Mutation Observer

(спостерігач за мутаціями)

https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver

https://uk.javascript.info/mutation-observer

Mutation Observer (спостерігач за мутаціями)

MutationObserver :

- це вбудований об’єкт, який спостерігає за елементом DOM
- при змінах викликає функцію-колбек
- дозволяє реагувати на зміну контента, зміну атрибутів, …

Кроки	Загальна форма	Приклад. Додавання спостерігача за елеметом `<div contenteditable="true" id="test"></div>`

Створення MutationObserver	`let observer = new MutationObserver(callback)`
	
	`callback` – колбек функція (приймає список з об’єктами-мутаціями `MutationRecord`)
		`function checkDiv(mutationRecords) {`
		`  . . . . . .`
		`}`
		
		`let observer = new MutationObserver(checkDiv)`

Підключення спостерігання	`observer.observe(node, config)`
	
	`node` – вузол, за яким треба вести спостереження
	`config` – об’єкт налаштувань спостерігача, який визначає, що саме треба відслідковувати
		`const config = {`
		`childList: true,`
		`subtree: true,`
		`characterData: true,`
		`attributes: true,`
		`characterDataOldValue: true,`
		`}`
		
		`observer.observe(document.getElementById('test'), config)`

Відключення спостерігання	`observer.disconnect()`	`observer.disconnect()`

Весь код прикладу. При зміні якщо довжина тексту більша за 5 робити його зеленим, інаше - червоним

```
<!DOCTYPE html>
<html lang="en">
  <head>
     . . . . . .
    <script>
      function checkDiv(nutationRecords) {
        const record = nutationRecords[0]

        if (record.target.textContent.length > 5)
          document.getElementById('test').style.color = 'green'
        else document.getElementById('test').style.color = 'red'
      }

      window.onload = function () {
        let observer = new MutationObserver(checkDiv)

        const config = {
          childList: true,
          subtree: true,
          characterData: true,
          attributes: true,
          characterDataOldValue: true,
        }

        observer.observe(document.getElementById('test'), config)
        observer.disconnect()
      }
    </script>
  </head>
  <body>    <div contenteditable="true" id="test"></div>    </body>
</html>
```

Властивості об’єкта `config` (`observer.observe(node, config)`)

Властивість	Значення. Зміни чого спостерігати?
`childList`	зміни у безпосередніх нащадках вузла node
`subtree`	у всіх нащадках вузла node
`attributes`	атрибути вузла node
`attributeFilter`	масив назв атрибутів, щоб спостерігати лише за певними з них
`characterData`	чи спостерігати за змінами в node.data (текстовий вміст)
`attributeOldValue`	якщо вказана як true – до функції зворотного виклику буде передано і нове, і старе значення атрибута (дивіться нижче), а інакше передаватиме лише нове значення (потребує вказаної опції `attributes`)
`characterDataOldValue`	якщо дорівнює true – до функції зворотного виклику буде передано і нове, і старе значення node.data (дивіться нижче), а інакше передаватиме лише нове значення (потребує вказаної опції ***characterData***)

Властивості об’єкта ***MutationRecord***

Властивість	Значення. Зміни чого спостерігати?
type	тип мутації – що саме змінилось : "attributes": змінився атрибут, "characterData": змінилися дані, використовуються для текстових вузлів, "childList": додані/прибрані дочірні елементи,
target	де саме відбулася зміна (елемент, або текстовий вузол)
addedNodes	вузли, які було додано
removedNodes	вузли, які було видалено
previousSibling	попередній елемент відносно доданих/видалених вузлів
nextSibling	наступний елемент відносно доданих/видалених вузлів
attributeName	ім’я зміненого атрибута
oldValue	попереднє значення, лише для змін в атрибуті або тексті, за умови, що встановлено відповідний параметр attributeOldValue/characterDataOldValue

Приклад. Кожну секунду у деякий контейнер вставляється div з випадковим числом (від 1 до 20). Якщо число більше за 10, то видалити його через 1 секунду.

Опис команд	Реалізація
Генерування div з випадковим числом (від 1 до 20)	
Додавання кожну секунду	
Функція перевірки вставлених елементів	
Створення спостерігача	
Підключення спостерігача	

```
function createDivWithRandomNumber(minValue = 1, maxValue = 20) {
  const div = document.createElement('div')
  div.innerText =
    minValue + Math.floor(Math.random() * (maxValue - minValue + 1))
  return div
}
```

Приклад. Кожну секунду у деякий контейнер вставляється div з випадковим числом (від 1 до 20). Якщо число більше за 10, то видалити його через 1 секунду.

Опис команд	Реалізація
Генерування div з випадковим числом (від 1 до 20)	
Додавання кожну секунду	
Функція перевірки вставлених елементів	
Створення спостерігача	
Підключення спостерігача	

```
function createDivWithRandomNumber(minValue = 1, maxValue = 20) {
  const div = document.createElement('div')
  div.innerText =
    minValue + Math.floor(Math.random() * (maxValue - minValue + 1))
  return div
}
```

```
setInterval(() => {
  document.getElementById('test').append(createDivWithRandomNumber())
}, 1000)
```

Приклад. Кожну секунду у деякий контейнер вставляється div з випадковим числом (від 1 до 20). Якщо число більше за 10, то видалити його через 1 секунду.

Опис команд	Реалізація
Генерування div з випадковим числом (від 1 до 20)	
Додавання кожну секунду	
Функція перевірки вставлених елементів	
Створення спостерігача	
Підключення спостерігача	

```
function createDivWithRandomNumber(minValue = 1, maxValue = 20) {
  const div = document.createElement('div')
  div.innerText =
    minValue + Math.floor(Math.random() * (maxValue - minValue + 1))
  return div
}
```

```
  setInterval(() => {
    document.getElementById('test').append(createDivWithRandomNumber())
  }, 1000)
```

```
function checkDiv(mutationRecords) {
  console.log(mutationRecords)
  mutationRecords.forEach((record) => {
    record.addedNodes.forEach((element) => {
      const number = parseInt(element?.innerText)
      if (number > 10) {
        setTimeout(() => {
          console.log(`removed - ${number}`)
          element.remove()
        }, 1000)
      }
    })
  })
}
```

Приклад. Кожну секунду у деякий контейнер вставляється div з випадковим числом (від 1 до 20). Якщо число більше за 10, то видалити його через 1 секунду.

Опис команд	Реалізація

Генерування div з випадковим числом (від 1 до 20)	`function createDivWithRandomNumber(minValue = 1, maxValue = 20) {`
	`  const div = document.createElement('div')`
	`  div.innerText =`
	`  minValue + Math.floor(Math.random() * (maxValue - minValue + 1))`
	`  return div`
	`}`

Додавання кожну секунду	`setInterval(() => {`
	`  document.getElementById('test').append(createDivWithRandomNumber())`
	`}, 1000)`

Функція перевірки вставлених елементів	`function checkDiv(mutationRecords) {`
	`  console.log(mutationRecords)`
	`  mutationRecords.forEach((record) => {`
	`  record.addedNodes.forEach((element) => {`
	`  const number = parseInt(element?.innerText)`
	`  if (number > 10) {`
	`  setTimeout(() => {`
	`  console.log(`removed - ${number}`)`
	`  element.remove()`
	`  }, 1000)`
	`  }`
	`  })`
	`  })`
	`}`

Створення спостерігача	`let observer = new MutationObserver(checkDiv)`
Підключення спостерігача	

Приклад. Кожну секунду у деякий контейнер вставляється div з випадковим числом (від 1 до 20). Якщо число більше за 10, то видалити його через 1 секунду.

Опис команд	Реалізація

Генерування div з випадковим числом (від 1 до 20)	`function createDivWithRandomNumber(minValue = 1, maxValue = 20) {`
	`  const div = document.createElement('div')`
	`  div.innerText =`
	`  minValue + Math.floor(Math.random() * (maxValue - minValue + 1))`
	`  return div`
	`}`

Додавання кожну секунду	`setInterval(() => {`
	`  document.getElementById('test').append(createDivWithRandomNumber())`
	`}, 1000)`

Функція перевірки вставлених елементів	`function checkDiv(mutationRecords) {`
	`  console.log(mutationRecords)`
	`  mutationRecords.forEach((record) => {`
	`  record.addedNodes.forEach((element) => {`
	`  const number = parseInt(element?.innerText)`
	`  if (number > 10) {`
	`  setTimeout(() => {`
	`  console.log(`removed - ${number}`)`
	`  element.remove()`
	`  }, 1000)`
	`  }`
	`  })`
	`  })`
	`}`

Створення спостерігача	`let observer = new MutationObserver(checkDiv)`

Підключення спостерігача	`const config = {`
	`  childList: true,`
	`}`
	
	`observer.observe(document.getElementById('test'), config)`

```
<script>
  function createDivWithRandomNumber(minValue = 1, maxValue = 20) {
    const div = document.createElement('div')
    div.innerText =
      minValue + Math.floor(Math.random() * (maxValue - minValue + 1))
    return div
  }

  function checkDiv(mutationRecords) {
    console.log(mutationRecords)
    mutationRecords.forEach((record) => {
      record.addedNodes.forEach((element) => {
        const number = parseInt(element?.innerText)
        if (number > 10) {
          setTimeout(() => {
            console.log(`removed - ${number}`)
            element.remove()
          }, 1000)
        }
      })
    })
  }

  window.onload = function () {
    let observer = new MutationObserver(checkDiv)

    const config = {
      childList: true,
    }

    observer.observe(document.getElementById('test'), config)

    setInterval(() => {
      document.getElementById('test').append(createDivWithRandomNumber())
    }, 1000)
  }
</script>
```