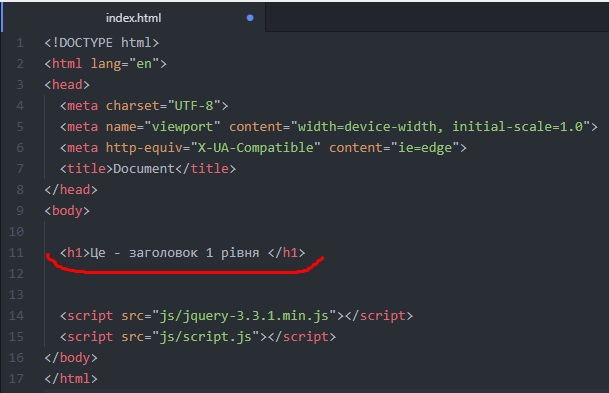
**Тема 02. Синтаксис jQuery**

Відкриємо файл index.html і в розділі body пропишемо якийсь тег, негай це буде

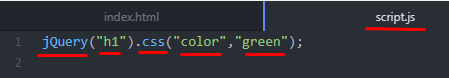


Використовуючи jQuery, ми можемо маніпулювати будь-яким елементом дерева DOM (Document Object Model – об’єктна модель документа).

DOM дерево являє собою ієрархічну структуру, де на самому верхньому рівні знаходиться розділ html, нижче – розділи head та body, ще нижче в даному випадку – розділ h1 і т. д.

Так от зараз спробуємо за допомогою jQuery виконати маніпуляції з елементом h1.

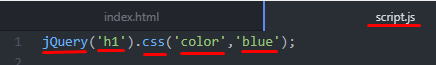
Спочатку вказуємо, що будемо діяти, використовуючи jQuery, далі – елемент, з яким будемо працювати (в даному випадку це – h1), ще далі – за якими правилами хочемо виконати маніпуляції (вказуємо: за правилами css), і конкретизуємо, яку властивість css беремо для роботи – в даному випадку властивість color, якій присвоюємо значення green.



Переглядаємо у браузері

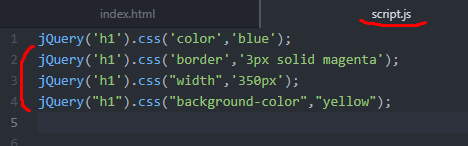


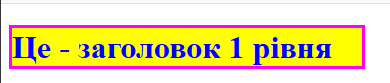
jQuery дозволяє замість подвійних лапок використовувати одинарні, перевіримо.





Спробуємо ще прописати якісь CSS-правила





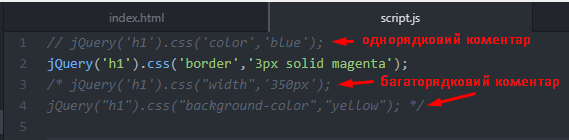
Як бачимо, в рядку 2 ми прописали суцільну рамку (solid) товщиною 3 пікселі (3px) і рожевий колір (magenta).

В рядку 3 вказали ширину (width) елемента h1 рівною 350 пікселів

В рядку 4 вказали жовтий фон (yellow).

Окремо ще раз переконались, що лапки можна використовувати і одинарні ('h1'), і подвійні ("h1").

В коді jQuery ми можемо використовувати коментарі типу C++, C#, JavaScript

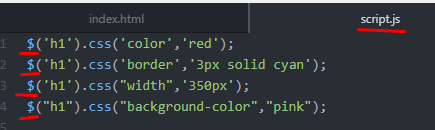




Залишився працювати лише рядок 2, який формує рамку.

Наступний момент – замість фрази «jQuery» можна писати просто значок «$»

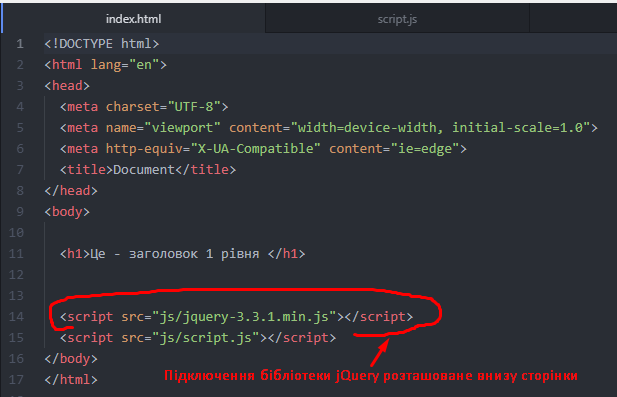
Перевіримо





Отже, значок долара («$») – це синонім фрази «jQuery».

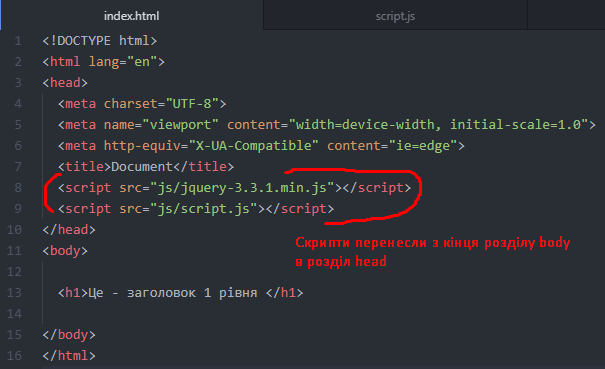
Звертаємо увагу на наступний момент – підключення бібліотеку jQuery ми розташували внизу файлу index.html



Якщо підключення бібліотеки jQuery розташовне внизу сторінки, то це означає, що спочатку завантажується в пам’ять весь html-документ, а лише потім до роботи приступає бібліотека jQuery.

Тобто спочатку завантажився елемент h1 (рядок 11), а лише потім з ним почали працювати скрипти (рядок 14 і рядок 15). І в такому випадку все коректно працює.

Перенесемо скрипти в розділ head

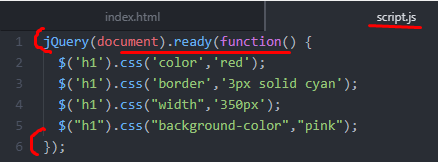


Перевіримо тепер, що буде в браузері



**В такому випадку скрипти не працюють!!!**

Але вихід є, потрібно прописати додатковий код

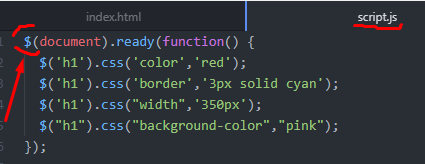




Тепер все працює. Цей код означає наступне:

«Дочекатись, коли документ завантажиться повністю, і лише потім завантажити тіло функції». Тут використовуються наступні ключові слова: document, ready («готовий», тобто коли документ буде готовий для роботи, повністю завантажиться), function.

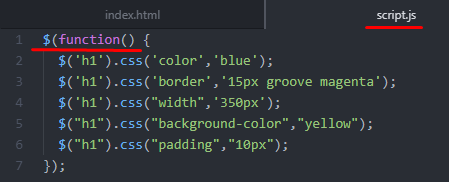
Далі наступне – замість фрази «jQuery» можна використати значок «$».



Перевіряємо



Розробники jQuery нам пропонують ще простіший варіант синтаксису:

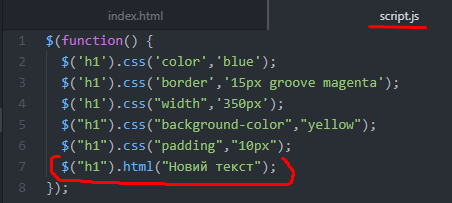




У js-файлі функцій може бути багато. Тому необов’язково весь функціонал «запихувати» в одну функцію.

*До цього ми працювали за правилами CSS.*

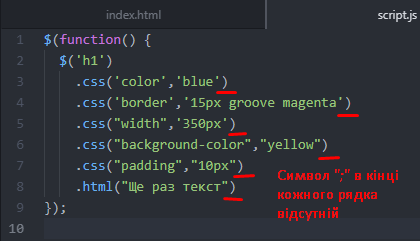
Тепер спробуємо працювати за правилами HTML





Трохи покращимо код. Рядки 2-7 починалися із фрази «Шукати елемент h1»

Напишемо цю фразу 1 раз і зробимо її спільною для цих рядків



Звертаємо увагу, що символ «;» в кінці кожного рядка обов’язково відсутній!!!

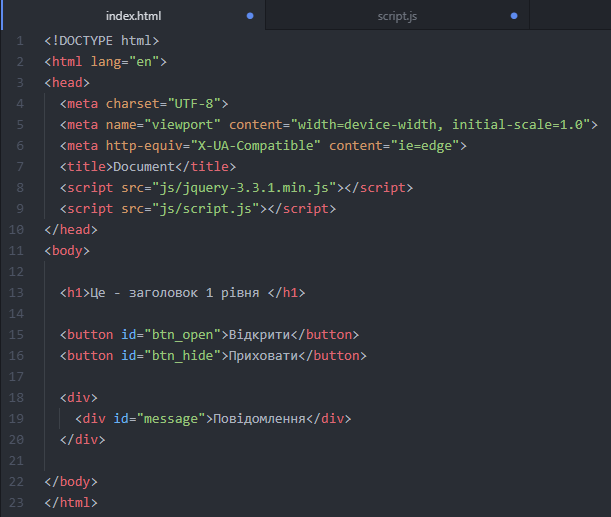


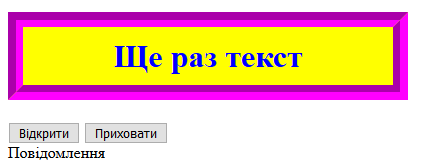
Вирівняємо текст по центру





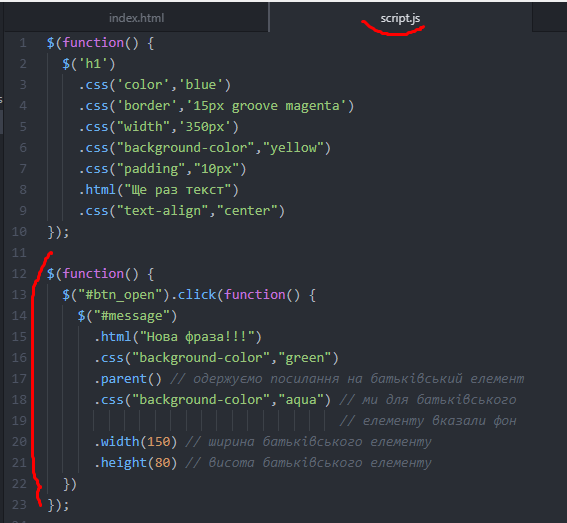
Розглянемо приклад з кнопками.

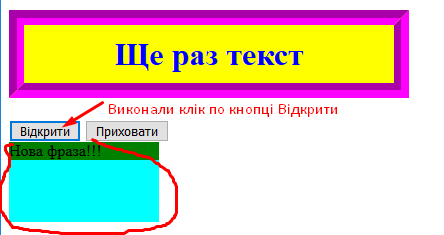




Форматувати нічого не будемо. *Займемось питанням роботи з подіями*.

Зокрема будемо працювати з *подією click*.





*Дамо пояснення для рядків файлу script.js*.

Рядок 12 (файлу script.js) – фактично є «jQuery(document).ready(function() {» - означає, що тіло функції буде виконуватись тільки тоді, коли повністю завантажиться розділ body.

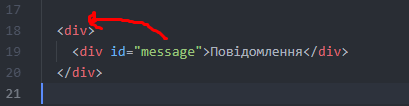
Рядок 13 (файлу script.js) – коли користувач виконає клік по елементу з id = “btn\_open”, то починає виконуватись тіло функції: рядки – 14-21.

Рядок 14 (файлу script.js) – вибирається елемент з id=”message”.

Рядок 15 (файлу script.js) – елементу з рядка 14 присвоюється текст «Нова фраза!!!».

Рядок 16 (файлу script.js) - елементу з рядка 14 присвоюється зелений фон.

Рядок 17 (файлу script.js) – відносно елемента з id=”message” переходимо на батьківський елемент



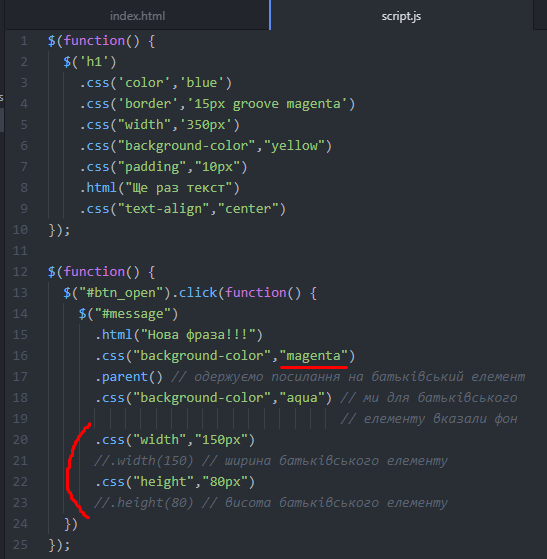
Це означає, що до цього поточним рядком був “div id=”message” (рядок 19 файлу index.html). Тепер поточним став div з рядка 18 файлу index.html.

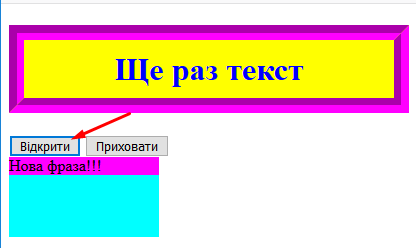
Рядок 18 (файлу script.js) – зафарбовуємо фон батьківського елемента в колір aqua.

Рядок 20 (файлу script.js) – робимо ширину батьківського елемента рівною 150 пікселів.

Рядок 21 (файлу script.js) – робимо висоту батьківського елемента рівною 80 пікселів.

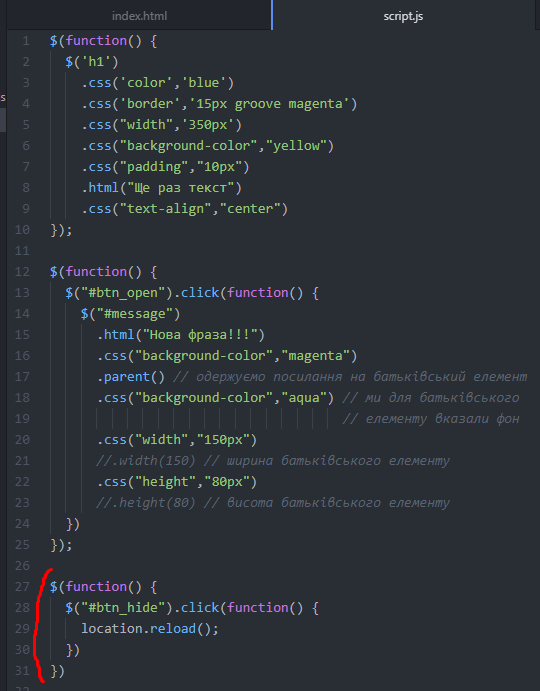
Тут звертаємо увагу на те, що рядки з шириною та висотою батьківського елемента можна було записати стандартним способом



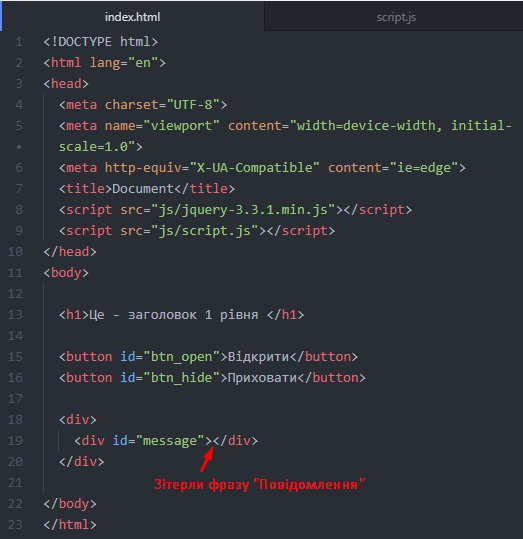


Отже, реакція на подію click для кнопки Відкрити працює.

Пропишемо реакцію на подію click для кнопки Приховати



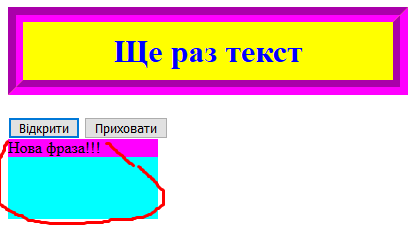
У файлі index.html зітремо фразу «Повідомлення»

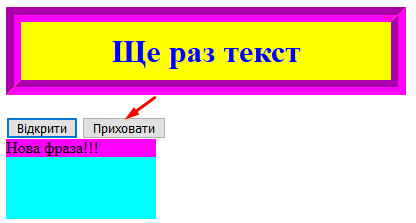


І перевіримо, як все працює.

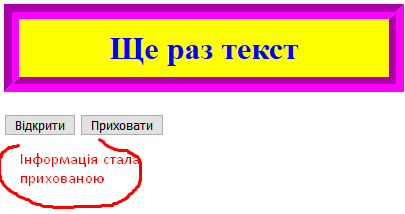


Одержуємо

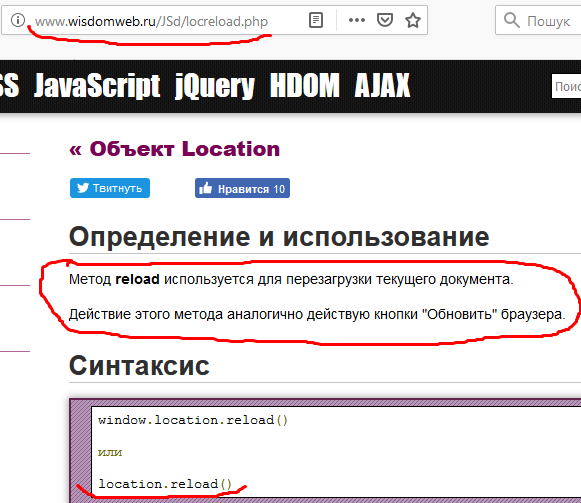




Одержуємо



Залишається розглянути метод location.reload();



Як бачимо, цей метод аналогічний дії кнопки Обновити вікна браузера. Повертається той стан, що був до події клік по кнопці Відкрити.