JAVASCRIPT IN BROWSER

Sunday, December 9, 2018 10:36 AM

DOM - document object model
Document - in DOM is the .html document
Object - in DOM is the object that models our .html document

QUERYSELECTOR:

Using the ***querySelector*** method under the DOM we can get any element in the .html file in f
javascript object and can manipulate it.
Only returns the first elements in the DOM structure.

```
<body>
    <h1>This is a Note taking application.</h1>
    <h4>Write once get/remember everywhere.</h4>



    <script src="notes-app.js"></script>
```

- Here inside the body tags we have two elments ***"h1 and h4"***
- Using the ***querySelector*** we can get the ***h4*** element and try to remove it in the following snippets

```
console.log('New application')

// delete an element from the .html file

const p = document.querySelector('h4') //
// p.innerHTML = "";
console.log(p)
// p.remove()
```

```
▼ h4
    accessKey: ""
    accessKeyLabel: ""
    align: ""
    assignedSlot: null
```

orm of a

```
attributes: NamedNodeMap []
baseURI: "http://127.0.0.1:8080/"
childElementCount: 0
childNodes: NodeList [ #text ]
children: HTMLCollection { length: 0 }
classList: DOMTokenList []
className: ""
clientHeight: 35
clientLeft: 0
clientTop: 0
clientWidth: 874
contentEditable: "inherit"
contextMenu: null
```

- ***h4*** is has various properties associated with it.

```
console.log('New application')

// delete an element from the .html file

const p = document.querySelector('h4') //
// p.innerHTML = "";
// console.log(p)
p.remove()
```

QUERYSELECTOR ALL:

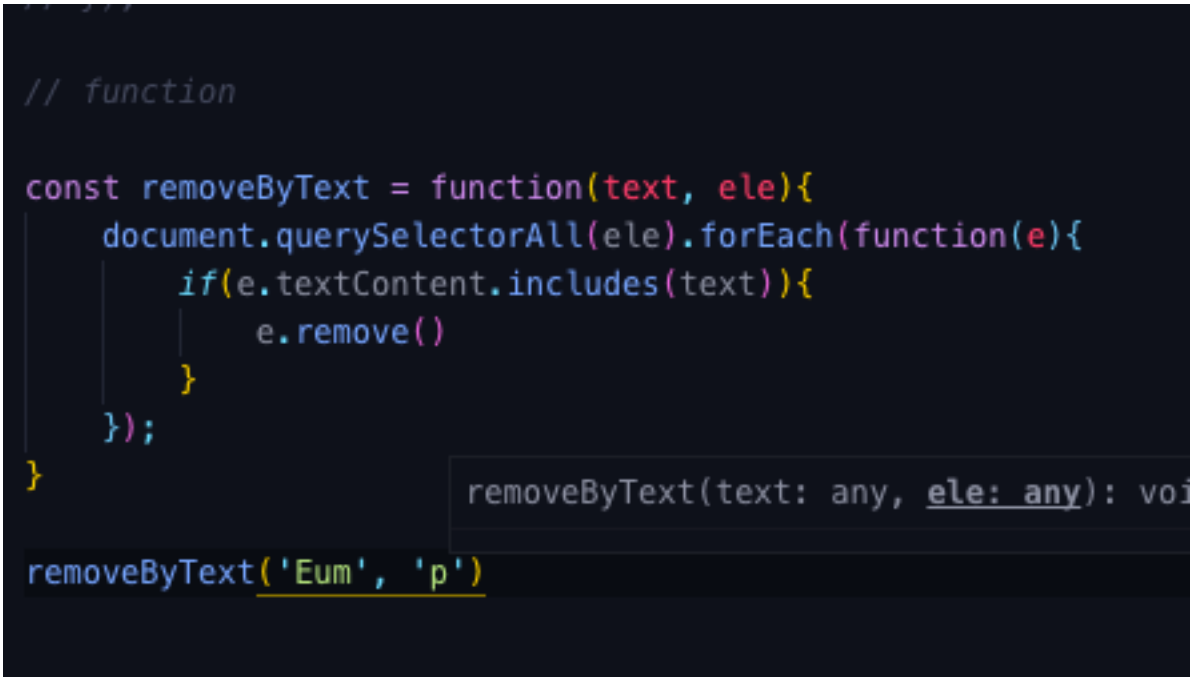
Returns an array of the elements present.

```
// query all and remove

const ps = document.querySelectorAll('p')
console.log(ps)
ps.forEach(function(p){
    p.remove()
})
```

- The ***textContent*** property is used to get the text content from an element and thus we can manipulate the DOM by changing the value or removing it.

Removing a elements which includes some specified text.



- Adding a new element to the .html document using the DOM
  - Create a element using the ***createElement()*** method.
  - Using the ***textContent*** property add some text to it.
  - Now using the ***querySelector*** method select the element under which we need t newly created element and then use ***the appendChild***() method add the newly element to it.

to put the
created

```
// adding a new element

const newPara = document.createElement('p') // returns DOM representation of the el
newPara.textContent = 'This is paragraph made by the js'
// select the body and then add the child using the append child.
document.querySelector('body').appendChild(newPara)
```

Displaying an array of object in the browser using the code we learnt above.

```
const todos = [
    {
        text: 'Todo 1',
        completed: true
    }, {
        text: 'Todo 2',
        completed: false
    }, {
        text: 'Todo 3',
        completed: false
    }, {
        text: 'Todo 4',
        completed: false
    }
]
// const summary = function(todos, element){
//     let count = 0;
//     todos.forEach(function(todo){
//         if(!todo.completed){
//             count = count + 1
//         }
//     });
//     const newElement = document.createElement(element);
//     newElement.textContent = `You have ${count} todos left!!`
//     document.querySelector('body').appendChild(newElement)
// }

const summaryFilter = function(todos, element){
    let c = todos.filter(function (todo){
        return !todo.completed
    })
    const newElement = document.createElement(element);
    newElement.textContent = `You have ${c.length} todos left!!`
    document.querySelector('body').appendChild(newElement)
}


summaryFilter(todos, 'h3')
```

```
// const pTodo = document.createElement('p');
// pTodo.textContent = `You have ${todosLeft} todos left!`;
// document.querySelector('body').appendChild(pTodo)

// function to add all the todos to the screen

const displayTodos = function(todos, element){
    todos.forEach(function(todo){
        const newElement = document.createElement(element);
        newElement.textContent = todo.text;
        document.querySelector('body').appendChild(newElement)
    });

}

displayTodos(todos, 'h5');
```

EVENT LISTENERS:

SCROLL OR CLICK ON SOMETHING.

```
document.querySelector('button').addEventListener('click', function(event){
    // console.log(event)
    // console.log('Did this work?')
    event.target.textContent = 'Button Clicked!'
})
```

When there are multiple similar elements inside a .html file we have the following two way disposal to add an event listener

```
<body>
    <h1>This is a Note taking application.</h1>
    <h4>Write once get/remember everywhere.</h4>

    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Et quo ex doloribu
    <p>3232 Lorem ipsum dolor sit amet consectetur, adipisicing elit. Et quo ex dol
    <button class="btn waves-effect waves-light">CREATE NOTE</button>
    <button class="btn waves-effect waves-light" id="cancelBtn">CANCEL</button>

    <script src="notes-app.js"></script>
```

```
document.getElementById('cancelBtn').addEventListener('click', function(event){
    console.log('cancel clicked...')
})

// ============================== or ==========================================
```

s at our

```
document.querySelectorAll('button')[1].addEventListener('click', function(){
    console.log('button clicked using the queryselector all!!')
})

// ============================ or ========================================

document.querySelector('#cancelBtn').addEventListener('click', function(){
    console.log('Button addededdddfdf')
})
```

- We can provide an ***id*** to the element inside the .html file and can use ***getElementById()*** method to get that element inside the js file.
- Or we can use the simple ***querySelectorAll()*** method and index to the button we want to manipulate it.

- Removing all the elements with the same class names when the remove button is clicked.

```
document.querySelector('#removeBtn').addEventListener('click', function(){
    document.querySelectorAll('.note').forEach(function(note){
        note.remove()
    })

})
```

- For input there are two ways in which we can add an EventListener when the input fie changed.

```
document.querySelector('#search-text').addEventListener('change', function(e){
    console.log(e.target.value)
})
// // ============================ or ========================================

document.querySelector('#search-text').addEventListener('input', function(e){
    console.log(e.target.value)
})
```

- By using ***change*** we can get the value but only when the cursor is moved to any other element
- But by using ***input*** we don't have that limitation i.e. with every letter

eld is

entered we can get it.

- Searching for notes inside the .html file.

```
<body class="container">
    <h1>This is a Note taking application.</h1>
    <h4>Write once get/remember everywhere.</h4>
    <input type="text" placeholder="Search" id="search-text">
    <div class="container" id="notes"></div>
    <button class="btn waves-effect waves-light">CREATE NOTE</button>
    <button class="btn waves-effect waves-light" id="removeBtn">Remove All</button>
```

```
const filters = {
    searchText: ''
}

const renderNotes = function(notes, filters){
    const filteredNotes = notes.filter(function(note){
        return note.title.toLowerCase().includes(filters.searchText.toLowerCase())
    });

    document.querySelector('#notes').innerHTML = '';

    filteredNotes.forEach(function(note){

        const noteElement = document.createElement('p');
        noteElement.textContent = note.title;
        document.querySelector('#notes').appendChild(noteElement);
    });

}

renderNotes(notes, filters)

document.querySelector('#search-text').addEventListener('input', function(e){
    filters.searchText = e.target.value;
    renderNotes(notes, filters)
})
// EVENTLISTENER

document.querySelector('button').addEventListener('click', function(event){
    // console.log(event)
    // console.log('Did this work?')
    event.target.textContent = 'Button Clicked!'
})
```

- Here we create a filters object where we keep our searchText.
- No we create a ***renderNotes()*** function where we use the the ***filter***() ***and includes()*** m notes array of objects to get the array where ***includes()*** method is true.
- Now for each note in the new array obtained through the filter function we will make to add that note.
- Now when the search text changes we will change the value of the ***searchText*** inside t object and then call the ***renderNotes()*** function.

Form and inputs

- Submitting form data.
- Consider the following form.
- In a form we give an ***id*** only to the form tags as through the ***name*** property of inputs we can get the values inside the form.

```
<form id="name-form">
    <input type="text" placeholder="First Name" name="firstName">
    <button class="btn waves-effect waves-light">Submit</button>
</form>
```

```
document.querySelector('#name-form').addEventListener('submit', function(event){
    event.preventDefault(); // prevent default behaviour with an updated url

    console.log(event.target.elements.firstName.value);
})
```

The value of what user typed resides inside the ***name*** property that we applied for the particular field and we can access that using the following code template

event.target.elements(refers to all the elements of the form).<namePropertyname>.value

```
const hideCompleteTodos = function(todos){
    document.querySelector('#todos').innerHTML = '';
```

ethod on the

an element

the ***filters***

```
    const incompleteTodos = todos.filter(function (todo){
        return !todo.completed;
    })

    incompleteTodos.forEach(function(todo){
        const newElement = document.createElement('h5');
        newElement.textContent = todo.text
        document.querySelector('#incomplete-todos').appendChild(newElement);
    });
}
```

- Checkbox to hide and show the completed and incomplete events.

```
document.querySelector('#showIncompleteTodos').addEventListener('change', function(event){
    event.preventDefault();
    if(event.target.checked){
        hideCompleteTodos(todos)
    } else {
        document.querySelector('#incomplete-todos').innerHTML = '';
        renderTodos(todos, filters)
    }
})
```

- Working with dropdowns