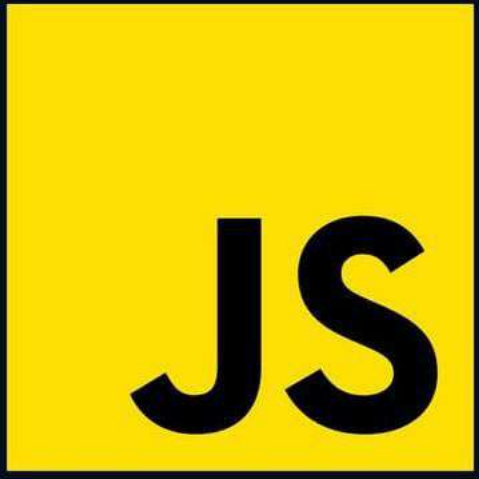
JavaScript Events Cheat Sheet



JS

Event Listener Registration

You can register event listeners to handle events triggered by HTML elements. This is typically done using the **addEventListener** method.

index.js

```
const button = document.getElementById('myButton');

button.addEventListener('click', function() {
  // Your event handling code here
});
```

JS

Event Types

There are various types of events, such as click, mouseover, keydown, and submit, to name a few. Choose the appropriate event type based on the user interaction you want to capture.

index.js

```
element.addEventListener('mouseover', function() {
  // Mouseover event handling
});
```

JS

Event Object

Event listeners receive an event object as an argument. This object contains information about the event, such as the target element and event type.

index.js

```
button.addEventListener('click', function(event) {
  console.log('Button clicked:', event.target);
});
```

JS

Event Bubbling

Events propagate upwards through the DOM tree by default. You can stop this propagation using the **stopPropagation** method.

index.js

```
parentElement.addEventListener('click', function() {
  // This event fires for child and parent elements
});

childElement.addEventListener('click', function(event) {
  event.stopPropagation(); // Stops event from reaching parent
});
```

JS

Prevent Default

You can prevent the default action associated with an event (e.g., form submission or link navigation) using the **preventDefault** method.

index.js

```
anchorElement.addEventListener('click', function(event) {
  event.preventDefault(); // Prevents link navigation
});
```

JS

Removing Event Listeners

You can remove event listeners using the removeEventListener method. Ensure the function reference matches the one you used to add the listener.

index.js

```
function eventHandler() {
  // Event handling code
}

element.addEventListener('click', eventHandler);
element.removeEventListener('click', eventHandler);
```

JS

Event Delegation

Event delegation is a technique where you attach a single event listener to a parent element to handle events for its children. This is efficient for dynamically generated content.

index.js

```
parentElement.addEventListener('click', function(event) {
  if (event.target.matches('button')) {
    // Handle button click
  }
});
```

JS

Keyboard Events

You can capture keyboard events, such as keydown, keyup, and keypress, to respond to user input from the keyboard.

index.js

```
document.addEventListener('keydown', function(event) {
  if (event.key === 'Enter') {
    // Handle Enter key press
  }
});
```