**Ext JS Event Handling**

**Presented To**

Sharath Kumar

**Group C**

**Submitted by**

Satyam Nayak – ([satyamn120@gmail.com](mailto:satyamn120@gmail.com))

Sushant Kumar Kapri – ([sushantkapri@gmail.com](mailto:sushantkapri@gmail.com))

Anunit Raj – ([anunitsingh@gmail.com](mailto:anunitsingh@gmail.com))

Sohaib Raza – ([razasohaib.muz@gmail.com](mailto:razasohaib.muz@gmail.com))

Rehan Rubin Yusuf khan – ([khanrubin004@gmail.com](mailto:khanrubin004@gmail.com))

**ACKNOWLEDGEMENT**

We, as a team, would like to express our sincere gratitude to everyone who helped us complete this document on Ext JS Event Handling. First, we thank our mentors and instructors for their guidance and support. Their insights helped us understand the key concepts of Ext JS event handling. We also appreciate the efforts of our team members—Sushant Kumar Kapri, Satyam Nayak, Anunit Raj, Rehan Rubin Yusuf Khan, and Sohaib Raza—for their dedication and teamwork in preparing this document. A special thanks to our peers for their valuable feedback and discussions, which helped improve the quality of this work. Lastly, we acknowledge the resources and documentation that provided useful information for our research. Thank you all for your support and encouragement.

**Introduction**

Ext JS provides a robust and efficient event-handling system, allowing developers to build interactive and dynamic web applications. Events are fundamental to modern web applications, enabling seamless user interactions, such as clicking buttons, hovering over elements, typing in text fields, and more. Ext JS simplifies event management by providing a structured approach to registering, handling, and propagating events within components.

This document delves into the core aspects of Ext JS event handling, covering essential topics such as event registration, propagation, and delegation. It explores how the framework utilizes the observable pattern, which allows components to fire and listen to events efficiently. Additionally, we discuss the role of event listeners, their attachment to DOM elements, and best practices for managing them to ensure optimal performance.

Understanding event propagation is crucial for managing event flow in an application. This document explains the bubbling and capturing phases, enabling developers to control how events move through the DOM hierarchy. Furthermore, we cover techniques such as relay events, which allow one component to listen and re-fire another component's events, enhancing modularity.

To ensure a smooth user experience, we also highlight the importance of animations in event handling. Animations provide visual feedback to user actions, making applications more engaging and responsive. By integrating animations effectively, developers can improve usability and accessibility.

Lastly, we outline the best practices for event handling in Ext JS, including proper event cleanup, using namespaces for organization, and optimizing performance by debouncing or throttling events. These techniques help prevent memory leaks and maintain an efficient, scalable application.

This document serves as a comprehensive guide to mastering event handling in Ext JS, equipping developers with the knowledge and best practices required to build robust, interactive applications.

**Methodology**

To implement event handling in Ext JS, we structured our approach by utilizing its event-driven architecture and built-in event management system. We included the Ext JS library in our HTML file and ensured all event handlers were registered within the Ext.onReady() function to execute after the DOM fully loads

**Implementation Steps:**

1. Event Listeners for Buttons – We added a click event on a button using Ext.get().on(), triggering an alert when clicked.
2. Form Handling – A submit event was attached to prevent default submission and retrieve input values dynamically.
3. Event Delegation – Instead of adding listeners to each child button, we used delegation on the parent <div>, improving performance.
4. Mouse Events – mouseover, mouseout, mouseenter, and mouseleave were applied to change element styles dynamically.
5. Keyboard Events – keypress, keydown, and keyup were used to log user input in real-time.
6. Focus & Blur Events – Implemented on input fields and buttons to highlight user interactions.
7. Context Menu Handling – Right-click was disabled on a section using contextmenu to prevent unwanted actions.
8. Resize Event – Captured window resize events to dynamically adjust elements.

This methodology ensures efficient event management, enhances user interaction, and improves the responsiveness of the application.

**1. Ext JS Event System**

Ext JS has a powerful event-handling system that allows components to respond to user interactions and system events. It follows an event-driven programming model.

**1.1 Event Registration**

* Events are registered using Ext.get().on() for elements.
* Example: Ext.get('myButton').on('click', function () { alert('Button Clicked!'); });

**1.2 Observable Pattern**

Ext JS follows the **observer pattern**, allowing components to listen for and respond to events.

**2. Understanding Events in Ext JS**

**2.1 Key Concepts and Definitions**

* **What are Events?** Events are occurrences in the browser that facilitate user interactions.
* **Event-driven Architecture** Applications dynamically respond to user actions based on events.
* **Listener**: A function that executes when an event occurs.
* **Event Delegation**: Handling events at a parent level to optimize performance.
* **Event Propagation** Events can bubble up or trickle down through the DOM hierarchy.
* **Types of Events** Includes mouse clicks, keyboard inputs, and hover actions.

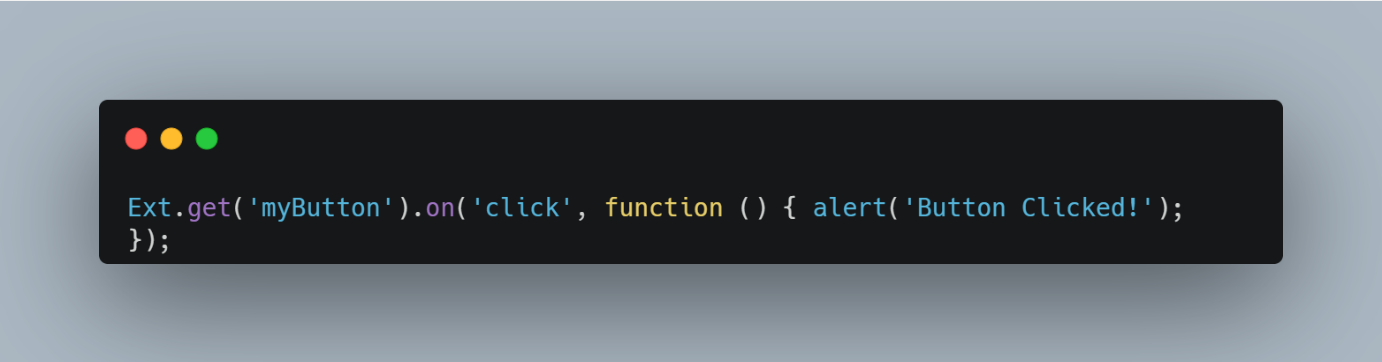
**3. Understanding Event Listeners**

**3.1 Definition and Purpose**

Event listeners are functions that execute specific actions when an event occurs. They improve interactivity and maintain a structured application workflow.

**3.2 Example Usage**

* Click event on a button:



**3.3 Separation of Concerns**

Event listeners keep logic separate from core application code, ensuring maintainability and clarity.

**4. Adding Event Listeners in Ext JS**

**4.1 Using the on Method**

* Adds an event listener to an element.



**4.2 Component Configuration**

Events can also be attached in component configurations during initialization.

**4.3 Using Animations**

* Example: Changing background color on mouseover.



**5. Understanding the Event Object**

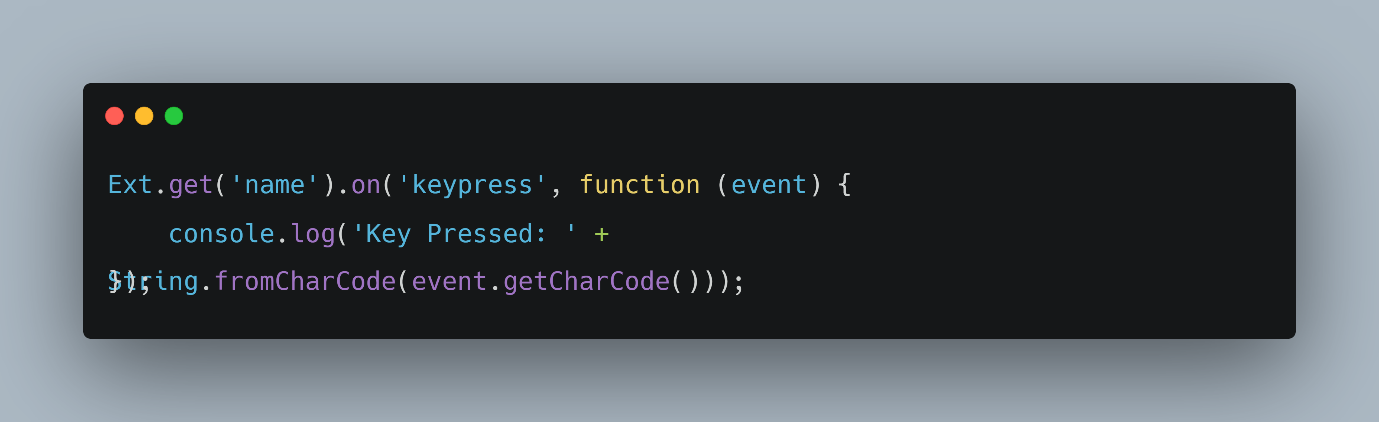
The **Event Object** in Ext JS provides detailed information about user interactions, such as keyboard and mouse actions. It helps in controlling event behavior efficiently.

* 1. **Key Attributes**

1. event.getCharCode()

Returns the character code of a pressed key in key events.

Example:



If a user presses "A", the output will be:

Key Pressed: A

1. event.preventDefault()

* Prevents the default action of an event.
* Example: Preventing form submission



This ensures the form does not refresh the page when submitted.

1. event.getKey()

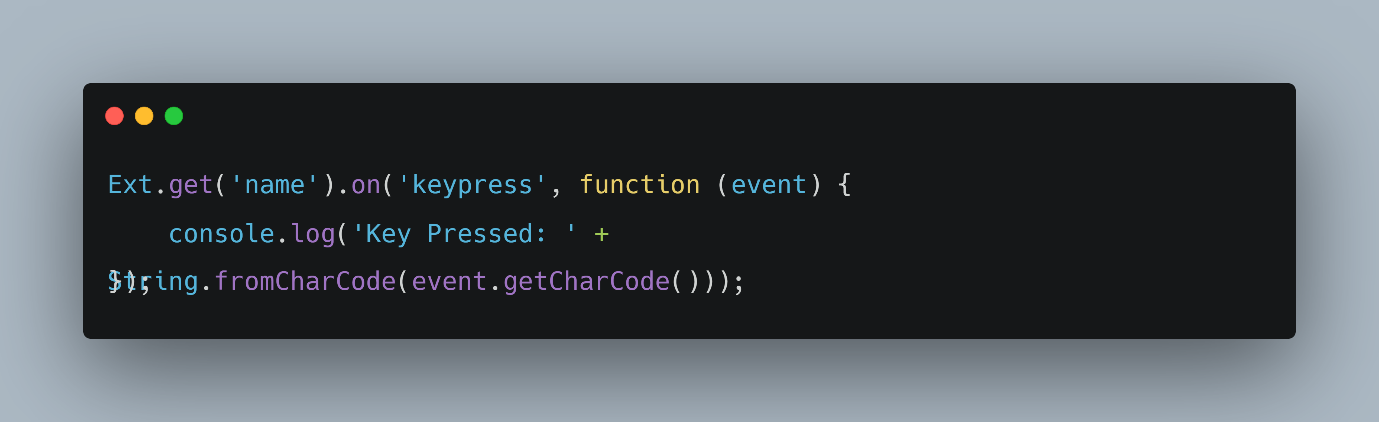
* Returns the key code of a pressed key.
* Example



Pressing "Enter" might log

Key Down: 13

**5.2 Example Usage**



**6. Enhancing User Experience with Animation**

Animations make interactions engaging and intuitive.

**6.1 Benefits of Animation**

* Provides visual feedback.
* Enhances user engagement.
* Improves retention and usability.

**6.2 Example Usage**



**7. Best Practices for Event Handling**

* 1. **Key Practices**

1. **Use Event Delegation for Dynamic Elements**

* Instead of adding event listeners to each element individually, attach the listener to a parent element and use delegation. This improves efficiency, especially for dynamically added elements.

1. **Keep Handlers Lightweight to Improve Performance**

* Avoid heavy computations inside event handlers. Keeping them optimized ensures a smooth user experience without unnecessary delays.

1. **Avoid Inline Event Handlers to Maintain Separation of Concerns**

* Writing event handlers directly in HTML (e.g., onclick="function()") makes code harder to maintain. Instead, define handlers separately in JavaScript for better organization.

1. **Use .on() for Better Flexibility Over Inline Event Attributes**

* The .on() method in Ext JS provides more control over event handling, allowing easy binding and unbinding of events when needed. This improves maintainability and scalability.

By implementing these practices, Ext JS applications can handle events efficiently while maintaining **scalability and performance**.

**8. Summary of Ext JS Event Handling**

* Events enable dynamic interactions in Ext JS applications.
* Understanding event types and propagation is essential.
* Managing event listeners efficiently improves performance.
* Using animations enhances the user experience.
* Following best practices ensures clean and maintainable code.

By mastering Ext JS event handling, We can build responsive and high-performing applications that provide a seamless user experience.

**: Thank You :**