



Mondragon
Unibertsitatea

Escuela Politécnica
Superior

Informatika
Ingeniaritza
Gradua

3rd course

Web Engineering I

JS - Introduction

Table of contents

1. What is JavaScript.
 - a. Introduction.
 - b. Client-side JavaScript.
 - c. Server-side JavaScript.
2. Java vs JavaScript.
3. Understanding JavaScript
 - a. Variables.
 - b. Constants.
 - c. Data Types.
 - d. Location.
 - e. Output/Input Data.
 - f. Events.
 - g. Getting Elements from the DOM.
 - h. Arrays.
 - i. Objects.
 - j. Front-end validation.
 - k. Take into Account
4. AJAX
5. JQuery
6. Exercise

1

What is JavaScript?

Introduction

What is JavaScript?

- JavaScript is a scripting language (not pre-compiled).
 - cross-platform.
 - object-oriented.
- Small & lightweight.
- Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.
- JavaScript contains:
 - A standard library of objects:
 - Array, Date, and Math, and
 - A core set of language elements:
 - Operators, control structures, and statements.
- Core JavaScript can be extended.

Client-side JavaScript

- Client-side JavaScript extends the core language by supplying objects to:
 - Control the browser
 - Control the Document Object Model (DOM).
- For example, client-side extensions allow an application to:
 - Place elements on an HTML form.
 - Respond to user events such as mouse clicks, form input, and page navigation

Server-side JavaScript

- Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server.
- For example, server-side extensions allow an application to:
 - Communicate with a database.
 - Provide continuity of information from one invocation to another of the application.
 - Perform file manipulations on a server.

*-side JavaScript

We will work with Client-side JavaScript in this course, as a programming language for the front-end.

2

Java vs Javascript

Do not mix them!

Java vs Javascript

Java is to JavaScript as car is to carpet.

javascript

IS NOT

java

Java vs Javascript

JavaScript compared to Java

JavaScript	Java
Object-oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.	Class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.
Variable data types are not declared (dynamic typing).	Variable data types must be declared (static typing).
Cannot automatically write to hard disk.	Can automatically write to hard disk.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript

Java vs JavaScript

- Borrows most of its syntax from **Java**, but is also influenced by **Awk**, **Perl** and **Python**.
- **Case-sensitive**.
- Uses the **Unicode** character set.

Java vs JavaScript

- Why is **Java** in its name then?
 - Marketing purposes.

3

Understanding JavaScript

Variables, data types, proper
objects...

3.1

Variables

- You use variables as symbolic names for values in your application.
- The names of variables, called identifiers, conform to certain rules:
 - Must start with a letter (a-zA-Z), underscore (_), or dollar sign (\$).
 - Subsequent characters can also be digits (0-9).
 - You can use ISO 8859-1 or Unicode letters such as å and ü.
 - You can also use the Unicode escape sequences as characters (\n, \t...).

Nevertheless, I would recommend to use the less “weird” characters as possible.

- You can declare variables in three ways:
 - With the keyword **var**.
 - For example: `var x = 42`.
 - This syntax can be used to declare both local and global variables.
 - By simply assigning it a value.
 - For example: `x = 42`.
 - This always declares a global variable.
 - It generates a strict JavaScript warning (You shouldn't use this variant).
 - With the keyword **let**.
 - For example: `let y = 13`.
 - This syntax can be used to declare a block scope local variable.
- **A variable declared using the var or let statement with no initial value specified has the value undefined.**

```
let x; -> x===undefined
```


Variables

- Let vs Var:

```
function varTest() {  
  var x = 1;  
  if (true) {  
    var x = 2; // same variable!  
    console.log(x); // 2  
  }  
  console.log(x); // 2  
}
```

```
function letTest() {  
  let x = 1;  
  if (true) {  
    let x = 2; // different variable  
    console.log(x); // 2  
  }  
  console.log(x); // 1  
}
```

Many static code analysis tools (Lint) recommend using let when possible.

3.2

Constants

Constants

- You can create a read-only, named constant with the `const` keyword.
- The syntax of a constant identifier is the same as for a variable identifier.

```
const prefix = '212';
```

Constants

- constants have to be initialized to a value.
- they cannot be changed through assignment or be re-declared while the script is running.
- You cannot declare a constant with the same name as a function or variable in the same scope.

```
const prefix = '212';
```

3.3

Data Types

- Six data types are primitive:
 - Boolean:
 - true or false.
 - null:
 - A special keyword denoting a null value. Because JavaScript is case-sensitive, null is not the same as Null, NULL, or any other variant.
 - Undefined:
 - A top-level property whose value is undefined.
 - Number:
 - 42, 3.14159, etc.
 - String:
 - “Hello”
 - Symbol (new in ECMAScript 6).
 - A data type whose instances are unique and immutable.

Data Type Conversion

- JavaScript is a dynamically typed language.
 - You don't have to specify the data type of a variable when you declare it.
 - Data types are converted automatically as needed during script execution.
- In expressions involving numeric and string values with the + operator, JavaScript converts numeric values to strings

```
x = "The answer is " + 42 // "The answer is 42"  
y = 42 + " is the answer" // "42 is the answer"
```

Data Type Conversion

- In expressions involving numeric and string values with the + operator, JavaScript converts numeric values to strings

```
x = "The answer is " + 42 // "The answer is 42"  
y = 42 + " is the answer" // "42 is the answer"
```

- In statements involving other operators, JavaScript does not convert numeric values to strings:

```
"37" - 7 // 30  
"37" + 7 // "377"
```

- Parsing is needed:

```
parseInt("37",10) + 7 // 44
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseInt

3.4

Location

Where do I put the code?

Where do I put the code?

- As tag attributes we can put the events and calls to the functions.
- In a `<script>` tag.
 - Inside the `<head>` tag.
 - Inside the `<body>` tag.
- Link to the external JS file (.js extension).
 - Easier to maintain.
 - Cache JavaScript files to speed up page loads.
 - `<script src='example.js'></script>`

3.5

Output/input Data

How to show/get data to/from
users using JavaScript

Example also available in [\[GitLab\]](#) JavaScript examples (1.1 & 2.3)

Output data

- Alert [[CodePen](#)]
 - Easy to use.
 - Intrusive output (users can see it, but makes difficult to use the web page).

```
alert("This is an alert!");
```

- Debug console log. [[CodePen](#)]
 - No intrusive output, only expert users can see it.
 - Useful for developers.

```
console.log("this is a log output");
```

- Writing in the DOM [[CodePen](#)]:
 - More complex output.
 - Non intrusive, the user can see the output.
 - You can change many things, even add new HTML elements.

```
var paragraph = document.getElementById("writeMe");  
paragraph.innerHTML = "This is the new content for the paragraph";
```

Input data

- Prompt [[CodePen](#)]

- Similar to alerts but waits for an input.

```
let text = prompt("Write your name");
```

- Events [[CodePen](#)]

- You can perform actions when a button is clicked.
- (More info about events in the next subsection.)

```
<button ... onclick="change_paragraph()">...</button>
```

```
function change_paragraph() {  
    ...  
}
```

- Reading the DOM [[CodePen](#)]:

- Non intrusive.
- You can read many things, even HTML elements.

```
let input-value = document.getElementById("my-input-text").value;
```

3.6

Events

How to interact with browser
events

- Browsers generate events in different cases.
 - When mouse is clicked.
 - When an element has the focus.
 - When the page has been loaded
 - ...
- JavaScript can get those events and act in consequence.

Events Exercise

- We will learn how to use the events with an exercise.
- It can be found on Mudle
 - Download “Events exercise (Starting point)”.
 - You have to complete the “writeText(str)” function:
 - Get the “writeMe” paragraph.
 - The message must be “Event: onclick”
 - Write the message on the paragraph.
 - Write the message on the console log.
- There is the onclick event done, you have to do the other ones.
 - When the document is loaded.
 - When the window is resized.
 - The ones that can be seen in the page (see html comments).

Testing DOM events

open console log to see the events

When focused/unfocused on me

When the text here is changed

When clicked

Similar but when I am pressed or released

When right-clicked on me

When double clicked on me

When the mouse is over me and when it is out of me

event: onclick

More info at [MDN - Event reference](#) or at [w3schools.com - HTML DOM Events](#)

3.7

Getting Elements from the DOM

How to interact with the DOM
elements

Getting elements from the DOM

- Classic
 - By ID
 - Returns a single element (Remember it is singular, “element”).
`var element = document.getElementById(“id”);`
 - By Class
 - Returns an array of elements (Remember it is plural, “elementS”).
`var elements = document.getElementsByClassName(“class”);`
 - By Tag
 - Returns an array of elements (Remember it is plural, “elementS”).
`var elements = document.getElementsByTagName(“HTML tag”);`

Getting elements from the DOM

- Modern
 - Due to influence of JavaScript libraries/frameworks (e.g. JQuery).
 - Using CSS selectors to select DOM elements.

```
var elements = document.querySelectorAll('.div > p');
```

```
var elements = document.querySelector('.div > p');
```
- `querySelector` => returns 1st matching element
- `querySelectorAll` => returns one Element object for each match.

Examples: <https://codepen.io/muaperez/pen/XWdVymv>

<https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector>

3.8

Arrays

List of elements

Getting elements from the DOM

- They are objects
- Can be from different types:

```
var myArray = ["text",15,myObject];
```

- They have attributes and methods:

```
myArray.length;  
myArray.sort();  
myArray.push(element);  
myArray[myArray.length] = element
```

```
myArray.forEach(myFunction);  
function myFunction(item, index){  
    item = index  
}
```

Example

- `querySelector + querySelectorAll + forEach()`

<https://codepen.io/muaperez/pen/ExjOjPG>

3.9

Objects

- Example (single object)

```
var contact = {  
  name: "Alain",  
  department: 11219,  
  ext: 8177  
};  
alert(contact.name);  
alert(contact['name']);
```

- Example (constructor for multiple):

```
function contact (name,department,ext) {  
    this.name=name;  
    this.department=department;  
    this.ext=ext;  
}  
  
var alain = new contact("Alain",11219,8177);  
var txomin = new contact("Txomin",2087,8563);
```

- Adding methods

```
function contact (name,department,ext) {  
    this.name=name;  
    this.department=department;  
    this.ext=ext;  
    this.changeExt = function (ext){  
        this.ext = ext;  
    }  
}  
  
var alain = new contact("Alain",11219,8177);  
alain.changeExt(8144);
```

- Adding methods

```
function contact (name,department,age) {  
    this.name=name;  
    this.department=department;  
    this.age=age;  
    this.birthYear = calcBirthYear;  
}  
function cacBirthYear(){  
    return 2021 - this.age;  
}
```

3.10

Front-End validation

HTML inputs can be validated
using JavaScript before submit

Front-end validation

- Validation for the regular user.
 - Many HTML5 input types auto validate.
 - Attributes.
 - “required”
 - “novalidate” in the form for development.
 - “pattern” => <http://html5pattern.com/>
 - [0-9]{9}
 - [0-9]{13-16}
 - [a-zA-Z]+
 - ...
 - Javascript.
 - No all browsers support html5 input type validations.
 - JavaScript can be executed onchange, onblur...

3.11

Take into account

Weird or not so obvious cases
and behaviours

Take into account

- Comparing value or value and type

```
5 == '5' => True  
5 === '5' => False
```

- Numbers, strings and sums

```
5 + 5 => 10  
'5' + '5' => '55'  
'5' + 5 => '55'  
parseInt('5') + 5 = 10
```

- Double '=' for comparisons:

```
if(name=="Alain") => OK  
if(name="Alain") => Probably wrong
```

- Multiple type arrays are possible:

```
var myArray = ["text",15,myObject];
```


Take into account

- WARNING!
 - JavaScript is a **single-threaded** programming language.
 - **Synchronous execution model**: processes one operation after another.
 - Execution can be blocked!
 - The browser environment has many Web APIs that JavaScript can access that are **asynchronous**.
 - E.g.: `setTimeout`
 - ECMAScript 2015 addition of *promises*, and the modern practice of using *async/await*.

3.12

Minimize the JS in the HTML

You can remove even event
attributes in the HTML

Minimize the JS in the HTML

- Use an external file.
 - Already explained how.
- Add “on-” events from the JavaScript.
 - Using `window.onload` & `addEventListener()`

[[GitLab](#)] JavaScript examples (3.1 & 3.2)

4

AJAX

Getting data without reloading
the page

- Asynchronous JavaScript and XML.
- Update web pages without reloading them.
 - Request, receive and send data to a server in the background.
- Libraries to make AJAX calls easier
 - jQuery (among other things, it adds AJAX functionalities)
 - <https://jquery.com/>
 - Axios (one of the best libraries for calling web services)
 - <https://www.npmjs.com/package/axios>
- W3school AJAX Tutorial.
- <http://www.w3schools.com/ajax/>

5

JQuery

Most extended JavaScript lib

- 1 of the most used JavaScript Library.
- It makes easy to use some JavaScript.
 - With querySelector some JQuery functionalities have been adapted.
- Not so fashioned nowadays.
 - There is a movement to reduce its use.
 - e.g.: Bootstrap 4 uses it but Bootstrap 5 will not use it.
- Very important to know to understand many web applications.
- Special object: \$

```
$(‘.hide-me’).hide() // this is similar to  
querySelectorAll(...).forEach(...)
```

- W3school jQuery Tutorial <http://www.w3schools.com/jquery/>

6

Mini Exercises

9 small exercises to practice
JavaScript

Mini Exercises

- 9 small exercises:
 - Multiply/Divide
 - URL
 - Amount of tags
 - Password fields
 - Changing style with JavaScript
 - Add rows to table
 - Add columns to table
 - Print
 - Date
- Result will be available after deadline on:
 - <https://gitlab.com/mgep-web-engineering-1/js/mini-exercises-result>

Multiply/Divide

- Make an HTML with 2 input fields (e.g. num1 & num2).
- Add 2 buttons:
 - 1 for dividing
 - 1 for multiplying
- When each button is clicked write the result in the console log and in a paragraph.

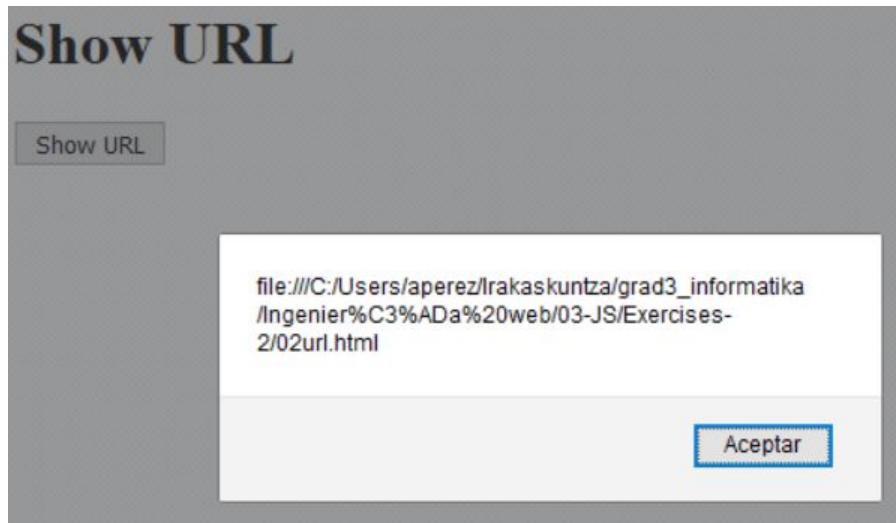
Multiply/divide

Insert the second number:

Insert the first number:

Result: 6

- Make a function that writes the URL of the document in an alert when clicked a button.
 - Search in the internet how to do it.



Count tags

- Create an HTML document with different type of content.
- Add an input text to write an html tag
- Add a button:
 - Take the tag name from the input text.
 - Search how many of those tags are in the document.
 - Print it in an alert and in the console log.

Tag name:

- This is a list item
- This is a list item

This is a paragraph

This is a paragraph

This is a paragraph

This is a paragraph with a span.

This is a paragraph with a span.

This is a paragraph with a span.

This is a paragraph with a span.

- This is a list item
- This is a list item

There are 4 'span' tags

Password fields

- Create a form with 2 password fields.
 - The first one should check if it is a **strong password** when it is changed (at least 10 characters for example).
 - The second one should check if **both password fields are the same**.
 - The output errors should be **printed** in a span element after each field.
 - If the error is solved, you should **delete** the error from the span element.
 - **Use css** to add a red color to the error text.

Check passwords

Password: The password is too short!

Repeat password: Both passwords are not the same!

Changing style with JavaScript

- Add a paragraph and a button in a HTML document
- When the button is clicked, the style of the paragraph should be changed as in the image.

Style the p element

This is the paragraph

Style the paragraph

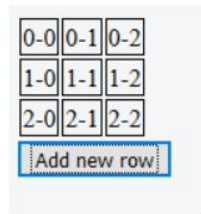
Style the p element

~~This is the paragraph~~

Style the paragraph

Add rows to tables

- Create a table in an html document.
- Add a button
 - It should add rows to the table.
- Do it in 2 ways (one button for each way):
 - Use **insertHTML** in the table with the full HTML code.
 - Use **insertRow()** & **insertCell()** & **textContent**



Add columns to tables

- Use the same HTML as in the previous exercise.
- Use only the second way
 - **insertRow() & insertCell() & textContent**

Adding rows to a table

0-0	0-1	0-2
1-0	1-1	1-2

Add new row

Adding rows to a table

0-0	0-1	0-2	0-3
1-0	1-1	1-2	1-3

Add new row

Adding rows to a table

0-0	0-1	0-2	0-3	0-4	0-5	0-6	0-7	0-8	0-9	0-10	0-11	0-12	0-13	0-14
1-0	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	1-11	1-12	1-13	1-14

Add new row

Hint:

```

var table = document.getElementById("myTableID");
var rows = table.getElementsByTagName("tr");
  
```


- Make an HTML document with a button.
- If you press the button, the whole document will be printed (you may need a PDF printer app for that).
- Add some extra elements that appear in the browser but not in the PDF.

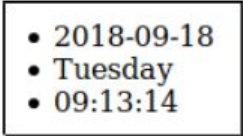
Hint:

```
window.print()
```

Hint 2:

Use mediaqueries for blocking the printing of elements.

- From an empty body, execute a JS function when the document is loaded.
- This function should create a list inside the document with the current date as in the image below.
 - Use **document.createElement()** and **document.createTextNode()** to create the list.



- 2018-09-18
- Tuesday
- 09:13:14

Hint 1:

Remember that Anglo-Saxon week starts on Sunday!!!

Interesting references

- Mozilla Contributors - MDN - JS:
 - <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Eskerrik asko
Muchas gracias
Thank you

Alain Perez Riaño

aperez@mondragon.edu

Goiru, 2
20500 Arrasate – Mondragon
Room: 11214
T. 647 50 44 76 (ext: 8177)
info@mondragon.edu