 Countdown-Widget (ioBroker VIS + MagicMirror)

Dieses Widget zeigt automatisch die zwei nächsten anstehenden Countdowns an (z. B. Weihnachten, Silvester, Urlaub, Touren usw.).

Abgelaufene Countdowns werden automatisch ausgeblendet.

⚠ WICHTIGE HINWEISE (bitte lesen!)

JEDER Countdown muss im versteckten Subscribe-Block stehen

KEINE Einzeiler mit { … } im JavaScript

Kein Array / kein Objekt-Literal

Code-Formatierung NICHT verändern

Sonst erscheint:

SyntaxError: Unexpected token 'null'

 1. Subscribe-Block (Pflicht!)

Hier meldet VIS alle benötigten Datenpunkte an.

```
{countdown.0.countdowns.Weihnachten.fullJson}
{countdown.0.countdowns.Silvester.fullJson}
{countdown.0.countdowns.SommerurlaubItalien.fullJson}
```

 Für jeden neuen Countdown MUSS hier eine Zeile ergänzt werden!

 2. Anzeige-HTML (2 Slots)

```
<div id="cd_text_xmas" style="font-size:28px;font-weight:700;line-
height:1.15;">
  Lade...
</div>
```

```
<div style="height:18px;"></div>
```

```
<div id="cd_text_ny" style="font-size:28px;font-weight:700;line-
height:1.15;">
  Lade...
</div>
```

Die IDs bleiben absichtlich so – sie sind MagicMirror-erprobt.

 3. JavaScript – Finale Version

⚠ NICHT kürzen, NICHT umformatieren

```
<!-- ============================================================
     VIS Subscribe-Block (Pflicht!)
     Jede OID, die du unten nutzt, MUSS hier stehen.
     ============================================================ -->
<div style="display:none;">
  {countdown.0.countdowns.Weihnachten.fullJson}
  {countdown.0.countdowns.Silvester.fullJson}
  {countdown.0.countdowns.SommerurlaubItalien.fullJson}
</div>

<div style="text-align:center;">
  <div id="cd_text_xmas" style="font-size:28px;font-weight:700;line-
height:1.15;">
    Lade...
  </div>

  <div style="height:18px;"></div>

  <div id="cd_text_ny" style="font-size:28px;font-weight:700;line-
height:1.15;">
    Lade...
  </div>
</div>

<script>
(function ()
{
  /* ============================================================
     KONFIGURATION
     ============================================================ */

  const EMPTY_TEXT = "…";

  // Wenn TRUE: Stunden werden ausgeblendet, sobald days > 1
  // (für days === 1 werden Stunden weiterhin angezeigt)
  const HIDE_HOURS_IF_MORE_THAN_ONE_DAY = true;

  // Wieviele Events sind AKTIV konfiguriert (max 8)
  const EVENT_COUNT = 3;

  /* ============================================================
     EVENT-DEFINITIONEN
     TYPE:
       1 = Weihnachten (Speziallogik)
       2 = Silvester   (Speziallogik)
       3 = Generisch   (Titel + Emoji)
     ============================================================ */

  // ---------------- Event 1 ----------------
  const E1_OID   = "countdown.0.countdowns.Weihnachten.fullJson";
  const E1_TYPE  = 1;
```

```
const E1_TITLE = "";
const E1_EMOJI = "";

// ---------------- Event 2 ----------------
const E2_OID   = "countdown.0.countdowns.Silvester.fullJson";
const E2_TYPE  = 2;
const E2_TITLE = "";
const E2_EMOJI = "";

// ---------------- Event 3 ----------------
const E3_OID   = "countdown.0.countdowns.SommerurlaubItalien.fullJson";
const E3_TYPE  = 3;
const E3_TITLE = "Sommerurlaub Italien";
const E3_EMOJI = "☀□□□";

// ---------------- Event 4..8 (Vorlagen) ----------------
const E4_OID   = "";
const E4_TYPE  = 3;
const E4_TITLE = "";
const E4_EMOJI = "";

const E5_OID   = "";
const E5_TYPE  = 3;
const E5_TITLE = "";
const E5_EMOJI = "";

const E6_OID   = "";
const E6_TYPE  = 3;
const E6_TITLE = "";
const E6_EMOJI = "";

const E7_OID   = "";
const E7_TYPE  = 3;
const E7_TITLE = "";
const E7_EMOJI = "";

const E8_OID   = "";
const E8_TYPE  = 3;
const E8_TITLE = "";
const E8_EMOJI = "";

/* ============================================================
   HILFSFUNKTIONEN (VIS-safe formatiert)
   ============================================================ */

function safeNum(x, fallback)
{
  const n = Number(x);

  if (Number.isFinite(n))
  {
```

```
      return n;
    }

    return fallback;
  }

  function parseFullJson(val)
  {
    if (val === null || val === undefined || val === "")
    {
      return null;
    }

    try
    {
      let t = val;

      if (typeof t === "string")
      {
        t = t.trim();

        if (t.startsWith('"') && t.endsWith('"'))
        {
          t = JSON.parse(t);
        }

        return JSON.parse(t);
      }

      return t;
    }
    catch (e)
    {
      return null;
    }
  }

  function subline(s)
  {
    return "<br><span style='font-size:18px;font-
weight:400;opacity:.85'>(noch " + s + ")</span>";
  }

  function dayWord(n)
  {
    if (n === 1)
    {
      return "Tag";
    }

    return "Tagen";
```

```
}

function hourWord(n)
{
  if (n === 1)
  {
    return "Stunde";
  }

  return "Stunden";
}

/* ============================================================
   EVENT-ZUGRIFF (ohne Arrays/Objekte)
   ============================================================ */

function getOid(i)
{
  if (i === 1) return E1_OID;
  if (i === 2) return E2_OID;
  if (i === 3) return E3_OID;
  if (i === 4) return E4_OID;
  if (i === 5) return E5_OID;
  if (i === 6) return E6_OID;
  if (i === 7) return E7_OID;
  if (i === 8) return E8_OID;

  return "";
}

function getType(i)
{
  if (i === 1) return E1_TYPE;
  if (i === 2) return E2_TYPE;
  if (i === 3) return E3_TYPE;
  if (i === 4) return E4_TYPE;
  if (i === 5) return E5_TYPE;
  if (i === 6) return E6_TYPE;
  if (i === 7) return E7_TYPE;
  if (i === 8) return E8_TYPE;

  return 0;
}

function getTitle(i)
{
  if (i === 1) return E1_TITLE;
  if (i === 2) return E2_TITLE;
  if (i === 3) return E3_TITLE;
  if (i === 4) return E4_TITLE;
  if (i === 5) return E5_TITLE;
```

```
    if (i === 6) return E6_TITLE;
    if (i === 7) return E7_TITLE;
    if (i === 8) return E8_TITLE;

    return "";
  }

  function getEmoji(i)
  {
    if (i === 1) return E1_EMOJI;
    if (i === 2) return E2_EMOJI;
    if (i === 3) return E3_EMOJI;
    if (i === 4) return E4_EMOJI;
    if (i === 5) return E5_EMOJI;
    if (i === 6) return E6_EMOJI;
    if (i === 7) return E7_EMOJI;
    if (i === 8) return E8_EMOJI;

    return "";
  }

  /* ============================================================
     RESTZEIT / ABGELAUFEN
     ============================================================ */

  function remainingSeconds(val)
  {
    const d = parseFullJson(val);

    if (!d)
    {
      return 999999999;
    }

    if (d.total && d.total.seconds !== undefined)
    {
      return safeNum(d.total.seconds, 999999999);
    }

    const days  = safeNum((d.total && d.total.days  !== undefined) ?
d.total.days  : d.days, 0);
    const hours = safeNum((d.total && d.total.hours !== undefined) ?
d.total.hours : d.hours, 0);
    const mins  = safeNum((d.total && d.total.minutes !== undefined) ?
d.total.minutes : d.minutes, 0);
    const secs  = safeNum((d.total && d.total.seconds !== undefined) ?
d.total.seconds : d.seconds, 0);

    return (days * 86400) + (hours * 3600) + (mins * 60) + secs;
  }
```

```
  function isExpired(val)
  {
    return (remainingSeconds(val) <= 0);
  }

  /* ============================================================
     FORMATTER: Zeittext mit optionalen Stunden
     ------------------------------------------------------------
     Gibt nur den "Zeitanteil" zurück, z.B.:
       - "3 Tagen" (wenn days>1 und hideHours aktiv)
       - "1 Tag und 5 Stunden"
     ============================================================ */

  function formatDaysHours(days, hours)
  {
    // Wenn konfiguriert: Stunden ausblenden sobald days > 1
    if (HIDE_HOURS_IF_MORE_THAN_ONE_DAY)
    {
      if (days > 1)
      {
        return String(days) + " " + dayWord(days);
      }
    }

    // Standard: Tage + Stunden
    return String(days) + " " + dayWord(days) + " und " + String(hours) + "
" + hourWord(hours);
  }

  /* ============================================================
     RENDERER
     ============================================================ */

  function renderWeihnachten(val)
  {
    const data = parseFullJson(val);
    const now  = new Date();

    if (now.getMonth() === 11 && now.getDate() >= 24 && now.getDate() <= 26)
    {
      return "□ Frohe Weihnachten □<br><span style='font-
size:26px;'>□□□</span>";
    }

    if (!data)
    {
      return EMPTY_TEXT;
    }

    const days  = safeNum((data.total && data.total.days !== undefined) ?
data.total.days : data.days, 0);
```

```
    const hours = safeNum((data.hours !== undefined) ? data.hours :
(data.total ? data.total.hours : undefined), 0);
    const words = (data.inWords && data.inWords.long) ? data.inWords.long :
"";

    if (now.getMonth() === 11 && now.getDate() === 23)
    {
      return "🎄 Weihnachten morgen" + subline(words);
    }

    if (days >= 1)
    {
      return "🎄 Weihnachten in " + formatDaysHours(days, hours);
    }

    return "🎄 Frohe Weihnachten 🎄";
  }

  function renderSilvester(val)
  {
    const data = parseFullJson(val);
    const now  = new Date();

    if (now.getMonth() === 11 && now.getDate() === 31)
    {
      return "🎆 Silvester heute";
    }

    if (now.getMonth() === 0 && now.getDate() === 1)
    {
      return "🎆🎆 Frohes neues Jahr! 🎆🎆";
    }

    if (!data)
    {
      return EMPTY_TEXT;
    }

    const days  = safeNum((data.total && data.total.days !== undefined) ?
data.total.days : data.days, 0);
    const hours = safeNum((data.hours !== undefined) ? data.hours :
(data.total ? data.total.hours : undefined), 0);
    const words = (data.inWords && data.inWords.long) ? data.inWords.long :
"";

    if (now.getMonth() === 11 && now.getDate() === 30)
    {
      return "🎆 Silvester morgen" + subline(words);
    }

    if (days >= 1)
```

```
    {
      return "🎆 Silvester in " + formatDaysHours(days, hours);
    }

    return "🎆 Silvester heute" + subline(words);
  }

  function renderGeneric(val, title, emoji)
  {
    const data = parseFullJson(val);

    if (!data)
    {
      return EMPTY_TEXT;
    }

    const days  = safeNum((data.total && data.total.days !== undefined) ?
data.total.days : data.days, 0);
    const hours = safeNum((data.hours !== undefined) ? data.hours :
(data.total ? data.total.hours : undefined), 0);
    const words = (data.inWords && data.inWords.long) ? data.inWords.long :
"";

    if (days === 1)
    {
      return emoji + " " + title + " morgen" + subline(words);
    }

    if (days >= 1)
    {
      return emoji + " " + title + " in " + formatDaysHours(days, hours);
    }

    if (words)
    {
      return emoji + " " + title + subline(words);
    }

    return emoji + " " + title + " bald";
  }

  function renderByIndex(i, val)
  {
    const t = getType(i);

    if (t === 1)
    {
      return renderWeihnachten(val);
    }

    if (t === 2)
```

```
    {
      return renderSilvester(val);
    }

    return renderGeneric(val, getTitle(i), getEmoji(i));
  }

  /* ============================================================
     AUSWAHL: 2 nächste Events
     ============================================================ */

  function pickTwoUpcoming()
  {
    let best1_i = -1;
    let best1_r = 999999999;

    let best2_i = -1;
    let best2_r = 999999999;

    let i = 1;

    while (i <= EVENT_COUNT)
    {
      const oid = getOid(i);

      if (oid)
      {
        const val = vis.states.attr(oid + ".val");

        if (!isExpired(val))
        {
          const r = remainingSeconds(val);

          if (r < best1_r)
          {
            best2_i = best1_i;
            best2_r = best1_r;

            best1_i = i;
            best1_r = r;
          }
          else
          {
            if (r < best2_r)
            {
              best2_i = i;
              best2_r = r;
            }
          }
        }
      }
```

```
    i = i + 1;
  }

  return String(best1_i) + "|" + String(best2_i);
}

function renderSlot(slotIndex, el)
{
  const picks = pickTwoUpcoming().split("|");
  let idx = -1;

  if (slotIndex === 1)
  {
    idx = Number(picks[0]);
  }
  else
  {
    idx = Number(picks[1]);
  }

  if (idx < 1)
  {
    el.innerHTML = EMPTY_TEXT;
    return;
  }

  const oid = getOid(idx);
  const val = vis.states.attr(oid + ".val");

  el.innerHTML = renderByIndex(idx, val);
}

/* ============================================================
   START & BINDING
   ============================================================ */

function start()
{
  const el1 = document.getElementById("cd_text_xmas");
  const el2 = document.getElementById("cd_text_ny");

  if (typeof vis !== "undefined" && vis.states && el1 && el2)
  {
    function updateSlot1()
    {
      renderSlot(1, el1);
    }

    function updateSlot2()
    {
      renderSlot(2, el2);
```

```
      }

      if (E1_OID) vis.states.bind(E1_OID + ".val", updateSlot1);
      if (E2_OID) vis.states.bind(E2_OID + ".val", updateSlot1);
      if (E3_OID) vis.states.bind(E3_OID + ".val", updateSlot1);
      if (E4_OID) vis.states.bind(E4_OID + ".val", updateSlot1);
      if (E5_OID) vis.states.bind(E5_OID + ".val", updateSlot1);
      if (E6_OID) vis.states.bind(E6_OID + ".val", updateSlot1);
      if (E7_OID) vis.states.bind(E7_OID + ".val", updateSlot1);
      if (E8_OID) vis.states.bind(E8_OID + ".val", updateSlot1);

      if (E1_OID) vis.states.bind(E1_OID + ".val", updateSlot2);
      if (E2_OID) vis.states.bind(E2_OID + ".val", updateSlot2);
      if (E3_OID) vis.states.bind(E3_OID + ".val", updateSlot2);
      if (E4_OID) vis.states.bind(E4_OID + ".val", updateSlot2);
      if (E5_OID) vis.states.bind(E5_OID + ".val", updateSlot2);
      if (E6_OID) vis.states.bind(E6_OID + ".val", updateSlot2);
      if (E7_OID) vis.states.bind(E7_OID + ".val", updateSlot2);
      if (E8_OID) vis.states.bind(E8_OID + ".val", updateSlot2);

      updateSlot1();
      updateSlot2();
    }
    else
    {
      setTimeout(start, 300);
    }
  }

  if (document.readyState === "complete")
  {
    start();
  }
  else
  {
    window.addEventListener("load", start);
  }

})();
</script>
```

 Neuen Countdown hinzufügen (Checkliste) Beispiel: countdown.0.countdowns.Herbsturlaub.fullJson
 Schritt 1 – Subscribe-Block {countdown.0.countdowns.Herbsturlaub.fullJson}

 Schritt 2 – OID definieren const OID_4 = „countdown.0.countdowns.Herbsturlaub.fullJson“;

 Schritt 3 – Renderfunktion ergänzen if (i === 4) return renderGeneric(val, „Herbsturlaub“, „□□“);

 Schritt 4 – remainingSeconds / pickTwo erweitern

(analog zu Event 3)