Below is a **clean, corrected, internally accurate, exam-ready, and conceptually deep rewrite** of **JavaScript Window Events vs Document Events**, aligned with how browsers **actually work internally**.
I've removed misleading parts (like excessive alerts on scroll/resize), clarified **load vs DOMContentLoaded**, and explained **why & when** to use each.

🌐 JavaScript Window Events vs Document Events

(Deep + Internal + Practical)

1️⃣ What Are Window Events?

Window events are events that occur at the **browser window level**.
They are associated with the `window` **object**, which represents the **entire browser viewport**, not individual HTML elements.

📘 Exam Definition

> Window events are JavaScript events triggered by actions that affect the browser window as a whole, such as loading, resizing, scrolling, or closing the page.

2️⃣ Internal Working: How Window Events Fire 🧠

When a page is opened:

```
pgsql

Browser creates window object
        ↓
HTML parsing begins
        ↓
DOM constructed
        ↓
External resources load (CSS, images, fonts)
        ↓
window events fire (load, resize, scroll)
```

📌 Window events **do not belong to elements**
📌 They describe **global browser state**

3️⃣ Most Common Window Events

Event	When It Fires	Key Use
`load`	Page + all resources loaded	Final initialization
`beforeunload`	Just before leaving page	Warnings, confirmations

Event	When It Fires	Key Use
`unload`	Page is unloading	Cleanup (limited use)
`resize`	Window size changes	Responsive logic
`scroll`	Page is scrolled	Lazy loading, effects

4️⃣ `load` Event (Very Important ⚠️)

What It Means

- Fires **after**:
  - HTML parsed
  - CSS loaded
  - Images loaded
  - Fonts loaded
- **Slowest** lifecycle event

Example

```
html

<script>
window.addEventListener("load", () => {
 console.log("Page fully loaded");
});
</script>
```

When to Use

✔️ Measuring layout sizes
✔️ Canvas rendering
✔️ Image-dependent logic

❌ Avoid for normal DOM access

5️⃣ `resize` Event — Viewport Changes

Triggered whenever:

- Window resized
- Orientation changes (mobile)

⚠️ Important Performance Rule

Resize fires **many times per second**

👉 Always throttle or debounce in real apps

Example

```
js

window.addEventListener("resize", () => {
  console.log(window.innerWidth, window.innerHeight);
});
```

6️⃣ `scroll` Event — User Scrolling

Triggered when user scrolls the document.

Common Uses

- Infinite scrolling
- Lazy image loading
- Sticky headers
- Scroll animations

Example (Safe Version)

```
js

window.addEventListener("scroll", () => {
  if (window.scrollY > 300) {
    console.log("Scrolled down");
  }
});
```

📌 **Never use alert() in scroll** — it freezes UI

7️⃣ `beforeunload` Event — Exit Warning

Fires when:

- User refreshes
- User closes tab
- User navigates away

Example

```
js

window.addEventListener("beforeunload", (event) => {
  event.preventDefault();
```

```
  event.returnValue = "";
});
```

📌 Browser shows its **own message** (custom text ignored)

8️⃣ What Are Document Events?

Document events are events that occur **inside the HTML document**, related to:

- Elements
- User interaction
- DOM lifecycle

They are attached to the `document` **object**.

📘 Exam Definition

> Document events are DOM events that occur within the HTML document and allow JavaScript to respond to user interaction with page content.

9️⃣ DOMContentLoaded (EXTREMELY IMPORTANT ⚠️)

What It Means

- Fires when:
  - HTML fully parsed
  - DOM tree created
- **Does NOT wait** for images or CSS

Faster than `load`

Example

```
js

document.addEventListener("DOMContentLoaded", () => {
  console.log("DOM ready");
});
```

Best Practice

✔️ Use this instead of `load` for most scripts

🔥 `DOMContentLoaded` vs `load`

Feature	DOMContentLoaded	load
Waits for HTML	✅	✅

Feature	DOMContentLoaded	load
Waits for images	❌	✅
Speed	Fast	Slow
DOM access	✅	✅
Best practice	⭐⭐⭐	⭐

🔟 Common Document Events

Event	Trigger
`click`	Mouse click
`submit`	Form submission
`keydown`	Key pressed
`keyup`	Key released
`change`	Input changed
`input`	Typing occurs

1️⃣1️⃣ Document Event Example (Clean)

```html
<form id="form">
  <input id="name">
  <button>Submit</button>
</form>

<script>
document.addEventListener("DOMContentLoaded", () => {
  console.log("DOM loaded");
});

document.getElementById("form").addEventListener("submit", (e) => {
  e.preventDefault();
  console.log("Form submitted");
});
```

```
document.addEventListener("keydown", (e) => {
  console.log("Key:", e.key);
});
</script>
```

1️⃣2️⃣ Window vs Document — Core Differences (EXAM GOLD)

Aspect	Window Events	Document Events
Scope	Entire browser	HTML document
Object	`window`	`document`
Concern	Browser state	Page content
Examples	load, resize	click, submit
DOM required	❌	✅

1️⃣3️⃣ Event Flow Relationship

```
sql

User Action
  ↓
Document Event
  ↓
Event Bubbles
  ↓
Window
```

📌 Window events **do not bubble up from elements**

1️⃣4️⃣ Common Mistakes ❌

❌ Using `load` instead of `DOMContentLoaded`
❌ Heavy logic inside scroll/resize
❌ Using alerts inside global events
❌ Confusing window & document scope

1️⃣5️⃣ One-Line Interview Answers

Q: Difference between window and document events?

👉 Window controls browser state, document controls page content.

Q: Which is faster — load or DOMContentLoaded?

👉 DOMContentLoaded.

Q: Do window events bubble?

👉 No.

1️⃣6️⃣ Key Takeaways ✅

✔️ Window events = browser lifecycle
✔️ Document events = DOM interaction
✔️ Prefer DOMContentLoaded
✔️ Throttle scroll & resize
✔️ Use document for UI logic

⏭️ What Should We Do Next?

Best logical continuation:

1️⃣ **Event Bubbling vs Capturing (Deep Internals)**
2️⃣ **Event Delegation (Real-world patterns)**
3️⃣ **preventDefault vs stopPropagation**
4️⃣ **Custom Events (Advanced JS)**

👉 Type `next` and we continue step-by-step 🚀