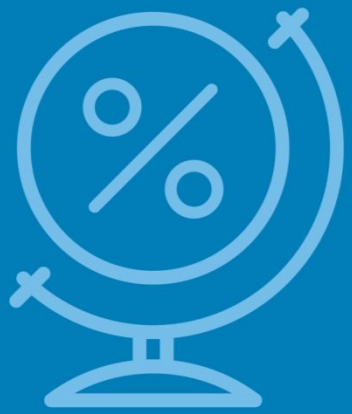
SoftServe | Empowering your Business through Software Development



Using jQuery. Creating Animations, Working with Graphics, and Accessing Data

Vyacheslav Koldovskyy
Last update: 09-Dec-2015

Agenda

- Introducing DOM
- Manipulating DOM with JavaScript
- Storages
- jQuery
- Useful links

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>
```

- <table>
  - <tbody>
    - <tr>
      - <td>
        - Some
      - <td>
        - Text
    - <tr>
      - <td>
        - in a
      - <td>
        - Table
- <head>
  - <title>
    - DOM Sample

Finding HTML Elements

1. By id: `var t = document.getElementById('target');`
   or use `target` as variable name (new in HTML5)
2. By element name: `var p = document.getElementsByTagName('p');`
3. By class name: `var p = document.getElementsByClassName('target');`

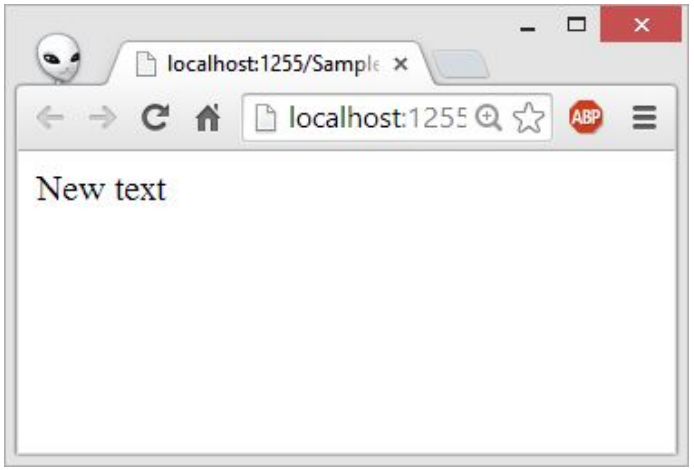
```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p id="target" class="target">Sample Target</p>
        <p class="target">Another Paragraph</p>
        <p>Last Paragraph</p>
    </body>
</html>
```

https://jsfiddle.net/koldovsky/6gb5zgrp/

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>
     target.innerHTML = 'New text';
    </script>
  </body>
</html>
```

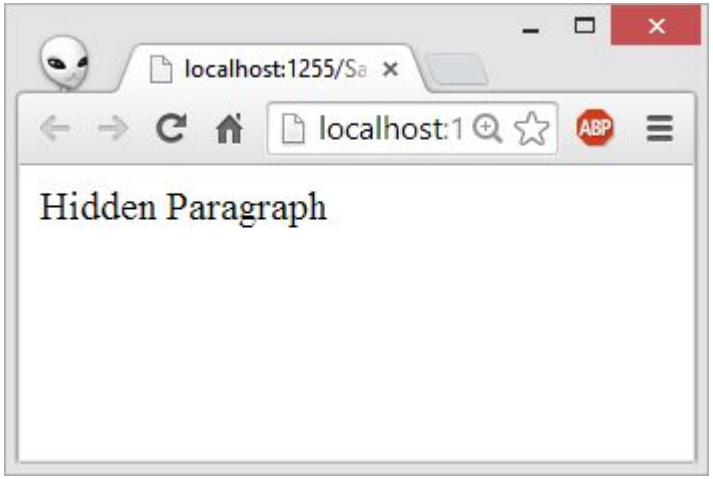


https://jsfiddle.net/koldovsky/8pdwvcy7/

Changing the Value of an Attribute

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

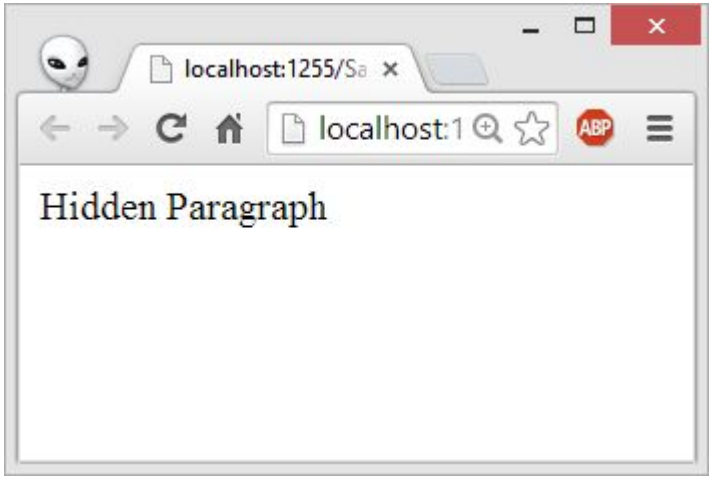
```
<html>
  <body>
    <p id="target" hidden>Hidden Paragraph</p>
    <script>
     target.hidden = '';
    </script>
  </body>
</html>
```



https://jsfiddle.net/koldovsky/yproq6t9/

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>
```

Accessing Form Elements

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Accessing form controls with JavaScript</title>
  </head>
  <body>
    <h1>Accessing form controls with JavaScript</h1>
    <form>
      <label for="inpt">Input string here: </label>
      <input id="inpt" type="text">
      <input id="btn" type="submit">
    </form>
    <p>Result: <span id="result"></span></p>
    <script src="script.js"></script>
  </body>
</html>
```


```javascript
btn.addEventListener('click', function(e) {
  var str = inpt.value;  // read value from input with id="str"
  var res = str.split('').reverse().join(''); // reverse symbols in a string
  result.innerHTML = res; // write result to span with is="result"
  e.preventDefault(); // prevent form submission
});
```

http://plnkr.co/edit/rGiBPab0hwWTg8fyMmJU?p=preview

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
- **window.sessionStorage** - stores data for one session (data is lost when the tab is closed)

Using Storage Objects

- There are methods to use storage objects:
  - **`localStorage['itemName'] = data`** // writes data
  - **`data = localStorage['itemName']`** // reads data
- Methods are identical for **`localStorage`** and **`sessionStorage`**

Sample

```
<script>
    function countClicks() {
        localStorage.clickcount = (localStorage.clickcount) ? Number(localStorage.clickcount) + 1: 1;
        update();
    }
    function update() {
        target.innerHTML = localStorage.clickcount || 0;
    }
</script>
<p>You have clicked the button < span id='target'></span> time(s).</p>
<button type="button" onclick="countClicks()">Count</button>
<script>
    update();
</script>
```

https://jsfiddle.net/koldovsky/d4j5zuk6/

jQuery: Write Less, Do More

- Query is a fast, small, and feature-rich JavaScript library.
- It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.
- jQuery is so popular that it often treated as an integral part of JavaScript but actually it isn't.

Linking jQuery

- jQuery is available at official website: http://jQuery.com
- You may download jQuery to local folder but it is recommended to use CDN link. There is an official CDN from jQuery but also there are alternate CDN's from Google, Microsoft and other companies.
- jQuery CDN link from http://code.jquery.com:

```
<script src='//code.jquery.com/jquery-2.1.1.min.js'></script>
```

- Important notes:
  - always to link to some specific version of jQuery but not to most recent version without number, as it may break your project because of future changes;
  - to accelerate page load use minified version of the library, you may link to uncompressed library while developing project to make code debugging possible;
  - CDN link shown above does not include protocol, it means that current protocol (http or https) will be used, but it will fail if html page loads locally, so it is better sometimes to add protocol to link

$ Alias

- jQuery uses '$' symbol as alias to object 'jQuery'
- This symbol available as a global variable
- You may reference library both by '$' alias and 'jQuery' variable name, but using '$' symbol is much more common and results in a cleaner and compact code

Basic jQuery Syntax

- jQuery syntax is focused on selecting html elements and performing some actions on the elements
- Basic syntax is: **$(*selector*).*action*()**
- Comments:
  - A **$** sign to define/access jQuery
  - A **(*selector*)** to 'query (or find)' HTML elements
  - A jQuery ***action*()** to be performed on the element(s)

window.onload

- Very common task in web programming is to run JS code when the browser finishes loading the document
- To achieve this task a developer may attach code to window.onload event:

```
window.onload = function() {
    alert( 'hello' );
}
```

- Unfortunately, the code doesn't run until all images are finished downloading, including banner ads

jQuery Ready Event

- To run code as soon as the document is ready to be manipulated, jQuery has a statement known as the ready event

```
$(document).ready(function() {
    alert('welcome');
});
```

- This approach is preferred over assigning window.onload event handler

- Alternative form (more preferred):

```
$(function() {
    alert('welcome');
});
```

Basic Selectors

- jQuery selectors work same as CSS selectors
  - To select all elements: **$('div')**
  - To select some specific element with id 'demo': **$('#demo')**
  - To select all elements with class 'demo': **$('.demo')**

- Example: hide all elements on target element click:

```
$(document).ready(function() {
    $('#target').click(function(){
        $('div').hide();
    });
});
```

CSS Selectors

- jQuery allows to use same selectors as used in CSS versions 1–3
- Incomplete list of some popular selectors:
  - 'X + Y': adjacent selector, select only the element that is immediately preceded by the former element;
  - 'X > Y': selects direct children of an element;
  - 'X ~ Y': sibling combinator, similar to 'X + Y' but allows selection of element 'Y' even if it is not immediately follows 'X' but just follows it;
  - 'X[title]': attribute selector, selects element 'X' if it has attribute 'title';
  - 'X[title=value]': attribute value selector, selects element 'X' if it has attribute 'title' with value 'value'
- Selector separated by comma treated as combination of selectors, selectors separated by space matched against descendants
- For complete list of selectors see jQuery manual: http://api.jquery.com/category/selectors/

Reading and Changing HTML Contents

- Method .html() allows to get or set HTML contents of elements
- When used as getter this method does not accept arguments and return contents of the first element in the set of matched elements:

Sample:
```
var htmlString = $('#mydiv').html();
```

- When used as setter this method sets the HTML contents of each element in the set of matched elements. Any content that was in that element is completely replaced by the new content.

Sample:
```
$('div').html('<p>Hello!</p>');
```

Reading and Changing Class Info

- jQuery allows to read, add or remove information about class for any element. This may be useful to change how elements shown based on predefined CSS styles assigned to the class and much more
- These methods are:
  - .addClass() – adds class:
    ```
    $('p').addClass('myClass');
    ```
  - .removeClass() – removes class:
    ```
    $('p').removeClass('myClass');
    ```
  - .hasClass() – checks if class is assigned:
    ```
    $('#myp').hasClass('myClass');
    ```
  - .toggleClass() – adds class if it is not assigned and vice versa:
    ```
    $('#myp').toggleClass('myClass');
    ```
- These methods, like other methods of jQuery may be chained to each other:

```
$('p').removeClass('myClass noClass').addClass('yourClass');
```

jQuery Event Basics

- It is very convenient to use jQuery to set up event-driven responses on page elements.
- These events are often triggered by the end user's interaction with the page, such as when text is entered into a form element or the mouse pointer is moved.
- In some cases, such as the page load and unload events, the browser itself will trigger the event.
- jQuery offers convenience methods for most native browser events. These methods are — including .click(), .focus(), .blur(), .change(), etc.

Setting Up Browser onclick Event

- Next example setups onclick event handler for all paragraphs on a page:

```
// Event setup using a convenience method
$( 'p' ).click(function() {
    console.log( 'You clicked a paragraph!' );
});
```

Setting Up Browser onclick Event With .on() Method

- Using .on() method we may setup any native browser event as well as custom events:

```
$( 'p' ).on( 'click', function() {
    console.log( 'click' );
});
```

- Or multiple events:

```
$( 'input' ).on('click change',  // bind listeners for
multiple events
    function() {
        console.log( 'An input was clicked or changed!' )
    }
);
```

Showing and Hiding Content

- jQuery can show or hide content instantaneously with .show() or .hide().
- When jQuery hides an element, it sets its CSS display property to none. This means the content will have zero width and height; it does not mean that the content will simply become transparent and leave an empty area on the page.

```
// Instantaneously hide all paragraphs
$( 'p' ).hide();
// Instantaneously show all divs that have the hidden
style class
$( 'div.hidden' ).show();
```

Animated Showing and Hiding

- jQuery can also show or hide content by means of animation effects.
- Simplest way is to pass argument of 'slow', 'normal', or 'fast' to .show() and .hide() methods:

```
// Slowly hide all paragraphs
$( 'p' ).hide( 'slow' );
```

- It is possible also to pass desired duration of animation in milliseconds:

```
// Show all divs that have the hidden style class over
0.5 sec
$( 'div.hidden' ).show( 500 );
```

Fade and Slide Animations

- jQuery uses combination of fade and slide effects while showing and hiding elements. It is possible to use this effects separately.
- Slide animation:

```
// Hide all paragraphs using a slide up animation over 0.8 seconds
$( 'p' ).slideUp( 800 );
// Show all hidden divs using a slide down animation over 0.6 seconds
$( 'div.hidden' ).slideDown( 600 );
```

- Fade animation:

```
// Hide all paragraphs using a fade out animation over 1.5 seconds
$( 'p' ).fadeOut( 1500 );
// Show all hidden divs using a fade in animation over 0.75 seconds
$( 'div.hidden' ).fadeIn( 750 );
```

Changing Display Based on Current Visibility State

- jQuery can also let you change a content's visibility based on its current visibility state. Method .toggle() will show content that is currently hidden and hide content that is currently visible. You can pass the same arguments to .toggle() as you pass to any of the effects methods above.

```
// Instantaneously toggle the display of all paragraphs
$( 'p' ).toggle();
// Slowly toggle the display of all images
$( 'img' ).toggle( 'slow' );
```

- There are also .slideToggle() and .fadeToggle() methods:

```
// Toggle the display of all ordered lists over 1 second using
slide up/down
$( 'ol' ).slideToggle( 1000 );
// Toggle the display of all blockquotes over 0.4 seconds using
fade in/out
$( 'blockquote' ).fadeToggle( 400 );
```

Doing Something After an Animation Completes

- If we want to do something after animation completes, we can't use such code because it won't wait for completion:

```
// Incorrect: Fade in all hidden paragraphs; then add a style class
to them $( 'p.hidden' ).fadeIn( 750 ).addClass( 'lookAtMe' );
```

- To defer an action until after an animation has run to completion, you need to use an animation callback function. You can specify your animation callback as the second argument passed to any of the animation methods discussed above. For the code snippet above, we can implement a callback as follows:

```
// Fade in all hidden paragraphs; then add a style class to them
$( 'p.hidden' ).fadeIn( 750, function() {
    // this = DOM element which has just finished being animated
    $( this ).addClass( 'lookAtMe' );
});
```

Practice Task

Advanced Topics

Cookies

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

- 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.

```
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;
}
```

Sample Function to Get a Cookie

- 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 ''.

```
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 '';
}
```

Setting Up Events to Run Only Once

- Sometimes you need a particular handler to run only once – after that, you may want no handler to run, or you may want a different handler to run.
- jQuery provides the .one() method for this purpose:

```
// Switching handlers using the `$.fn.one` method
$( 'p' ).one( 'click', firstClick );
function firstClick() {
    console.log( 'You just clicked this for the first time!' );
}
```

Tearing Down Event Listeners

- To remove an event listener, you use the .off() method and pass in the event type to off.

```
// Tearing down all click handlers on a selection
$( 'p' ).off( 'click' );
```

- If you attached a named function to the event, then you can isolate the event tear down to just that named function by passing it as the second argument.

Reading and Changing Styles

- jQuery provides method `.css()` that allows to read or set style data.
- If this method used as getter, it returns CSS property value of a first element that matches selector.
  Syntax: `.css('propertyName')`
  Sample: `var color = $('#myDiv').css('background-color');`
- If this method used as setter, it sets CSS property values for all elements that match selector.
  Syntax:
  - `.css(propertyName, value);  // value - a value to set for the property`
  - `.css(propertyName, function); // function - a function returning // the value to set`
  - `.css(properties);  // properties - an object of // property-value pairs to set`

Using Different Handlers for Multiple Events

- In the example below shown how to use different event handlers for multiple events:

```
// Binding multiple events with different
handlers
$( 'p' ).on({
    'click': function() { console.log( 'clicked!'
); },
    'mouseover': function() { console.log(
'hovered!' ); }
});
```

Contacts

Europe Headquarters

52 V. Velykoho Str.
Lviv 79053, Ukraine

Tel: +380-32-240-9090
Fax: +380-32-240-9080

E-mail: info@softserveinc.com
Website: **www.softserveinc.com**

US Headquarters

12800 University Drive, Suite 250
Fort Myers, FL 33907, USA

Tel: 239-690-3111
Fax: 239-690-3116

Thank You!