JS DOM

Some of examples and definitions are from very good JS docs - https://developer.mozilla.org/ .

Hello!

I am Mateusz Choma

I am a scientific mind, passionate of technology, an engineer "squared" - a graduate of two universities in Lublin :)
As well, I am a JS developer, entrepreneur and owner of small software house - Amazing Design.

1.
Inserting JS to HTML

Inserting JS to HTML
Script tag

Directly in HTML, JavaScript code must be inserted between <script> and </script> tags.

```
<script>
    alert('Hello world!');
</script>
```

Inserting JS to HTML
Script tag

We can include a .js file using a script tag ant src attribute.

```
<script src="js/script.js"></script>
```

Inserting JS to HTML
Script tag

If we put scripts at the bottom the web pages will be displayed to users first, then load javascripts - the pages renders faster.

```
<html>
  <head>
    ......
  </head>
  <body>
    ......
    <script></script>
  </body>
</html>
```

Scripts run from top to bottom so pay attention to to the order!

Additional info: https://stackoverflow.com/questions/28635141/sequence-in-which-html-page-loads

2.
DOM?

DOM

What is DOM?

The Document Object Model (DOM) connects web pages to scripts or programming languages. Usually that means JavaScript.

The DOM model represents a document with a logical tree.

Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them you can change the document's structure, style or content.

Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.

DOM
Window

The **window object** represents a window containing a DOM document.

The document property points to the DOM document loaded in that window.

A window for a given document can be obtained using the **document.defaultView** property.

WINDOW IS A GLOBAL SCOPE FOR JS IN THE BROWSER!

DOM
Window methods

window.alert() - display a message in browser

window.open() - opens new window

window.close() - close the window

window.prompt() - display a prompt

DOM
Window methods usage

window.alert(message); - display a message in browser

window.prompt(message, value); - display a prompt

window.confirm(message); - display a confirm dialog

window.open(url); - opens new window

window.close(); - close the window

DOM
Window methods

```
window.alert('Hello world!');

// true or false
var confirmed = window.confirm('It is ok?');

// string
var answer = window.prompt('It is ok?','Yes!');
```

Task 0

Declare a function that asks user if he/she want to answer a question (confirm), if yes ask a question (prompt) and displays the answer in an alert!

DOM
Window location

```
// return Location object - check it in browser

window.location

// we can set new location

window.location = “http://google.com”
```

Task 1

Create a prompt that take URL from user and go to that URL.

Try to open that URL in new tab/window.

DOM
Document

The **document object** represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree.

It provides functionality global to the document, like how to obtain the page's URL and create new elements in the document.

DOM
Document methods

The **document object** most important methods are used to **select, modify and create DOM elements**.

We describe them one by one in next slides.

3.
Selecting DOM elements

Selecting DOM elements
document.getElementById()

Returns a reference to the element by its ID attribute.

```
<html>
    <head></head>
    <body id="example”></body>
</html>

var elem = document.getElementById(‘example’);
elem.style.backgroundColor = 'red';
```

Selecting DOM elements
document.getElementById()

Returns a reference to the element by its ID attribute.

```
<html>
    <head></head>
    <body id="example”></body>
</html>

var elem = document.getElementById(‘example’);
elem.style.backgroundColor = 'red';
```

Selecting DOM elements
document.getElementsByClassName()

getElementsByClassName() returns an HTMLCollection of elements with the given class name.

An HTMLCollection object is an array-like list (collection) of HTML elements. It does not inherit from Array object, so it does not have all the standard Array methods (like forEach).

```
var elements = document.getElementsByClassName("p");
```

The elements in the collection can be accessed by an index number, as an array.

```
var element = elements[1];
```

Task 2

Make 3 HTML paragraphs in the body with class="p". Select first and last one and assign them to variables. Console.log that variables.

Selecting DOM elements
document.getElementsByClassName()

You may also call **getElementsByClassName()** on any element; it will return only elements which are descendants of the specified root element with the given class names.

Task 3

Make 2 HTML divs in the body one in another. First with id = “first”. Select second.

Selecting DOM elements
document.getElementsByTagName()

document.getElements**ByTagName()** returns an HTMLCollection of elements with the given tag name.

```
var elements = document.getElementsByTagName("p");
```

Selecting DOM elements
document.querySelector()

document.querySelector() Returns the first Element within the document that matches the specified selector, or group of selectors.

```
var element = document.querySelector("p");
```

Selecting DOM elements
document.querySelectorAll()

document.querySelectorAll() returns a list of the elements within the document that match the specified group of selectors. The object returned is a NodeList (that is similar to HTMLColleciton that is similar to array, and have forEach method).

```
var elements = document.querySelectorAll("p");
```

The elements in the collection can be accessed by an index number, as an array.

```
var elements = elements[1];
```

Task 4

Create an HTML element and select it with all method you know, then assign effects to variables, then console.log them!

4.
Navigation through DOM elements

Navigation through DOM elements
element.parentNode

The **element.parentNode** property returns the parent of the specified node in the DOM tree.

Navigation through DOM elements
element.firstChild

The **element.firstChild** property returns the node's first child in the tree, or **null** if the node has **no children**.

Navigation through DOM elements
element.nextSibling

The **element.nextSibling** property returns the node immediately following the specified one in its parent's childNodes list, or null if the specified node is the last node in that list.

Navigation through DOM elements
element.previousSibling

The **element.previousSibling** property returns the node immediately preceding the specified one in its parent's childNodes list, or null if the specified node is the first in that list.

Task 5

Create some divs and divs in divs and try to traverse this structure using methods from previous slides.

Navigation through DOM elements
element.remove()

The **element.remove()** method removes the object from the tree it belongs to.

Navigation through DOM elements
document.createElement()

document.createElement() method creates the HTML element specified by tagName.

```
var element = document.createElement('div');
```

Navigation through DOM elements
element.appendChild()

element.appendChild() method adds a node to the end of the list of children of a specified parent node. If the given child is a reference to an existing node in the document, appendChild() moves it from its current position to the new position.

Navigation through DOM elements
element.insertBefore()

element.insertBefore() method inserts the specified node before the reference node as a child of the current node.

```
var insertedEl = parentEl.insertBefore(newEl, referenceEl);
```

If referenceEl is null, the newEl is inserted at the end of the list of child nodes.

Navigation through DOM elements
document.createTextNode()

document.createTextNode() creates a new text node, that can be append to HTML Element.

```
var newtext = document.createTextNode(text);
var p1 = document.getElementById("p1");
p1.appendChild(newtext);
```

Navigation through DOM elements
element.hasChildNodes()

element.hasChildNodes() method returns a Boolean value indicating whether the current Elemenet has child nodes or not.

Navigation through DOM elements
addElement custom function

```
function addElement () {
  var newDiv = document.createElement("div");
  var newContent = document.createTextNode("Hi there!");
  newDiv.appendChild(newContent);

  var currentDiv = document.getElementById("div1");
  document.body.insertBefore(newDiv, currentDiv);
}
```

Navigation through DOM elements
addElement custom function

```
function addElement (tag, text, target) {
  var newEl = document.createElement(tag);
  var newContent = document.createTextNode(text);
  newEl.appendChild(newContent);

  target.appendChild(newEl)
}
```

Navigation through DOM elements
innerHTML & innerText

The **element.innerHTML** property of the sets or returns the DOM tree inside the element. It can have problems with different browsers.

element.innerText is a property that represents the "rendered" text content of a node and its descendants. As a getter, it approximates the text the user would get if they highlighted the contents of the element with the cursor and then copied to the clipboard.

Task 6

Create div tag with another div inside. Create text ‘Hello’ in second div. Use only JS.

5.
Managing class and attributes

Managing class and attributes
element.classList

The **element.classList** is property which returns a collection (similar to array) of the class attributes of the element.

It has two important methods:
add - that add specified class values, if these classes already exist in attribute of the element, then they are ignored.
remove - that remove specified class values.

Managing class and attributes
element.className

element.className gets and sets the value of the class attribute of the specified element.

Task 7

Create div tag in HTML. Add and remove some class to it by className and classList.add and classList.remove.

Managing class and attributes
element.setAttribute()

element.setAttribute() sets the value of an attribute on the specified element. If the attribute already exists, the value is updated; otherwise a new attribute is added with the specified name and value.

```
var b = document.querySelector("button");
b.setAttribute("name", "helloButton");
b.setAttribute("disabled", "");
```

Managing class and attributes
element.removeAttribute()

element.removeAttribute() removes an attribute from the specified element.

```
// <div id="div1" align="left" width="200px">
document.getElementById("div1").removeAttribute("align");
// now: <div id="div1" width="200px">
```

Task 8

Create input tag in HTML. Add and remove disabled attribute and name attribute with value to it.

6.
CSS styles

CSS styles
HTMLElement.style

The **HTMLElement.style** property is used to get as well as set the inline style of an element. While getting, it returns a CSSStyleDeclaration object that contains a list of all styles properties for that element with values assigned for the attributes that are defined in the element's inline style attribute.

Styles can be set by assigning values to the properties of HTMLElement.style. For adding specific styles to an element without altering other style values, it is preferred to use the individual properties of style

```
var div = document.createElement('div');
div.style.color = 'red' ;
```

CSS styles
HTMLElement.style

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference

Task 9

Change the body background-color to ‘red’.

7.
Managing forms

Managing forms
Input values

We can get and set **input and textareas** values using **value** property.

```
// <input name = “text” />
input.value = 'Hello input';
input.value; // 'Hello input'
```

Managing forms
Submitting forms

We can submit form as easy as:

```
var form = document.querySelector('form');
form.submit();
```

8.
Simple events & events listeners

Simple events & events listeners

Common HTML events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Simple events & events listeners

Common HTML events

We can attach events by attribute in HTML but that method isn't the best.

```
<button onclick="alert('Click!')">Click!</button>
```

Task 10

Make an alert on button click!

Simple events & events listeners
EventTarget.addEventListener()

The **EventTarget.addEventListener()** method adds the specified EventListener-compatible object to the list of event listeners for the specified event type on the EventTarget on which it is called. The event target may be an Element in a document, the Document itself, a Window, or any other object that supports events (such as XMLHttpRequest).

Simple events & events listeners
EventTarget.addEventListener()

Syntax is:

```
target.addEventListener(type, listener[, options]);
```

type - is string representing the event type to listen for
listener - is the object which receives a notification, in most cases a JavaScript function

Simple events & events listeners
EventTarget.addEventListener()

```
// anonymous function attached to event listener
element.addEventListener("click", function(event){
    alert("Hello World!");
});

// named function attached to event listener
element.addEventListener("click", myFunction);
function myFunction(event) {
    alert ("Hello World!");
}
```

Simple events & events listeners
EventTarget.addEventListener()

We can add multiple event listeners to one target!

Task 11

Make a button and implement an onclick method that alerts ‘Heeeelo!’.