DOM Selection API Cheatsheet

A quick reference guide to the HTML DOM Selection API, covering essential methods, properties, and event handling for managing text selections in web applications.

Selection Basics

Obtaining the Selection Object

Use `document.getSelection()` to retrieve the Selection object representing the range of text selected by the user or the current position of the caret.

```
var selection = document.getSelection();
```

Selection Object Properties

`selection.anchorNode`	The node in which the selection begins.
`selection.anchorOffset`	The offset into the `anchorNode` at which the selection begins.
`selection.focusNode`	The node in which the selection ends.
`selection.focusOffset`	The offset into the `focusNode` at which the selection ends.
`selection.isCollapsed`	A boolean indicating whether the selection's start and end points are at the same position.
`selection.rangeCount`	The number of ranges in the selection.

Example

```
<p id="myText">This is some text.</p>
<button onclick="getSelectionInfo()">Get Selection Info</button>

<script>
function getSelectionInfo() {
  var sel = document.getSelection();
  console.log('Anchor Node:', sel.anchorNode);
  console.log('Anchor Offset:', sel.anchorOffset);
  console.log('Focus Node:', sel.focusNode);
  console.log('Focus Offset:', sel.focusOffset);
  console.log('Is Collapsed:', sel.isCollapsed);
}
</script>
```

Manipulating Selections

Adding and Removing Ranges

`selection.addRange(range)`	Adds a `Range` object to the selection.
`selection.removeRange(range)`	Removes a `Range` object from the selection.
`selection.removeAllRanges()`	Removes all ranges from the selection, effectively deselecting everything.

Creating and Using Ranges

```
var range = document.createRange();
range.setStart(startNode, startOffset);
range.setEnd(endNode, endOffset);

selection.addRange(range);
```

Example: Selecting part of a text node.

```
<p id="textContainer">Hello, world!</p>
<button onclick="selectText()">Select 'world'</button>

<script>
function selectText() {
  var textContainer = document.getElementById('textContainer');
  var range = document.createRange();
  range.setStart(textContainer.firstChild, 7);
  range.setEnd(textContainer.firstChild, 12);

  var selection = document.getSelection();
  selection.removeAllRanges();
  selection.addRange(range);
}
</script>
```

Selection State Manipulation

Collapsing Selections

`selection.collapse(node, offset)`	Collapses the selection to a single point at the specified `node` and `offset`.
`selection.collapseToStart()`	Collapses the selection to its start point.
`selection.collapseToEnd()`	Collapses the selection to its end point.

Checking Node Containment

`selection.containsNode(node, partlyContained)` - Indicates if the specified node is part of the selection.

- `node`: The node to check.
- `partlyContained` (optional): A boolean. If `true`, the method returns `true` if part of the node is contained in the selection. If `false` (default), only returns `true` if the entire node is contained in the selection.

```
var isContained = selection.containsNode(element, true);
```

Advanced Selection Techniques

Deleting Content from the Document

`selection.deleteFromDocument()` - Deletes the selected content from the document.

```
selection.deleteFromDocument();
```

Example: Delete selected text when a button is clicked.

```
<p id="deletableText">Select this text and click delete.</p>
<button onclick="deleteSelected()">Delete Selection</button>

<script>
function deleteSelected() {
  var selection = document.getSelection();
  selection.deleteFromDocument();
}
</script>
```

Listening for Selection Changes

Use the `selectionchange` event to monitor changes to the document's selection.

```
document.addEventListener('selectionchange', () => {
  console.log('Selection changed');
  var selection = document.getSelection();
  console.log('Selected text:', selection.toString());
});
```