```
const http  = require('http');
const https = require('https');
const fs    = require('fs');
const path  = require('path');
const url   = require('url');

const PORT = process.env.PORT || 3000;
const MELLOJOY_HOST = 'www.mellojoyjapan.com';

function cors(res) {
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET, OPTIONS');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
}

function json(res, status, data) {
  cors(res);
  res.writeHead(status, { 'Content-Type': 'application/json; charset=utf-8' });
  res.end(JSON.stringify(data));
}

function proxyProduct(handle, res) {
  const opts = {
    hostname: MELLOJOY_HOST,
    path: `/products/${encodeURIComponent(handle)}.js`,
    method: 'GET',
    headers: {
      'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 17_0 like Mac OS X) Apple
      'Accept': 'application/json',
      'Accept-Language': 'ja,en;q=0.9',
      'Referer': `https://${MELLOJOY_HOST}/`,
    }
  };
  const req = https.request(opts, shopifyRes => {
    let body = '';
    shopifyRes.on('data', c => body += c);
    shopifyRes.on('end', () => {
      try {
        const d = JSON.parse(body);
        const variants = d.variants || [];
        const avail = variants.filter(v => v.available);
        json(res, 200, {
          id: d.id,
          title: d.title,
          handle: d.handle,
          price_display: `¥${(d.price / 100).toLocaleString('ja-JP')}`,
          image: d.featured_image ? `https:${d.featured_image}` : null,
          url: `https://${MELLOJOY_HOST}/products/${d.handle}`,
          available: avail.length > 0,
          available_count: avail.length,
          available_variants: avail.map(v => ({
            id: v.id,
            title: v.title,
            cart_url: `https://${MELLOJOY_HOST}/cart/add?id=${v.id}&quantity=1&re
          })),
          checked_at: new Date().toISOString(),
        });
        if (avail.length > 0) console.log(`在庫あり: ${d.title}`);
      } catch (e) {
        json(res, 502, { error: 'Parse error' });
      }
    });
  });
  req.on('error', e => json(res, 500, { error: e.message }));
  req.setTimeout(8000, () => { req.destroy(); json(res, 504, { error: 'Timeout' }
  req.end();
}

const MIME = {
  '.html': 'text/html; charset=utf-8',
  '.js':   'application/javascript; charset=utf-8',
  '.css':  'text/css',
  '.json': 'application/json',
  '.webmanifest': 'application/manifest+json',
  '.png':  'image/png',
  '.ico':  'image/x-icon',
  '.svg':  'image/svg+xml',
};

function serveFile(filePath, res) {
  const ext  = path.extname(filePath);
  const mime = MIME[ext] || 'application/octet-stream';
  fs.readFile(filePath, (err, data) => {
    if (err) { res.writeHead(404); res.end('Not Found'); return; }
    const headers = { 'Content-Type': mime };
    if (ext === '.html') headers['Cache-Control'] = 'no-cache';
    else headers['Cache-Control'] = 'public, max-age=86400';
    res.writeHead(200, headers);
    res.end(data);
  });
}
```

```
// ファイルを public/ → ルート の順で探す
function findAndServe(pathname, res) {
  const name = pathname === '/' ? 'index.html' : pathname.replace(/^\//, '');

  // 候補パスリスト
  const candidates = [
    path.join(__dirname, 'public', name),
    path.join(__dirname, name),
    // icons/ フォルダなしフォールバック
    path.join(__dirname, 'public', path.basename(name)),
    path.join(__dirname, path.basename(name)),
  ];

  for (const p of candidates) {
    if (fs.existsSync(p) && fs.statSync(p).isFile()) {
      return serveFile(p, res);
    }
  }

  // どこにもなければ index.html を返す（SPA fallback）
  const index = fs.existsSync(path.join(__dirname, 'public', 'index.html'))
    ? path.join(__dirname, 'public', 'index.html')
    : path.join(__dirname, 'index.html');
  serveFile(index, res);
}

const server = http.createServer((req, res) => {
  const { pathname, query: qs } = url.parse(req.url, true);

  if (req.method === 'OPTIONS') { cors(res); res.writeHead(204); res.end(); retur

  if (pathname === '/api/product') {
    if (!qs.handle) { json(res, 400, { error: 'handle required' }); return; }
    proxyProduct(qs.handle, res);
    return;
  }

  if (pathname === '/api/health') {
    json(res, 200, {
      ok: true,
      server: 'Mellojoy Sniper Cloud',
      version: '3.1',
      env: process.env.RENDER ? 'render' : 'local',
      time: new Date().toISOString(),
    });
    return;
  }

  findAndServe(pathname, res);
});

server.listen(PORT, '0.0.0.0', () => {
  console.log(`Mellojoy Sniper 起動 PORT:${PORT}`);
});

server.on('error', err => { console.error(err); process.exit(1); });
```