Școala
informală
de IT

JavaScript BOM

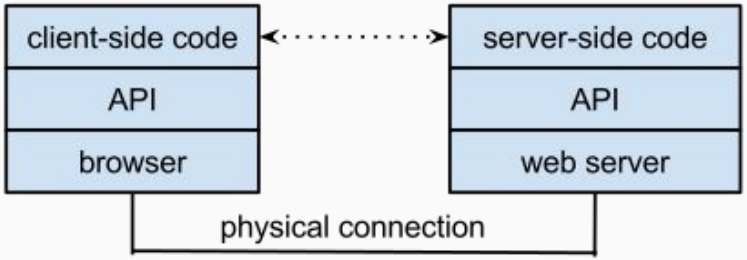
Agenda

- Window
- Screen
- Timing
- Location
- History & navigation
- Cookies

JavaScript and The Browser

JavaScript and The Browser

- The browser provides a set of Web APIs (Web Application Programming Interfaces) that can be used to perform various tasks (via Javascript), such as:
  - Manipulating the DOM
  - Manipulating browser's capabilities
  - Playing audio and video
  - etc.



- Web API = programmatic interface to extend functionality within a web browser or other HTTP client

Manipulating the browser

- The set of Web APIs that are used to manipulate the browser represents the *Browser Object Model*
- There are no official standards for the BOM
- Such APIs are:
  - Window
  - History
  - Location

Window Object

Window Object

- The Window object = a window containing a DOM
- Each tab (in a tabbed browser) contains its own window object
- Its properties provide access to
  - the DOM
  - screen properties
  - cursor and scroll
- *Verify if the properties are supported in a certain browser before using them!*

Generic Window Properties

- Window.console - references the console object
- Window.document - references the document that the window contains
- Window.frames - an array of the subframes in current window
- Window.name - gets/sets the name of the window
- Window.parent - references the parent of the current window

Window Screen Properties

- **Window.screen - references the associated screen object**
- **Window.screenX - the horizontal distance of the left border of user's browser from the left side of the screen**
- **Window.screenY - the vertical distance of the top border of user's browser from the top side of the screen**
- **Window.fullScreen - indicates if it is displayed in fullscreen or not**
- **Window.orientation - the orientation in degrees (90 degree increments) of the viewport relative to device's natural orientation - DEPRECATED**

Window Width/Height Properties

- Window.innerHeight - the height of the content area of the browser window including, if rendered, the horizontal scrollbar
- Window.innerWidth - the width of the content area of the browser window including, if rendered, the vertical scrollbar
- Window.outerHeight - the height of the outside of the browser window
- Window.outerWidth - the width of the outside of the browser window

Window Scroll Properties

- Window.scrollbars - the scrollbars object (can toggle its visibility)
- Window.scrollX = Window.pageXOffset - the number of pixels that the document has already been scrolled horizontally
- Window.scrollY = Window.pageYOffset - the number of pixels that the document has already been scrolled vertically

Window Generic Methods

- Window.open() - opens a new window
- Window.close() - closes the current window
- Window.blur() - sets focus away from window
- Window.focus() - sets focus on the current window
- Window.openDialog() - opens a new dialog window
- Window.print() - opens the print dialog to print current document

Window Sizing and Positioning Methods

- Window.resizeBy() - resizes the window by a certain amount
- Window.resizeTo() - dynamically resizes window
- Window.sizeToContent() - sizes window according to its content
- Window.moveBy() - moves the current window by an amount
- Window.moveTo() - moves the window to the specified coordinates

Window Scroll Methods

- Window.scroll() - scrolls the window to a particular place in the document
- Window.scrollBy() - scrolls the window by a given amount
- Window.scrollByLines() - scrolls by a specified number of lines
- Window.scrollByPages() - scrolls by a specified number of pages
- Window.scrollTo() - scrolls to a set of coordinates in the document

Window Generic Events

- onload - called after all resources and the DOM are fully loaded (not when the page is loaded from cache)
- onunload - called when the user navigates away
- onbeforeunload - event handler property for before-unload events on the window
- onclose - called after the window is closed
- onerror - called when a resource fails or in the case of a runtime error

Window Size and Orientation Events

- onresize - called continuously as resizing the window
- onscroll - called when the scrollbar is moved via ANY means
- ondeviceeorientation - when the orientation is changed
- ondevicemotion - when the accelerometer detects a change

Timing on Window object

Timing

- setTimeout() - schedules a function to execute in a given amount of time
- clearTimeout() - cancels the delayed execution set using setTimeout()
- setInterval() - schedules a function to execute every time a given number of milliseconds elapses
- clearInterval() - cancels the repeated execution set using setInterval()

Set and clear timeout/interval

```
let reminderTimeout = setTimeout(function() {
    console.log(“after 5 seconds”);
}, 5000);

clearTimeout(reminderTimeout);
```

```
let reminderInterval = setInterval(function() {
    console.log(“every 2 seconds”);
}, 2000);

clearInterval(reminderInterval);
```

Location

Location

- = the location (URL) of the object it is linked to
  - Window.location
  - Document.location
- Properties
  - href, protocol, host, hostname, port, pathname, search, hash, origin
- Methods
  - assign(), reload(), replace(), toString()

Location properties

```
// navigate to
https://www.google.ro/search?q=window+location&client=firefox-b#q=
window+location&start=10&*
console.log(url.href);      // same url as above
console.log(url.protocol); // https:
console.log(url.host);      // www.google.ro
console.log(url.hostname); // www.google.ro
console.log(url.port);      // (blank-https assumes 443)
console.log(url.pathname); // /search
console.log(url.search);    // ?q=window+location&client=firefox-b
console.log(url.hash);      // q=window+location&start=10&*
console.log(url.origin);    // https://www.google.ro
```

Location methods

- assign()
  - Loads the resource at the URL provided in parameter
- reload()
  - Reloads the resource from the current URL
- replace()
  - Replaces the current resource with the one at the provided URL; after using replace(), the current page will not be saved in session History
- toString()
  - returns a string containing the whole URL

History

History

- Allows to manipulate the browser session history (i.e., the pages visited in the tab or frame that the current page is loaded in)
- Properties: length, state
- Methods: back(), forward(), pushState(), replaceState()
- Event: onpopstate
  - A popstate event is dispatched to the window every time the active history entry changes between two history entries for the same document. pushState and replaceState don't trigger a popstate event!

Manipulating the browser history

- Traveling through history

```
window.history.back();
window.history.forward();
```

```
window.history.go(-1);
window.history.go(1);
```

- Determining the number of pages in history stack

```
var numberOfPages = history.length;
```

- Reading the current state

```
var currentState = history.state;
```

Manipulating the browser history

- Adding and modifying history entries
  - History.pushState() pushes the given data onto the session history with the specified title and URL
  - This causes the URL bar to display host/next.html but won't cause the browser to load next.html
  - onpopstate event is triggered when going back to that page

```
var stateObj = { current: “next” };
history.pushState(stateObj, “next page”, “next.html”);
```

Manipulating the browser history

- Adding and modifying history entries
  - History.replaceState() modifies the current history entry instead of creating a new one
  - This doesn't prevent the creation of a new entry in the global event history
- replaceState() is particularly useful when updating the state object or URL of the current history entry in response to some user action

```
var stateObj = { current: "next" };
history.replaceState(stateObj, "third page", "third.html");
```

Manipulating the browser history - Demo

- **Let’s see some code!**

Cookies

Reminder: HTTP cookie

- = small piece of data that a server sends to the user's web browser, that may store it and send it back with the next request to the same server
- Remembers stateful information for the stateless HTTP protocol
- 3 main purposes:
  - Session management (user logins, shopping carts)
  - Managing user preferences
  - Tracking (analyzing user behavior)

Document.cookie

- Gets and sets the cookies associated with the current document
- Get all cookies:

```
allCookies = document.cookie;
```

- Write a new cookie:

```
document.cookie = newCookie;
```

- Attributes:
  - path, domain, max-age (in seconds), expires (date in GMTString format), secure

Hands on Cookies

1. Set a cookie on a document
2. Read all cookies
3. Add two radio buttons with two available languages (e.g., en-US, ro-RO)
   - The one whose value equals cookie's value should be "pre-selected"
   - When the user selects the other radio button, his option should be preserved in the cookie

Homework

Homework

Cookies: The user can have a preference whether to see the weather is in C or in F. Store this in a cookie and use it to display the temperature for the user based on that cookie

Resources

https://developer.mozilla.org/en-US/docs/Web/API/Window

https://developer.mozilla.org/en-US/docs/Web/API/Location

https://developer.mozilla.org/en-US/docs/Web/API/History

https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie

http://www.quirksmode.org/js/cookies.html

https://www.tutorialspoint.com/javascript/javascript_cookies.htm