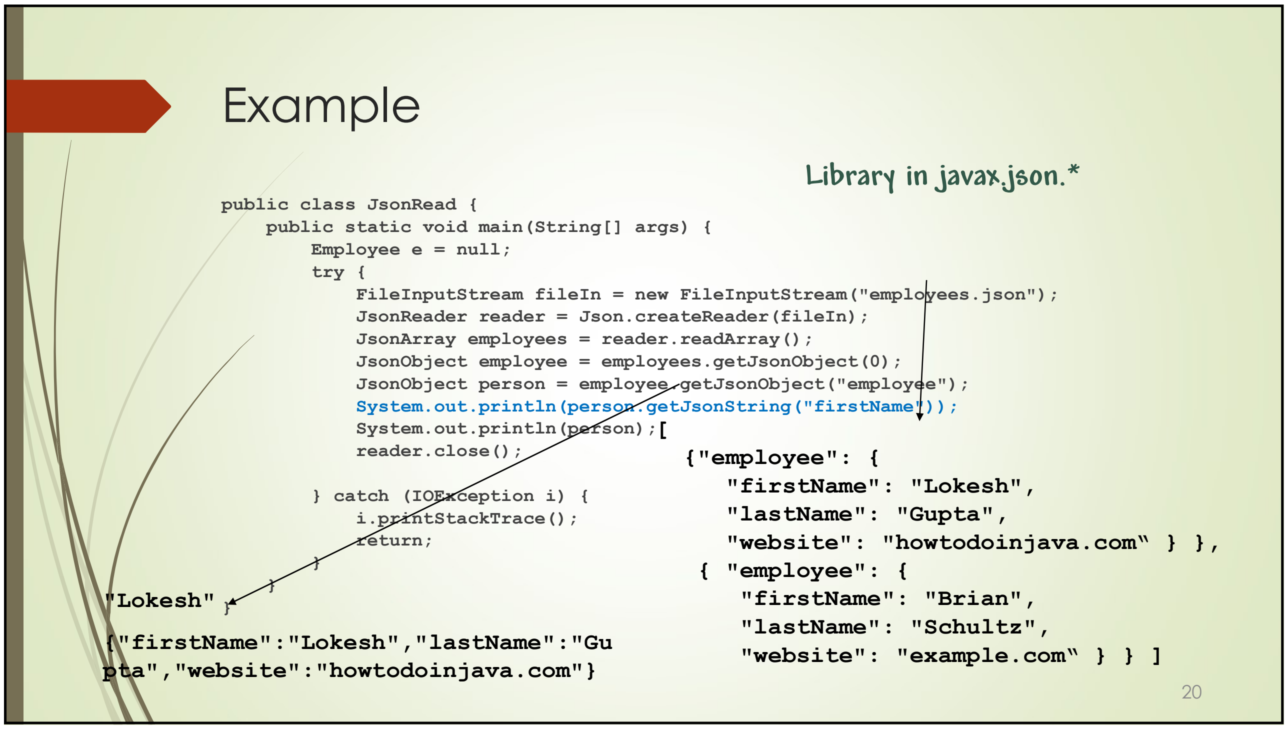
18



Read a JSON File in Java

- JSON is also used to access data from a file
- A few libraries are available
- Example uses jsavax.json.*

19



Example

Library in javax.json.*

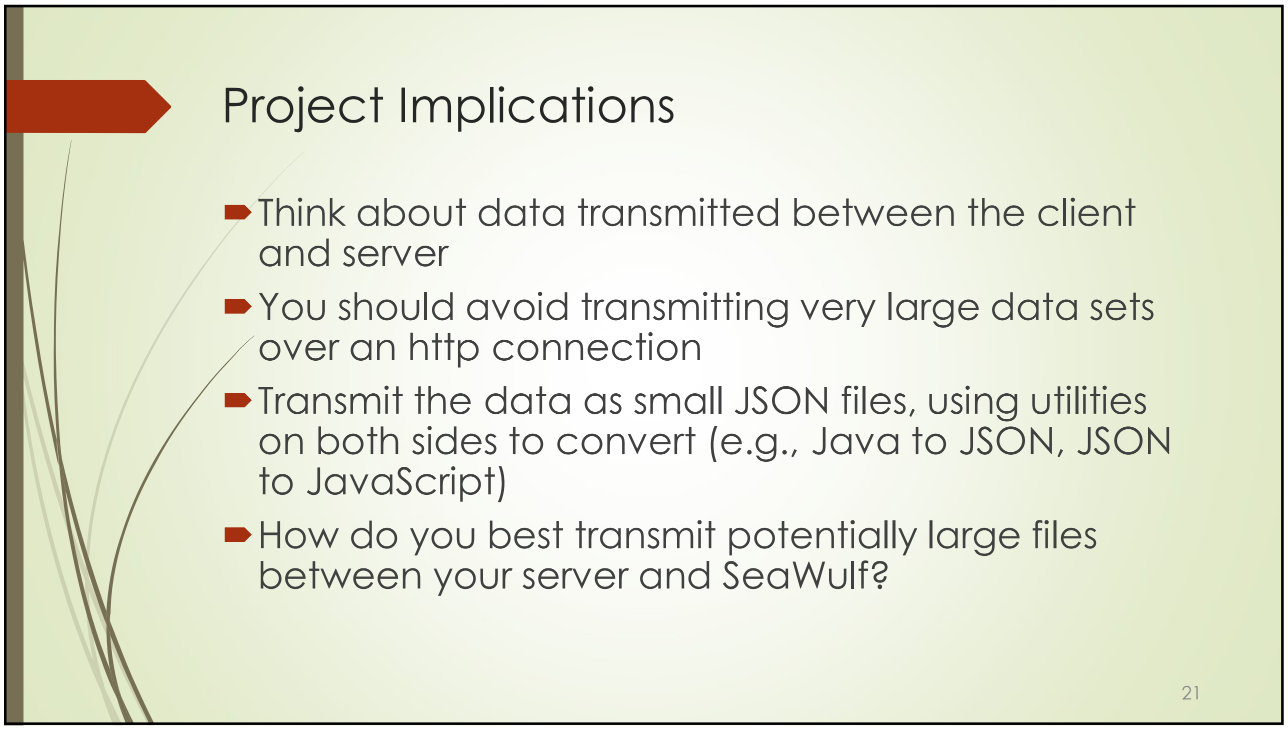
```
public class JsonRead {
    public static void main(String[] args) {
        Employee e = null;
        try {
            FileInputStream fileIn = new FileInputStream("employees.json");
            JsonReader reader = Json.createReader(fileIn);
            JsonArray employees = reader.readArray();
            JsonObject employee = employees.getJsonObject(0);
            JsonObject person = employee.getJsonObject("employee");
            System.out.println(person.getJsonString("firstName"));
            System.out.println(person);
            reader.close();

        } catch (IOException i) {
            i.printStackTrace();
            return;
        }
    }
}
```

```
[
{"employee": {
   "firstName": "Lokesh",
   "lastName": "Gupta",
   "website": "howtodoinjava.com" } },
 { "employee": {
    "firstName": "Brian",
    "lastName": "Schultz",
    "website": "example.com" } } ]
```

```
"Lokesh"
{"firstName":"Lokesh","lastName":"Gu
pta","website":"howtodoinjava.com"}
```

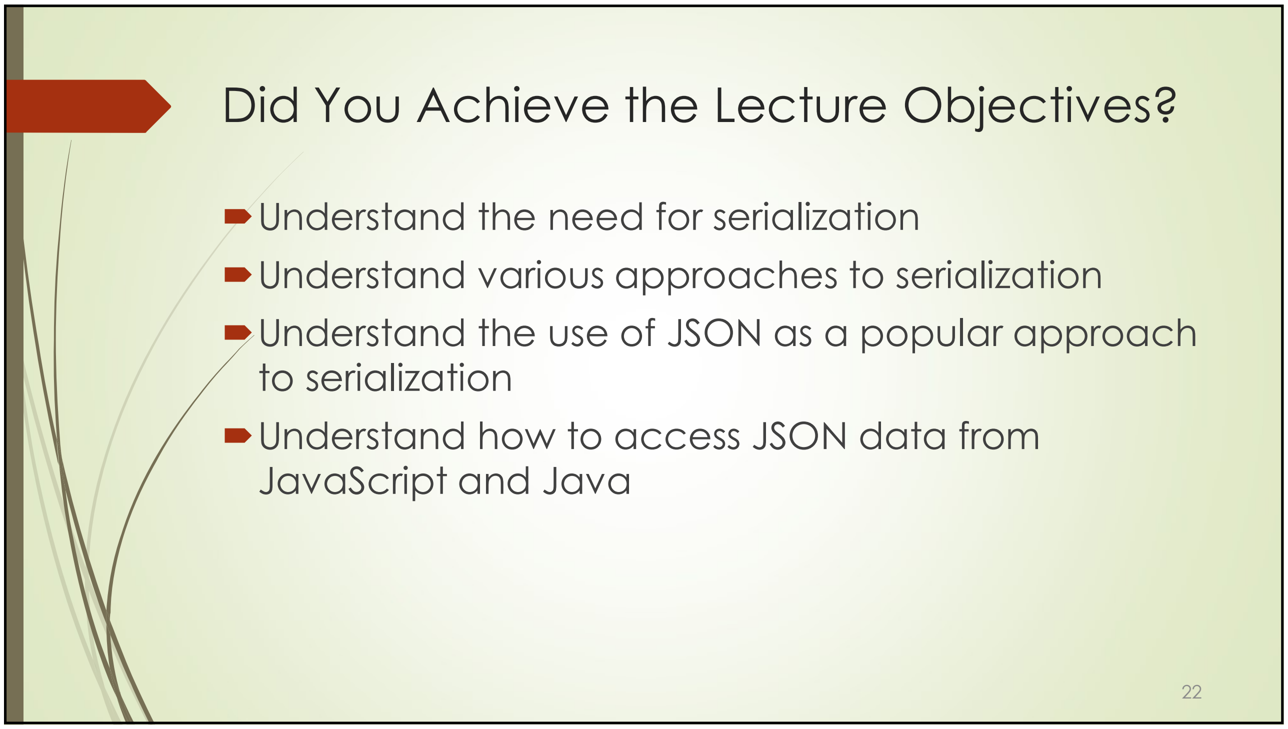
20



Project Implications

- Think about data transmitted between the client and server
- You should avoid transmitting very large data sets over an http connection
- Transmit the data as small JSON files, using utilities on both sides to convert (e.g., Java to JSON, JSON to JavaScript)
- How do you best transmit potentially large files between your server and SeaWulf?

21



Did You Achieve the Lecture Objectives?

- Understand the need for serialization
- Understand various approaches to serialization
- Understand the use of JSON as a popular approach to serialization
- Understand how to access JSON data from JavaScript and Java

22