Co-funded by the
Erasmus+ Programme
of the European Union

# JavaScript Training Materials
# Subchapter 3 – JS & BOM

WP3: Code4SP Training Materials

Prepared by: social hackers academy

Co-funded by the
Erasmus+ Programme
of the European Union

# **Subchapter 3** – JS & BOM

Co-funded by the
Erasmus+ Programme
of the European Union

# The Browser Object Model (BOM)

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser. There are no official standards for the Browser Object Model (BOM). Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

Co-funded by the
Erasmus+ Programme
of the European Union

# The Window Object

The window object is supported by all browsers. It represents the browser's window. All global JavaScript objects, functions, and variables automatically become members of the window object. Global variables are properties of the window object. Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

```
window.document.getElementById("header");
```

Co-funded by the
Erasmus+ Programme
of the European Union

# Window Size

Two properties can be used to determine the size of the browser window. Both properties return the sizes in pixels: window.innerHeight

- the inner height of the browser window (in pixels) window.innerWidth
- the inner width of the browser window (in pixels)

The browser window (the browser viewport) is NOT including toolbars and scrollbars.

```
let w = window.innerWidth;
let h = window.innerHeight;
```

Co-funded by the
Erasmus+ Programme
of the European Union

# Window Screen

The window.screen object can be written without the window prefix. Properties:

- screen.width
- screen.height
- screen.availWidth
- screen.availHeight
- screen.colorDepth
- screen.pixelDepth

Co-funded by the
Erasmus+ Programme
of the European Union

# Window Location

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page. The window.location object can be written without the window prefix. Some examples:

- window.location.href returns the href (URL) of the current page
- window.location.hostname returns the domain name of the web host
- window.location.pathname returns the path and filename of the current page
- window.location.protocol returns the web protocol used (http: or https:)
- window.location.assign() loads a new document

Co-funded by the
Erasmus+ Programme
of the European Union

# Window History

The window.history object can be written without the window prefix. To protect the privacy of the users, there are limitations to how JavaScript can access this object. Some methods:

- history.back() - same as clicking back in the browser
- history.forward() - same as clicking forward in the browser

Co-funded by the
Erasmus+ Programme
of the European Union

# Window History Back

The history.back() method loads the previous URL in the history list. This is the same as clicking the Back button in the browser.

Example

Create a back button on a page:

```
<html>
<head>
<script>
function goBack() {
  window.history.back()
}
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

The output of the code above will be:

Back

Co-funded by the Erasmus+ Programme of the European Union

# Window Navigator

The window.navigator object contains information about the visitor's browser. The window.navigator object can be written without the window prefix. Some examples:

- navigator.appName
- navigator.appCodeName
- navigator.platform

```
document.getElementById("demo").innerHTML =
"navigator.appName is " + navigator.appName;
```

Co-funded by the
Erasmus+ Programme
of the European Union

# JavaScript Popup Boxes

## Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed. The window.alert() method can be written without the window prefix.

```
alert("I am an alert box!");
```

Co-funded by the
Erasmus+ Programme
of the European Union

# Confirm Box

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false. The window.confirm() method can be written without the window prefix.

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

# Prompt Box

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

```
let person = prompt("Please enter your name", "Harry Potter");
let text;
if (person == null || person == "") {
  text = "User cancelled the prompt.";
} else {
  text = "Hello " + person + "! How are you today?";
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

# Timing Events

The window object allows execution of code at specified time intervals. These time intervals are called timing events. The two key methods to use with JavaScript are:

- setTimeout(function, milliseconds) Executes a function, after waiting a specified number of milliseconds.
- setInterval(function, milliseconds) Same as setTimeout(), but repeats the execution of the function continuously.

The setTimeout() and setInterval() are both methods of the HTML DOM Window object.

Co-funded by the
Erasmus+ Programme
of the European Union

# JavaScript Cookies

Cookies are data, stored in small text files, on your computer. When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user. Cookies were invented to solve the problem "how to remember information about the user":

- When a user visits a web page, his/her name can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" his/her name.

Cookies are saved in name-value pairs like:

```
username = John Doe
```

Co-funded by the
Erasmus+ Programme
of the European Union


# THANK YOU!

Co-funded by the
Erasmus+ Programme
of the European Union