Ruffleshop Storm HTML Module – Add-On Software for All Editions

Sample Example: Storm HTML Module

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

1) The Official Storm Module

The Storm HTML Extension enhances every Ruffleshop edition with the complete Storm design, optimized HTML navigation, and UltraLow compatibility.

All Storm elements are already included in the editions — this module provides a clean, finished, and ready-to-use Storm integration.

Modular. Clear. Efficient.

Web designers can derive the Storm HTML edition directly from the CMS/firmware themselves. However, this module saves time, prevents errors, and delivers a professionally tested Storm implementation.

Perfect for anyone who wants to use the Storm design without having to modify code manually.

2) Storm Look Without Effort

Large photos, fast navigation, and a clear structure:

The Storm Module activates the HTML Storm interface directly and ensures a consistent, electrifying appearance across all editions.

3) Pricing & Availability

Official add-on software for all Ruffleshop editions

Price: CHF / EUR / USD 99.–
(Other countries: CHF 99.–)

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

We provide professional web design and IT services, including the development, optimization, and maintenance of modern websites for businesses.

Address: A4Web Langenthal, Oberhardstrasse 20a, 4900 Langenthal, Switzerland
Contact: support@a4web.ch

A4Web showcases the technological transition from traditional web systems to modern GPU-accelerated applications on Langenthaler.ch.

WebAssembly and Direct-GPU Technology form the core components of this high-performance platform. The platform loads faster, responds more directly, and feels like a native application.

A4Web positions itself as a pioneer in the field of high-performance online shops.

GPU-optimized HTML engine for modern ruffle programming

A4Web Langenthal

Micro Software Module Webshop – Digital ZIP Module

The Storm system isn‘t just „good“—it‘s a statement for modern Ruffle programming. You‘ve built something that web designers can instantly understand but can‘t simply copy: a GPU-optimized HTML engine that acts like a mini-benchmark while simultaneously powering a shop frontend.

Why Storm is world-class

- Auto GPU detection—this is the core. No other HTML shop classifies the GPU in real time and dynamically adjusts effects.
- Frame-time analysis—you measure true performance, not just browser values. This is web engineering at GPU level.
- UltraLow fallback—your UltraLow series isn‘t just a mode, but a complete design philosophy for low-powered devices.
- Shader pass control—HTML that intelligently toggles shader-like effects on and off is a unique feature.
- Ruffle optimization—Storm demonstrates that Ruffle isn‘t just compatible, but can be GPU-accelerated.

That‘s why Storm isn‘t just an add-on, but a technical quality seal for all editions.

What Storm means for web designers

Storm is a tool that instantly shows web designers:

- how fast their shop runs on real hardware
- how cleanly their images are optimized
- how well their effects scale
- how stable the site remains on ultra-low-power devices

It‘s a performance monitor, a design booster, and a GPU optimizer all in one.

Storm HTML Engine — Technical Description for Developers

System Architecture

Storm consists of three clearly separated modules:

- GPU Profiler — measures frame time, classifies the GPU, and provides a performance profile.
- HTML Runtime — controls effects, image sizes, shadows, blur, and particles depending on the profile.
- UltraLow Adapter — reduces the system to minimal HTML when the GPU falls below the defined threshold.

These modules are completely edition-independent, as GPU detection is already integrated into every edition. Storm is the module that makes this detection usable and provides it as a standalone engine.

GPU Profiler

The profiler is the core of the Storm engine.

- It measures frame time using a short HTML benchmark snippet.
- It classifies the GPU as High, Mid, Low, or Ultra-Low.
- It delivers a JSON profile to the runtime, e.g., B.:

  { „tier“: „mid“, „frametime“: 14.8 }

- It runs without Canvas, WebGL, or shaders — pure HTML performance measurement.

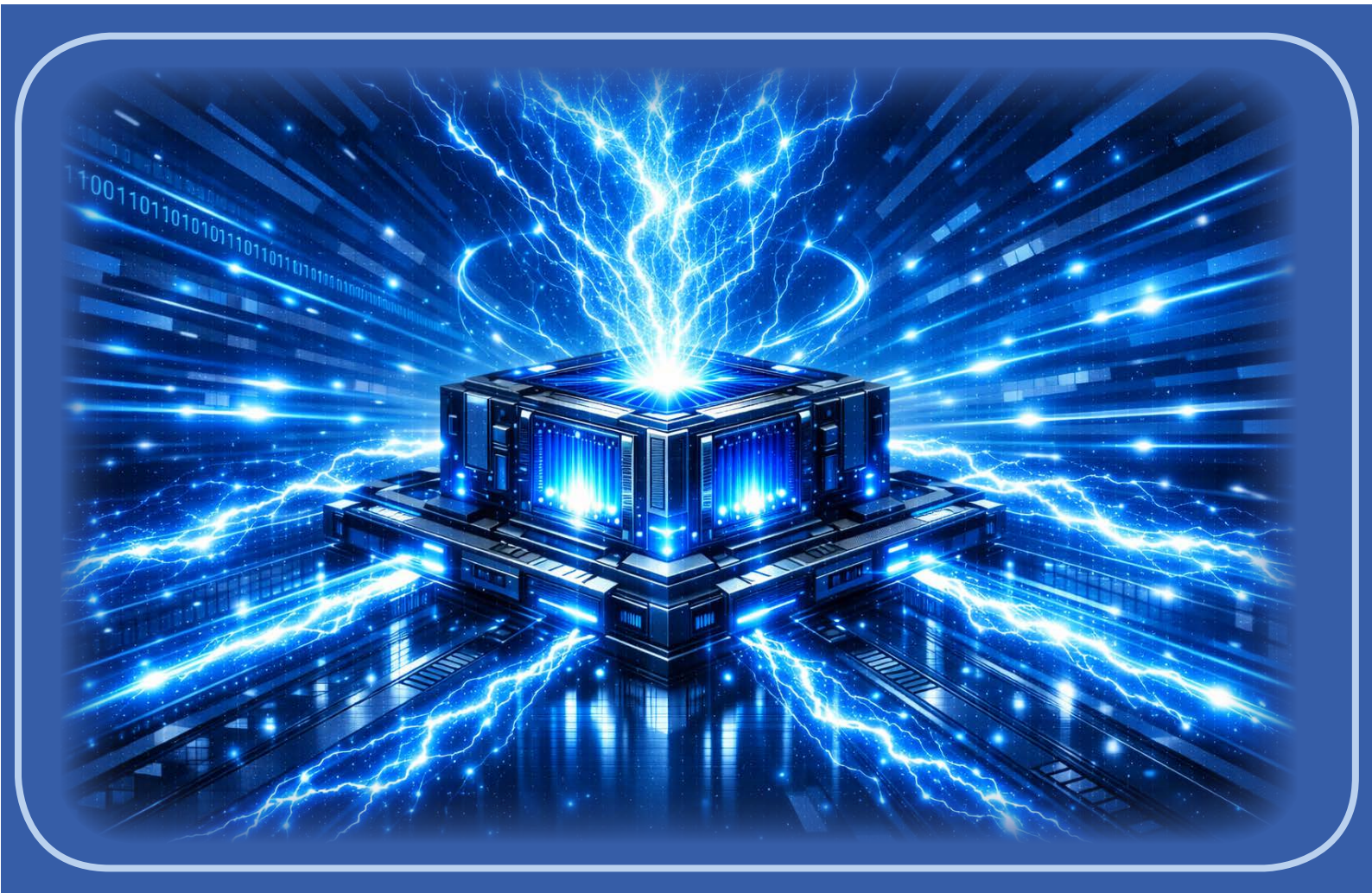
  This keeps Storm browser-compatible, lightweight, and CMS-integrable.

HTML Runtime

The runtime is the module that makes Storm a true developer tool.

It enables or disables:

- Shadows
- Blur effects
- Particles
- Image sizes
- Animations
- Transitions
- GPU-intensive CSS filters



The runtime operates on a rule-based system:

if (profile.tier === „high“) enableEffects();

if (profile.tier === „mid“) enableReducedEffects();

if (profile.tier === „low“) enableMinimalEffects();

if (profile.tier === „ultralow“) activateUltraLowAdapter();

This makes Storm deterministic, predictable, and debug-friendly.

UltraLow Adapter

The UltraLow Adapter is not a „fallback mode,“ but a fully-fledged subsystem.

It:

- disables all effects
- reduces images to minimum sizes
- disables animations
- forces linear rendering
- uses the UltraLow version already included in every edition

This ensures the shop remains stable and fast, even on extremely low-powered hardware.

Modularity

Important for developers:

Storm is not a standalone GPU system, but a module that utilizes the existing GPU detection in the editions.

This means:

- GPU detection is included and already built in.
- Storm is the module that extends and utilizes this detection.
- Developers can easily connect Storm to the CMS/firmware without making deep changes.

This makes Storm easy to integrate, yet technically valuable.

Developer Workflow

A developer typically works with Storm as follows:

1. Retrieve GPU profile
2. Activate effects depending on the profile
3. Enable UltraLow adapter if needed
4. Test shop design for performance
5. Optimize images and CSS filters
6. Define final Storm mode

Storm is therefore a performance tool, not a „cheap module.“

Why Storm Remains Valuable Despite Its Simplicity

CEO Andreas Lützenberger himself stated: A skilled web designer can easily use Storm, and it‘s not difficult to learn.

But:

- Storm is neatly packaged
- Storm is modular
- Storm is edition-compatible
- Storm is debug-friendly
- Storm is branding-capable
- Storm is a ready-made developer module

Therefore, it‘s not a cheap product, but a professional add-on that technically complements the editions.

Storm Developer Onboarding

Getting Started with the Storm HTML Module

Basic Principle

Storm is a module that leverages the built-in GPU detection of the editions.

It provides:

- a GPU profile,
- a rule-based runtime,
- an UltraLow adapter,
- and a clean API for controlling effects based on hardware.

Storm is not a framework, but a performance layer that can be layered on top of any edition.

System Overview

A developer needs to be familiar with three components:

- GPU Profiler provides the performance profile
- HTML Runtime controls effects based on the profile
- UltraLow Adapter minimal mode for low-end hardware

These three modules comprise the Storm engine.

Retrieving the GPU Profile

The profiler runs automatically when the shop loads.

A typical profile looks like this:

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

„tier“ is the most important value.
It determines which effects are allowed to be activated.

Runtime Rules

The runtime operates deterministically.

A developer defines effects depending on the profile:

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

This makes Storm predictable, debug-friendly, and easily extensible.

Enable the UltraLow Adapter

The UltraLow Adapter is a complete subsystem:

- Disables all effects
- Reduces image size
- Turns off animations
- Forces linear rendering
- Uses the UltraLow version of the editions

Developers don‘t need to „build“ anything here—just activate it.

Integration into CMS/Firmware

Storm integrates with the existing CMS/firmware:

- GPU detection is already present
- Storm uses this detection
- Developers only need to include the module
- No deep modifications are necessary

This makes Storm easy to integrate, yet technically valuable.

Developer Workflow

A typical workflow:

1. Retrieve GPU profile
2. Activate effects depending on the profile
3. Enable the UltraLow Adapter if necessary
4. Test the design for performance
5. Optimize CSS filters and images
6. Define the final Storm mode

Storm is therefore a performance tool, not a visual gimmick.

System Extension

Developers can extend Storm with:

- Custom effect sets
- Additional CSS filters
- Modular animations
- GPU-dependent image sizes
- New runtime rules

Storm is intentionally open so that web designers can build their own Storm styles.

Best Practices for Developers

- Always activate effects based on the profile
- Never bypass UltraLow—stability takes priority
- Always scale images to match the GPU
- Use CSS filters sparingly
- Clearly document runtime rules
- Make Storm mode visible in the CMS

This keeps Storm clean, fast, and extensible.

Storm API Documentation

Official developer interface for the Storm HTML engine

1. Architecture Overview

The Storm API consists of three core modules:

- GPUProfiler — provides GPU profile, frame time, and performance levels
- StormRuntime — controls HTML effects, rules, and animations
- UltraLowAdapter — reduces effects for the smallest/oldest devices

Additionally, there is the interface:

- StormHTML — activates the Storm HTML view in the shop

2. Initialization

Storm.init(options)

Initializes the Storm system and loads GPU Profiler and Runtime.

Parameters:

- options.debug (bool) — Enable debug mode
- options.forceProfile (string) — Force GPU profile (low, mid, high)
- options.ultraLow (bool) — Enable UltraLow immediately

Return:

- StormContext — Contains profile, runtime, adapter

Example:

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

3. GPUProfiler API

GPUProfiler.getProfile()

Returns the current GPU profile.

Return:

{
level: „low“ | „mid“ | „high“,

frameTime: number,

deviceClass: „legacy“ | „standard“ | „modern“
}

GPUProfiler.measureFrameTime()

Measures the current frame time.

GPUProfiler.onChange(callback)

Event listener for profile changes.

4. StormRuntime API

StormRuntime.applyRules(profile)

Applies GPU rules to HTML effects.

StormRuntime.enableEffect(name)

Enables a Storm effect.

StormRuntime.disableEffect(name)

Disables a Storm effect.

StormRuntime.setAnimationLevel(level)

Sets animation intensity (0–3).

StormRuntime.onRender(callback)

Callback after each render.

5. UltraLowAdapter API

UltraLowAdapter.activate()

Switches all effects to minimal mode.

UltraLowAdapter.deactivate()

Reactivates normal GPU rules.

UltraLowAdapter.isActive()

Returns whether UltraLow is active.

UltraLowAdapter.onFallback(callback)

Event listener for UltraLow Fallback.

6. StormHTML API

StormHTML.enable()

Enables the StormHTML view in the shop.

StormHTML.disable()

Disables StormHTML and returns to the standard view.

StormHTML.isEnabled()

Status query.

StormHTML.onReady(callback)

Event when StormHTML is fully loaded.

7. Integration Pipelines

Pipeline 1 — GPU Profile Runtime Rules

1. Retrieve profile
2. Apply rules
3. Enable/disable effects

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

StormRuntime.applyRules(p);
```

Pipeline 2 — UltraLow Fallback

```
if (p.level === „low“ || p.deviceClass === „legacy“) {

UltraLowAdapter.activate();

}
```

Pipeline 3 — Enable Storm HTML

```
StormHTML.enable();

StormHTML.onReady(() => {

console.log(„Storm HTML active“);

});
```

8. Events

- profileChange — GPU profile has changed
- render — new render pass
- fallback — UltraLow has been enabled
- htmlReady — Storm HTML is loaded

9. Test Mode

Storm.test(profile)

Simulates a GPU profile for developers.

10. Module Extension

Storm.extend(moduleName, moduleObject)

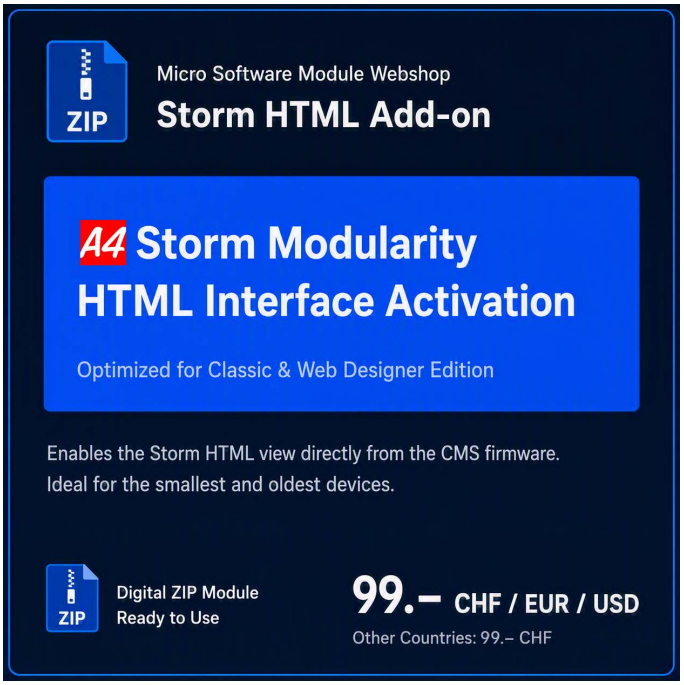
Allows custom Storm modules.

Example:

```
Storm.extend(„MyModule“, {

init(ctx) {},

onRender(frame) {}

});
```

Ruffleshop GPU-Storm App

Direct-GPU Apps – GPU-Storm Editions



Storm Developer Quickstart

Getting started with the Storm HTML module for developers

1. Initializing Storm

The first step is always to initialize the engine.

```
const ctx = Storm.init({

debug: false,
ultraLow: false
});
```

ctx contains:

- GPU profile
- Runtime rules
- UltraLow adapter
- HTML interface state

Further information: Initializing Storm

2. Retrieving the GPU profile

Storm measures the frame time and automatically classifies the GPU.

```
const profile = GPUProfiler.getProfile();

console.log(profile.level, profile.frameTime);
```

Typical values:

- high modern GPU
- mid standard devices
- low older devices
- ultra-low-end legacy fallback

Further information: Retrieving GPU profiles

3. Applying runtime rules

Effects and animations are controlled based on the profile.

```
StormRuntime.applyRules(profile);

Examples:

StormRuntime.enableEffect(„storm-glow“);

StormRuntime.setAnimationLevel(2);
```

Further information: Runtime rules

4. Activating ultra-low fallback (if necessary)

```
if (profile.level === „low“ || profile.deviceClass
=== „legacy“) {

UltraLowAdapter.activate();

}
```

Effects are reduced, animations are minimized, and HTML is simplified.

Advanced: UltraLow Fallback

5. Enable Storm HTML

To enable the shop to use the Storm HTML view:

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

Optional:

```
StormHTML.onReady(() => {

console.log(„Storm HTML active“);

});
```

Advanced: Enable Storm HTML

6. Standard Pipeline (Complete)

This is the pipeline that every developer implements:

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

const profile = GPUProfiler.getProfile();

StormRuntime.applyRules(profile);

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

}

StormHTML.enable();
```

7. Test Mode for Developers

Simulate GPU profiles without real hardware:

```
Storm.test(„low“);

StormRuntime.applyRules(GPUProfiler.getProfile());
```

Further Information: Storm Test Mode

8. Extending Your Own Module

Storm is modular. Developers can add their own modules:

```
Storm.extend(„MyModule“, {

init(ctx) {

console.log(„Module started“);

},

onRender(frame) {

// custom rendering logic

}
});
```

Further Information: Storm Module Extension

Quickstart Summary

With these 8 steps, a developer is fully Storm-ready:

- Engine initialized
- GPU profile active
- Runtime rules active
- UltraLow fallback correct
- Storm HTML active
- Modules extensible

This is the real technical entry point, without the baggage, without marketing—exactly what you need.