Вы можете найти любой элемент на странице по его *CSS*-селектору.

Давайте посмотрим, как это работает. Перед нами задача — изменить текст в определенном *html-*элементе на тот, который пользователь введёт с помощью prompt(). В нашей задаче это будет блок заголовка с текстом «Тест консоли».

**Шаг 1. Подготовка**

Для этого создадим простую веб-страницу:

* 1. Откройте *VScode;*
  2. Создайте новый файл *script.js;*
  3. Пропишите в нем следующий текст:
  4. console.log('Здравствуй, Мир!');
  5. Сохраните изменения в файле *script.js;*
  6. Создайте файл*index.html;*
  7. Пропишите в нем следующий текст:
  8. <!DOCTYPE html>
  9. <html lang="en">
  10. <head>
  11. <meta charset="UTF-8">
  12. <title>Тест консоли</title>
  13. </head>
  14. <body>
  15. <h2>Тест консоли</h2>
  16. <script src="script.js"></script>
  17. </body>
  18. </html>
  19. Сохраните изменения в файле*index.html;*

Теперь откройте страницу в браузере и посмотрите, что получилось. Если заглянуть в консоль, то можно увидеть текст «Здравствуй, Мир!»

## Шаг 2. prompt()

Добавим в файл script.js:

**const** userText = prompt('Введите текст');

console.log(userText);

Как и ожидается, пользователь вводит текст, текст выводится в консоль.

Но при этом, разумеется, текст «Тест консоли» не изменился.

**Шаг 3. Поиск элемента**

Для начала, требуется найти этот текст.

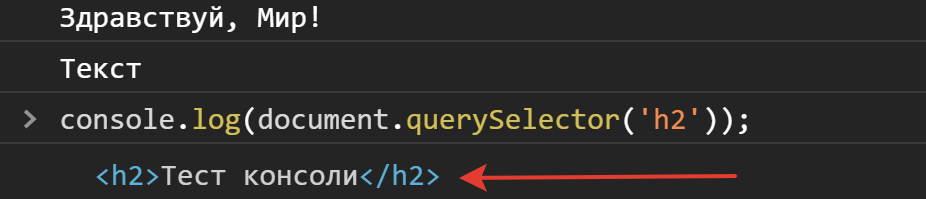
Этот текст находится в блоке h2:

<h2>Тест консоли</h2>

Эту информацию можно использовать, чтобы *JS*-интерпретатор нашёл этот блок на странице с помощью метода document.querySelector(). Напишите в консоли (на созданной нами странице):

console.log(document.querySelector('h2'));

Ожидается следующий результат:



Элемент найден, теперь требуется изменить его текст.

**Шаг 4. Изменение свойства()**

Для этого можно использовать свойство найденного элемента textContent. Замените код в*script.js* на следующий:

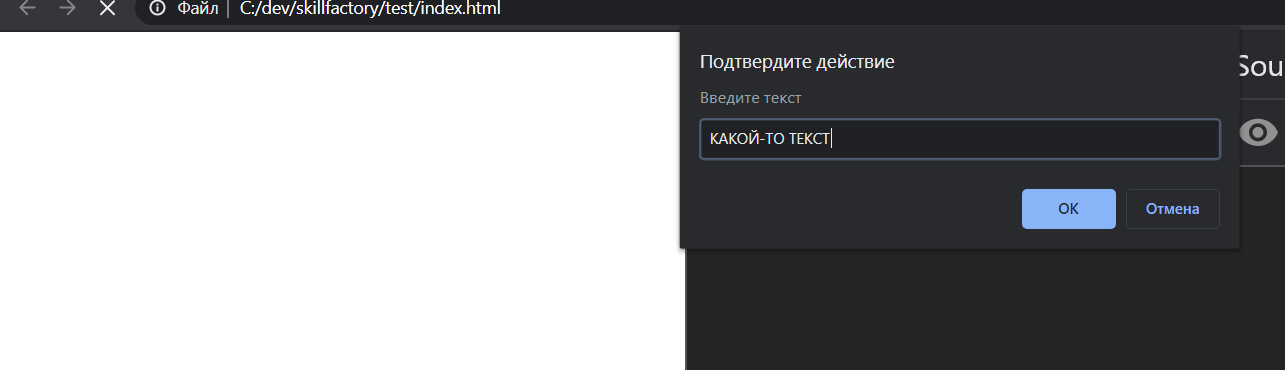
**const** userText = prompt('Введите текст');

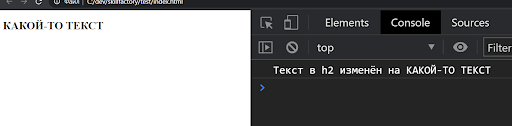
**const** userTextField = document.querySelector('h2');

userTextField.textContent = userText;

console.log('Текст в h2 изменён на', userText);

И обновите страницу. Если всё выполнено верно, то результат должен быть примерно следующий:





Разбираем построчно:

* 1. В константу userText записываем текст, который пользователь ввел в поле ввода.
  2. **const** userText = prompt('Введите текст');
  3. Нашли элемент в  документе по тегу h2 и записали его в userTextField.
  4. **const** userTextField = document.querySelector('h2');
  5. Присвоили у найденного элемента свойство textContent значению переменной userText. Это присвоение позволяет изменить текст внутри найденного элемента.
  6. userTextField.textContent = userText;
  7. Вывели в консоль сообщение о том, что текст изменён
  8. console.log('Текст в h2 изменён на', userText);

**Шаг 5. Модификация — поиск элемента по id()**

Попробуем изменить немного наш код так, что поиск осуществлялся не по тегу, а по *id*. Для этого требуется изменить восьмую строку в файле*index.html* на:

<h2 id="userTextField">Тест консоли</h2>

А код в файле*script.js* требуется изменить на:

**const** userText = prompt('Введите текст');

**const** userTextField = document.querySelector('#userTextField');

userTextField.textContent = userText;

console.log('Текст в блоке с id userTextField изменён на', userText);

Разумеется, в этом случае абсолютно не важен *id*, главное, чтоб искомый *id* в *JS*-коде совпадал с *id* элемента на *html-*странице.

**querySelector**

Существует два современных метода для поиска элемента веб-страницы:

* + document.querySelector(),
  + document.querySelectorAll().

Метод document.querySelector() производит поиск элемента веб-страницы, соответствующего указанному селектору. Если найден элемент — возвращает его. Если нет — возвращает null. Поиск осуществляется только до первого попавшегося подходящего элемента. Если элементов с указанным селектором несколько, нельзя точно сказать, какой будет возвращен. Если вы уверены на 100% что такой элемент должен быть один — используйте этот метод.

Метод document.querySelectorAll() используется, если искомых элементов может быть более одного. Метод производит поиск всех элементов, соответствующих указанному селектору. Все найденные элементы возвращаются массивом. Соответственно, если элементов, соответствующих селектору не найдено — массив возвращается пустой.

Оба метода принимают строку — селектор, такой же как и*CSS*-селектор. Поиск не выполняется только для селекторов с псевдоэлементом, так как результат всегда будет возвращать пустые значения. Используя эти методы можно искать элементы, вложенные в элемент. Например, у вас на веб-странице есть такой блок:

<div class="parent">

<div class="child"></div>

</div>

Можно выполнять при этом поиск следующим образом (не по document, а по родительскому элементу):

**const** parent = document.querySelector('.parent');

**const** child = parent.querySelector('.child');

**Устаревшие методы**

История *JS* очень богата. Многие операции имеют несколько способов реализации. Поиск элемента на странице достаточно важная операция, и потому в браузере есть много способов найти элемент. Мы ознакомились с современными методами — document.querySelector(), document.querySelectorAll(). Но устаревшие методы, которые распространены, требуется знать, потому что в работе вы вполне можете с ними встретиться.

Один из самых старых (и плохих) способов найти элемент — обратится к нему по*ID* через глобальную переменную. Предположим, что у нас есть элемент с id=”textField”:

<p id="textField">Текстовый блок</p>

В таком случае мы можем обратиться к нему, как к глобальной переменной textField:

console.log(textField);

Этот способ работает до сих пор, но возможно, вы с ним никогда не встретитесь.

Есть еще ряд устаревших способов, которые встречаются намного чаще:

* + document.getElementById() — поиск элемента по атрибуту *id,*
  + document.getElementsByClassName() — поиск элемента по *CSS*-классу (атрибут *class),*
  + document.getElementsByName() — поиск элемента по атрибуту *name,*
  + document.getElementsByTagName() — поиск элемента по тегу,
  + document.getElementsByTagNameNS() — поиск элемента по тегу, который принадлежит определенному пространству имен.

Самое важное, что нужно знать про старые методы — **при встрече смело заменяйте их на более современные.**