**Selection и Range**

## Range

В основе выделения лежит [**Range**](https://dom.spec.whatwg.org/#ranges) – диапазон. Он представляет собой пару «граничных точек»: начало и конец диапазона. Каждая точка представлена как родительский DOM-узел с относительным смещением от начала. Если этот узел – DOM-элемент, то смещение – это номер дочернего элемента, а для текстового узла смещение – позиция в тексте.

<p id="p">Example: <i>italic</i> and <b>bold</b></p>

From <input id="start" type="number" value=1> – To <input id="end" type="number" value=4>

<button id="button">Click to select</button>

<script>

button.onclick = () => {

let range = new Range();

range.**setStart**(p, start.value);

range.**setEnd**(p, end.value);

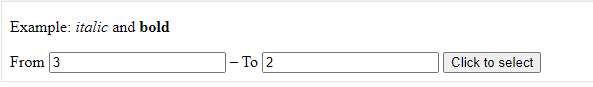
// применим выделение, объясняется далее

**document.getSelection().removeAllRanges();**

**document.getSelection().addRange(range);**

};

</script>



Не обязательно использовать один и тот же элемент в setStart и setEnd. Диапазон может охватывать множество не связанных между собой элементов. Важно лишь чтобы конец шёл после начала.

### **Свойства Range**

* startContainer, startOffset – узел и начальное смещение,
* endContainer, endOffset – узел и конечное смещение,
* collapsed – boolean, **true**, если диапазон начинается и заканчивается на одном и том же месте (следовательно, в диапазон ничего не входит),
* commonAncestorContainer – ближайший общий предок всех узлов в пределах диапазона, в примере выше: <p>

### **[Методы Range](https://learn.javascript.ru/selection-range" \l "metody-range)**

Существует множество удобных методов для манипулирования диапазонами.

**Установить начало диапазона**:

* setStart(node, offset) установить начальную границу в позицию offset в node
* setStartBefore(node) установить начальную границу прямо перед node
* setStartAfter(node) установить начальную границу прямо после node

**Установить конец диапазона** (похожи на предыдущие методы):

* setEnd(node, offset) установить конечную границу в позицию offset в node
* setEndBefore(node) установить конечную границу прямо перед node
* setEndAfter(node) установить конечную границу прямо после node

**Другие:**

* selectNode(node) выделить node целиком
* selectNodeContents(node) выделить всё содержимое node
* collapse(toStart) если указано toStart=true, установить конечную границу в начало, иначе установить начальную границу в конец, схлопывая таким образом диапазон
* cloneRange() создать новый диапазон с идентичными границами

Чтобы манипулировать содержимым в пределах диапазона:

* deleteContents() – удалить содержимое диапазона из документа
* extractContents() – удалить содержимое диапазона из документа и вернуть как **DocumentFragment**
* cloneContents() – склонировать содержимое диапазона и вернуть как DocumentFragment
* insertNode(node) – вставить node в документ в начале диапазона
* surroundContents(node) – обернуть node вокруг содержимого диапазона. Чтобы этот метод сработал, диапазон должен содержать как открывающие, так и закрывающие теги для всех элементов внутри себя: не допускаются частичные диапазоны по типу <i>abc.

## DocumentFragment

DocumentFragment является специальным DOM-узлом, который служит обёрткой для передачи списков узлов.

Например, getListContent ниже генерирует фрагмент с элементами <li>, которые позже вставляются в <ul>:

<ul id="ul"></ul>

<script>

function getListContent() {

let fragment = new DocumentFragment();

for(let i=1; i<=3; i++) {

let li = document.createElement('li');

li.append(i);

fragment.append(li);

}

return fragment;

}

ul.append(getListContent()); // (\*)

</script>

## Selection

Range это общий объект для управления диапазонами выделения. Мы можем создавать и передавать подобные объекты. Сами по себе они ничего визуально не выделяют. Выделение в документе представлено объектом **Selection**, который может быть получен как window.getSelection() или document.getSelection().

Выделение может включать **ноль или более диапазонов**. По крайней мере, так утверждается в Спецификации Selection API. На практике же выделить несколько диапазонов в документе можно только в **Firefox**, используя Ctrl+click (Cmd+click для Mac).

### **Свойства Selection**

Аналогично диапазону, выделение имеет начальную границу, именуемую «якорем», и конечную, называемую «фокусом».

* anchorNode – узел, с которого начинается выделение,
* anchorOffset – смещение в anchorNode, где начинается выделение,
* focusNode – узел, на котором выделение заканчивается,
* focusOffset – смещение в focusNode, где выделение заканчивается,
* isCollapsed – true, если диапазон выделения пуст или не существует.
* rangeCount – количество диапазонов в выделении, максимум 1 во всех браузерах, кроме Firefox.

Конец выделения может быть в документе до его начала. Это отличается от объектов Range, которые всегда направлены вперёд: начало диапазона не может стоять после его конца.

### **События при выделении**

* elem.onselectstart – когда с elem начинается выделение, например пользователь начинает двигать мышкой с зажатой кнопкой.
* preventDefault() отменяет начало выделения.
* document.onselectionchange – когда выделение изменено. Заметьте: этот обработчик можно поставить только на document.

### **Методы Selection**

* getRangeAt(i) – взять i-ый диапазон, начиная с 0. Во всех браузерах, кроме Firefox, используется только 0.
* addRange(range) – добавить range в выделение. Все браузеры, кроме Firefox, проигнорируют этот вызов, если в выделении уже есть диапазон.
* removeRange(range) – удалить range из выделения.
* removeAllRanges() – удалить все диапазоны.
* empty() – сокращение для removeAllRanges.

Также существуют **методы управления диапазонами выделения напрямую**, без обращения к Range:

* collapse(node, offset) – заменить выделенный диапазон новым, который начинается и заканчивается на node, на позиции offset.
* setPosition(node, offset) – то же самое, что collapse (дублирующий метод-псевдоним).
* collapseToStart() – схлопнуть (заменить на пустой диапазон) к началу выделения,
* collapseToEnd() – схлопнуть диапазон к концу выделения,
* extend(node, offset) – переместить фокус выделения к данному node, с позиции offset,
* setBaseAndExtent(anchorNode, anchorOffset, focusNode, focusOffset) – заменить диапазон выделения на заданные начало anchorNode/anchorOffset и конец focusNode/focusOffset. Будет выделено всё содержимое между этими границами
* selectAllChildren(node) – выделить все дочерние узлы данного узла node.
* deleteFromDocument() – удалить содержимое выделения из документа.
* containsNode(node, allowPartialContainment = false) – проверяет, содержит ли выделение node (частично, если второй аргумент равен true)

Чтобы что-то выделить, сначала **снимите текущее выделение.** Если выделение уже существует, сначала снимите его, используя **removeAllRanges()**, и только затем добавляйте новые диапазоны. В противном случае все браузеры, кроме Firefox, проигнорируют добавление. Исключением являются некоторые методы выделения, которые заменяют существующее выделение, например, **setBaseAndExtent**.

## Выделение в элементах форм

Элементы форм, такие как input и textarea, предоставляют отдельное API для выделения. Так как значения полей представляют собой простой текст, а не HTML, и нам не нужны такие сложные объекты, как Range и Selection

### **Свойства:**

* input.selectionStart – позиция начала выделения (это свойство можно изменять),
* input.selectionEnd – позиция конца выделения (это свойство можно изменять),
* input.selectionDirection – направление выделения, одно из: «forward» (вперёд), «backward» (назад) или «none» (без направления, если, к примеру, выделено с помощью двойного клика мыши).

### **События:**

* input.onselect – срабатывает, когда выделение завершено.

### **Методы:**

* input.select() – выделяет всё содержимое input (может быть textarea вместо input),
* input.setSelectionRange(start, end, [direction]) – изменить выделение, чтобы начиналось с позиции start, и заканчивалось end, в данном направлении direction (необязательный параметр).
* input.setRangeText(replacement, [start], [end], [selectionMode]) – заменяет выделенный текст в диапазоне новым. Если аргументы start и end указаны, то они задают начало и конец диапазона, иначе используется текущее выделение.

Последний аргумент, **selectionMode**, определяет, как будет вести себя выделение после замены текста. Возможные значения**:**

* + "**select**" – только что вставленный текст будет выделен.
  + "**start**" – диапазон выделения схлопнется прямо перед вставленным текстом (так что курсор окажется непосредственно перед ним).
  + "**end**" – диапазон выделения схлопнется прямо после вставленного текста (курсор окажется сразу после него).
  + "**preserve**" – пытается сохранить выделение. Значение по умолчанию.

### **Существуют три способа сделать что-то невыделяемым:**

1) Используйте CSS-свойство user-select: none.

<style>

#elem {

user-select: none;

}

</style>

<div>Можно выделить <div id="elem">Нельзя выделить</div> Можно выделить</div>

Это свойство не позволяет начать выделение с elem, но пользователь может начать выделять с другого места и включить elem. После этого elem станет частью document.getSelection(), так что на самом деле выделение произойдёт, но его содержимое обычно игнорируется при копировании и вставке.

2) **Предотвратить действие по умолчанию** в событии onselectstart или mousedown.

<div>Можно выделить <div id="elem">Нельзя выделить</div> Можно выделить</div>

<script>

elem.onselectstart = **() => false**;

</script>

Этот способ также не даёт начать выделение с elem, но пользователь может начать с другого элемента, а затем расширить выделение до elem. Это удобно, когда есть другой обработчик события на том действии, которое запускает выделение (скажем, mousedown). Так что мы отключаем выделение, чтобы избежать конфликта.

А содержимое elem при этом может быть скопировано.

3) Мы также можем очистить выделение после срабатывания с помощью document.getSelection().empty(). Этот способ используется редко, так как он вызывает нежелаемое мерцание при появлении и исчезновении выделения.