SoftServe | Empowering your Business through Software Development

Module 7:
Accessing DOM with JavaScript

Agenda

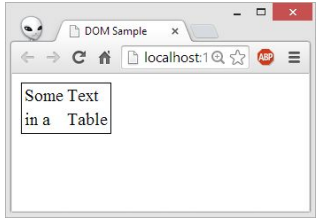
- Introducing DOM
- Manipulating DOM with JavaScript
- Cookies and Storages
- Useful links

Introducing DOM

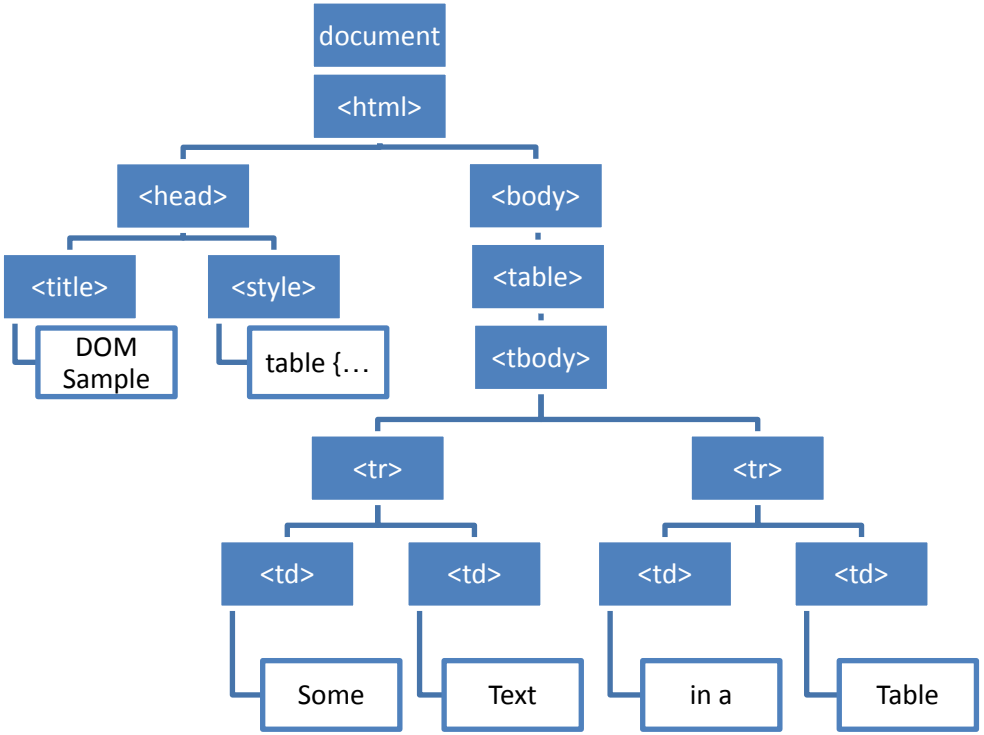
What is "DOM"?

- **DOM** – an acronym for **Document Object Model**.
- It's an interface that provides browser to allow scripts on a webpage to dynamically access and update the content, structure and style of documents.
- When browser prepares webpage to be shown to user, it constructs tree of objects from all elements of a page according to it's HTML structure
- JavaScript code can access the tree and modify it, browser reacts on changes and updates HTML page shown to the user.
- Changing HTML with JavaScript using DOM interface is also called as **Dynamic HTML.**

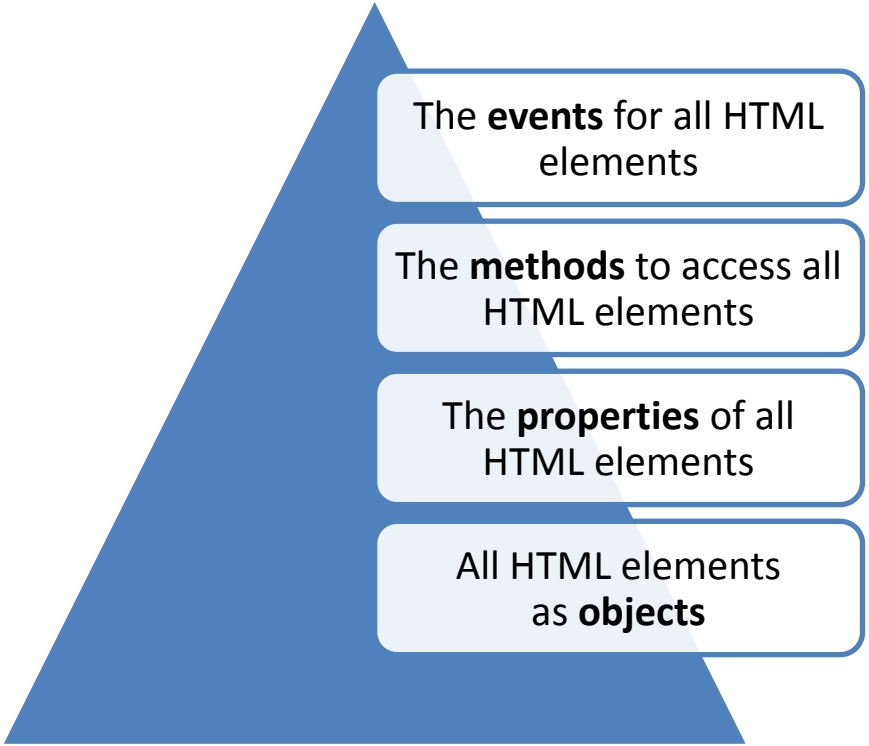
DOM Tree




```
<!DOCTYPE html>
<html>
    <head>
        <title>DOM Sample</title>
        <style type="text/css">
            table {
                border: 1px solid black;
            }
        </style>
    </head>
    <body>
        <table>
            <tbody>
                <tr>
                    <td>Some</td>
                    <td>Text</td>
                </tr>
                <tr>
                    <td>in a</td>
                    <td>Table</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```



What DOM Defines?



What can do JavaScript with DOM?

JavaScript can change all the HTML elements in the page

JavaScript can change all the HTML attributes in the page

JavaScript can change all the CSS styles in the page

JavaScript can remove existing HTML elements and attributes

JavaScript can add new HTML elements and attributes

JavaScript can react to all existing HTML events in the page

JavaScript can create new HTML events in the page

Manipulating DOM with JavaScript

Finding Elements

Finding HTML elements by id

Finding HTML elements by tag name

Finding HTML elements by class name

Finding HTML Elements by id

- `var t = document.getElementById('target');`
- Will find *one* element with id "target"

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p id="target">Sample Target</p>
        <p>Another Paragraph</p>
    </body>
</html>
```

Finding HTML Elements by Tag Name

- `var p = document.getElementsByTagName('p');`
- Will find *all* paragraphs on a page

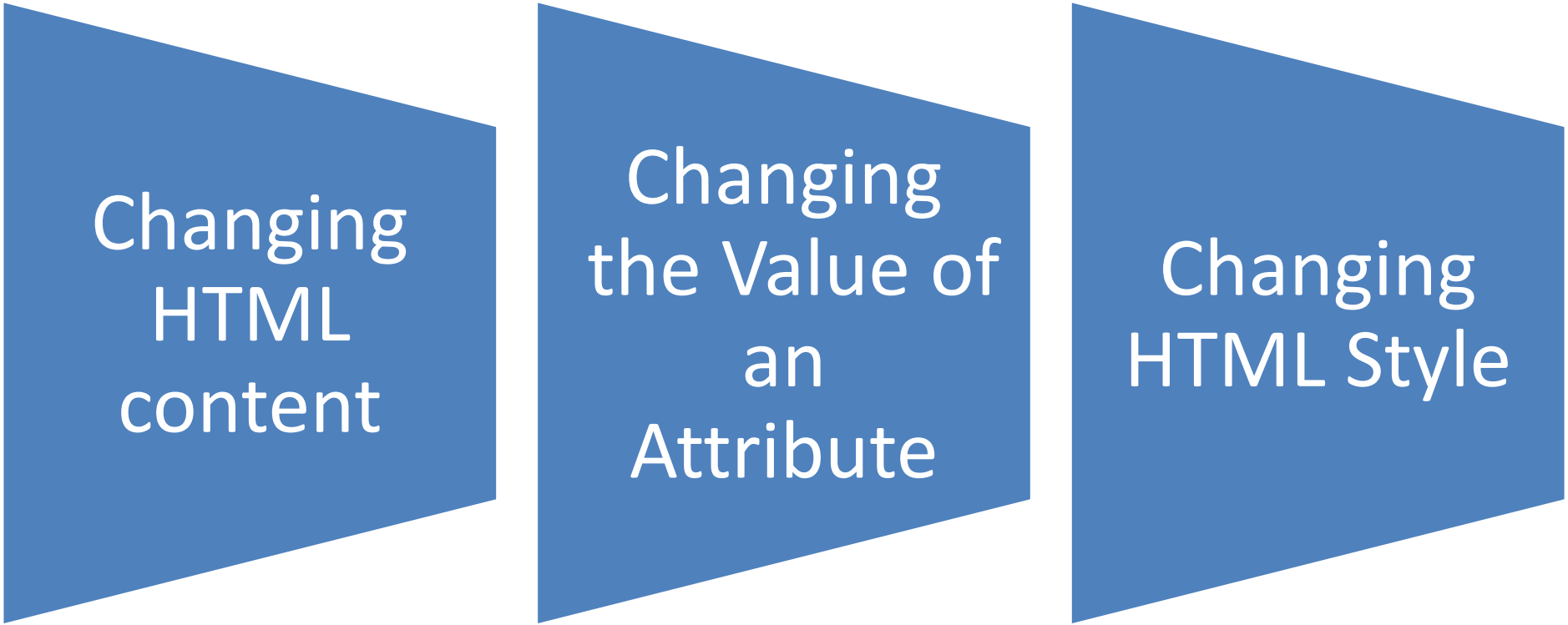
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p id="target">Sample Target</p>
        <p>Another Paragraph</p>
    </body>
</html>
```

Finding HTML Elements by Class Name

- `var p = document.getElementsByClassName('target');`
- Will find *all* elements with class 'target' on a page

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p class="target">Sample Target</p>
        <p class="target">Another Paragraph</p>
    </body>
</html>
```

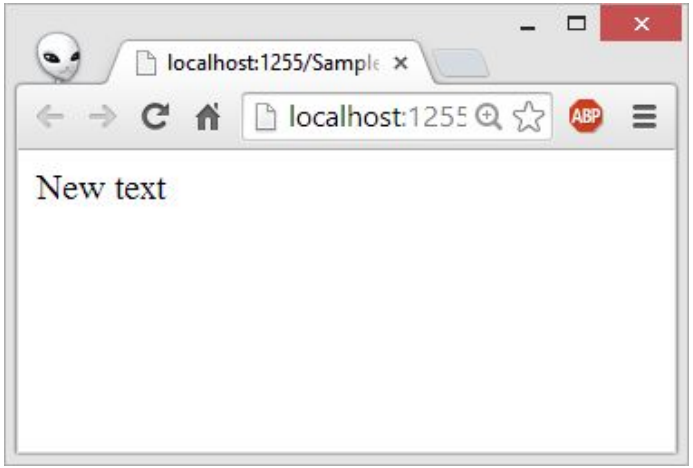
Changing HTML



Changing HTML Content

- `document.getElementById(id).innerHTML = New value`
- Will replace inner content of an element

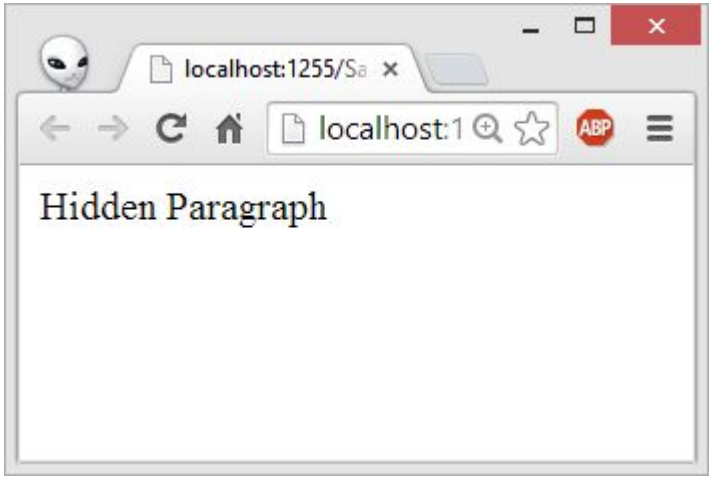
```
<html>
<body>
    <p id='target'>Old text</p>
    <script>
document.getElementById('target').innerHTML =  "New
text";
    </script>
</body>
</html>
```



Changing the Value of an Attribute

- `document.getElementById(id).attribute = New value`
- Will replace inner content of an element

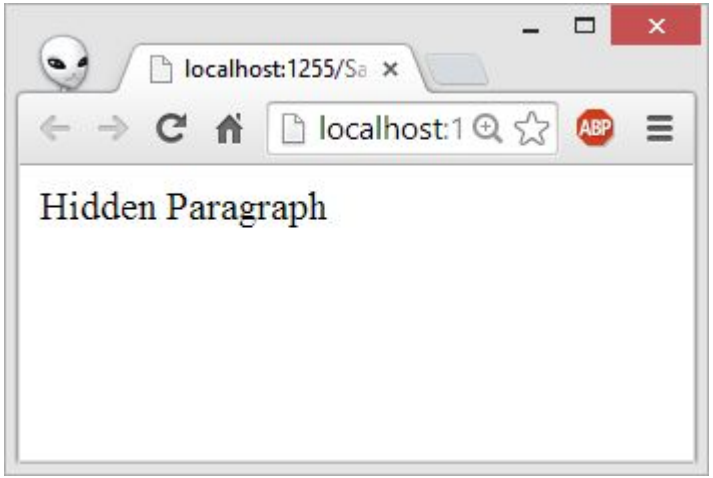
```
<html>
<body>
    <p id="target" hidden>Hidden Paragraph</p>
    <script>
        document.getElementById('target').hidden = '';
    </script>
</body>
</html>
```



Changing HTML Style

- `document.getElementById(id).style.property = New value`
- Will replace inner content of an element

```
<html>
<body>
    <p id="target" style="display: none">Hidden
Paragraph</p>
    <script>
document.getElementById('target').style.display = '';
    </script>
</body>
</html>
```

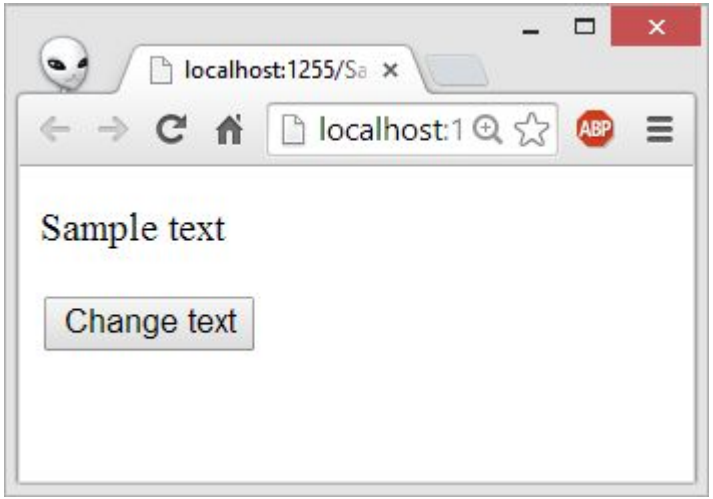


Using Events

- A JavaScript can be executed when an event occurs, examples of HTML events:
  - When a user clicks the mouse
  - When a user strokes a key
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted

Sample onclick() Event Handler

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function changeText() {
                document.getElementById('target').innerHTML =
                                'New text';
            }
        </script>
    </head>
    <body>
        <p id='target'>Sample text</p>
        <button type="button" onclick=changeText()>
            Change text
        </button>
    </body>
</html>
```



Cookies and Storages

What are Cookies?

- **Cookies** are data, stored in small text files, on client computer.
- There is a problem: 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:
  - When a user visits a web page, his ID can be stored in a cookie.
  - Next time the user visits the page, the cookie "remembers" his ID

Create a Cookie with JavaScript

- JavaScript can create, read, and delete cookies with the **document.cookie** property.
- A cookie can be created like this:

```
document.cookie = "ID=123456789";
```

- To save the cookie between browser sessions, we may add expiry date:

```
document.cookie = "ID=123456789; expires=Wed, 01 Jul 2015 12:00:00 GMT";
```

- By default, cookie belongs to the page that created it, path parameter allows to set what path the cookie belong to:

```
document.cookie = "ID=123456789; expires=Wed, 01 Jul 2015 12:00:00 GMT; path=/";
```

Read a Cookie

- To read a cookie:

```
var x = document.cookie;
```

- This code will return all cookies in one string in **`name=value`** pairs
- To find the value of one specified cookie, we must write a JavaScript function that searches for the cookie value in the cookie string.

Changing and Deleting Cookie

- Changing cookie is made same way as creating it:

```
document.cookie = "ID=123456789; expires=Wed, 01 Jul 2015 12:00:00 GMT; path=/";
```

- To delete a cookie we have to set **`expires`** parameter to a passed date:

```
document.cookie = "ID=123456789; expires=Thu, 01 Jan 1970 00:00:00 GMT";
```

Sample Function to Set a Cookie

```
function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
    var expires = "expires=" + d.toGMTString();
    document.cookie = cname + "=" + cvalue + "; " + expires;
}
```

- The parameters of the function above are the name of the cookie (cname), the value of the cookie (cvalue), and the number of days until the cookie should expire (exdays).
- The function sets a cookie by adding together the cookiename, the cookie value, and the expires string.

Sample Function to Get a Cookie

```
function getCookie(cname) {
    var name = cname + '=';
    var ca = document.cookie.split(';');
    for (var i = 0; i < ca.length; i++) {
        var c = ca[i].trim();
        if (c.indexOf(name) == 0) return c.substring(name.length, c.length);
    }
    return '';
}
```

- Take the cookiename as parameter (cname).
- Create a variable (name) with the text to search for (cname + '=').
- Split document.cookie on semicolons into an array called ca (ca = document.cookie.split(';')).
- Loop through the ca array (i=0;i<ca.length;i++), and read out each value trimmed (c=ca[i].trim()).
- If the cookie is found (c.indexOf(name) == 0), return the value of the cookie (c.substring(name.length,c.length).
- If the cookie is not found, return ''.

HTML5 Web Storage

- With HTML5, web pages can store data locally within the user's browser alternatively to cookies. Web Storage is more secure and faster. The data is not included with every server request, but used only when asked for.
- The data is stored in name/value pairs, and a web page can only access data stored by itself.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

HTML5 Web Storage Objects

HTML5 Web Storage provides two new objects for storing data on the client

- **window.localStorage** - stores data with no expiration date
- **code.sessionStorage** - stores data for one session (data is lost when the tab is closed)

Initial Check

- Before using web storage, check browser support for **`localStorage`** and **`sessionStorage`**:

```
if (typeof (Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // No Web Storage support
}
```

Using Storage Objects

- There are methods to use storage objects:
  - **`.setItem()`** – writes data
  - **`.getItem()`** – reads data
- Methods are identical for **`localStorage`** and **`sessionStorage`**

Sample Use of localStorage

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function countClicks() {
                if (localStorage.clickcount) {
                    localStorage.clickcount = Number(localStorage.clickcount) + 1;
                } else {
                    localStorage.clickcount = 1;
                }
                document.getElementById('target').innerHTML = localStorage.clickcount;
            }
        </script>
    </head>
    <body>
        <p>You have clicked the button <span id='target'></span> time(s).</p>
        <button type="button" onclick=countClicks()>
            Count
        </button>
    </body>
</html>
```

Useful links

Useful Links

- HTML DOM на сайті Wikipedia:
  http://en.wikipedia.org/wiki/Document_Object_Model
  W3Schools JavaScript HTML DOM:
  http://www.w3schools.com/js/js_htmldom.asp
- Специфікація HTML DOM на сайті W3C:
  http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/

Thank you!