Building Super Powered HTML Forms with JavaScript

What Are Super Powers?

User experience improvements that do **NOT** negatively impact:

- native functionality
- accessibility
- semantice
- performance
- security

AKA: Progressive enhancement

First Rule Of JavaScript

Know when to use it

JavaScript Has Inherent Cost

HTML and CSS are *generally* going to be faster than JavaScript

- Extra request
- More data
- Render blocking
- Runtime performance
- Unexpected errors
- JavaScript *may* be disabled 😱

HTML Gives Us A Lot

- State management (`value`/`checked`)
- Clickable labels
- Accessibility
  - Keyboard navigation
  - Focusable
  - Screen readers support
- Consistent experience
  - Radios, checkboxes, selects
  - "Power users"
  - Implicit submissions
- Validation (0kb)
  - `required`, `minlength`, `maxlength`, `min`, `max`, `type`, `pattern` (regex)
  - a11y hints

Inputs

We Have `Options`

There are 24 different kinds of form controls:

- button
- checkbox
- color
- date
- datetime-local
- email
- file
- hidden
- image
- month
- number
- password
- radio
- range
- reset
- search
- submit
- tel
- text
- time
- url
- week
- select
- textarea

Why Do I Still See This?

```
<div
  class="checkbox"
  onclick="toggleWithClick"
>
  I'm a checkbox
</div>
```

Styling Forms 🤮

2019 survey by Greg Whitworth (@gregwhitworth)

- 1400 respondents
- The most common reason people recreate native controls: styling
- The most common custom control: `select`
- www.gwhitworth.com/surveys/controls-components

"...the amount of work it takes to implement an accessible alternative with complete feature parity is massive."

Real Example

```
<div
  class="checkbox"
  role="checkbox"
  aria-checked="false"
  aria-label="I'm a checkbox"
  tabindex="0"
  onclick="toggleWithClick"
  onkeydown="toggleWithKey"
>
  I'm a checkbox
</div>
```

Compared to

```
<input id="checkbox" name="checkbox" class="checkbox">
<label for="checkbox">I'm a checkbox</label>
```

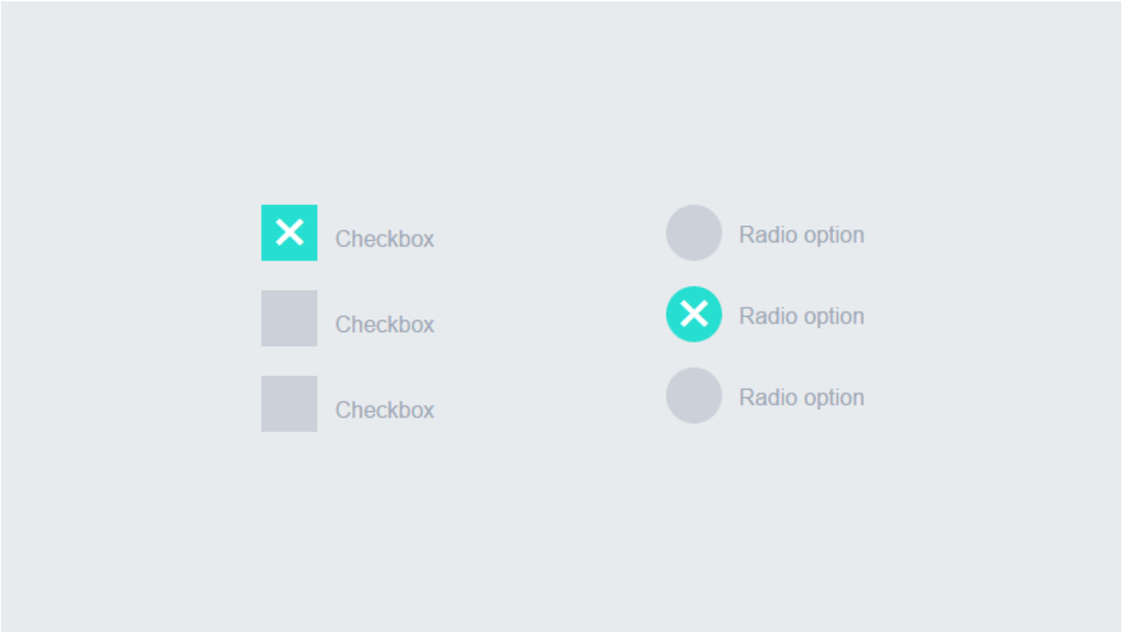
(This doesn't even show the event handlers, or other more complex inputs like radios 😱)

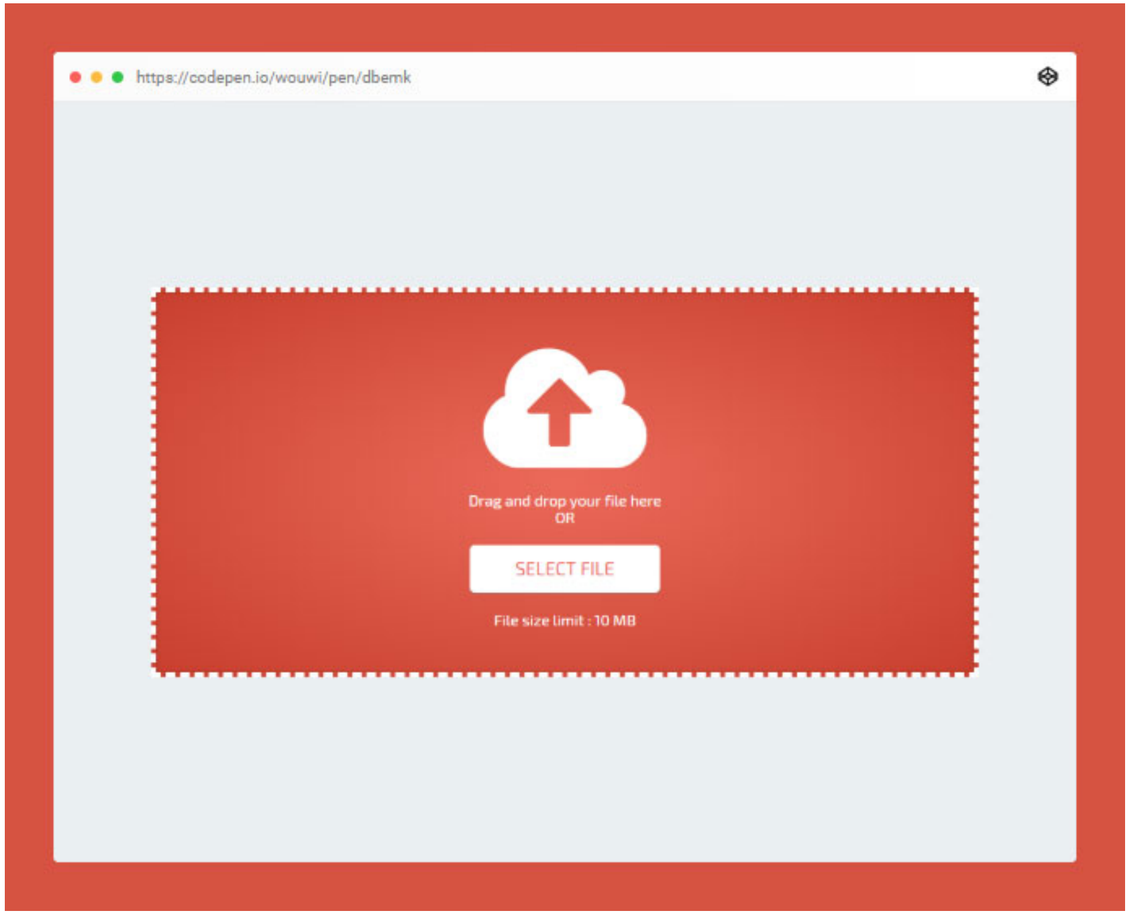
Good News

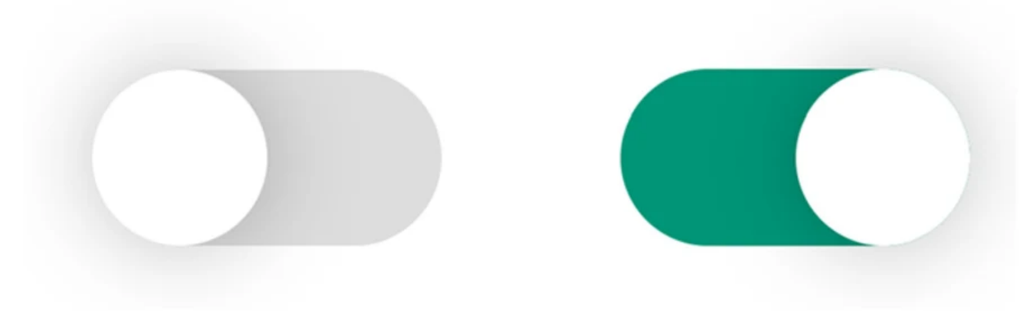
Much better than it used to be.

- CSS pseudoclasses
  - `:valid`
  - `:invalid`
  - `:required`
  - `:focus`
  - `:focus-within`
  - `:placeholder-shown` (hack)
  - `:out-of-range` (`min`/`max`)
- `appearance: none` + `:before`
  - Checkboxes & Radios
- Visually hidden input + sibling selector (`+`, `~`)

More at austingil.com/build-html-forms-right-styling


Checkbox
Checkbox
Checkbox
Radio option
Radio option
Radio option


https://codepen.io/wouwi/pen/dbemk
Drag and drop your file here
OR
SELECT FILE
File size limit : 10 MB

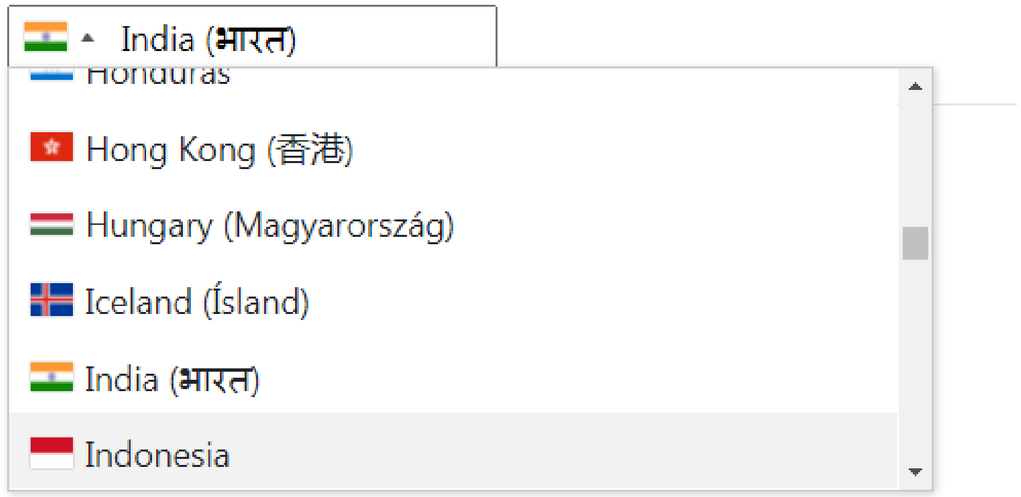


The Future Is Bright

New pseudoselectors & Parts

```
select:open { /* styles */ }

select::part(button) { /* styles */ }
```

Named slots



More Details

Smart people at Open UI (open-ui.org) are making things happen

Editorial Proposals:

- Select
- Checkbox
- File

I thought this was a JavaScript conference 🤔

Inputs + JS 😍

We can improve native inputs without breaking them

- Customize native validation
  - `input.setCustomValidity()`
  - `input.reportValidity()`
- Improve accessibility
  - Toggling `aria-invalid` on inputs
  - Toggling `aria-disabled` on submit button
- UX improvements
  - Toggle input `type` (show /hide password)
  - Auto-expand textarea
  - Input masking & formatting
- Custom UI validation messages

Custom Validation UI

- ValidityState Web API: `input.validity`

```
{
  `valid`: Boolean, // all validators
  `typeMismatch`: Boolean, // `type`
  `valueMissing`: Boolean, // `required`
  `rangeUnderflow`: Boolean, // `min`
  `rangeOverflow`: Boolean, // `max`
  `tooShort`: Boolean, // `min-length`
  `tooLong`: Boolean, // `max-length`
  `patternMismatch`: Boolean, // `pattern`
  `stepMismatch`: Boolean, // `step` (range input)
}
```

- Toggle `aria-invalid`
- Add accessible errors with `aria-describedby` + ARIA live regions

Forms

Semantic Forms

Almost every input will work better when wrapped in a form. It gives us more features, and less work to do.

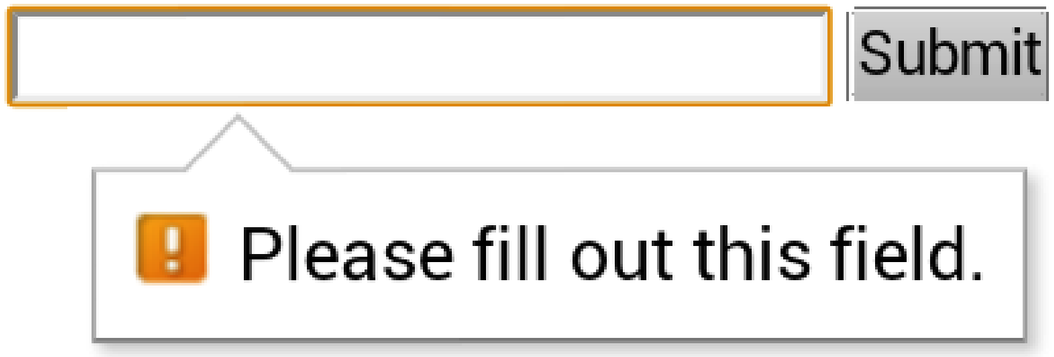
- Native validation
- Implicit return
- Simpler JS submission
- Resiliency

Forms + JS 😍

We can also improve forms without breaking anything

- Keyboard shortcuts (`ctrl`+`enter`)
- Repeater input fields
- Drag and drop ordering
- Custom validation UX

Native Validation



Actually quite useful

- Focuses first invalid input
- Scrolls to the focused input
- Shows users errors

Just one problem...

Can’t customize native UI 🤬

Why Not Both 🤷

Use JavaScript to enhance the native attribute constraints

- Works if JS is disabled
- Less to learn (compared to library)
- Less to download (compared to library)
  - Performance
  - Maintenance
  - Security
- Validation needs to happen on the backend anyway

** You may still consider a 3rd party library for special occasions.

Custom Validation 💅

- Prevent default validation

```
document.addEventListener("DOMContentLoaded", (event) ⇒ {
  form.noValidate = true
})
```

- Scroll to first invalid input on submit

```
form.addEventListener('submit', (event) ⇒ {
  const form = event.target

  if (!form.checkValidity()) {
    form.querySelector(':invalid').focus()
    return
  }

  // Valid submit logic
  event.preventDefault()
})
```

Enhanced Submissions

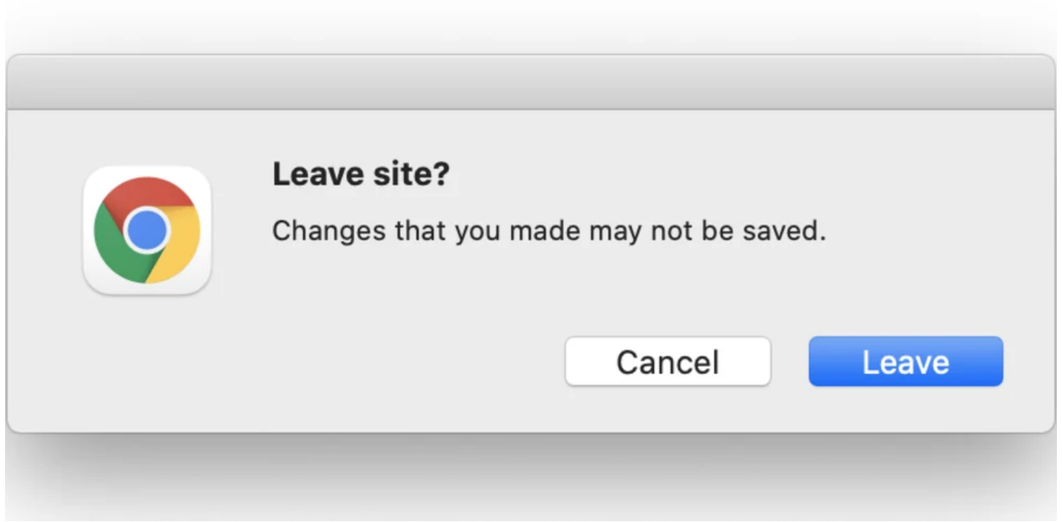
```
function jsSubmitForm(event) {
  const form = event.target

  let url = form.action
  const options = {
    method: form.method,
  }

  const formData = new FormData(form)
  const searchParams = new URLSearchParams(formData)
  const isMultipart = form.enctype === 'multipart/form-data' || !!form.querySelector('input[type="file"]')

  if (options.method.toUpperCase() === 'GET') {
    url += searchParams.toString()
  } else {
    options.body = isMultipart ? formData : searchParams
  }

  event.preventDefault()
  return fetch(url, options)
}
```

Caveats

- Only GET & POST
  - Affects API endpoints
- Requires JS detection from
  - `Sec-Fetch-Mode` header === `navigate`
  - `Accept` header includes `application/json`
  - Respond with 301 redirect or JSON.
- Doesn’t work for complex data
  - Nested objects (JSON)
  - GraphQL

Prevent Data Loss

Accidental refresh or navigation with `beforeunload`




```
window.addEventListener("beforeunload", (event) => {
  if (okToRefreshPage) return
  event.preventDefault()
  event.returnValue = ""
})
```

austingil.com/prevent-browser-refresh-url-changes-route-navigation-vue

Keep Backups

We can improve UX by restoring unsaved work from `localStorage`

```
form.addEventListener('change', function(event) {
  const formData = new FormData(this)
  const dataObect = Object.fromEntries(formData)
  const lsData = JSON.stringify(dataObject)

  window.localStorage.setItem(uniqueKey, lsData)
})

document.addEventListener("DOMContentLoaded", (event) => {
  const savedData = window.localStorage.getItem(uniqueKey)
  if (!savedData) return

  const inputValues = JSON.parse(savedData);

  // Loop over form inputs and assign value from object above (sorry, too long)
})

form.addEventListener('submit', (event) => {
  window.localStorage.removeItem(uniqueKey, JSON.stringify(dataObject))
})
```

Component Frameworks

Real super powers unlocked

Benefits

- Simplify form creation
- Repeatable quality
- Easier maintenance
- Enforce best practices
  - Required props (label, name)
  - Default fallbacks (ID, POST)

Vue Input Component

```
<script>
import { generateId } from './utils.js'

export default {
  props: {
    label: {
      type: String,
      required: true
    },
    name: {
      type: String,
      required: true
    },
    id: {
      type: String,
      default: () => generateId()
    }
  },
  // ...
}
</script>
```

```
<script>
export default {
  // ...
  data: () => ({
    errors: []
  }),
  methods: {
    validateInputOnBlur(event) {
      const input = event.target
      const validityState = input.validityState
      const errors = []
      for (const [property, isInvalid] of Object.entries(validityState)) {
        if (!isInvalid) return

        if (property === 'rangeUnderflow') { // min attribute
          errors.push(`Must be greater than ${this.$attrs.min}`)
        }
        // insert other validator logic ...
      }
      this.errors = errors
    }
  }
}
</script>
```

```
<template>
  <div>
    <label :for="id">{{ label }}</label>

    <span v-if="$attrs.required" class="color-red" aria-hidden="true">*</span>

    <input
      :id="id"
      :aria-describedby="`${id}-description`"
      @blur="validateInput"
      v-bind="$attrs"
      v-on="$listeners"
    />

    <div
      v-if="errors.length"
      :id="`${id}-description`"
      role="alert"
    >
      {{ errors.join(' ') }}
    </div>
  </div>
</template>
```

Example: vuetensils.austingil.com/components/Input.html

Vue Form Component

```
<script>
export default {
  methods: {
    async onSubmit(event) {
      const form = event.target

      if (!form.checkValidity()) {
        form.querySelector(':invalid').focus()
        this.$emit('invalidSubmit', event)
        return
      }

      this.$emit('validSubmit', event)
    }
  }
}
</script>
```

```
<template>
  <form :method="$attrs.method || 'POST'">
    <slot />

    <button type="submit">Submit</button>
  </form>
</template>
```

Example: vuetensils.austingil.com/components/Form.html

Putting It All Together

```
<script>
import { jsSubmit } from './utils.js'
export default {
  methods: {
    onValidSubmit: jsSubmit,
    onInvalidSubmit: console.log // Be better
  }
}
</script>
<template>
  <MyForm action="/api/login" @validSubmit="onValidSubmit" @invalidSubmit="onInvalidSubmit">
    <MyInput
      label="Email"
      type="email"
      name="email"
    />
    <MyInput
      label="Password"
      type="password"
      name="password"
    />
  </MyForm>
</template>
```

In The End We Get

- Consistent user experience across all browsers
- Accessible to everyone (including AT)
- Minimal performance impact (compared to only JS)
- Enhanced with JS but work without
- Compartmentalize logic in components

Thanks 😘

HTML forms series: austingil.com/how-to-build-html-forms-right-semantics

Vue.js library: vuetensils.austingil.com

Newsletter: austingil.com/newsletter

Twitter: @Stegosource

Follw my dog on Instagram now, plz, thx: instagram.com/NuggetTheMighty/