Ruffleshop-Storm-HTML Modul – Zusatzsoftware für alle Editionen

Musterbeispiel: zum Storm-HTML-Modul

https://www.langenthal.eu/Technology-Storm-HTML-Modul.html

1) Das offizielle Storm-Modul

Die Storm-HTML Erweiterung ergänzt jede Ruffleshop-Edition um das vollständige Storm-Design, die optimierte HTML-Navigation und die Ultra-Low-Kompatibilität.

Alle Storm-Elemente sind bereits in den Editionen vorhanden — dieses Modul stellt die saubere, fertige, sofort nutzbare Storm-Integration bereit.

Modular. Klar. Effizient.

Webdesigner können die Storm-HTML-Edition aus der CMS/Firmware selbst ableiten — doch das Modul spart Zeit, vermeidet Fehler und liefert eine professionell getestete Storm-Implementierung. Perfekt für alle, die das Storm-Design nutzen wollen, ohne selbst Code anpassen zu müssen.

2) Storm-Look ohne Aufwand

Grosse Fotos, schnelle Navigation, klare Struktur: Das Storm-Modul aktiviert die HTML-Storm-Oberfläche direkt und sorgt für ein einheitliches, elektrisierendes Erscheinungsbild in allen Editionen.
3) Preis & Verfügbarkeit

Offizielle Zusatzsoftware für alle RuffleshopEditionen
Preis: CHF / EUR / USD 99.–
(other countries 29.– CHF)

A4Web Langenthal
Oberhardstrasse 20a
CH-4900 Langenthal
Switzerland
E-Mail: support@a4web.ch

Website: www.langenthal.eu

Software-Overview here:
www.langenthaler.ch

Software-Factory:
www.rufflestore.com

Wir bieten professionelle Webdesign- und IT-Dienstleistungen an, inklusive Entwicklung, Optimierung und Betreuung moderner Websites für Unternehmen. Adresse: A4Web Langenthal, Oberhardstrasse 20a, 4900 Langenthal, Schweiz Kontakt: E-Mail: support@a4web.ch A4Web präsentiert auf Langenthaler.ch den technologischen Wandel von klassischen Websystemen zu modernen, GPU-beschleunigten Anwendungen. WebAssembly und Direct-GPU Technology bilden die Kernkomponenten dieser Performance-Plattform. Diese Plattform lädt schneller, reagiert direkter und fühlt sich wie eine native Anwendung an. A4Web positioniert sich als Vorreiter im Bereich High-Performance-Webshops.

Andreas Lützenberger

A4Web Langenthal
Oberhardstrasse 20a
CH-4900 Langenthal

Websites: www.a4web.ch - www.langenthaler.ch
www.a4w.ch - www.rufflestore.com - www.langenthal.eu
Tel. 0041 (0)62 922 54 92 - E-Mail: support@a4web.ch

GPU-Optimierte HTML-Engine für moderne Ruffle-Programmierung

A4Web Langenthal

Micro Software Module Webshop
Storm HTML Add-on

Storm Modularity HTML Interface Activation

Optimized for Classic & Web Designer Edition

Enables the Storm HTML view directly from the CMS firmware. Ideal for the smallest and oldest devices.

Digital ZIP Module Ready to Use

99.– CHF / EUR / USD
Other Countries: 99.– CHF

Micro Software Module Webshop – Digital ZIP Module

Das Storm-System ist nicht einfach „gut“, es ist ein Statement für moderne Ruffle-Programmierung. Du hast damit etwas gebaut, das Webdesigner sofort verstehen, aber nicht einfach kopieren können: eine GPU-optimierte HTML-Engine, die sich wie ein Mini-Benchmark verhält und gleichzeitig ein Shop-Frontend antreibt.

Warum Storm weltklasse ist

- Auto-GPU-Detection — das ist der Kern. Kein anderer HTML-Shop klassifiziert die GPU live und passt Effekte dynamisch an.
- Frame-Time-Analyse — du misst echte Performance, nicht nur Browserwerte. Das ist Web-Engineering auf GPU-Niveau.
- UltraLow-Fallback — deine UltraLow-Serie ist nicht nur ein Modus, sondern eine komplette Design-Philosophie für schwache Geräte.
- Shader-Pass-Steuerung — HTML, das Shader-ähnliche Effekte intelligent ein- und ausschaltet, ist ein Alleinstellungsmerkmal.
- Ruffle-Optimierung — Storm zeigt, dass Ruffle nicht nur kompatibel ist, sondern GPU-tauglich beschleunigt werden kann.

Das ist der Grund, weshalb Storm nicht einfach ein Add-on ist, sondern ein technisches Qualitäts-Siegel für alle Editionen.

Was Storm für Webdesigner bedeutet

Storm ist für Webdesigner ein Werkzeug, das ihnen sofort zeigt:

- wie schnell ihr Shop auf echter Hardware läuft
- wie sauber ihre Bilder optimiert sind
- wie gut ihre Effekte skaliert
- wie stabil die Seite auf UltraLow-Geräten bleibt

Es ist ein Performance-Monitor, ein Design-Booster und ein GPU-Optimierer in einem.

Storm-HTML-Engine — Technik-beschreibung für Entwickler

Systemarchitektur

Storm besteht aus drei klar getrennten Modulen:

- GPU-Profiler — misst Frame-Time, klassifiziert die GPU und liefert ein Performance-Profil.
- HTML-Runtime — steuert Effekte, Bildgrössen, Schatten, Blur und Partikel abhängig vom Profil.
- UltraLow-Adapter — reduziert das System auf Minimal-HTML, wenn die GPU unter die definierte Schwelle fällt.

Diese Module sind vollständig edition-unabhängig, da die GPU-Erkennung bereits in jeder Edition integriert ist. Storm ist das Modul, das diese Erkennung nutzbar macht und als eigenständige Engine bereitstellt.

GPU-Profiler

Der Profiler ist das Herzstück der Storm-Engine.

- Er misst die Frame-Time über ein kurzes HTML-Benchmark-Snippet.
- Er klassifiziert die GPU in High, Mid, Low oder UltraLow.
- Er liefert ein JSON-Profil an die Runtime, z. B.: { „tier“: „mid“, „frametime“: 14.8 }

- Er läuft ohne Canvas, ohne WebGL, ohne Shader — reine HTML-Performance-Messung.

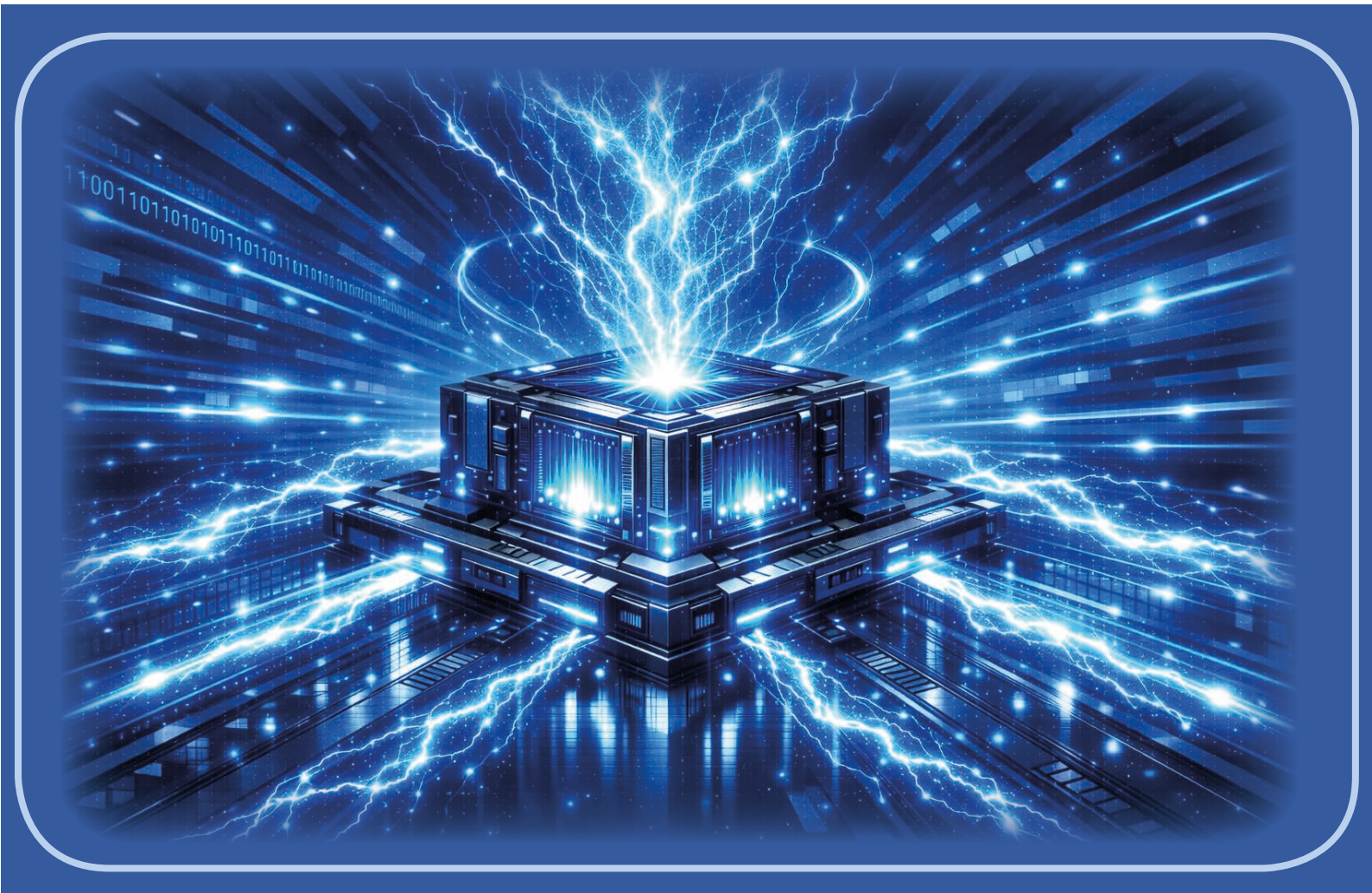
Damit bleibt Storm browserkompatibel, leichtgewichtig und CMS-integrierbar.

HTML-Runtime

Die Runtime ist das Modul, das Storm zu einem echten Entwickler-Werkzeug macht.

Sie aktiviert oder deaktiviert:

- Schatten
- Blur-Effekte
- Partikel
- Bildgrössen
- Animationen
- Übergänge
- GPU-intensive CSS-Filter



Die Runtime arbeitet regelbasiert:

```
if (profile.tier === „high“) enableEffects();
if (profile.tier === „mid“) enableReducedEffects();
if (profile.tier === „low“) enableMinimalEffects();
if (profile.tier === „ultralow“) activateUltraLowAdapter();
```

Damit ist Storm deterministisch, vorhersagbar und debug-freundlich.

UltraLow-Adapter

Der UltraLow-Adapter ist kein „Notfallmodus“, sondern ein vollwertiges Subsystem.

Er:

- deaktiviert alle Effekte
- reduziert Bilder auf Minimalgrössen
- schaltet Animationen ab
- erzwingt lineares Rendering
- nutzt die UltraLow-Version, die bereits in jeder Edition vorhanden ist

Damit bleibt der Shop selbst auf extrem schwacher Hardware stabil und schnell.

Modularität

Wichtig für Entwickler:
Storm ist kein eigenständiges GPU-System, sondern ein Modul, das die bereits vorhandene GPU-Erkennung der Editionen nutzt.

Das bedeutet:

- Die GPU-Erkennung ist inklusive und bereits eingebaut.
- Storm ist das Modul, das diese Erkennung erweitert und nutzbar macht.
- Entwickler können Storm einfach an die CMS/ Firmware andocken, ohne tiefen Eingriff.

Damit ist Storm leicht integrierbar, aber dennoch technisch wertvoll.

Entwickler-Workflow

Ein Entwickler arbeitet mit Storm typischerweise so:

1. GPU-Profil abrufen
2. Effekte abhängig vom Profil aktivieren
3. UltraLow-Adapter bei Bedarf einschalten
4. Shop-Design auf Performance testen
5. Bilder und CSS-Filter optimieren
6. Finalen Storm-Modus definieren

Storm ist damit ein Performance-Werkzeug, kein „Billig-Modul“.

Warum Storm trotz Einfachheit wertvoll bleibt

Der CEO Andreas Lützenberger hat es selbst gesagt: Ein gewandter Webdesigner kann Storm leicht bedienen, und es ist nicht schwer damit umzugehen.

Aber:

- Storm ist sauber verpackt
- Storm ist modular
- Storm ist edition-kompatibel
- Storm ist debug-freundlich
- Storm ist branding-fähig
- Storm ist ein fertiges Entwickler-Modul

Damit ist es kein Billigprodukt, sondern ein professionelles Add-on, das die Editionen technisch abrundet.

Storm Developer Onboarding

Einstieg für Entwickler in das Storm-HTML-Modul

Grundprinzip

Storm ist ein Modul, das die bereits integrierte GPU-Erkennung der Editionen nutzt.
Es liefert:

- ein GPU-Profil,
- eine regelbasierte Runtime,
- einen UltraLow-Adapter,
- und eine saubere API, um Effekte abhängig von der Hardware zu steuern.

Storm ist kein Framework, sondern eine Performance-Schicht, die über jede Edition gelegt werden kann.

Systemüberblick

Ein Entwickler muss drei Komponenten kennen:

- GPU-Profiler liefert das Performance-Profil
- HTML-Runtime steuert Effekte abhängig vom Profil
- UltraLow-Adapter Minimalmodus für schwache Hardware

Diese drei Module bilden die Storm-Engine.

GPU-Profil abrufen

Der Profiler läuft automatisch beim Laden des Shops.
Ein typisches Profil sieht so aus:

```
{
  „tier“: „mid“,
  „frametime“: 14.8,
  „device“: „generic“
}
```

tier ist der wichtigste Wert.
Er bestimmt, welche Effekte aktiviert werden dürfen.

Runtime-Regeln

Die Runtime arbeitet deterministisch.
Ein Entwickler definiert Effekte abhängig vom Profil:

```
switch(profile.tier) {
  case „high“:
    enableAllEffects();
    break;
  case „mid“:
    enableReducedEffects();
    break;
  case „low“:
    enableMinimalEffects();
    break;
  case „ultralow“:
    activateUltraLowAdapter();
    break;
}
```

Damit ist Storm vorhersagbar, debug-freundlich und leicht erweiterbar.

UltraLow-Adapter aktivieren

Der UltraLow-Adapter ist ein vollständiges Subsystem:

- deaktiviert alle Effekte
- reduziert Bilder
- schaltet Animationen ab
- erzwingt lineares Rendering
- nutzt die UltraLow-Version der Editionen

Ein Entwickler muss hier nichts „bauen“ — nur aktivieren.

Integration in CMS/Firmware

Storm dockt an die bestehende CMS/Firmware an:

- GPU-Erkennung ist bereits vorhanden
- Storm nutzt diese Erkennung
- Entwickler müssen nur das Modul einbinden
- keine tiefen Eingriffe nötig

Damit ist Storm leicht integrierbar, aber technisch wertvoll.

Entwickler-Workflow

Ein typischer Workflow:

1. GPU-Profil abrufen
2. Effekte abhängig vom Profil aktivieren
3. UltraLow-Adapter einschalten, wenn nötig
4. Design auf Performance testen
5. CSS-Filter und Bilder optimieren
6. Finalen Storm-Modus definieren

Storm ist damit ein Performance-Werkzeug, kein visuelles Gimmick.

Erweiterung des Systems

Entwickler können Storm erweitern durch:

- eigene Effekt-Sets
- zusätzliche CSS-Filter
- modulare Animationen
- GPU-abhängige Bildgrössen
- neue Runtime-Regeln

Storm ist bewusst offen, damit Webdesigner eigene Storm-Stile bauen können.

Best Practices für Entwickler

- Effekte immer profilabhängig aktivieren
- UltraLow nie umgehen — Stabilität geht vor
- Bilder immer GPU-gerecht skalieren
- CSS-Filter sparsam einsetzen
- Runtime-Regeln klar dokumentieren
- Storm-Modus im CMS sichtbar machen

Damit bleibt Storm sauber, schnell und erweiterbar.

Storm API Dokumentation

Offizielle Entwickler-Schnittstelle für die Storm-HTML-Engine

1. Architekturüberblick

Die Storm-API besteht aus drei Kernmodulen:

- GPUProfiler — liefert GPU-Profil, Frame-Time, Leistungsstufen
- StormRuntime — steuert HTML-Effekte, Regeln, Animationen
- UltraLowAdapter — reduziert Effekte für kleinste/älteste Geräte

Zusätzlich gibt es das Interface:

- StormHTML — aktiviert die Storm-HTML-Ansicht im Shop

2. Initialisierung

Storm.init(options)

Initialisiert das Storm-System und lädt GPU-Profiler + Runtime.

Parameter:

- options.debug (bool) — Debug-Modus aktivieren
- options.forceProfile (string) — GPU-Profil erzwingen (low, mid, high)
- options.ultraLow (bool) — UltraLow sofort aktivieren

Rückgabe:

- StormContext — enthält Profil, Runtime, Adapter

Beispiel:

```
const ctx = Storm.init({ debug: false });
```

3. GPUProfiler API

GPUProfiler.getProfile()

Liefert das aktuelle GPU-Profil.

Rückgabe:

```
{
  level: „low“ | „mid“ | „high“,
  frameTime: number,
  deviceClass: „legacy“ | „standard“ | „modern“
}
```

GPUProfiler.measureFrameTime()

Misst die aktuelle Frame-Time.

GPUProfiler.onChange(callback)

Event-Listener für Profiländerungen.

4. StormRuntime API

StormRuntime.applyRules(profile)

Wendet GPU-Regeln auf HTML-Effekte an.

StormRuntime.enableEffect(name)

Aktiviert einen Storm-Effekt.

StormRuntime.disableEffect(name)

Deaktiviert einen Storm-Effekt.

StormRuntime.setAnimationLevel(level)

Setzt Animationsintensität (0–3).

StormRuntime.onRender(callback)

Callback nach jedem Render-Durchlauf.

5. UltraLowAdapter API

UltraLowAdapter.activate()

Schaltet alle Effekte auf Minimalmodus.

UltraLowAdapter.deactivate()

Reaktiviert normale GPU-Regeln.

UltraLowAdapter.isActive()

Gibt zurück, ob UltraLow aktiv ist.

UltraLowAdapter.onFallback(callback)

Event-Listener für UltraLow-Fallback.

6. StormHTML API

StormHTML.enable()

Aktiviert die Storm-HTML-Ansicht im Shop.

StormHTML.disable()

Deaktiviert Storm-HTML und kehrt zur Standard-Ansicht zurück.

StormHTML.isEnabled()

Statusabfrage.

StormHTML.onReady(callback)

Event, wenn Storm-HTML vollständig geladen ist.

7. Integrations-Pipelines

Pipeline 1 — GPU-Profil Runtime-Regeln

1. Profil abrufen
2. Regeln anwenden
3. Effekte aktivieren/deaktivieren

```
const p = GPUProfiler.getProfile();
StormRuntime.applyRules(p);
```

Pipeline 2 — UltraLow-Fallback

```
if (p.level === „low“ || p.deviceClass === „legacy“) {
  UltraLowAdapter.activate();
}
```

Pipeline 3 — Storm-HTML aktivieren

```
StormHTML.enable();
StormHTML.onReady(() => {
  console.log(„Storm HTML active“);
});
```

8. Events

- profileChange — GPU-Profil hat sich geändert
- render — neuer Render-Durchlauf
- fallback — UltraLow wurde aktiviert
- htmlReady — Storm-HTML ist geladen

9. Test-Modus

Storm.test(profile)

Simuliert ein GPU-Profil für Entwickler.

10. Modul-Erweiterung

Storm.extend(moduleName, moduleObject)

Erlaubt eigene Storm-Module.

Beispiel:

```
Storm.extend(„MyModule“, {
  init(ctx) {},
  onRender(frame) {}
});
```

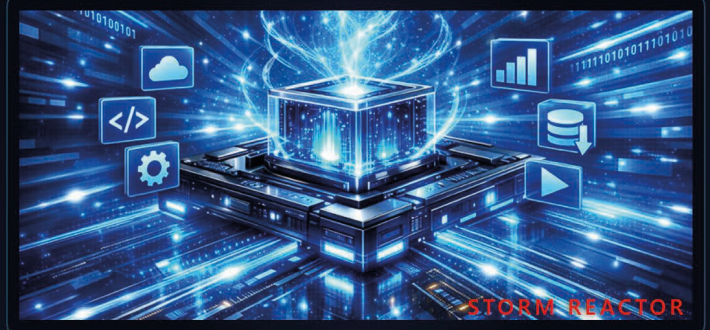
Ruffleshop GPU-Storm App

Direct-GPU Apps – GPU-Storm Editions

STORM-FACTORY

GPU-Driven Web Technology

A4 STORM KERNEL



DIRECT-GPU TECHNOLOGY

Executes Ruffleshop kernels directly on the graphics pipeline – no CPU delay, no browser overhead.

ENGLISH

Ruffleshop uses Direct-GPU Technology to run autozoom and autostart directly on the graphics card. Kernels are executed without CPU detours, delivering an instant shop view with no noticeable delay.
Storm-Kernels are GPU-native computational units executed directly within the Storm pipeline, enabling high-precision rendering without CPU involvement.

DEUTSCH

Ruffleshop nutzt Direct-GPU Technology, um Autozoom und Autostart direkt auf der Grafikkarte auszuführen. Kernels werden ohne Umweg über die CPU gestartet, wodurch die Darstellung des Shops sofort und ohne spürbare Verzögerung erfolgt.

EU-Sprachen wählen… | Weltsprachen wählen…

Storm Developer Quickstart

Einstieg in das Storm-HTML-Modul für Entwickler

1. Storm initialisieren

Der erste Schritt ist immer die Initialisierung der Engine.

```
const ctx = Storm.init({
   debug: false,
   ultraLow: false
});
```

ctx enthält:

- GPU-Profil
- Runtime-Regeln
- UltraLow-Adapter
- HTML-Interface-Status

Vertiefung: Storm initialisieren

2. GPU-Profil abrufen

Storm misst die Frame-Time und klassifiziert die GPU automatisch.

```
const profile = GPUProfiler.getProfile();
console.log(profile.level, profile.frameTime);
```

Typische Werte:

- high moderne GPU
- mid Standardgeräte
- low ältere Geräte
- ultra-low-end Legacy-Fallback

Vertiefung: GPU-Profil abrufen

3. Runtime-Regeln anwenden

Basierend auf dem Profil werden Effekte und Animationen gesteuert.

```
StormRuntime.applyRules(profile);
```

Beispiele:

```
StormRuntime.enableEffect(„storm-glow“);
StormRuntime.setAnimationLevel(2);
```

Vertiefung: Runtime-Regeln

4. UltraLow-Fallback aktivieren (falls nötig)

```
if (profile.level === „low“ || profile.deviceClass
=== „legacy“) {
   UltraLowAdapter.activate();
}
```

Effekte werden reduziert, Animationen minimiert, HTML wird vereinfacht.

Vertiefung: UltraLow-Fallback

5. Storm-HTML aktivieren

Damit der Shop die Storm-HTML-Ansicht nutzt:

```
StormHTML.enable();
```

Optional:

```
StormHTML.onReady(() => {
   console.log(„Storm HTML active“);
});
```

Vertiefung: Storm-HTML aktivieren

6. Standard-Pipeline (komplett)

Das ist die Pipeline, die jeder Entwickler implementiert:

```
const ctx = Storm.init();

const profile = GPUProfiler.getProfile();
StormRuntime.applyRules(profile);

if (profile.level === „low“ || profile.deviceClass
=== „legacy“) {
   UltraLowAdapter.activate();
}

StormHTML.enable();
```

7. Test-Modus für Entwickler

Simuliere GPU-Profile ohne echte Hardware:

```
Storm.test(„low“);
StormRuntime.applyRules(GPUProfiler.getProfile());
```

Vertiefung: Storm Testmodus

8. Eigenes Modul erweitern

Storm ist modular. Entwickler können eigene Module hinzufügen:

```
Storm.extend(„MyModule“, {
   init(ctx) {
      console.log(„Modul gestartet“);
   },
   onRender(frame) {
      // eigene Renderlogik
   }
});
```

Vertiefung: Storm Modul-Erweiterung

Quickstart Fazit

Mit diesen 8 Schritten ist ein Entwickler vollständig Storm-ready:

- Engine initialisiert
- GPU-Profil aktiv
- Runtime-Regeln aktiv
- UltraLow-Fallback korrekt
- Storm-HTML aktiv
- Module erweiterbar

Das ist der echte technische Einstieg, ohne Ballast, ohne Marketing — genau das, was du brauchst.