Univerzitet Singidunum

Web programming

www.singidunum.ac.rs

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format.
- It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999.
- JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.
- These properties make JSON an ideal data-interchange language.

JSON

- JSON is built on two structures:
- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

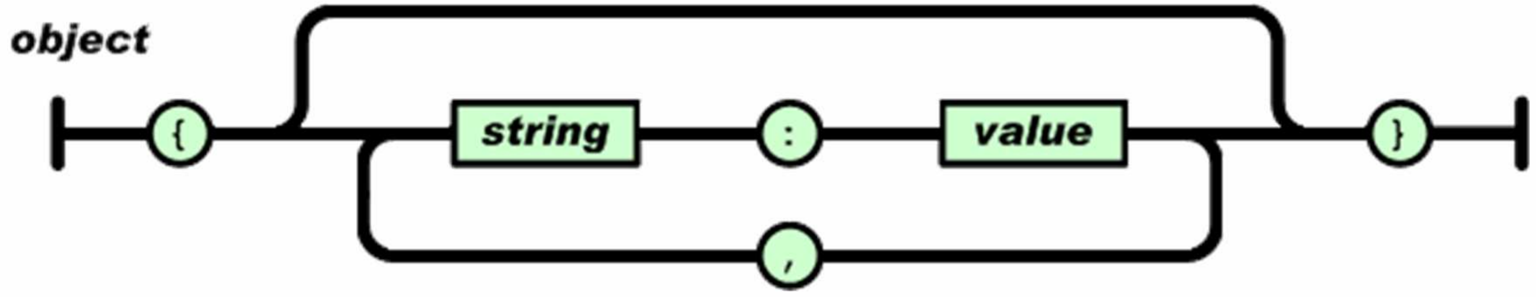
JSON

- These are universal data structures. Virtually all modern programming languages support them in one form or another.
- It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

- In JSON, they take on the following forms:

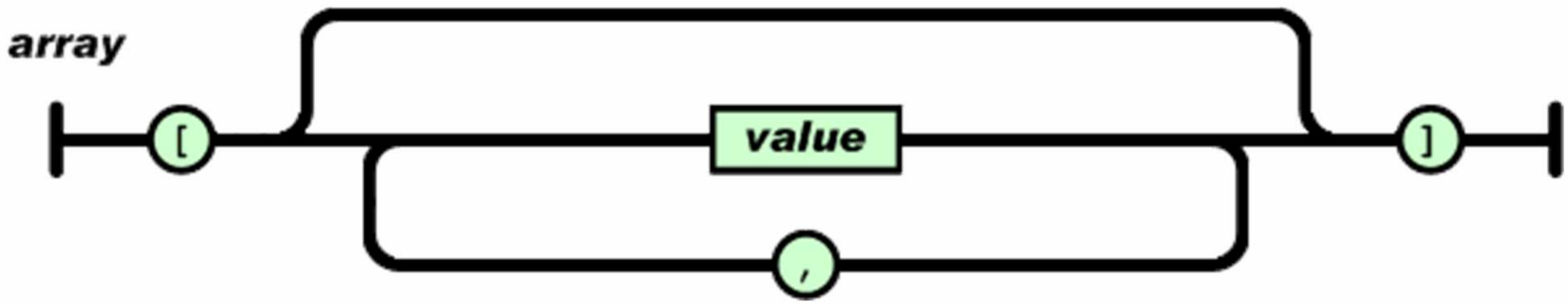
JSON

- An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



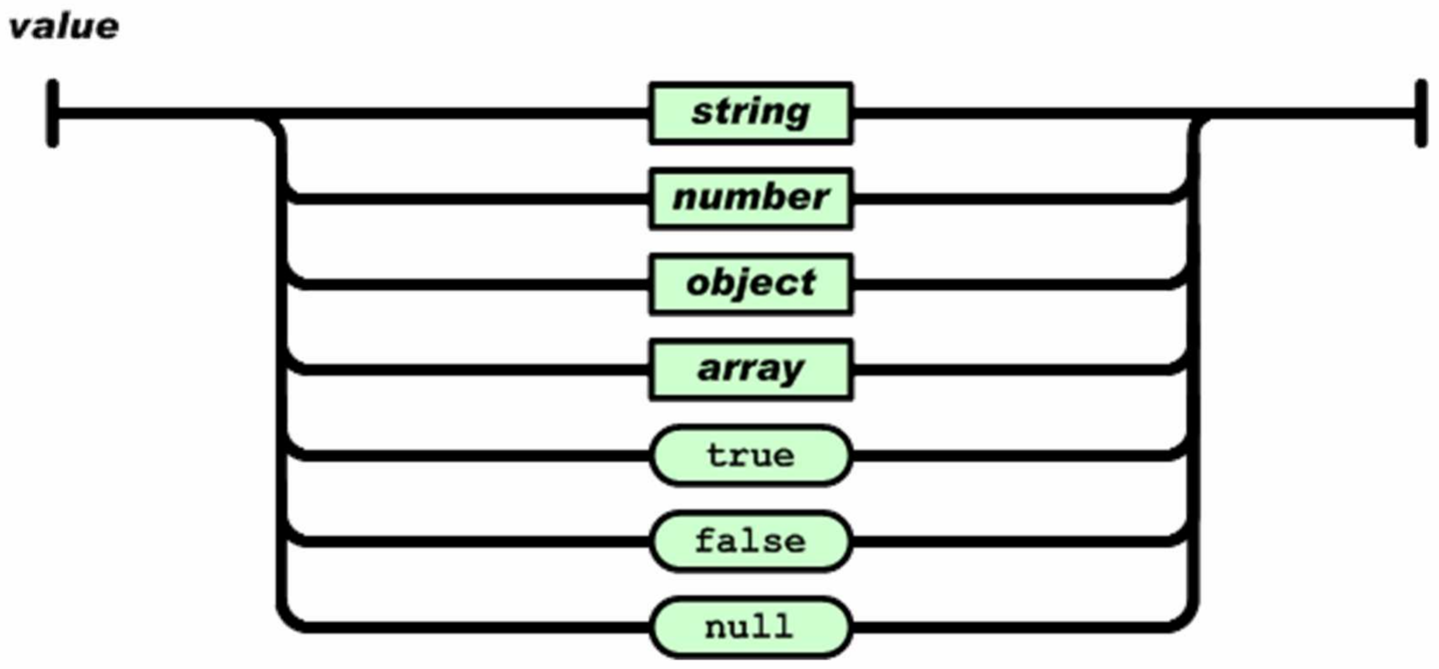
JSON

- An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).



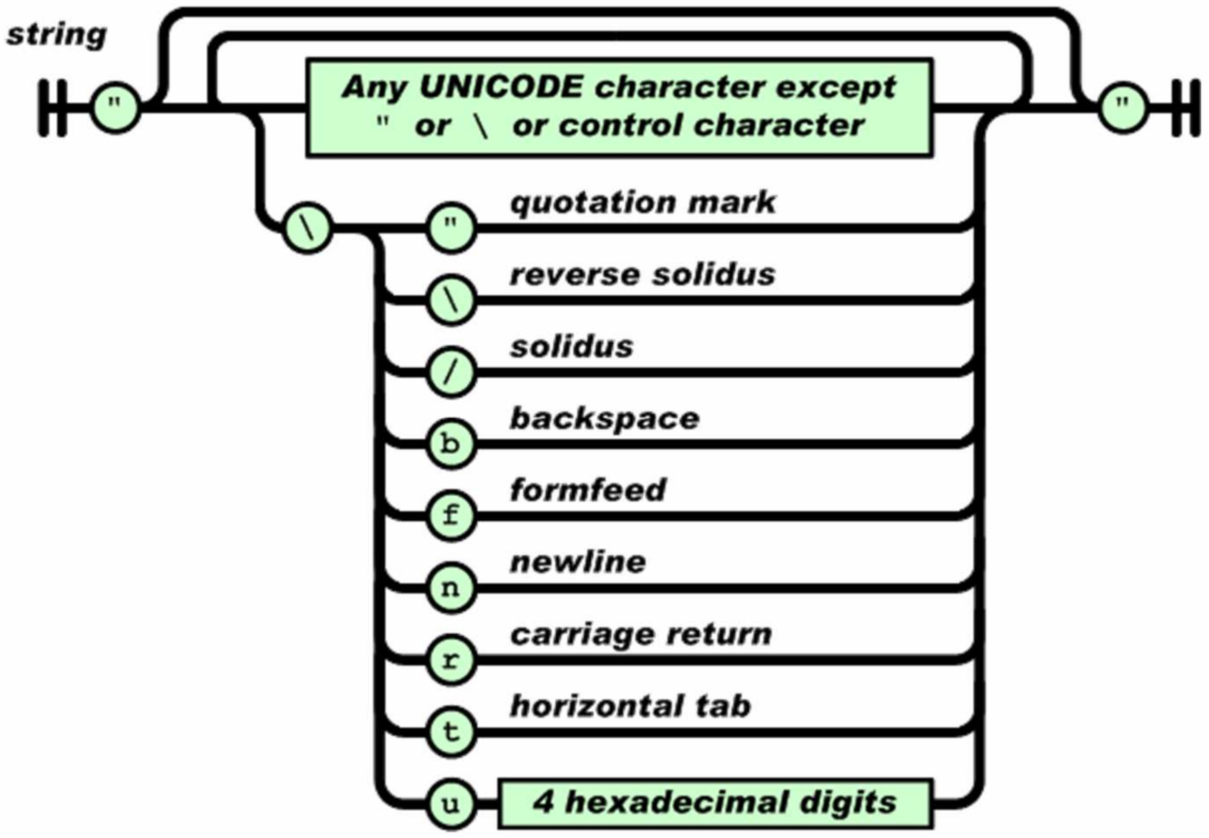
JSON

- A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.



JSON

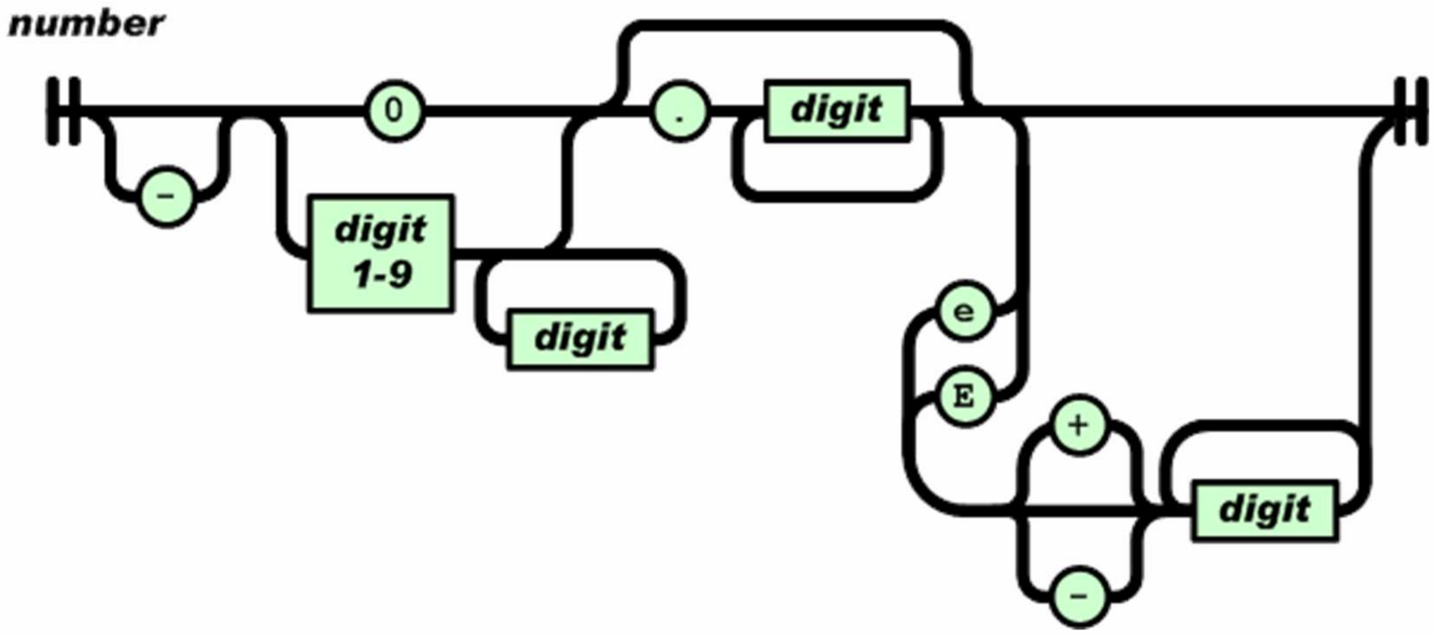
- A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.



Univerzitet Singidunum

JSON

- A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used.



JSON

- Whitespace can be inserted between any pair of tokens.
- Excepting a few encoding details, that completely describes the language.
- Defined on: https://www.json.org/
- JSON Files:
  - The file type for JSON files is ".json"
  - The MIME type for JSON text is "application/json

JSON

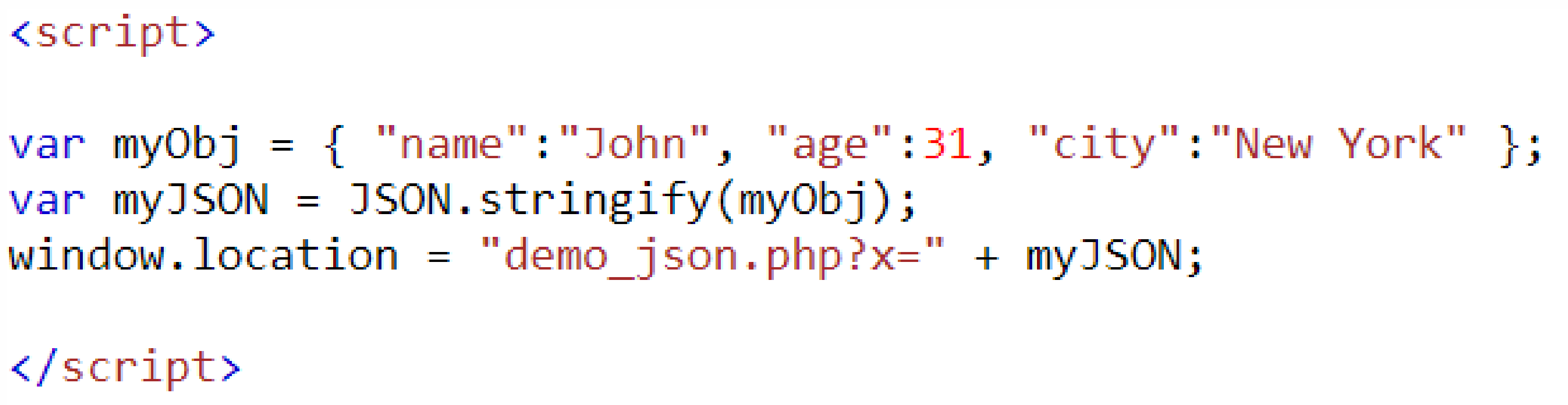
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
- Main usage – exchanging data between systems

JSON

- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

JSON

- Sending Data
- If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:



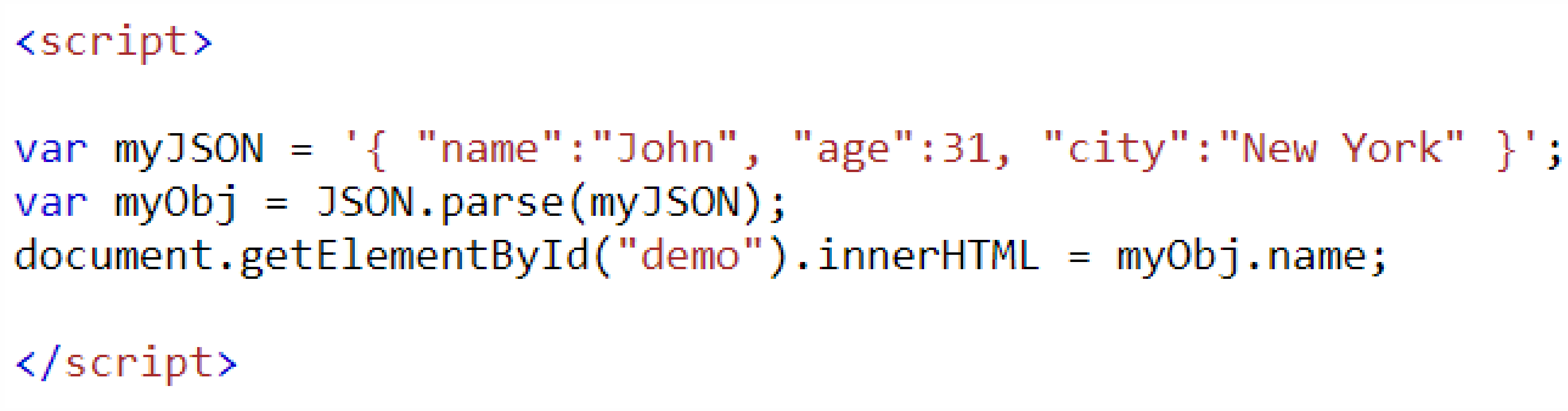
```
<script>

var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;

</script>
```

JSON

- Receiving Data
- If you receive data in JSON format, you can convert it into a JavaScript object:



```
<script>

var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;

</script>
```

JSON

- The JSON syntax is a subset of the JavaScript syntax.
- JSON syntax is derived from JavaScript object notation syntax:

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

JSON

- JSON data is written as name/value pairs.
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

  "name":"John“
- JSON names require double quotes. JavaScript names don't.

Univerzitet Singidunum

JSON

- JSON - Evaluates to JavaScript Objects
- The JSON format is almost identical to JavaScript objects.
- In JSON, keys must be strings, written with double quotes:

```
{ "name":"John" }
```

- In JavaScript:

```
{ name:"John" }
```

JSON

- In JSON, values must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null
- In JavaScript values can be all of the above, plus any other valid JavaScript expression, including:
  - a function
  - a date
  - undefined

JSON

- JSON Uses JavaScript Syntax
- Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.

JSON

- With JavaScript you can create an object and assign data to it, like this:

```
var person = { "name":"John", "age":31, "city":"New York" };
```

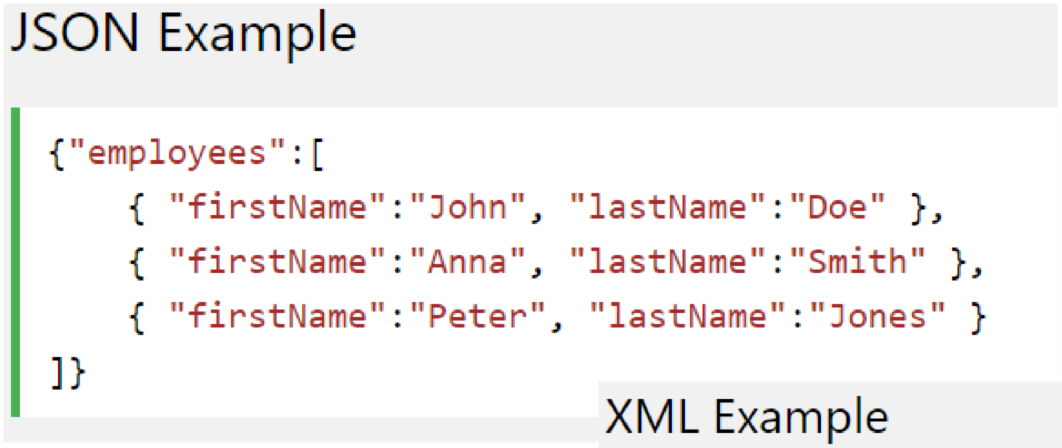
- You can access a JavaScript object like this:

```
// returns John
person.name;
```

JSON

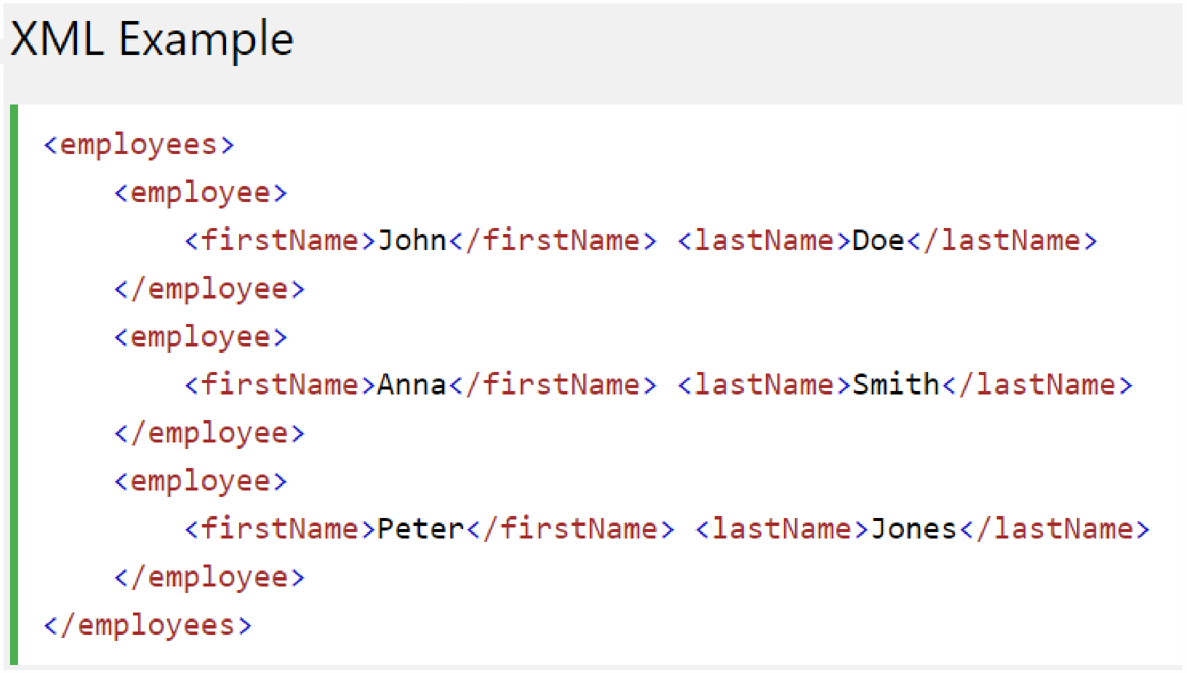
- JSON vs XML
- Both JSON and XML can be used to receive data from a web server.
- The following JSON and XML examples both defines an employees object, with an array of 3 employees

JSON



JSON Example

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```



XML Example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON

- JSON is Like XML Because
  - Both JSON and XML are "self describing" (human readable)
  - Both JSON and XML are hierarchical (values within values)
  - Both JSON and XML can be parsed and used by lots of programming languages
  - Both JSON and XML can be fetched with an XMLHttpRequest

Univerzitet Singidunum

JSON

- JSON is Unlike XML Because
  - JSON doesn't use end tag
  - JSON is shorter
  - JSON is quicker to read and write
  - JSON can use arrays
  - The biggest difference is:
- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

JSON

- Why JSON is Better Than XML?
- XML is much more difficult to parse than JSON.
- JSON is parsed into a ready-to-use JavaScript object.

JSON

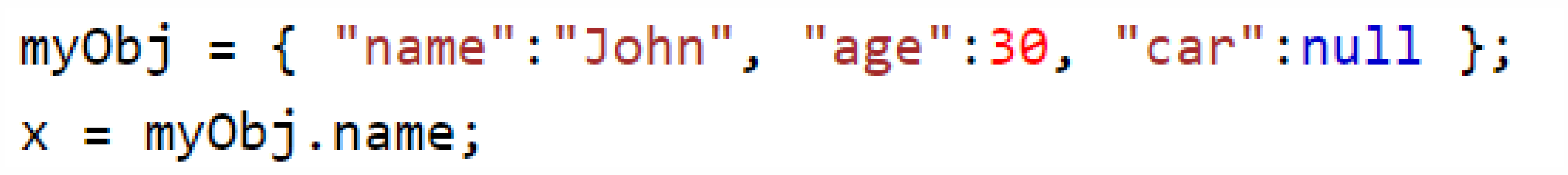
- For AJAX applications, JSON is faster and easier than XML:
- Using XML
  - Fetch an XML document
  - Use the XML DOM to loop through the document
  - Extract values and store in variables
- Using JSON
  - Fetch a JSON string
  - JSON.Parse the JSON string

JSON Objects

- Object Syntax
- { "name":"John", "age":30, "car":null }
- JSON objects are surrounded by curly braces {}.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.

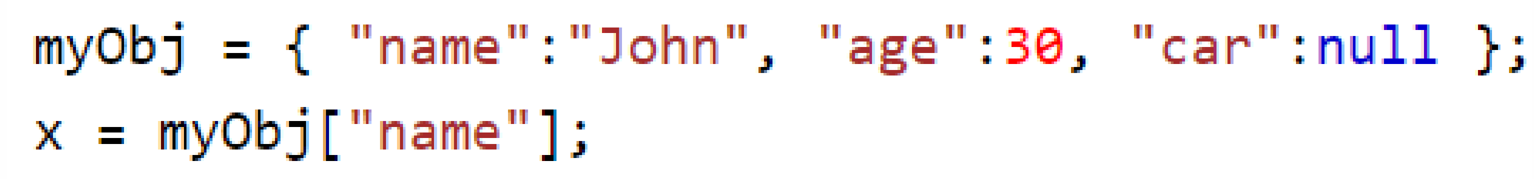
JSON Objects

- Accessing Object Values
- You can access the object values by using dot (.) notation:



```
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;
```

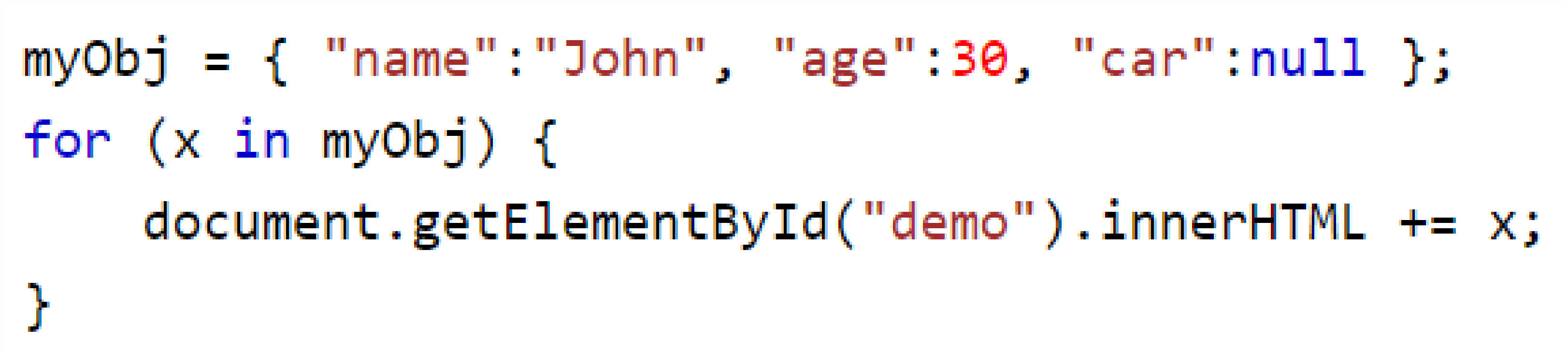
- You can also access the object values by using bracket ([]) notation:



```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

JSON Objects

- Looping an Object
- You can loop through object properties by using the for-in loop:



```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
    document.getElementById("demo").innerHTML += x;
}
```

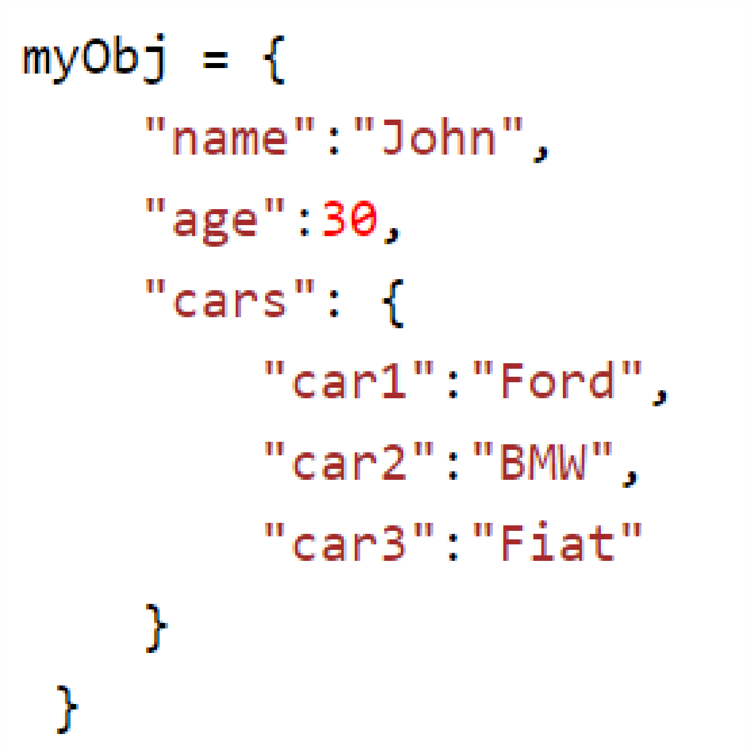
JSON Objects

- In a for-in loop, use the bracket notation to access the property values:

```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
    document.getElementById("demo").innerHTML += myObj[x];
}
```

JSON Objects

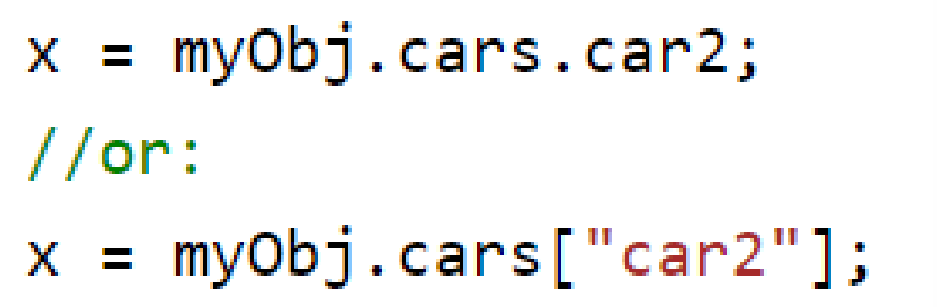
- Nested JSON Objects
- Values in a JSON object can be another JSON object.



```
myObj = {
    "name":"John",
    "age":30,
    "cars": {
        "car1":"Ford",
        "car2":"BMW",
        "car3":"Fiat"
    }
 }
```

JSON Objects

- You can access nested JSON objects by using the dot notation or bracket notation:



```
x = myObj.cars.car2;
//or:
x = myObj.cars["car2"];
```

JSON Objects

- Modify Values
- You can use the dot notation to modify any value in a JSON object:

```
myObj.cars.car2 = "Mercedes";
```

- You can also use the bracket notation to modify a value in a JSON object:

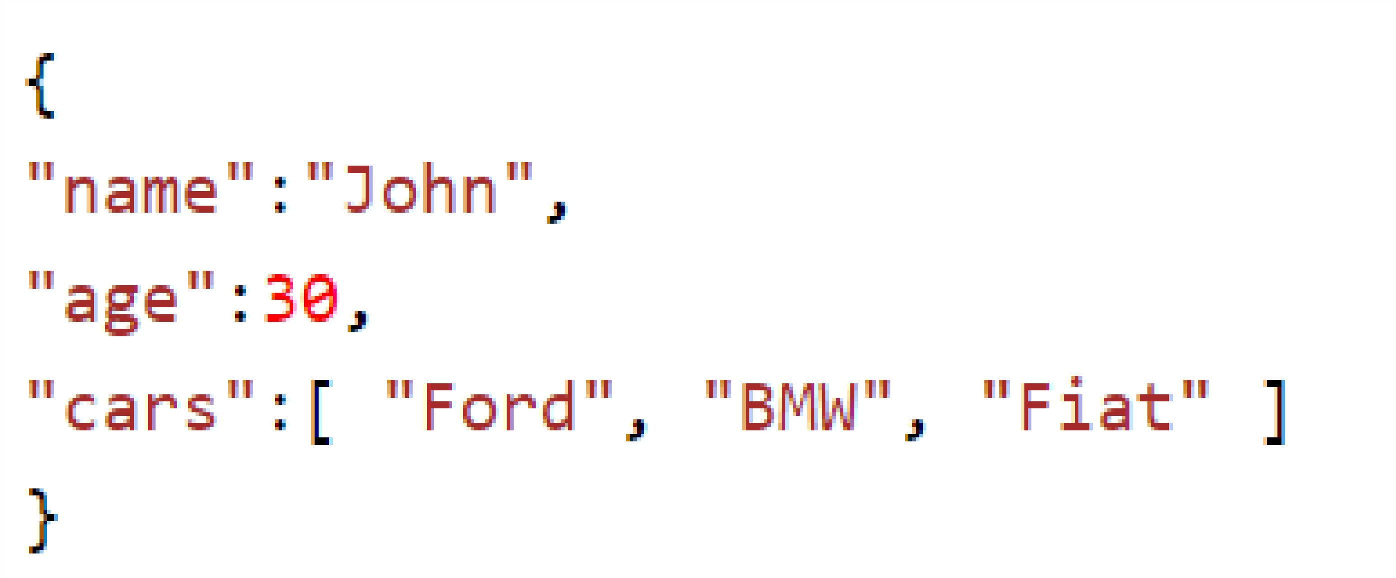
```
myObj.cars["car2"] = "Mercedes";
```

JSON Arrays

- Arrays as JSON Objects
- ["Ford", "BMW", "Fiat"]
- Arrays in JSON are almost the same as arrays in JavaScript.
- In JSON, array values must be of type string, number, object, array, boolean or null.
- In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and undefined.

JSON Arrays

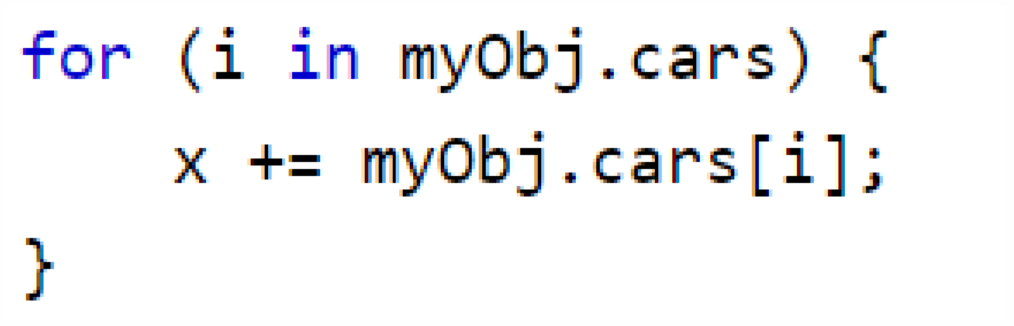
- Arrays can be values of an object property:



```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

JSON Arrays

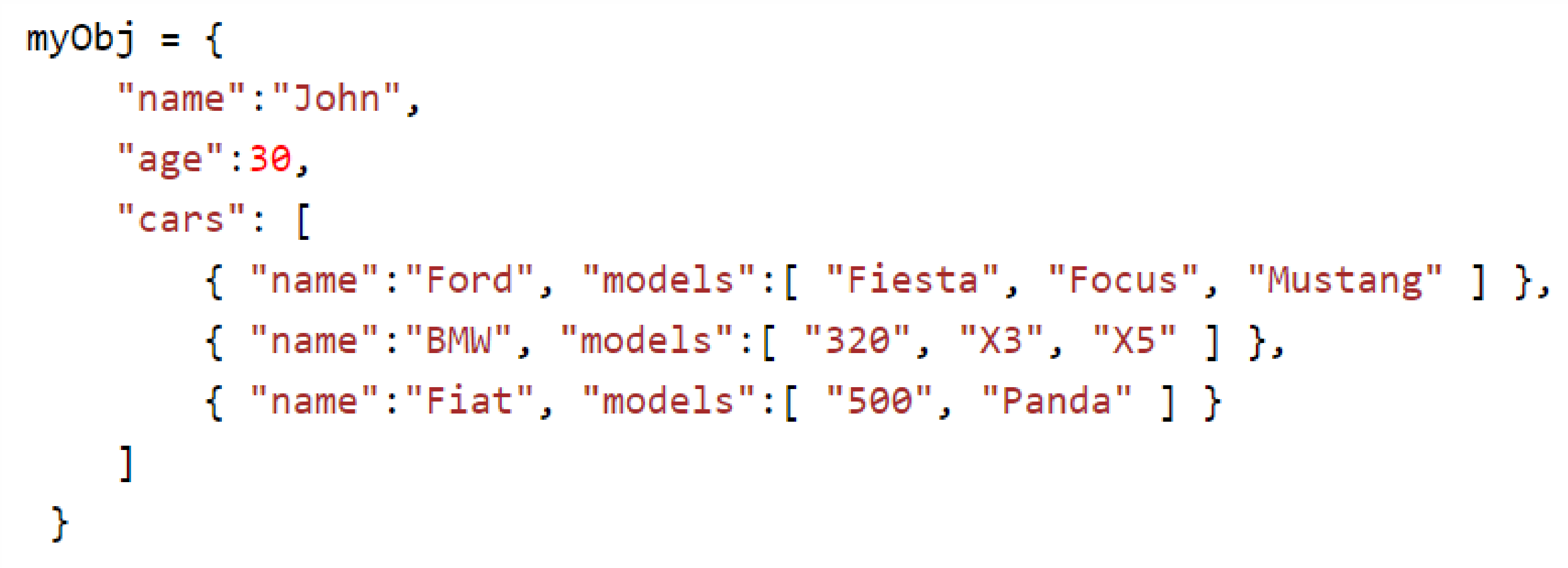
- Looping Through an Array
- You can access array values by using a for-in loop:



```
for (i in myObj.cars) {
    x += myObj.cars[i];
}
```

JSON Arrays

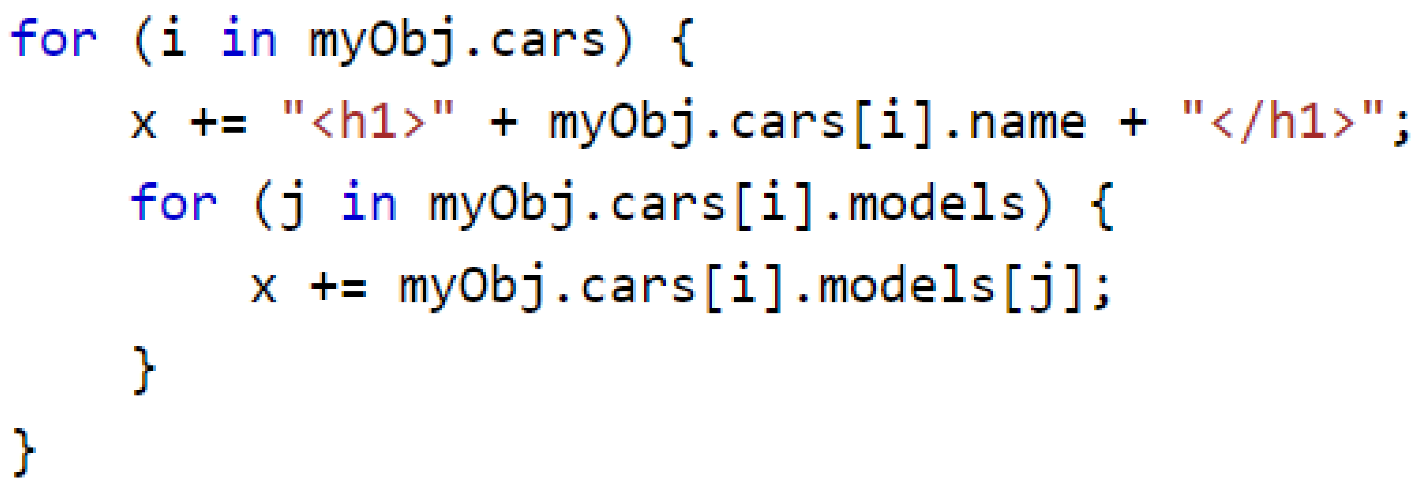
- Nested Arrays in JSON Objects
- Values in an array can also be another array, or even another JSON object:



```
myObj = {
    "name":"John",
    "age":30,
    "cars": [
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },
        { "name":"Fiat", "models":[ "500", "Panda" ] }
    ]
 }
```

JSON Arrays

- To access arrays inside arrays, use a for-in loop for each array:



```
for (i in myObj.cars) {
    x += "<h1>" + myObj.cars[i].name + "</h1>";
    for (j in myObj.cars[i].models) {
        x += myObj.cars[i].models[j];
    }
}
```

JSON.parse()

- A common use of JSON is to exchange data to/from a web server.
- When receiving data from a web server, the data is always a string.
- Parse the data with JSON.parse(), and the data becomes a JavaScript object.

JSON.parse()

- Imagine we received this text from a web server:

```
'{ "name":"John", "age":30, "city":"New York"}'
```

- Use the JavaScript function JSON.parse() to convert text into a JavaScript object:

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

- Make sure the text is written in JSON format, or else you will get a syntax error.

JSON.parse()

- You can request JSON from the server by using an AJAX request
- As long as the response from the server is written in JSON format, you can parse the string into a JavaScript object.

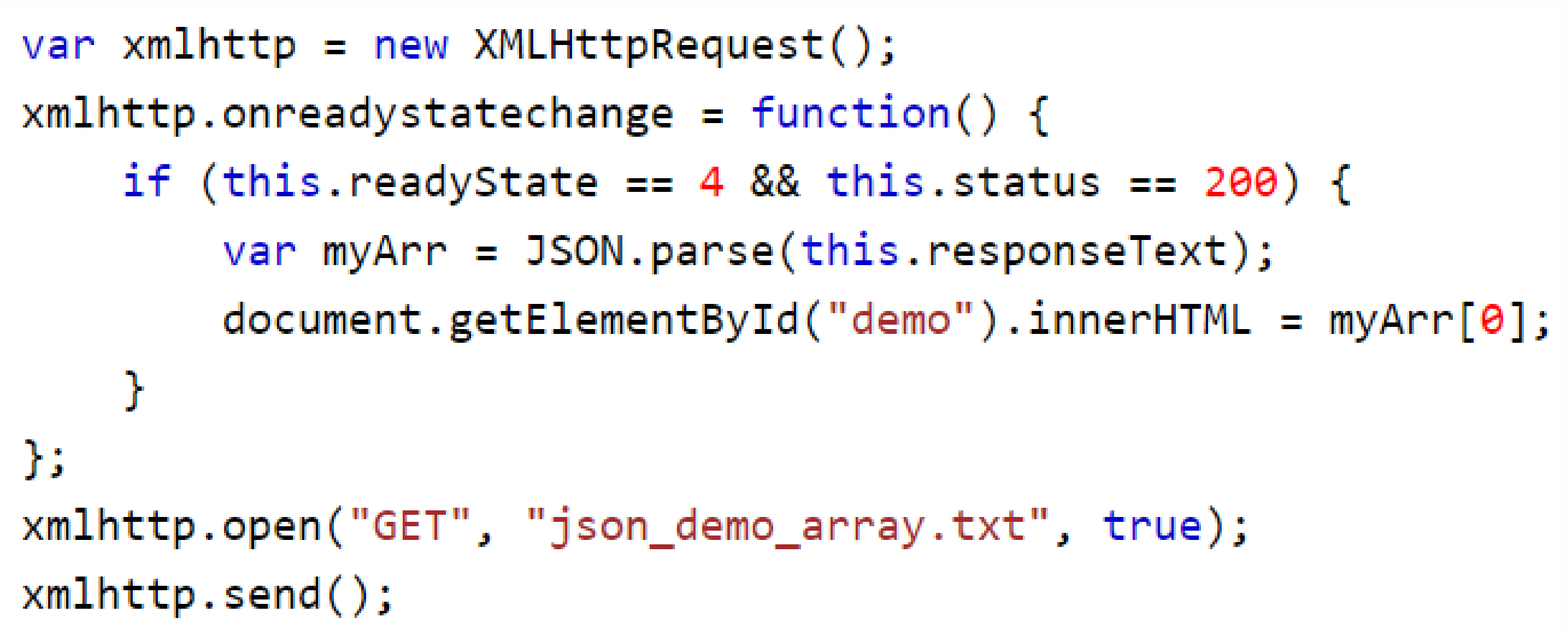
JSON.parse()

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

JSON.parse()

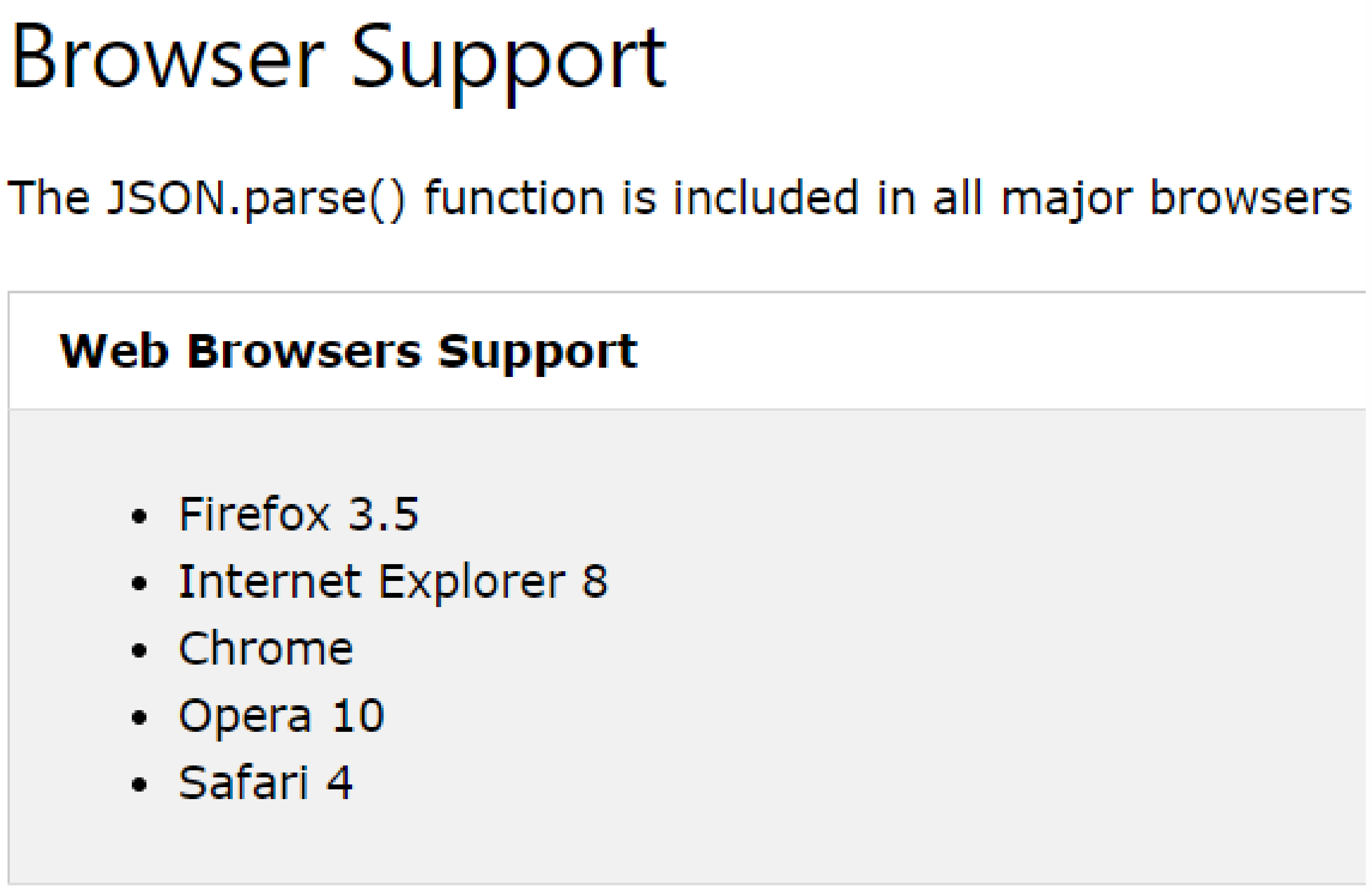
- Array as JSON
- When using the JSON.parse() on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

JSON.parse()



```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myArr[0];
    }
};
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

JSON.parse()



Browser Support

The JSON.parse() function is included in all major browsers

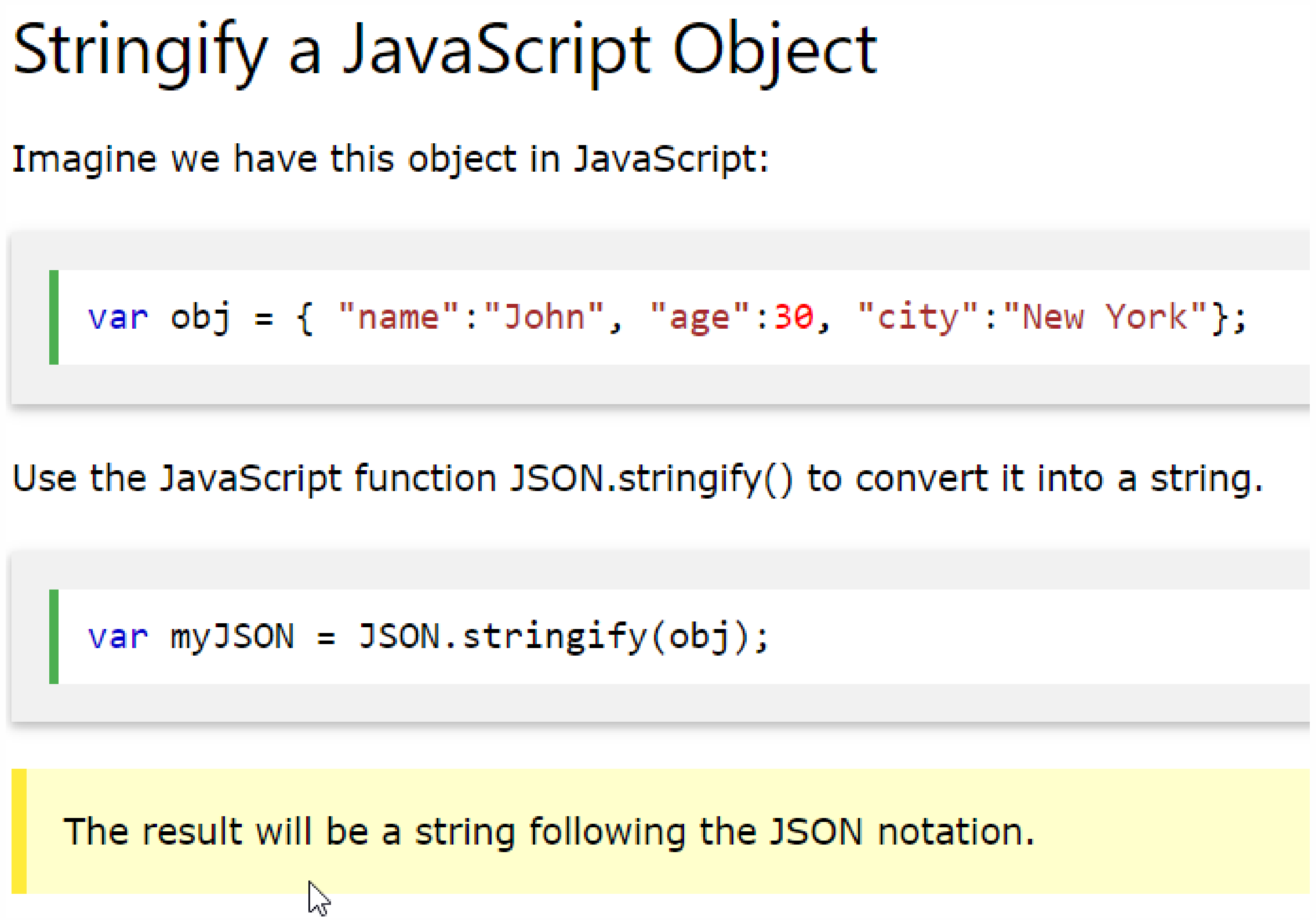
Web Browsers Support

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

JSON.stringify()

- A common use of JSON is to exchange data to/from a web server.
- When sending data to a web server, the data has to be a string.
- Convert a JavaScript object into a string with JSON.stringify().

JSON.stringify()



Stringify a JavaScript Object

Imagine we have this object in JavaScript:

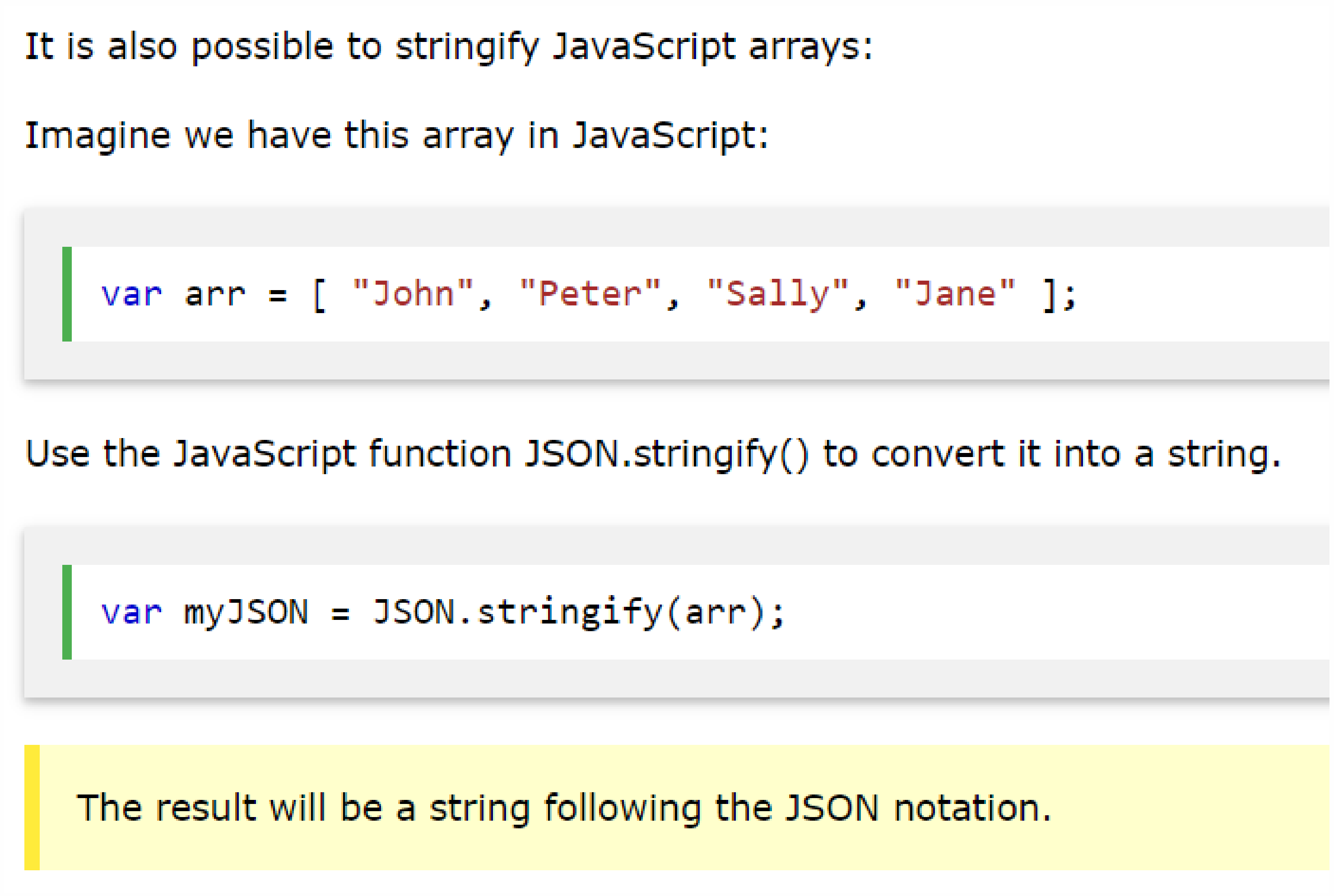
```
var obj = { "name":"John", "age":30, "city":"New York"};
```

Use the JavaScript function JSON.stringify() to convert it into a string.

```
var myJSON = JSON.stringify(obj);
```

The result will be a string following the JSON notation.

JSON.stringify()



It is also possible to stringify JavaScript arrays:

Imagine we have this array in JavaScript:

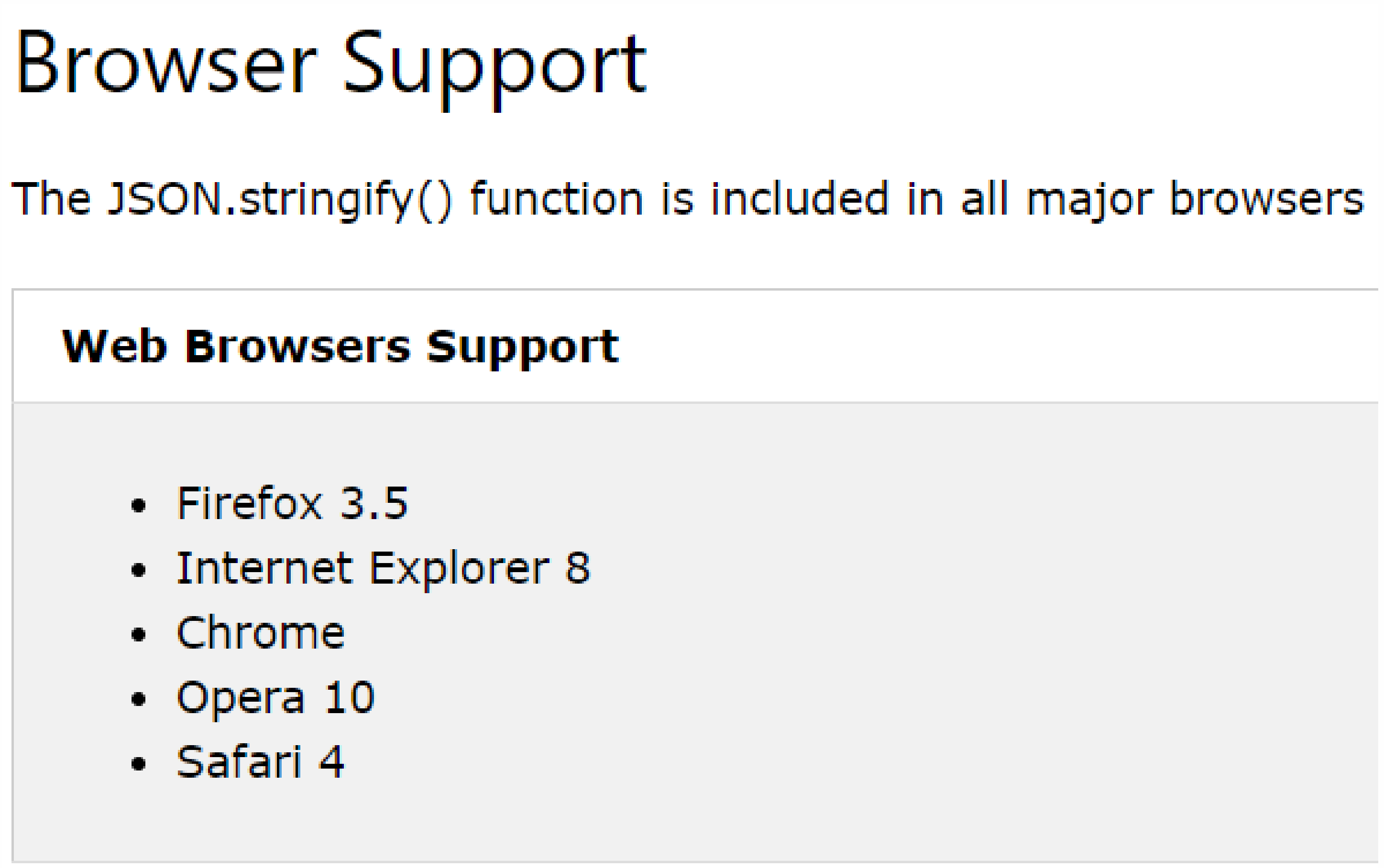
```
var arr = [ "John", "Peter", "Sally", "Jane" ];
```

Use the JavaScript function JSON.stringify() to convert it into a string.

```
var myJSON = JSON.stringify(arr);
```

The result will be a string following the JSON notation.

JSON.stringify()



Browser Support

The JSON.stringify() function is included in all major browsers

Web Browsers Support

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

JSON

- JSON can very easily be translated into JavaScript.
- JavaScript can be used to make HTML in your web pages.
- Make an HTML table with data received as JSON:

Univerzitet Singidunum

JSON

```
obj = { "table":"customers", "limit":20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        myObj = JSON.parse(this.responseText);
        txt += "<table border='1'>"
        for (x in myObj) {
            txt += "<tr><td>" + myObj[x].name + "</td></tr>";
        }
        txt += "</table>"
        document.getElementById("demo").innerHTML = txt;
    }
}
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

JSON

- Make an HTML drop down list with data received as JSON:

```
obj = { "table":"customers", "limit":20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        myObj = JSON.parse(this.responseText);
        txt += "<select>"
        for (x in myObj) {
            txt += "<option>" + myObj[x].name;
        }
        txt += "</select>"
        document.getElementById("demo").innerHTML = txt;
    }
}
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```