Index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <title>Contact — Client Form Validation Demo</title>
    <style>
      body { font-family: Arial, sans-serif; margin: 2rem; }
      .field { margin-bottom: 1rem; }
      .error { color: #b00020; font-size: 0.9rem; display:block; }
      .invalid { border-color: #b00020; }
      #formStatus { margin-top: 1rem; }
    </style>
  </head>
  <body>
    <h1>Contact Us</h1>

    <form id="contactForm" novalidate>
      <div class="field">
        <label for="name">Name</label><br />
        <input id="name" name="name" type="text" required minlength="2" />
        <span class="error" data-for="name" aria-live="polite"></span>
      </div>

      <div class="field">
        <label for="email">Email</label><br />
        <input id="email" name="email" type="email" required />
        <span class="error" data-for="email" aria-live="polite"></span>
      </div>

      <div class="field">
        <label for="phone">Phone (optional)</label><br />
        <input id="phone" name="phone" type="tel" />
        <span class="error" data-for="phone" aria-live="polite"></span>
      </div>

      <div class="field">
        <label for="message">Message</label><br />
        <textarea id="message" name="message" required
minlength="5"></textarea>
        <span class="error" data-for="message" aria-live="polite"></span>
      </div>

      <div id="formStatus" role="status" aria-live="polite"></div>

      <button id="submitBtn" type="submit">Send</button>
    </form>
    <script src="/js/form-validation.js" defer></script>
  </body>
</html>
```

Index.js

```
// Minimal Express server for the client-form-validation-demo
const express = require('express');
const path = require('path');

const app = express();
const PORT = process.env.PORT || 3000;

// Parse JSON bodies
app.use(express.json());

// Serve static files from the public directory
app.use(express.static(path.join(__dirname, '..', 'public')));

// Simple server-side validation helper
function validateContact(body) {
  const errors = [];
  const { name, email, phone, message } = body || {};

  if (!name || String(name).trim().length < 2) {
    errors.push({ param: 'name', msg: 'Name must be at least 2 characters.' 
});
  }

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!email || !emailRegex.test(String(email))) {
    errors.push({ param: 'email', msg: 'Invalid email address.' });
  }

  const phoneVal = phone ? String(phone).trim() : '';
  const phoneRegex = /^\+?[0-9\-()\s]{7,20}$/;
  if (phoneVal && !phoneRegex.test(phoneVal)) {
    errors.push({ param: 'phone', msg: 'Invalid phone number.' });
  }

  if (!message || String(message).trim().length < 5) {
    errors.push({ param: 'message', msg: 'Message must be at least 5 
characters.' });
  }
  return errors;
}

// POST /api/contacts - validate and echo back (placeholder for saving to DB)
app.post('/api/contacts', (req, res) => {
  const errors = validateContact(req.body);
  if (errors.length) {
    return res.status(422).json({ errors });
  }

  // In a real app you'd persist the contact here.
  const { name, email, phone, message } = req.body;
  return res.status(201).json({ message: 'Contact received', data: { name,
email, phone, message } });
});

// Fallback route to ensure index.html is served for unknown routes (optional)
app.get('*', (req, res, next) => {
  if (req.path.startsWith('/api/')) return next();
  res.sendFile(path.join(__dirname, '..', 'public', 'index.html'));
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

Form- validation.js

```
// Client-side form validation and submit (plain JS)
(function () {
  const form = document.getElementById('contactForm');
  if (!form) return;
  const statusNode = document.getElementById('formStatus');
  const submitBtn = document.getElementById('submitBtn');

  // Validation rules
  const rules = {
    name: { required: true, min: 2 },
    email: { required: true, pattern: /^[^\s@]+@[^\s@]+\.[^\s@]+$/ },
    phone: { required: false, pattern: /^\+?[0-9\-()\s]{7,20}$/ },
    message: { required: true, min: 5 }
  };

  function getErrorNode(name) {
    return form.querySelector('.error[data-for="' + name + '"]');
  }

  function validateField(el) {
    const name = el.name;
    const val = String(el.value || '').trim();
    const rule = rules[name];
    if (!rule) return true;

    if (rule.required && !val) return 'This field is required.';
    if (rule.min && val.length < rule.min) return `Must be at least
${rule.min} characters.`;
    if (rule.pattern && val && !rule.pattern.test(val)) {
      if (name === 'email') return 'Enter a valid email address.';
      if (name === 'phone') return 'Enter a valid phone number.';
      return 'Invalid format.';
    }
    return true;
  }

  function showError(el, msg) {
    const node = getErrorNode(el.name);
    if (msg) {
      el.classList.add('invalid');
      el.setAttribute('aria-invalid', 'true');
      if (node) node.textContent = msg;
    } else {
      el.classList.remove('invalid');
      el.removeAttribute('aria-invalid');
      if (node) node.textContent = '';
    }
  }

  function validateForm() {
    const elements = Array.from(form.elements).filter(e => e.name);
    let firstInvalid = null;
    let ok = true;
    elements.forEach(el => {
      const res = validateField(el);
      if (res !== true) {
        ok = false;
        showError(el, res);
        if (!firstInvalid) firstInvalid = el;
      } else {
        showError(el, '');
      }
    });
    return { ok, firstInvalid };
  }
```

```
// Debounce helper
function debounce(fn, ms) {
  let t;
  return function () {
    const args = arguments;
    clearTimeout(t);
    t = setTimeout(() => fn.apply(null, args), ms);
  };
}

form.addEventListener('input', debounce(function (e) {
  if (!e.target.name) return;
  const res = validateField(e.target);
  showError(e.target, res === true ? '' : res);
  if (submitBtn) submitBtn.disabled = !validateForm().ok;
}, 200));

async function submitJSON(data) {
  if (submitBtn) {
    submitBtn.disabled = true;
    submitBtn.textContent = 'Sending...';
  }
  statusNode.textContent = '';
  try {
    const res = await fetch('/api/contacts', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(data)
    });
    const payload = await res.json().catch(() => ({}));
    if (res.status === 201) {
      statusNode.textContent = 'Message sent.';
      form.reset();
      // clear errors
      Array.from(form.elements).forEach(el => { if (el.name) showError(el,
''); });
    } else if (res.status === 422 && payload.errors) {
      statusNode.textContent = 'Please fix highlighted fields.';
      payload.errors.forEach(err => {
        const el = form.querySelector('[name="' + err.param + '"]');
        if (el) showError(el, err.msg);
      });
    } else {
      statusNode.textContent = payload.error || 'Server error. Try again.';
    }
  } catch (err) {
    statusNode.textContent = 'Network error. Try again.';
    } finally {
      if (submitBtn) {
        submitBtn.disabled = !validateForm().ok;
        submitBtn.textContent = 'Send';
      }
    }
  }

  form.addEventListener('submit', function (e) {
    e.preventDefault();
    const { ok, firstInvalid } = validateForm();
    if (!ok) {
      if (firstInvalid) firstInvalid.focus();
      return;
    }

    const data = {
      name: form.name.value.trim(),
      email: form.email.value.trim(),
      phone: form.phone.value.trim(),
      message: form.message.value.trim()
    };
    submitJSON(data);
  });

  // initial submit button state
  if (submitBtn) submitBtn.disabled = !validateForm().ok;
})();
```

Package.json

```
{
  "name": "client-form-validation-demo",
  "version": "1.0.0",
  "description": "Demo: client-side form validation with a minimal Express
backend",
  "main": "server/index.js",
  "scripts": {
    "start": "node server/index.js",
    "dev": "nodemon server/index.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  },
  "devDependencies": {
```

```json
    "nodemon": "^2.0.22"
  },
  "license": "MIT"
}
```

.gitignore

```
node_modules/
```

README.md

~~~markdown
# client-form-validation-demo

Minimal demo showing client-side form validation with a tiny Express backend.

Quick start (Windows PowerShell):

1. Install dependencies

```powershell
npm install
```

2. Run in development (restarts on change)

```powershell
npm run dev
```
~~~