Тема 4. Основи JQuery

4.1 Що таке jQuery, основні поняття і можливості

jQuery - це JavaScript -бібліотека, що фокусується на взаємодії JavaScript, HTML і CSS.

Що уміє jQuery

- Звертатися до будь-якого елементу DOM (об'єктній моделі документу) і не лише звертатися, але і маніпулювати ними.
- Працювати з подіями.
- Легко здійснювати різні візуальні ефекти.
- Працювати з AJAX (дуже корисна технологія, що дозволяє спілкуватися з сервером без перезавантаження сторінки, але доки ми її чіпати не будемо).
- Має величезну кількість JavaScript плагінів, призначених для створення елементів призначених для користувача інтерфейсів.

Як це працює

Спочатку потрібно викачати саму бібліотеку на сайті розробників, при необхідності розархівувати і перенести її (бібліотеку) в ту ж теку, де лежать наші html-сторінки (це необов'язково, але адреси для усіх подальших прикладів будуть написані, виходячи з такої структури).

Тепер нам потрібно підключити jQuery до html -сторніки. Для цього, як ви пам'ятаєте, в html існує тег *<script>*, який і відповідає за підключення зовнішніх файлів скриптів (html - урок 2). Додамо цей тег в html -сторінку (так само ми підключали сторінку script.js з функціями js) :

```
<html>
  <head>
    <title>jQuery</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery-3.5.1.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
  </body>
</html>
```

* Зверніть увагу на ім'я файлу. Тут використовується бібліотека jquery-3.5.1.js (станом на 09.2020р), але Ви можете викачати свіжішу версію

Отже, у нас в теці лежать html -сторінка, сторінка із стилями (style.css), сторінка з js -функціями (script.js) і бібліотека jQuery (jquery-3.5.1.js). Давайте зробимо ось такий приклад (поклацуйте по кнопках) :

Приблизно синтаксис оператора jQuery можна представити таким чином:

$(“Селектор”).дія(“Властивості_дії”);

де *селектор* - елемент або елементи, з якими ми що-небудь робитимемо.

***дія* - що саме ми робитимемо з вибраними елементами. Ми можемо додати який-небудь**

ефект, css -стиль, змінити html -код і так далі. Тут же можуть бути вказані які-небудь події.

властивості дії **- якщо вони передбачені дією.**

4.2 JQuery селектори

Селектори - це елементи, з якими ми хочемо що-небудь зробити. Давайте подивимося, як їх можна задавати:

- *#id* - вибирає єдиний елемент з переданим ідентифікатором (id).
- Приклад:

```
$("#lok").css("border", "1 px solid red");
```

Цей оператор вибере елемент з id="lok" і додасть йому css -стиль, в даному випадку рамку, шириною в 1 піксел, суцільну і червону.

- *.class* - вибирає усі елементи з переданим класом.
- Приклад:

```
$(".lok").css("border", "1 px solid red");
```

Цей оператор вибере усі елементи з class="lok" і додасть їм css -стиль, в даному випадку рамку, шириною в 1 піксел, суцільну і червону.

- *elements* - вибирає усі елементи з вибраним ім'ям.
- Приклад:

```
$("div").css("border", "1 px solid red");
```

Цей оператор вибере усі div -и і обведе їх в рамки.

- * - вибирає усі елементи, включаючи head і body.
- Приклад:

```

$("*").css("border", "1 px solid red");
```

Цей оператор додасть рамку до усіх елементів.

```
$("*", document.body).css("border", "1 px solid red");
```

Цей оператор додасть рамку до усіх елементів в тегах body.

- *selector1, ..., selectorN* - вибирає усі елементи, передані в селекторах.
- Приклад:

```

$("div, span, p.lok").css("border", "1 px solid red");
```

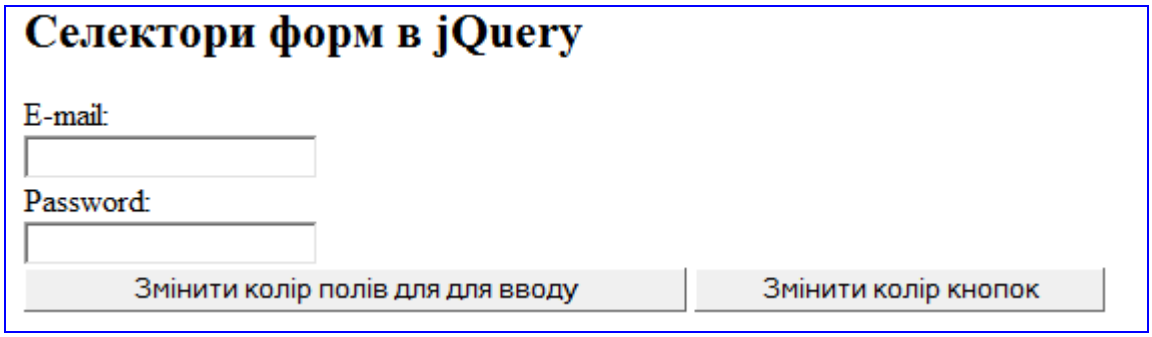
Цей оператор додасть рамку до усіх div, span, і абзацам, що мають клас lok. Таким чином, через кому можна вказати будь-які селектори.

Селектори форм

Спочатку просто їх перерахуємо:

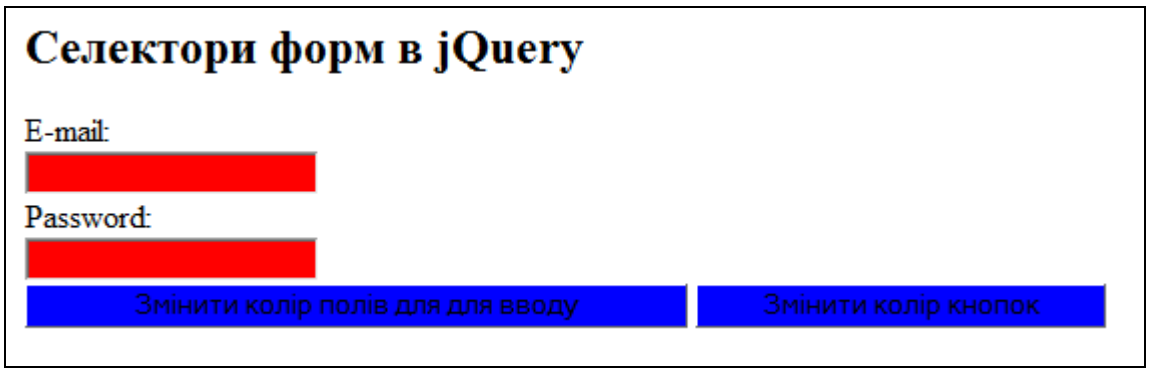
- *:input* - вибираються усі елементи input.
- *:text* - вибираються усі елементи input типу text.
- *:password* - вибираються усі елементи input типу password.
- *:radio* - вибираються усі елементи input типу radio.
- *:checkbox* - вибираються усі елементи input типу checkbox.
- *:submit* - вибираються усі елементи input типу submit.
- *:reset* - вибираються усі елементи input типу reset.
- *:button* - вибираються усі елементи input типу button.
- *:image* - вибираються усі елементи input типу image.
- *:file* - вибираються усі елементи input типу file.
- *:hidden* - вибираються усі елементи input типу hidden або просто приховані.

Давайте зробимо, наприклад, ось таку форму (поклацуйте по кнопках) :



Код html -сторінки :

```
<html>
  <head>
    <title>jQuery</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery-3.5.1.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
     <h2>Селектори форм в jQuery</h2>
      <form>
          <label>E-mail:</label><br>
          <input type="text"  name="mail"><br>
          <label>Password:</label><br>
          <input type="password" name="pas"><br>
          <input  type="button"  id="b1"  value="Змінити  колір  полів
для для введенню" onclick="addColor 1();">
          <input  type="button"  id="b2"  value="Змінити  колір  кнопок"
onclick="addColor 2();">
       </form>
  </body>
</html>
```



Самі функції:

```
function addColor1(){
  $(":text,:password").css("background", "red");
}
function addColor2(){
  $(":button").css("background", "blue");
}
```

Ось так усе просто! До речі, якби ми захотіли поміняти колір тільки у однієї кнопки, наприклад, у першої, то ми зверталися б до неї по її id:

```
function addColor2(){
  $("#b1").css("background", "blue");
}
```

Тобто, селектори форм допомагають здійснювати масові дії з елементами форми.

4.3 Фільтри

:first - вибирає перший елемент відповідного селектора.

Приклад:

```
$("tr:first").css("font-style", "italic");
```

Ця інструкція напише курсивом перший рядок таблиці.

:last - вибирає останній елемент відповідного селектора.

Приклад:

```
$("tr:last").css("font-style", "italic");
```

Ця інструкція напише курсивом останній рядок таблиці.

:even - вибирає парні елементи, починаючи з нуля.

Приклад:

```
$("tr:even").css("font-style", "italic");
```

Ця інструкція напише курсивом усі парні рядки таблиці (оскільки нумерація йде з нуля, то зорово здається, що непарні рядки).

:odd - вибирає непарні елементи, починаючи з нуля.

Приклад:

```
$("tr:odd").css("font-style", "italic");
```

Ця інструкція напише курсивом усі непарні рядки таблиці (оскільки нумерація йде з нуля, то зорово здається, що парні рядки).

:eq(index) - вибирає елемент по його індексу (починаючи з нуля).

Приклад:

```
$("td:eq(2)").css("font-style", "italic");
```

Ця інструкція напише курсивом текст другого елементу таблиці (оскільки нумерація йде з нуля, то зорово здається, що третього осередку).

:gt(index) - вибирає елементи з індексом більше за вказане.

Приклад:

```
$("td:gt(2)").css("font-style", "italic");
```

Ця інструкція напише курсивом текст в усіх елементах таблиці, починаючи з другої (оскільки нумерація йде з нуля, то зорово здається, що з третього осередку).

:lt(index) - вибирає елементи з індексом менше за вказане.

Приклад:

```
$("td:lt(4)").css("font-style", "italic");
```

Ця інструкція напише курсивом текст в перших п'яти елементах таблиці.

4.4. Методи для роботи з CSS -стилями

У jQuery є три категорії методів : одні маніпулюють з елементами, відповідними за шаблоном; другі повертають значення елементу, а треті змінюють самі елементи.

Отже, щоб додати якому-небудь елементу стиль, необхідно скористатися наступним методом:

.css(name, value)

Приклад:

```
$("div").css("border", "1px solid blue");
```

Ця інструкція обведе div синьою рамкою.

Як параметри тут використовуються назви і значення, застосовні в CSS : background, border, font-style, color і так далі.

Якщо необхідно задати для елементу декілька CSS-правил, то краще використати наступну конструкцію:

.css({properties})

Приклад:

```
$("div").css({
        border:"1px solid blue",
        fontWeight:"bolder",
        backgroundColor:"red"
        });
```

Ця інструкція обведе div синьою рамкою, зробить фон червоним, а текст - жирним.

Зверніть увагу, що для складених властивостей CSS ніби font - weight і background - color використовуються иэ еквіваленти з JS: fontWeight, backgroundColor і так далі

Перерахуємо інші методи для роботи із стилями:

- *.addClass(class)*
-
- Приклад:
-
-

```
$("p:last").addClass("main");
```

Ця інструкція додасть клас main до останнього елементу параграфа.

- *.removeClass(class)*
-
- Приклад:
-

```
$("p:even").removeClass("main");
```

Ця інструкція видалить клас main з усіх парних параграфів.

- *.toggleClass(class)*
-
- Приклад:
-
-

```
$("p").toggleClass("main");
```

Ця інструкція видалить клас main з усіх параграфів, якщо він є присутнім. І додасть цей клас, якщо він відсутній.

- *.offset()*
-
- Приклад:
-
-

```
var offDiv=$("div").offset();
```

Ця інструкція дозволяє отримати відступи ліворуч і згори для елементу. Щоб отримати значення конкретної властивості, треба використати наступні властивості: offset.left для відступу ліворуч і offset.top - для відступу згори.

- *.height(value)*
-
- Приклад:
-
- ```
  $("div").height();
  $("div").height(200);
  ```

Ця інструкція дозволяє отримати (перший рядок) і задати (другий рядок) висоту елементу.

- *.width(value)*
-
- Приклад:
-
- ```
  $("div").width();
  $("div").width(200);
  ```

Ця інструкція дозволяє отримати (перший рядок) і задати (другий рядок) ширину елементу.

Ще приклад:

```
var widDiv=$("div").width();

$("div.fir").width(300);
```

Перший рядок запише в змінну widDiv значення ширини першого div -а. Друга інструкція задасть div -ам класу fir ширину в 300 пікселів.

Це дуже цікава особливість методів jQuery : вони використовуються, як для завдання параметрів (коли приймаються 2 аргументи), так і для набуття значень цих параметрів (якщо передається один аргумент).

4. 5. Методи для роботи з html

Спочатку розглянемо основні методи роботи з html, а потім вже методи для маніпуляції з цими елементами.
Отже, основні методи:

- *html(val) - додає html -код у вибрані елементи.*
- Приклад:
-
- ```
  $("div.sp").html("Привіт");
  ```

  Ця інструкція додасть в усі div -и з класом sp, span з текстом "Привіт".

- *text() / val()* - повертає текстовий вміст елементу / значення елементу.
- Приклад:
-
- ```
  $("p").text();
  ```

  Ця інструкція поверне текст з параграфа.

- *text(val) / val(val)* - встановить текст для елементу / значення для елементу.
- Приклад:
-
- ```
  $("p").text("Привіт"!);
  ```

  Ця інструкція в параграфі напише слово "Привіт"!.

Давайте для прикладу, зробимо так, щоб при натисненні по кнопці "Повторити", текст з першого абзацу дублювався в другий абзац.

Текст першого абзацу!

Html -код:

```
<html>
  <head>
    <title>jQuery html</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery - 1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
      Текст першого абзацу!

      <input type="button" value="Повторити"! onclick="addHtml();">
  </body>
</html>
```

Фукція виглядатиме так:

```
function addHtml(){
    var tp1=$("#p1").text();
    $("#p2").text(tp1);
```
```

```
}
```

Тобто спочатку в змінну tp1 ми запишемо текстовий вміст першого абзацу (перший рядок), а потім передамо його в другий абзац (другий рядок).

Тепер давайте розглянемо методи, які дозволяють маніпулювати елементами, що вставляються.

- *append(content)* - додає content всередину усіх вибраних елементів ПІСЛЯ існуючого контенту.
- Приклад:
-
-
```
$("p").append("<b>Привіт</b>");
```

  Ця інструкція додасть в кінець параграфа текст "Привіт" жирним шрифтом.

- *appendTo(content)* - зворотна операція, додає в content усі вибрані елементи.
- Приклад:
-
-
```
$("p").appendTo("#main");
```

  Ця інструкція додасть параграф в кінець елементу з ідентифікатором "main".

  Іншими словами:

  $(A).append(B) - додасть У в кінець А.

  $(A).appendTo(B) - додасть А в кінець В.

- *prepend(content)* - додає content всередину усіх вибраних елементів ДО існуючого контенту.
- Приклад:
-
-
```
$("p").prepend("<b>Привіт</b>");
```

  Ця інструкція вставить в початок параграфа текст "Привіт" жирним шрифтом.

- *prependTo(content)* - зворотна операція, додає В content усі вибрані елементи.
- Приклад:
-

- ```
  $("p").prependTo("#main");
  ```

Ця інструкція вставить параграф в початок елементу з ідентифікатором "main".

Іншими словами:

$(A).prepend(B) - додасть У в початок А.

$(A).prependTo(B) - додасть А в початок В.

- *after(content)* - додає content ПІСЛЯ усіх вибраних елементів (зверніть увагу ПІСЛЯ елементу, а не в кінець елементу, як у випадку з append.
- Приклад:
-
- ```
  $("p").after("<b>Привіт</b>");
  ```

Ця інструкція вставить після параграфа текст "Привіт" жирним шрифтом.

- *befor(content)* - додає content ДО усіх вибраних елементів (зверніть увагу ДО елементу, а не в початок елементу, як у випадку з prepend.
- Приклад:
-
- ```
  $("p").befor("Привіт");
  ```

Ця інструкція вставить перед параграфом текст "Привіт" жирним шрифтом.

- *insertAfter(content)* - вставляє усі вибрані елементи ПІСЛЯ content -а .
- Приклад:
-
- ```
  $("p").insertAfter("#main");
  ```

Ця інструкція вставить параграф після елементу з ідентифікатором "main".

Іншими словами:

$(A).after(B) - вставить В послові А.

$(A).insertAfter(B) - вставить А після В.

- *insertBefore(content)* - вставляє усі вибрані елементи ПЕРЕД content -му .
- Приклад:
-
-
```
$("p").insertBefore("#main");
```

Ця інструкція вставить параграф перед елементом з ідентифікатором "main".

Іншими словами:

$(A).before(B) - вставить B перед A.

$(A).insertBefore(B) - вставить A перед B.

Розглянемо приклад. Давайте зробимо так, щоб при натисненні на кнопку "Додати" в червоному прямокутнику з'являлися жовті квадратики (один клік - один квадратик).

Код html -страницы складається з одного div -а і однієї кнопки :

```
<html>
  <head>
    <title>jQuery html</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery - 1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <div id="red_sk"></div>
    <input type="button" value="Додати"! onclick="addYellowSquare();">
  </body>
</html>
```

На сторінці style.css як завжди визначуваний стилі:

```
#red_sk{
    width :700 px;
    height :100 px;
    background: red;
}
.yellow_square{
    float: left;
    width :80 px;
    height :80 px;
    background: yellow;
    margin :5 px;
}
```

Ну а код самої функції зовсім короткий:

```
function addYellowSquare(){
    $("#red_sk").prepend("<div class='yellow_square'></div>");
}
```

Ось так усе просто. Але продовжимо розглядати самі методи:

- *wrap(html)* - обертає кожен елемент в елемент обгортку.
- Приклад:
-
-
```
$("p").wrap("<span></span>");
```

Ця інструкція оберне кожен параграф span -му.

- *wrapAll(html)* - обертає усі елементи в ОДИН елемент обгортку.
- Приклад:
-
-
```
$("p").wrapAll("<span></span>");
```

Ця інструкція оберне усі параграфи одним span -му.

- *wrapInner(html)* - обертає внутрішньо зміст кожного елементу.
- Приклад:
-
-
```
$("p").wrapInner("<em></em>");
```

Ця інструкція виділить вміст кожного параграфа курсивом.

- *replaceWith(content)* - заміщає одні елементи іншими.
- Приклад:
-
-
```
$("span").replaceWith("<div></div>");
```

Ця інструкція замінить усе span -ы div -ами.

- *replaceAll(selector)* - зворотна операція, тобто усе selector -ы будуть поміняні на елементи.
- Приклад:
-
-
```
$("span").replaceAll("<div></div>");
```

Ця інструкція замінить усе div -ы span -ами.

- *empty()* - видаляє з елементу усі вузли-нащадки, включаючи текст.
- Приклад:
-
- ```
  $("span").empty();
  ```

  Ця інструкція видалить увесь вміст span -ов.

- *remove()* - видаляє самі елементи.
- Приклад:
-
- ```
  $("span").remove();
  ```

  Ця інструкція видалить усе span -ы.

- *clone()* - клонує елементи.
- Приклад:
-
- ```
  $("b").clone().append("div");
  ```

  Ця інструкція візьме елементи, виділені жирним і додасть їх в div.

У останньому прикладі, ми торкнулися ще одного аспекту jQuery, а саме підтримка цією бібліотекою ланцюжка викликів. Т. е. ви можете викликати декілька методів в одному рядку, відділяючи їх точкою. Наскільки це зручно? Залежить від Вас. Якщо це Вас не плутає, то використайте на здоров'ї. Надалі ми часто використовуватимемо такий вид запису.

Наостанок, давайте зробимо ще один приклад. Нехай у нас є деякий список, і ми хочемо дозволити користувачам додавати в нього пункти:
Наші міста:

- Архангельськ
- Москва
- Іркутськ

Доповніть список:
Код html -сторніки :

```
<html>
  <head>
    <title>jQuery html</title>
    <link rel="stylesheet" type="text/css" href="style.css">
```
```

```
    <script type="text/javascript" src="jquery - 1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    Наші міста:
    <ul id="jq">
    <li>Архангельськ</li>
    <li>Москва</li>
    <li>Іркутськ</li>
    </ul>
    Доповніть список:
    <input type="text" id="user_text" size="20" maxlength="50">
    <input type="button" value="Додати" onclick="addSpisok();">
  </body>
</html>
```

А код функції :

```
function addSpisok(){
    var jq=$('#user_text').val();
    $('#jq').append('<li>'+jq+'</li>');
}
```

4.5 Методи для роботи з атрибутами і фільтрами.

attr(name) - забезпечує доступ до значення вказаного атрибуту першого елементу в наборі.

- Приклад:
-
-
```
var a=$("i").attr("title");
$("div").text(a);
```

Ця інструкція знайде перший елемент в тегах і, знайде атрибут title цього елементу і додасть його значення в div.

- *attr(properties)* - встановить атрибути в усіх відібраних елементах.
- Приклад:
-
-
```
$("img").attr({src :" images/pict.gif",alt :"рисунок"});
```

Ця інструкція знайде усі картинки і встановить їм відповідні атрибути.

- *attr(key, value)* - встановить значення (value) атрибуту (key) для усіх відібраних елементів.
- Приклад:
-
-
```
$("button").attr("disabled", "disabled");
```

Ця інструкція встановить для усіх кнопок значення "disabled" атрибуту "disabled".

- *removeAttr(name)* - видалить вказаний атрибут в усіх елементів.
- Приклад:
-
-
```
$("img").removeAttr("alt");
```

Ця інструкція видалить атрибут "alt" у усіх картинок.

- *hasClass(class)* - повертає істину (true), якщо вказаний клас є присутнім хоч би в одному з елементів.
- Приклад:
-
-
```
if ($("p: last").hasClass("selected"))
    $(this).css("background", "blue");
```

Ця інструкція зробить колір фону останнього абзацу синім, якщо у нього клас "selected".

- *filter(expr)* - обмежує елементи, до яких слід що-небудь застосувати.
- Приклад:
-
-
```
$("p").filter(".blue").css("background", "blue");
```

Ця інструкція зробить колір фону синім, тільки у тих абзаців, які мають клас "blue".

- *not(expr)* - означає елементи, до яких не слід що-небудь застосувати.
- Приклад:
-
-
```
$("p").not(".blue").css("background", "blue");
```

Ця інструкція зробить колір фону синім у усіх параграфів окрім тих, що мають клас "blue".

- *is(expr)* - повертає істину (true), якщо хоч би один з елементів відповідає вираженню.
- Приклад:

```
if ($(this).is(": last - child"))
    $("p").text("останній");
```

Ця інструкція додасть в параграф текст "останній", тільки якщо елемент останній, що перевіряється.

- *slice(start, end)* - відбирає піднабір з набору елементів.
- Приклад:

```
$("p").slice(1,4).css("background", "blue");
```

Ця інструкція зробить колір фону синім у усіх параграфів з 1 по 4.

4.6 Методи по обробці подій.

ready(fn) - призначає функцію, яка виконуватиметься, коли документ готовий до роботи.

- Приклад:

```
$(document).ready(init);
```

Ця інструкція говорить браузеру, що відразу після завантаження документу повинна спрацювати функція init.

- *bind(type, fn)* - зв'язує обробник події з самою подією.
- Приклад:

```
$(div).bind('click ', init);
```

Ця інструкція говорить браузеру, що при клацанні по div -у повинна спрацювати функція init (тут click - подія, а init - функція, обробник події).

Давайте розглянемо приклад. Створимо список уроків по jQuery і зробимо так, щоб при клацанні по якому-небудь з них, в полі нижче з'являвся опис вибраного уроку :
Урок:

Опис:

Отже створимо сам список і поле для виведення описів на html -странице:

```
<html>
  <head>
    <title>jQuery</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery - 1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
      Урок: <select id="lesson">
          <option  value='1'>Урок 1</option>
          <option  value='2'>Урок 2</option>
          <option  value='3'>Урок 3</option>
      </select><br><br>
      Опис: <span id="desc"></span>
  </body>
</html>
```

Отже, ми могли б кожному пункту списку задати обробник події - функцію desc, яка і задаватиме описи залежно від вибраного пункту. Але jQuery дозволяє зробити це ще простіше.

На сторінці script.js нам досить призначити функцію, яка спрацьовуватиме кожного разу, коли документ готовий до роботи (назвемо її init). Сама ж функція init викликатиме функцію desc кожного разу, коли вибраний який-небудь пункт списку (тобто настає подія change) :

```
        $(document).ready(init);

        function init(){
          $('#lesson').bind('change ', desc);
        }

        function desc(){
        var op=$('#lesson').val();
        switch (op)
         {
         case '1 ': $('#desc').text('Перший урок по jQuery знайомить з основними
поняттями
                                                                  і можливостями цієї
бібліотеки.'); break;
         case '2 ': $('#desc').text('Другий урок по jQuery знайомить з таким
поняттям, як селекторы.');

break;
         case '3 ': $('#desc').text('Третій урок по jQuery знайомить з таким
поняттям, як фільтри.');

break;
         }
        }
```

От і все. Можете викачати початковий код прикладу і подивитися, як він працює. Продовжуємо вивчати методи по обробці подій :

- *one(type, fn)* - зв'язує обробник події з самою подією, але виконується він тільки один раз.
- Приклад:
-
-
```
$(div).one('click ', init);
```

Ця інструкція говорить браузеру, що при клацанні по div -у повинна спрацювати функція init, але інструкції цієї функції будуть виконані тільки один раз.

Якщо в нашому прикладі зі списком уроків ми замінимо bind на one, то функція desc спрацює тільки один раз, тобто при першому виборі уроку зі списку ми побачимо опис, але далі, скільки б ми не клацали за пунктами списку, опис мінятися не буде.

- *hover(over, out)* - коли покажчик миші знаходиться над об'єктом, спрацьовує функція over, а коли покажчик миші виходить за об'єкт, то функція out.
- Приклад:

  перший абзац

  другий абзац

  третій абзац

Отже, html -код:

```
<html>
  <head>
    <title>jQuery</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery - 1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
      <p>перший абзац</p>
      <p>другий абзац</p>
      <p>третій абзац</p>
  </body>
</html>
```

Код функції :

```
$(document).ready(init);

function init(){
$('p').hover(
  function(){$(this).css("background-color", "blue");},
  function(){$(this).css("background-color", "white");}
  );
}
```

Таким чином, коли покажчик миші над абзацом, його фон - синій, а при виході - білий.

Зверніть увагу, тут ми застосували короткий запис, тобто використали функції без назв, а могли розписати і так:

```
$(document).ready(init);

function init(){
 $('p').hover(hOver, hOut);
}

function hOver(){
 $(this).css("background-color", "blue");
}

function hOut(){
 $(this).css("background-color", "white");
}
```

Вибирайте ту форму запису, який вам більше подобається.

- *toggle(fn1, fn2, ...fn)* - перемикач між функціями. Клацання по елементу викликає функцію fn1, повторне клацання - функцію fn2, третє клацання - функцію fn3 і так далі
- Приклад:

```
$('p').toggle(
   function(){$(this).css("background-color", "blue");},
   function(){$(this).css("background-color", "white");}
);
```

Ця інструкція говорить браузеру, що при клацанні по абзацу, його фон стане синім, при повторному клацанні - білим.

- *click(fn)* - функція fn зв'язується з подією click.
- Приклад:

```
$('div').click(
   function(){$(this).css("background-color", "blue");}
);
```

Ця інструкція говорить браузеру, що при клацанні по div -у, його фон стане синім.

click() - емулюється виникнення події click.

Приклад:

```
$('div').click(
   function(){$(this).css("background-color", "blue");}
);
$('div: first').click();
```

Ця інструкція говорить браузеру, що при клацанні по div -у його фон стане синім і емулюватиме цю подію для першого div -а.

От як це виглядає на практиці:

перший div

другий div

третій div

Іншими словами, виклик методу з функцією в якості аргументу, наприклад click(fn), призначає обробник події, виклик без аргументу, наприклад click(), емулює виникнення цієї події.

Аналогічні методи визначені і для інших подій, підтримуваних javascript, наприклад: blur(), blur(fn), focus(), focus(fn) і так далі. Якщо забули, які події бувають і коли настає те або інша подія, то подивитеся тут.

4.7 Візуальні ефекти

Методи видимості - hide(), show() і toggle()

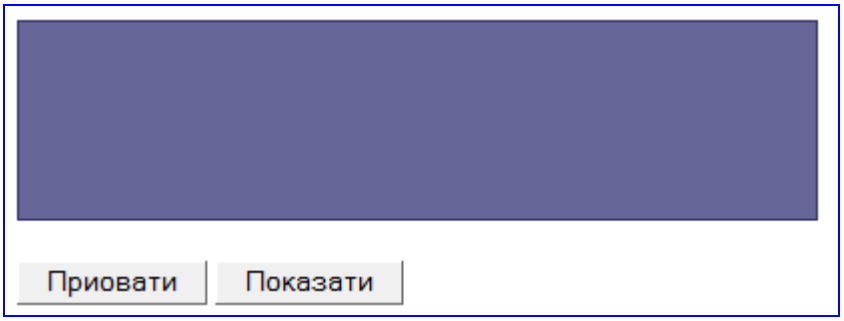
Ці методи відповідають за видимість елементів, працюють за принципом приховати - показати:

hide() - приховати

show() - показати,

toggle() - показати, якщо приховані і приховати, якщо видимі.

Приклад:



Html -код:

```
<html>
 <head>
  <title>jQuery ефекти</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <script type="text/javascript" src="jquery-3.5.1.js"></script>
  <script type="text/javascript" src="script.js"></script>
 </head>
 <body>
  <div id="les 8_ex1"></div>
  <input type="button" value="Приховати" onclick="hideDiv();">
  <input type="button" value="Показати" onclick="showDiv();">
 </body>
</html>
```

Код функцій (на сторінці script.js) :

```
function hideDiv(){
    $('#les8_ex1').hide();
}
function showDiv(){
    $('#les8_ex1').show();
}
```

Ну і стилі на style.css в принципі на ваш розсуд:

```
#les8_ex1{
```

```
        width :400 px;
        height :100 px;
        background:#666699;
        border :1 px solid #333366;
        margin - bottom :20 px;
}
```

Сторінка style.css тут приведена не випадково. Вся річ у тому, що приховати елемент можна і за допомогою таблиці стилів, а не методом hide().

Наприклад, якби ми хотіли, щоб при завантаженні сторінки наш div був прихований, то ми могли б це зробити двома способами:

Перший, використати метод hide() :

```
$(document).ready(init);

function init(){
    $('#les8_ex1').hide();
}
function hideDiv(){
    $('#les8_ex1').hide();
}
function showDiv(){
    $('#les8_ex1').show();
}
```

А другий - використати властивість display CSS :

```
#les8_ex1{
        width :400 px;
        height :100 px;
        background:#666699;
        border :1 px solid #333366;
        margin - bottom :20 px;
        display: none;
}
```

Результат буде однаковий.

Ці ж методи набагато цікавіше використати з анімацією:

hide(speed, callback) - приховати

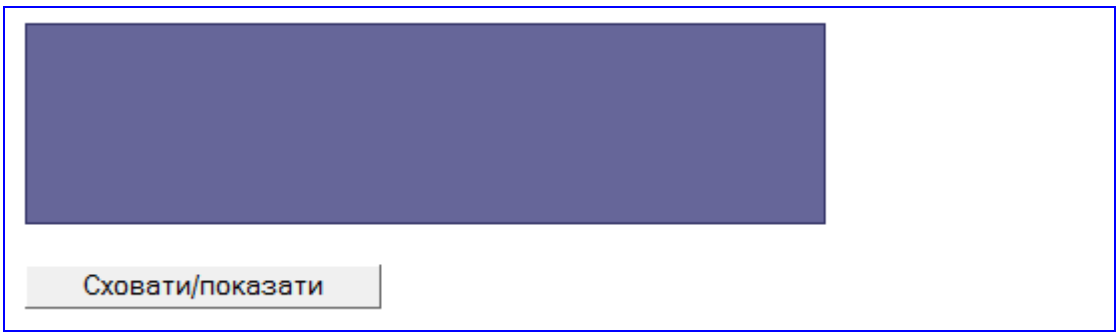
show(speed, callback) - показати

toggle(speed, callback) - перемкнути (показати, якщо приховані і навпаки), де:

speed - швидкість зміни висоти, ширини або властивості opacity (прозорість) елементу. Може приймати три значення: slow (повільно), normal (нормально) або fast (швидко), а також значення в мілісекундах.

callback - функція, яка виконуватиметься після завершення анімації. Її присутність зовсім необов'язкова.

Подивимося той же приклад, тільки приховувати і відображати div будемо повільно (порівняйте з першим прикладом) і, використовуючи метод toggle().



Отже, html -код:

```
<html>
 <head>
  <title>jQuery ефекти</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <script type="text/javascript" src="jquery-3.5.1.js"></script>
  <script type="text/javascript" src="script.js"></script>
 </head>
 <body>
  <div id="les 8_ex2"></div>
  <input type="button" id="but1" value="Приховати/Відобразити" onclick="hideShowDiv();">
 </body>
</html>
```

Код функції :

```
function hideShowDiv(){
        $('#les8_ex2').toggle('slow');
}
```

Методи згортання - slideDown(), slideUp() і slideToggle()

Ці методи також відповідають за видимість елементів, але працюють за принципом згорнути елемент плавно знизу-вгору - розгорнути елемент плавно свеху-вниз:

slideUp() - згорнути

slideDown() - розгорнути,

toggle() - розгорнути, якщо прихований і згорнути, якщо бачимо.

Приклад:

Html -код:

```
<html>
  <head>
    <title>jQuery ефекти</title>
    <link rel="stylesheet" type="text/css" href="style 1.css">
    <script type="text/javascript" src="jquery-3.5.1.js"></script>
    <script type="text/javascript">
function slideUpDiv(){
    $('#les8_ex1').slideUp();
}
function slideDownDiv(){
    $('#__
}


</script>
  </head>
  <body>
<div id="les8_ex1"></div>
    <input type="button" value="Згорнути" onclick="slideUpDiv();">
    <input type="button" value="Розгорнути" onclick="slideDownDiv();">

  </body>
</html>
```

СтилиСтилі на style.css:

```
#les8_ex3{
    width :100 px;
    height :200 px;
    background: #666699;
    border: 1 px solid #333366;
    margin-bottom :20 px;
}
```

Ці методи також можна використати з анімацією:

slideUp(speed, callback) - згорнути

slideDown(speed, callback) - розгорнути

slideToggle(speed, callback) - перемкнути (розгорнути, якщо приховані і навпаки), де:

speed - швидкість зміни висоти елементу. Може приймати три значення: slow (повільно), normal (нормально) або fast (швидко), а також значення в мілісекундах.

callback - функція, яка виконуватиметься після завершення анімації. Її присутність необов'язкова.

Подивимося той же приклад, тільки приховувати і відображати div будемо за 7 секунд і, використовуючи метод toggle().

Методи зникнення - fadeTo(), fadeOut() і fadeIn()

fadeTo(speed, opacity, callback) - зменшує властивість opacity (прозорість) до заданого значення

fadeOut(speed, callback) - зменшує властивість opacity (прозорість)

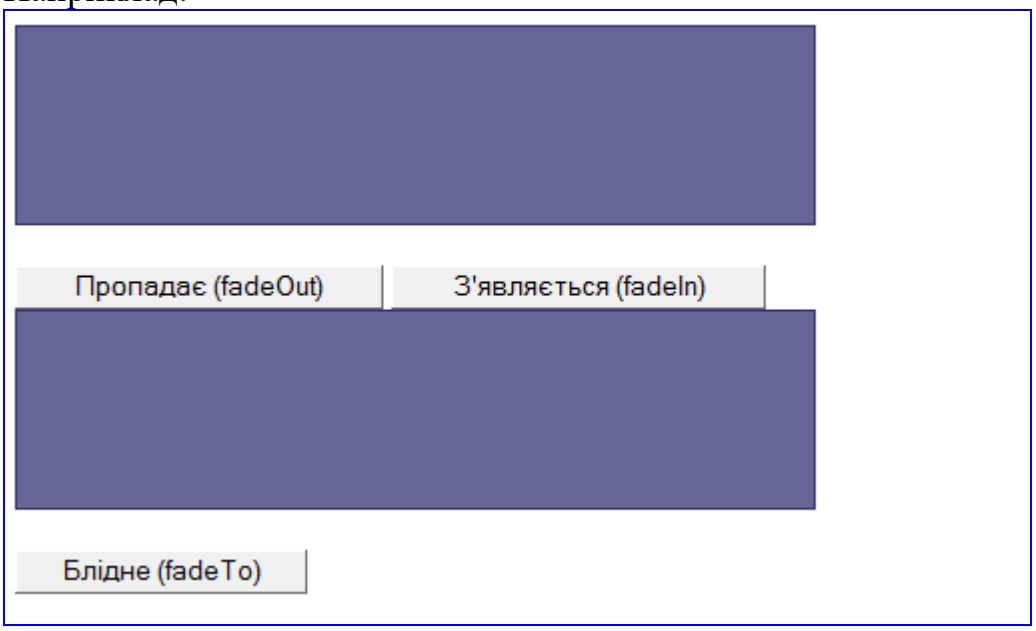
fadeIn(speed, callback) - збільшує властивість opacity (прозорість), де:

speed - швидкість зміни прозорості елементу. Може приймати три значення: slow (повільно), normal (нормально) або fast (швидко), а також значення в мілісекундах.

opacity - значення прозорості, до якого воно буде зменшено (число від 0 до 1).

callback - функція, яка виконуватиметься після завершення анімації. Її присутність необов'язкова.

Наприклад:



Html-код:Html -код:

```
<html>
 <head>
  <title>jQuery ефекти</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <script type="text/javascript" src="jquery-1.2.6.js"></script>
  <script type="text/javascript" src="script.js"></script>
 </head>
 <body>
  <div id="les 8_ex5"></div>
  <input type="button" value="Пропадає (fadeOut)" onclick="fadeOutDiv();">
  <input type="button" value="З' являється (fadeIn)" onclick="fadeInDiv();">
  <div id="les 8_ex6"></div>
  <input type="button" value="Блідне (fadeTo)" onclick="fadeToDiv();">
 </body>
</html>
```

Код функцій (на сторінці script.js) :

```
function fadeOutDiv(){
        $('#les8_ex5').fadeOut(5000);
}
function fadeInDiv(){
        $('#les8_ex5').fadeIn(5000);
}
function fadeToDiv(){
        $('#les8_ex6').fadeTo(5000, 0.5);
}
```

Коментувати тут нічого.

Методи анімації - animate() і stop()

animate(params, options) - анімує стильові властивості, де:

params - атрибути css, які хочемо анімувати ("width", "top", "border"...).

options - властивості анімації (у тому числі швидкість).

stop() - зупиняє анімацію.

Наприклад:

Html -код:

```
<html>
  <head>
    <title>jQuery ефекти</title>
    <link rel="stylesheet" type="text/css" href="style.css">
```

```
    <script type="text/javascript" src="jquery-3.5.1.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
   <div id="les 8_ex7"></div><div id="les 8_ex8"></div>
    <input type="button" value="Почати анімацію" onclick="animateDiv();">
    <input type="button" value="Зупинити анімацію" onclick="stopDiv();">
  </body>
</html>
```

Стилі (на сторінці style.css) :

```
#les8_ex7{
    width :100 px;
    height :100 px;
    background:#666699;
}
#les8_ex8{
    width :400 px;
    height :200 px;
    background:#666699;
}
```

Код функцій (на сторінці script.js) :

```
function animateDiv(){
      $('#les8_ex7').animate({
          width :"400px"
          height :"200px"
      }, 3000 );
      $('#les8_ex8').animate({
          width :"100px"
          height :"100px"
      }, 3000 );
}
function stopDiv(){
      $('#les8_ex7').stop();
      $('#les8_ex8').stop();
}
```

Декілька нюансів:

- властивості мають бути позначені без пропусків, подальше слово з великої букви, тобто "borderWidth" замість "border-width",

- підтримуються тільки ті властивості, значення яких виражаються числами.

- також в якості значень властивостей можуть бути використані значення "hide", "show" і "toggle".