Web Programming

CS-406

Lecture # 05

JavaScript

Today’s Lecture

- JS Objects
- ES6
- Classes
- JSON
- Jquery
- Common Mistakes
- Performance

JavaScript Objects

- JavaScript variables are containers for data values.
- var car = "Fiat";

- Objects are variables too. But objects can contain many values.
- var car = {type:"Fiat", model:"500", color:"white"};

- The values are written as name:value pairs (name and value separated by a colon).
- JavaScript objects are containers for named values called properties or methods.

Properties and Methods

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

JS Versions

- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMAScript is the official name of the language.
- ECMAScript versions have been abbreviated to ES1, ES2, ES3, ES5, and ES6.
- Since 2016 new versions are named by year (ECMAScript 2016 / 2017 / 2018).

ES-6

- ECMAScript 6 was the second major revision to JavaScript.
- ECMAScript 6 is also known as ES6 and ECMAScript 2015.
- Introduced some new features and concept of Classes

New Features in ES6:

- The **let** keyword //variable with block scope
- The const keyword // block scope + constant value
- JavaScript Arrow Functions // =>
- JavaScript For/of
- JavaScript Classes
- JavaScript Promises
- JavaScript Symbol
- Default Parameters
- Function Rest Parameter
- Array.find()
- Array.findIndex()
- New Math Methods
- New Number Properties
- New Number Methods
- New Global Methods
- JavaScript Modules

Arrow Function

- Arrow functions allows a short syntax for writing function expressions.
- You don't need the function keyword, the return keyword, and the curly brackets.

- ```
  var x = function(x, y) {
     return x * y;
  }

  // ES6
  const x = (x, y) => x * y;
  ```
  ```

The For/of loop

- The JavaScript for/of statement loops through the values of an iterable objects.
- for/of lets you loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more.

```
for (variable of iterable) {
  // code block to be executed
}
```

```
var cars = ["BMW", "Volvo", "Mini"];
var x;

for (x of cars) {
  document.write(x + "<br >");
}
```

JS Classes

- ECMAScript 2015, also known as ES6, introduced JavaScript Classes.
- JavaScript Classes are templates for JavaScript Objects.
- A JavaScript class is not an object.
- It is a template for JavaScript objects.

- class Car { // use of class keyword
- constructor(name, year) { //Constructor function is a must
- this.name = name;
- this.year = year;
- }
- }

JSON

- JSON stands for Javascript Object Notation
- It is a lightweight data-interchange format that provides a compact syntax for storing, processing and communicating structured data.
- It keeps data in name value pairs
- Its supported by all the browsers
- The functions are a lot simpler to process the data
- It is used primarily to transmit data between a server and web application, serving as an alternative to XML.

- 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.
- Web services and APIs use JSON format to provide public data over the network.

Structure

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.

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

{ "name":"John“ , “age”:20 }

Books.json

```
{
    “title” : “Object oriented programming”,
    “author” : “Robert Lafore”,
    “Year” : 2007
}
```

Exchanging Data

- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
- 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.

Sending Data

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

- Example

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

Receiving Data

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

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

Storing Data

- When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.
- JSON makes it possible to store JavaScript objects as text.
- Example
- Storing data in local storage

```
// Storing data:
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

JSON vs XML

- The previous dominant approach for storing and communicating data was XML
- Both JSON and XML are self describing
- Both are hierarchical
- Both can be processed with programming language
- Both are used to communicate data across different machines
- But, JSON is shorter, quicker and can use arrays!

Data types

The values can be,

- String
- Number
- Boolean
- Object
- Array
- null

JSON Objects

- { "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.

Accessing Object Values

- Object values can be accessed by using dot (.) notation or square brackets []:

- Example

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

Looping an object

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

- Example

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

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

- Example

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

Hierarchical Structure

Can have nested objects

Books.json

```
{
    “title” : “Object-oriented programming”,
    “author” : {
    “firstName” : “Robert”,
    “secondName” : “Lafore”
    }
    “year” : 2007
}
```

Arrays

Books.json

```
{
    “title” : “Object-oriented Programming”,
    “authors” : “Robert Lafore”,
    “year” : 2007,
    “chapters” : [“Introduction”, “Classes”, “Inheritance”]
}
```

Modifying object

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

- Example

myObj.cars.car2 = "Mercedes";

Delete Object properties

- Use the delete keyword to delete properties from a JSON object:

- Example

delete myObj.cars.car2;

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.

Parse

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

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().

Stringify

- var obj = { name: "John", age: 30, city: "New York" };
  var myJSON = JSON.stringify(obj);
  document.getElementById("demo").innerHTML = myJSON;

Stringify Dates

- In JSON, date objects are not allowed. The JSON.stringify() function will convert any dates into strings.
- Example

```
var obj = { name: "John", today: new Date(), city : "New York" };
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
```

Stringify functions

- In JSON, functions are not allowed as object values.
- The JSON.stringify() function will remove any functions from a JavaScript object, both the key and the value:

- Example

```
var obj = { name: "John", age: function () {return 30;}, city: "New York"};
var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;
```

Local Storage

- var bookJSON = {
  “title” : “Object-oriented Programming”,
  “author” : {
  “firstName” : “Robert”,
  “seondName” : “Lafore”
  }
  “year” : 2007,
  “chapters” : [“Introduction”, “Classes”, “Inheritance”]
- }

Storing Data

- # Storing book data
- var bookStr = JSON.stringify(bookJSON);
-
- localStorage.setItem(“book”, bookStr);

Retrieving data

- #Retrieving book data
- var bookStr = localStorage.getItem(“book”);
- var bookJSON = JSON.parse(bookStr);

JQuery

Do It Yourself

Common Mistakes

- https://www.w3schools.com/js/js_mistakes.asp

Performance

- Reduce activity in loop.
- Reduce DOM access.
- Avoid Unnecessary Variables.
- Reduce DOM Size
- Putting your scripts at the bottom of the page body lets the browser load the page first.
- If possible, you can add your script to the page by code, after the page has loaded.

```
<script>
window.onload = function() {
  var element = document.createElement("script");
  element.src = "myScript.js";
  document.body.appendChild(element);
};
</script>
```

Loop Activity

- Bad code:

```
var i;
for (i = 0; i < arr.length; i++) {
```

- Good code:

```
var i;
var l = arr.length;
for (i = 0; i < l; i++) {
```

DOM Access

- Accessing the HTML DOM is very slow, compared to other JavaScript statements.

- If you expect to access a DOM element several times, access it once, and use it as a local variable

- ```
  var obj;
  obj = document.getElementById("demo");
  obj.innerHTML = "Hello";
  ```
  ```

References

- https://www.w3schools.com
- https://www.w3schools.com/js
- https://www.json.org/json-en.html
- https://www.w3schools.com/js/js_json_intro.asp
- https://en.wikipedia.org/wiki/JSON
- https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON
- https://www.tutorialrepublic.com/php-tutorial/php-json-parsing.php