How to make a book with HTML & CSS

Using Bindery.js v2.3.1

Intro

Bindery.js is a javascript library to create printable books with HTML and CSS.

First, content is divided across multiple pages. From there, you can set up rules to generate running headers, spreads, and footnotes, based on the content that fits on each page. Turn links into page numbers to create a table of contents or index. When you're ready to print, configure bleed, crop marks, and booklet ordering.

If you're a web designer, you can think about books as an extension of responsive design. If you're a print designer, you can express layouts programmatically, without the need for InDesign.

Get started →[1] Github ↗[2]

1: bindery.info/learn
2: github.com/evnbr/bindery

Learn

Getting Started

Bindery is intended for web designers and developers of every skill level. Just include the script tag with your content, and you're ready to go.

```
<div id="content">
  <!-- The contents of your book -->
</div>
<script src="https://unpkg.com/bindery@2.3.0/">
</script>
<script>
  Bindery.makeBook({ content: '#content' });
</script>
```

You can also install bindery from npm[1], or download directly.

```
npm install --save bindery
```

↓ Download bindery.min.js [2]

1: www.npmjs.com/package/bindery
2: unpkg.com/bindery/dist/bindery.min.js

Styling

Use your existing web CSS if you've got it—96 CSS pixels equals 1 CSS inch. That should be around 1 inch while printed, although some browsers might adjust the scale a little. CSS also supports points (`pt`), pica (`pc`), inches (`in`), and millimeters (`mm`).

When using media queries or viewport-specific units like `vh` or `vw`, note that these refer to the browser width, not width of a book page. Also, avoid changing sizes or layout when using `@media print`. Bindery won't be able to use that information when flowing content across pages, and you won't be able to see them in the preview.

Preparing Content

Your book content is probably pretty long, so you may want to keep it in a separate file. This also prevents any flash of unstyled content you might have seen before bindery runs. You can fetch content by passing in the URL and selector, like this:

```
Bindery.makeBook({
  content: {
    selector: '#content'
    url: '/book-content.html',
  },
});
```

(Keep in mind—your browser won't fetch content from a different web server, since that wouldn't be secure. Make sure you're loading both your current file and your content from the same server. If your browser says `file://` in the URL bar, you aren't using a server.)

You don't need to do anything special with your HTML content, as long as it all ends up in a single file. For example, you could use the Wikipedia API to make a book from an article[1] without changing the markup.

1: bindery.info/examples/6_wikipedia

You may want to use your CMS or templating language to generate a file by looping over you site's posts or pages. For example, if you're using Jekyll[1], it might look something like this.

```
{% raw %}
<section id="content">
  {% for post in site.posts %}
  <h2>{{ post.title | escape }}</h2>
  <div class="post-content">
    {{ post.content | markdownify }}
  </div>
  {% endfor %}
</section>
{% endraw %}
```

Rules

You've now got content flowing across your pages. Next, you'll probably want to add page breaks, spreads, running headers, and the other elements of a usable book. You can do that by creating rules that apply to selectors, like you would with CSS.

1: jekyllrb.com

```
<div id="content">
  <h2>Chapter 1</h2>
  <p></p>
  <figure class="big-figure">
    <img src="figure1.png" />
  </figure>
</div>

<script>
  Bindery.makeBook({
    content: '#content',
    rules: [
      Bindery.PageBreak({ selector: 'h2', position:
'before' }),
      Bindery.FullBleedSpread({ selector: '.big-
figure' }),
    ],
  });
</script>
```

Book Components

For rules that create new elements on the page, you can pass in your own function. You can use whatever other tools or libraries you like, as long as you return an HTML element or line of text.

```
let linksAsFootnotes = Bindery.Footnote({
  selector: 'p > a',
  render: (element, number) => `${number}: Link to
${element.href}`;
});

let runningHeaders = Bindery.RunningHeader({
  render: (page) => page.isLeft
    ? `${page.number} · Jan Tschichold`
    : `The Form of the Book · ${page.number}`;
});

Bindery.makeBook({
  content: {
    selector: '#content'
    url: '/content.html',
  },
  rules: [ linksAsFootnotes, runningHeaders ]
});
```

In the example above, we return a string. We could also create an element ourselves in javascript. In plain javascript, that would look like:

```
let myCustomFootnote = Bindery.Footnote({
  selector: 'p > a',
  render: (element, number) => {
    let myFootnote = document.createElement('div');
    myFootnote.classList.add('note');
    myFootnote.textContent = `${number}: Link to
${element.href}`;
    return myFootnote;
  },
});
```

You can use any library that creates HTML elements, for example nanohtml[1]:

```
const html = import 'nanohtml';

let myCustomFootnote = Bindery.Footnote({
  selector: 'p > a',
  render: (element, number) => html`
    <div class="note">${number}: Link to
${element.href}</div>
  `;
});
```

1: github.com/choojs/nanohtml

Next Steps

To learn more about available rules and options, check out the documentation[1] or view some examples[2].

View Docs →[3] View Examples →[4]

1: bindery.info/docs
2: bindery.info/gallery
3: bindery.info/docs
4: bindery.info/gallery

Docs

Setup

Use `Bindery.makeBook({ options })` to create a book and display it immediately on page load. It takes an object of options as described below.

```
// With required options
Bindery.makeBook({
  content: '#content',
});
```

content

If the content is on the same page, use a CSS selector or a reference to the node. If the content must be fetched from a remote page, pass an object in the form of `{ url: String, selector: String }`.

```
// CSS selector
Bindery.makeBook({
  content: '#content',
});

// Element
const el = document.getElementByID('content');
Bindery.makeBook({
  content: el,
});

// Fetch from a URL
Bindery.makeBook({
  content: {
    selector: '#content',
    url: '/posts.html',
  }
});
```

pageSetup

`size:`	Book size, in the form of `{ width: String, height: String }`. Values must include absolute CSS units.

`margin:` Book margin, in the form of `{ top: String, outer: String, bottom: String, inner: String }`. Values must include absolute CSS units.

```
Bindery.makeBook({
  content: '#content',
  pageSetup: {
    size: { width: '4in', height: '6in' },
    margin: { top: '12pt', inner: '12pt', outer:
'16pt', bottom: '20pt' },
  },
});
```

printSetup

Note that setting the paper size through bindery only works in Chrome and Opera[1] as of 2017. Users with other browsers must set the size in the system print dialog.

`layout:`

1: caniuse.com/#feat=css-paged-media

	PAGES One page per sheet, in numerical order `default`
	SPREADS Two pages per sheet, in numerical order
	BOOKLET Two pages per sheet, in booklet order. For printing double sided and folding into a saddle stitched booklet.
`paper:`	AUTO Sets paper to the size of the page or, if the layout is `spreads` or `booklet`, twice as wide as the page. Note that marks will not be visible. `default`
	AUTO_BLEED The size of the page plus the size of the bleed. Note that marks will not be visible.
	AUTO_MARKS The size of the page plus room for

	crop and bleed marks.
	`LETTER_PORTRAIT`
	`LETTER_LANDSCAPE`
	`A4_PORTRAIT`
	`A4_LANDSCAPE`
`marks:`	`NONE`
	`CROP` Note that crop marks are always outset by the bleed amount. `default`
	`BLEED`
	`BOTH`
`bleed:`	Amount of bleed. Values must include absolute CSS units. This affects the size of full-bleed pages[1] and spreads[2], and sets the position of bleed and crop marks.

1: bindery.info/book/#fullbleedpage
2: bindery.info/book/#fullbleedspread

```
Bindery.makeBook({
  content: '#content',
  printSetup: {
    layout: Bindery.Layout.BOOKLET,
    paper: Bindery.Paper.AUTO_BLEED,
    marks: Bindery.Marks.CROP,
    bleed: '12pt',
  },
});
```

Preview

`view:`

`PREVIEW`
shows the spreads of the book as they will appear when the book is trimmed and bound. If you choose this mode, Bindery will switch to `PRINT` before printing. `default`

`PRINT`
shows the complete printed sheet, which may include multiple pages,

marks, and bleed if those options are enabled. Note that when

printing a booklet, pages will appear out of order.
`FLIPBOOK`
shows a three-dimensional preview, making it easy to visualize which pages will end up on the backs of others. If you choose this mode, Bindery will switch to `PRINT` before printing.

```
Bindery.makeBook({
  content: '#content',
  view: Bindery.View.FLIPBOOK,
})
```

Flowing Content

Book content runs within the margins on the front and back of every page. You can set a series of rules that change the book flow. Rules are triggered by selectors, like CSS. For example, you might want to start all `h2` elements on a new page, and make all `.big-figure` elements into a full-bleed spread across two pages:

```
Bindery.makeBook({
  content: '#content',
  rules: [
    Bindery.PageBreak({ selector: 'h2', position:
'before' }),
    Bindery.FullBleedSpread({ selector: '.big-figure'
}),
  ],
});
```

You may prefer to create rules separately:

```
let breakRule = Bindery.PageBreak({
  selector: 'h2',
  position: 'before',
});

let spreadRule = Bindery.FullBleedSpread({
  selector: '.big-figure',
});

Bindery.makeBook({
  content: '#content',
  rules: [ breakRule, spreadRule ],
});
```

PageBreak

Adds or avoids page breaks for the selected element.

`selector:`	Which elements the rule should be applied to.
`position:`	`'before'`
	insert a break before the element,

so it starts on a new page

`'after'`

insert a break after the element

`'both'`

insert breaks before and after the element

`'avoid'`

prevents the element from breaking in the middle, by pushing it to the next page.

`continue:` will insert an extra break when appropriate so that the flow will resume on a specific page. `Optional`

`'next'`

`default`

`'left'`

`'right'`

```
// Make sure chapter titles always start on a
righthand page.
Bindery.PageBreak({
  selector: 'h2',
  position: 'before',
  continue: 'right'
})
```

Split

Add a class when an element splits across two pages, to customize the styling.

Bindery makes as few assumptions as possible about your intended design— by default, text-indent will be removed from `<p>`s that started on the previous page, and the bullet will be hidden for `<li>`s that started on the previous page. Everything else you should specify yourself—for example, you may want to remove margin, padding, or borders when your element splits. See example[1].

`selector:` Which elements the rule should be

1: bindery.info/examples/7_custom_split

applied to.

`toNext:`	Class applied to elements that will continue onto the next page. `Optional`
`fromPrevious:`	Class applied to elements that started on a previous page. `Optional`

```
Bindery.Split({
  selector: 'p',
  toNext: 'to-next',
  fromPrevious: 'from-previous',
}),
```

```
<!-- Before -->      <!-- Page 1 -->
<p>                  <p class='to-next'>
  Some books are       Some books are
  saddle stitched...</p>
</p>                 <!-- Page 2 -->
                     <p class='from-previous'>
                       saddle stitched...
                     </p>
```

Counter

Increment a counter as the book flows. This is useful for numbering figures or sections. Bindery's Counters can be used in place of CSS counters[1], which will not work as expected when the DOM is reordered to create a book.

`incrementEl:`	CSS Selector. Matching elements will increment the counter by 1.
`resetEl:`	CSS Selector. Matching elements will set the counter to 0. `Optional`
`replaceEl:`	CSS Selector. Matching elements will display the value of the counter.
`replace:`	A function that takes the selected element and the counter value, and returns an new element. By default, Bindery will simply replace the contents with the value of the

1: `developer.mozilla.org/en-US/docs/Web/CSS/CSS_Lists_and_Counters/Using_CSS_counters` counter. `Optional`

Here's how you could number all `<figure>`s, by replacing the existing `<span class='fig-num'>`s.

```
Bindery.Counter({
  incrementEl: 'figure',
  replaceEl: '.fig-num',
})
```

```
<!-- Before -->
<figure>
  <img />
  <figcaption>
    <span class='fig-num'></span>
    Here's the caption
  </figcaption>
</figure>
```

```
<!-- After -->
<figure>
  <img />
  <figcaption>
    <span class='fig-num'>1</span>
    Here's the caption
  </figcaption>
</figure>
```

Here's how you could number all `<p>`s, resetting each section, by inserting new markup. See example[1].

```
Bindery.Counter({
  incrementEl: 'p',
  replaceEl: 'p',
  resetEl: 'h2',
  replace: (el, counterValue) => {
    el.insertAdjacentHTML('afterbegin', `<i>P
${counterValue} </i>`);
    return el;
  }
}),
```

1: bindery.info/examples/10_counters

```html
<!-- Before -->
<h2>Section 1</h2>
<p>Some books are
saddle stitched.</p>
<p>Other books are
perfect bound.</p>

<h2>Section 2</h2>
<p>eBooks are different
altogether.</p>
```

```html
<!-- After -->
<h2>Section 1</h2>
<p><i>P 1</i>
Some books are
saddle stitched.</p>
<p><i>P 2</i>
Other books are
perfect bound.</p>

<h2>Section 2</h2>
<p><i>P 1</i>
eBooks are different
altogether.</p>
```

Page Elements

RunningHeader

An element added to each page. By default it will add a page number at the top right of each page, but you can use `render` to generate running headers using your section titles.

Note that you can't currently use multiple `RunningHeader` rules at the same time. However, you can create a custom rule[1] that works similarly.

```
Bindery.RunningHeader({
  render: (pageInfo) => pageInfo.isLeft
    ? `${pageInfo.number} · ${pageInfo.heading.h1}`
    : `${pageInfo.heading.h2} · ${pageInfo.number}`
})
```

`render:`	A function that takes a `PageInfo` and returns a string of HTML. You'll probably want to use the `number`, `isLeft`, `isEmpty`, and `heading` properties — see `PageInfo`[2] for details. `Optional`

Footnote

Add a footnote to the bottom of the flow area.

1: codepen.io/brsev/pen/yLVrwdw

2: bindery.info/book/#pageinfo

Footnotes cut into the area for text, so note that very large footnotes may bump the entire element to the next page.

`selector:`	Which elements the rule should be applied to.
`render:`	A function that takes an element and number, and returns the footnote for that element. This footnote will be inserted at the bottom of the flow area.
`replace:`	A function that takes the selected element and number, and returns an new element with a footnote indicator. By default, Bindery will simply insert the number as a superscript after the original element. `Optional`

```
Bindery.Footnote({
  selector: 'p > a',
  render: (element, number) => {
    return '<i>' + number + '</i>: Link to ' +
element.href;
  }
}),
```

FullBleedPage

Removes the selected element from the ordinary flow of the book and places it on its own page. Good for displaying figures and imagery. You can use CSS to do your own layout on this page—`width: 100%; height: 100%` will fill the whole bleed area.

`selector:`	Which elements the rule should be applied to.
`continue:`	Where to resume the book flow after adding the full bleed page. `Optional`

	`'same'` Continues on the previous page where the element would have been. This will fill the remainder of that page, avoiding a gap, though note that it results in a different order than your original markup. `default`
	`'next'` Continues on a new page
	`'left'` Continues on the next left page, inserting another page when appropriate
	`'right'` Continues on the next right page, inserting another page when appropriate
`rotate:`	Add a rotation the full-bleed content. `Optional`
	`'none'` `default`

`'clockwise'`
The top will become the left edge
`'counterclockwise'`
The top will become the right edge
`'inward'`
The top will become the outside edge
`'outward'`
The top will become the inside edge

```
Bindery.FullBleedPage({
  selector: '.big-figure',
  continue: 'same'
}),
```

FullBleedSpread

The same as `FullBleedPage`[1], but places the element across two pages.

`selector:` Which elements the rule should be

1: bindery.info/book/#fullbleedpage

applied to.

`continue:`	Where to resume the book flow after adding the full bleed element. `Optional`
	`'same'`
	`default` Continue where the element was, so there's not a blank gap before the spread.
	`'next'`
	Continues on a new page after the spread.
	`'left'`
	Continues on the next left page after the spread
	`'right'`
	Continues on the next right page after the spread

`rotate:`	Add a rotation the full-bleed content. `Optional`
	`'none'`
	`default`

`'clockwise'`
The top will become the left edge
`'counterclockwise'`
The top will become the right edge

```
Bindery.FullBleedSpread({
  selector: '.wide-figure',
  continue: 'next',
  rotate: 'clockwise',
}),
```

Referencing Pages

If your web content has internal links or navigation, you can use a `PageReference` to insert the page number the content will eventually end up on. You can use them to create traditional book navigation elements, like a table of contents, index, endnotes, without having to update them every time you change the page size or style.

PageReference

`selector:`	Which elements the rule should be applied to.
`replace:`	A function that takes an element and a page range, and must return a new element. By default, Bindery will insert the page range after the original element. `Optional`
`createTest:`	A function that takes your reference element and returns a test function. The test function receives a page element, and should return true if the reference can be found. By default, the test function will look for the anchor tag of the reference element's `href` property, which is useful for a table of contents. Use a custom function to create an index. `Optional`

Creating a Table of Contents

A table of contents is a PageReference[1] that points to a specific page. By default, PageReference will look for anchor links. To create a table of contents, do this:

```
Bindery.PageReference({
  selector'.toc a',
  replace: (element, number) => {
    let row = document.createElement('div');
    row.classList.add('toc-row');
    row.innerHTML = element.textContent;
    row.innerHTML += `<span class='num'>${number}
</span>`;
    return row;
  }
})
```

You can use any library that creates HTML elements, for example nanohtml[2]:

1: bindery.info/book/#pagereference
2: github.com/choojs/nanohtml

```
const html = import 'nanohtml';

Bindery.PageReference({
  selector'.toc a',
  replace: (element, number) => html`
    <div class="toc-row">
      <span>${element.textContent}</span>
      <span class="num">${number}</span>
    </div>
  `;
})
```

This will transform the HTML of your anchor links like this:

```
<!-- Before -->        <!-- After -->
<nav class='toc'>      <nav class='toc'>
  <a href='#chapter1'>   <div class='toc-row'>
    Chapter 1              <span>Chapter 1</span>
  </a>                     <span class='num'>5</span>
</nav>                   </div>
                       </nav>
```

Creating an Index

An index is a PageReference[1] that points to content on a range of pages. There are many way you might create an index. In the following example, rather than checking the `href`, Bindery will search the entire text of each page to see if contains the text of your reference element.

```
Bindery.PageReference({
  selector: '.index-content li',
  createTest: (el) => {
    const searchTerm =
el.textContent.toLowerCase().trim();
    return (page) => {
      const textOfPage =
page.textContent.toLowerCase();
      return textOfPage.includes(searchTerm);
    }
  },
})
```

This will transform the list items as below:

1: bindery.info/book/#pagereference

```html
<!-- Before -->
<p>
  Some books are
  saddle stitched...
</p>

<ul class='index-content'>
  <li>Saddle Stitch</li>
</ul>
```

```html
<!-- After -->
<p>
  Some books are
  saddle stitched...
</p>

<ul class='index-content'>
  <li>Saddle Stitch, 5</li>
</ul>
```

If you didn’t want to match on the exact string, you could use other selectors or attribute, or use a fuzzier method of searching. Just create your own testing function from the index entry.

```
Bindery.PageReference({
  selector: '[data-ref]',
  createTest: (el) => {
    let selector = el.getAttribute('data-ref');
    return (page) => page.querySelector(selector);
  },
})
```

```
<!-- Before -->
<p data-id='perfectBind'>
  Most books are perfect bound.
</p>

<ul>
  <li data-ref='perfectBind'>
    Binding, Perfect
  </li>
</ul>
```

```html
<!-- After -->
<p data-id='perfectBind'>
  Most books are perfect bound.
</p>

<ul>
  <li data-ref='perfectBind'>
    Binding, Perfect: 5
  </li>
</ul>
```

Note that we can’t know what page something will end up on until the book layout is complete, so make sure that your `replace` function doesn’t change the layout drastically.

Advanced

PageInfo

You may receive instances of this class when using custom rules, but will not create them yourself.

`number`	the page number, with the first page being 1
`heading`	The current hierarchy of headings from previous pages, in the form of `{ h1: String, h2: String, ... h6: String }`
`isEmpty`	`Bool` Whether the page includes flow content
`isRight`	`Bool` The page is on the right (the front)
`isLeft`	`Bool` The page is on the left (the back)

BookInfo

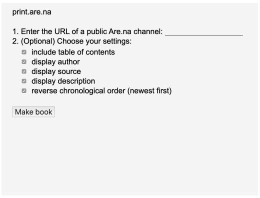
You may receive instances of this class when using custom rules, but will not create them yourself.

`pages`	Array of PageInfo[1]

1: bindery.info/book/#pageinfo

Gallery

Gallery



print.are.na

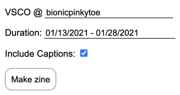
1. Enter the URL of a public Are.na channel: ___
2. (Optional) Choose your settings:
   - ☑ include table of contents
   - ☑ display author
   - ☑ display source
   - ☑ display description
   - ☑ reverse chronological order (newest first)

print.are.na

Generate a book from an Are.na channel

Mindy Seu, Charles Broskoski, Ekene Ijeoma, 2020

1



VSCO @ bionicpinkytoe

Duration: 01/13/2021 - 01/28/2021

Include Captions: ☑

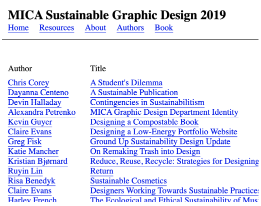
VSCO Zine

Create a zine from a VSCO profile.

Trudy Painter, 2020

2

1: print.are.na
2: vsco-zine.herokuapp.com



MICA Sustainable Graphic Design 2019

Home Resources About Authors Book

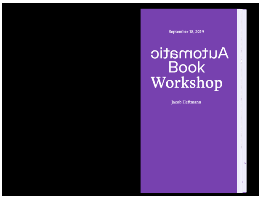
Author	Title
Chris Corey	A Student's Dilemma
Dayanna Centeno	A Sustainable Publication
Devin Halladay	Contingencies in Sustainabilitism
Alexandra Petrenko	MICA Graphic Design Department Identity
Kevin Guyer	Designing a Compostable Book
Claire Evans	Designing a Low-Energy Portfolio Website
Greg Fisk	Ground Up Sustainability Design Update
Katie Mancher	On Remaking Trash into Design
Kristian Bjørnard	Reduce, Reuse, Recycle: Strategies for Designing
Ruyin Lin	Return
Risa Benedyk	Sustainable Cosmetics
Claire Evans	Designers Working Towards Sustainable Practice
Harley French	The Ecological and Ethical Sustainability of Mus

MICA Sustainable Graphic Design

Collected thinking on design and the environment

Members of Sustainable Design, MICA Fall 2019

1



Automatic Book Workshop

At Hochschule der Bildenden Künste Saar

Jacob Heftmann, 2019

2

1: gd314-sgd.github.io/publication

2: www.jacobheftmann.com/teaching/automatic-book-workshop

Every Photo Taken Between July 10 and September15

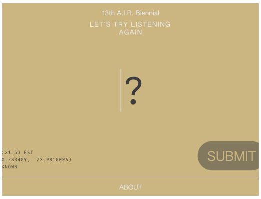
This site takes a Small Victories Feed and uses Bindery.js to automatically turn every iPhone photo I take into a formatted book.

Every Photo

A book generated from a Small Victories feed

Jacob Heftmann, 2019

1



Let's try listening again

Catalog for the 13th annual A.I.R. Biennial

Lukas Eigler-Harding, 2019

2

1: bindery-demo.smvi.co
2: letstrylisteningagain.org

Cita Press

Feminist indie press publishing public-domain books by women

Juliana Castro, 2017

1

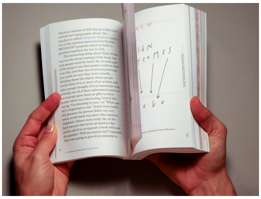
John Caserta

Book and site to document projects, writing, and teaching

John Caserta, 2017

2

1: citapress.org
2: johncaserta.com



for/with/in

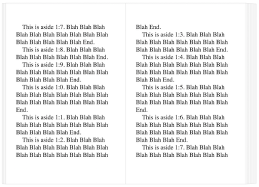
Book and site to explore the web browser as a design tool

Members of HTML Output, RISD Fall 2014

1

Examples to get started

1: htmloutput.risd.gd



This is aside 1:7. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:8. Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:9. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:0. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:1. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:2. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah

Blah End.
This is aside 1:3. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:4. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:5. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:6. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
This is aside 1:7. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah

Getting Started

Fetch content and flow it through pages

1

4 · Book

First chapter title · 5

First chapter title

This is paragraph 1:0. Blah Blah Blah Blah Blah End.[1]
This is paragraph 1:1. Blah Blah Blah Blah Blah Blah Blah Blah Blah End.[2]
This is paragraph 1:2. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.[3]

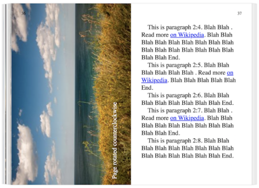
[1] Default footnote (Learn how to change it)
[2] Default footnote (Learn how to change it)
[3] Default footnote (Learn how to change it)

Configure Rules

Page breaks, footnotes, and a table of contents

2

1: bindery.info/examples/2_content_on_separate_page
2: bindery.info/examples/3_using_configuration_options



This is paragraph 2:4. Blah Blah . Read more on Wikipedia. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 2:5. Blah Blah Blah Blah Blah Blah . Read more on Wikipedia. Blah Blah Blah Blah Blah End.

This is paragraph 2:6. Blah Blah Blah Blah Blah Blah Blah Blah End.

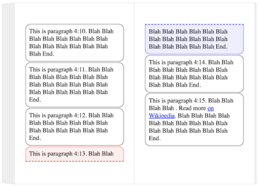
This is paragraph 2:7. Blah Blah . Read more on Wikipedia. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 2:8. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

Spreads and Images

Full-bleed, out-of-flow spreads

1



This is paragraph 4:10. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 4:11. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 4:12. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 4:13. Blah Blah

Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

This is paragraph 4:14. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

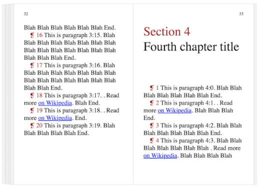
This is paragraph 4:15. Blah Blah Blah Blah . Read more on Wikipedia. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.

Handle a Split

Indicate that an element starts or ends on another page

2

1: bindery.info/examples/9_spreads_and_images
2: bindery.info/examples/7_custom_split


32

Blah Blah Blah Blah Blah Blah End.
¶ 16 This is paragraph 3:15. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
¶ 17 This is paragraph 3:16. Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah Blah End.
¶ 18 This is paragraph 3:17. . Read more on Wikipedia. Blah End.
¶ 19 This is paragraph 3:18. . Read more on Wikipedia. End.
¶ 20 This is paragraph 3:19. Blah Blah Blah Blah Blah End.

33

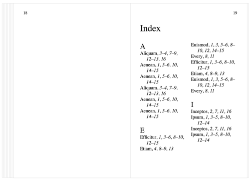
Section 4
Fourth chapter title

¶ 1 This is paragraph 4:0. Blah Blah Blah Blah Blah Blah Blah End.
¶ 2 This is paragraph 4:1. . Read more on Wikipedia. Blah Blah Blah End.
¶ 3 This is paragraph 4:2. Blah Blah Blah Blah Blah Blah Blah End.
¶ 4 This is paragraph 4:3. Blah Blah Blah Blah Blah Blah Blah . Read more on Wikipedia. Blah Blah Blah Blah

Counters

Number figures and sections (alternative to CSS Counters)

1


18

19

Index

A
Aliquam, *3–4, 7–9, 12–13, 16*
Aenean, *1, 5–6, 10, 14–15*
Aenean, *1, 5–6, 10, 14–15*
Aliquam, *3–4, 7–9, 12–13, 16*
Aenean, *1, 5–6, 10, 14–15*
Aenean, *1, 5–6, 10, 14–15*

E
Efficitur, *1, 3–6, 8–10, 12–15*
Etiam, *4, 8–9, 13*
Euismod, *1, 3, 5–6, 8–10, 12, 14–15*
Every, *8, 11*
Efficitur, *1, 3–6, 8–10, 12–15*
Etiam, *4, 8–9, 13*
Euismod, *1, 3, 5–6, 8–10, 12, 14–15*
Every, *8, 11*

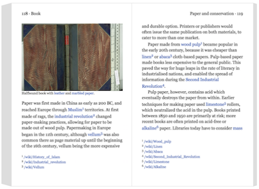
I
Inceptos, *2, 7, 11, 16*
Ipsum, *1, 3–5, 8–10, 12–14*
Inceptos, *2, 7, 11, 16*
Ipsum, *1, 3–5, 8–10, 12–14*

Index

Automatically list every page that matches a query

2

1: bindery.info/examples/10_counters
2: bindery.info/examples/8_index



118 - Book

Halfbound book with leather and marbled paper.

Paper was first made in China as early as 200 BC, and reached Europe through Muslim[1] territories. At first made of rags, the industrial revolution[2] changed paper-making practices, allowing for paper to be made out of wood pulp. Papermaking in Europe began in the 11th century, although vellum[3] was also common there as page material up until the beginning of the 16th century, vellum being the more expensive

[1] /wiki/History_of_Islam
[2] /wiki/Industrial_revolution
[3] /wiki/Vellum

Paper and conservation - 119

and durable option. Printers or publishers would often issue the same publication on both materials, to cater to more than one market.

Paper made from wood pulp[1] became popular in the early 20th century, because it was cheaper than linen[2] or abaca[3] cloth-based papers. Pulp-based paper made books less expensive to the general public. This paved the way for huge leaps in the rate of literacy in industrialised nations, and enabled the spread of information during the Second Industrial Revolution[4].

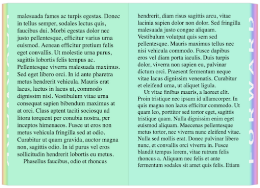
Pulp paper, however, contains acid which eventually destroys the paper from within. Earlier techniques for making paper used limestone[5] rollers, which neutralized the acid in the pulp. Books printed between 1850 and 1950 are primarily at risk; more recent books are often printed on acid-free or alkaline[6] paper. Libraries today have to consider mass

[1] /wiki/Wood_pulp
[2] /wiki/Linen
[3] /wiki/Abaca
[4] /wiki/Second_Industrial_Revolution
[5] /wiki/Limestone
[6] /wiki/Alkaline

Wikipedia Article

Create a book from HTML you didn't write

1



malesuada fames ac turpis egestas. Donec in tellus semper, sodales lectus quis, faucibus dui. Morbi egestas dolor nec justo pellentesque, efficitur varius urna euismod. Aenean efficitur pretium felis eget convallis. Ut molestie urna purus, sagittis lobortis felis tempus ac. Pellentesque viverra malesuada maximus. Sed eget libero orci. In id ante pharetra metus hendrerit vehicula. Mauris erat lacus, luctus in lacus ut, commodo dignissim nisl. Vestibulum vitae urna consequat sapien bibendum maximus at at orci. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Fusce ut eros non metus vehicula fringilla sed at odio. Curabitur ut quam gravida, auctor magna non, sagittis odio. In id purus vel eros sollicitudin hendrerit lobortis eu metus.

Phasellus faucibus, odio et rhoncus hendrerit, diam risus sagittis arcu, vitae lacinia sapien dolor non dolor. Sed fringilla malesuada justo congue aliquam. Vestibulum volutpat quis sem sed pellentesque. Mauris maximus tellus nec nisi vehicula commodo. Fusce dapibus eros vel diam porta iaculis. Duis turpis dolor, viverra non sapien eu, pulvinar dictum orci. Praesent fermentum neque vitae lacus dignissim venenatis. Curabitur et eleifend urna, ut aliquet ligula.

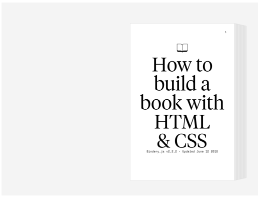
Ut vitae finibus mauris, a laoreet elit. Proin tristique nec ipsum id ullamcorper. In quis magna non lacus efficitur commodo. Ut quam leo, porttitor sed tortor eget, sagittis tristique quam. Nulla dignissim enim eget euismod aliquam. Maecenas pellentesque metus tortor, nec viverra nunc eleifend vitae. Nulla sed mollis erat. Donec pulvinar libero nunc, et convallis orci viverra in. Fusce blandit tempus lorem, vitae rutrum felis rhoncus a. Aliquam nec felis et ante fermentum sodales sit amet quis felis. Etiam

Create your own rules

Dynamic background and font size, fore-edge graphics

2

1: bindery.info/examples/6_wikipedia
2: bindery.info/examples/4_custom_rules


1

How to
build a
book with
HTML
& CSS

Bindery.js v2.2.2 · Updated June 12 2018

How to make a book with HTML & CSS

These docs as a book

1

1: bindery.info/examples

About

Approach

Bindery is intended for anyone new to web design. The documentation doesn't assume you know javascript, and you don't need a module bundler or build step to get started. If you need help, or something seems unclear in the documentation, feel free to open an issue[1] on Github or ask @bindery_js[2] on twitter.

Bindery relies on your browser's PDF generation. If you want to adjust a PDF setting, your browser must support it— different browsers and platforms may have different options.

Bindery is open source—report bugs, make suggestions, or lend a hand on Github[3].

History

Bindery.js 1.0 was developed in Spring 2014 for

1: github.com/evnbr/bindery/issues/new/choose
2: twitter.com/bindery_js
3: github.com/evnbr/bindery

for/with/in[1], a publication from participants in the graphic design course *HTML Output* at RISD[2]. Fellow course members Catherine Leigh Schmidt[3] and Lukas WinklerPrins[4] produced a Jekyll theme called Baby Bindery[5] for the Design Office[6] based on this initial version.

Bindery.js 2.0 has been developed since February 2017 by Evan Brooks[7]. It has been rewritten from scratch to be smaller, faster, more flexible, and more robust. With thanks to John Caserta[8] and Teddy Bradford[9] for contributions and feedback.

Web browsers may eventually support some of Bindery's features natively— see CSS Paged Media Level 3[10] and CSS Generated Content[11]. Note that

1: htmloutput.risd.gd
2: risd.edu
3: cath.land
4: ltwp.net
5: github.com/thedesignoffice/babybindery
6: thedesignoffice.org
7: evanbrooks.info
8: johncaserta.com
9: teddybradford.com
10: drafts.csswg.org/css-page-3
11: www.w3.org/TR/css-gcpm-3

CSS drafts and standards aren't guaranteed to be adopted. The initial version of Bindery was based on the CSS Regions[1] draft, which was criticized[2] and later abandoned by Chrome[3]. Cross-element layout in Bindery is currently handled by regionize.js[4].

1: drafts.csswg.org/css-regions
2: alistapart.com/blog/post/css-regions-considered-harmful
3: arstechnica.com/information-technology/2014/01/google-plans-to-dump-adobe-css-tech-to-make-blink-fast-not-rich
4: github.com/evnbr/regionize

Colophon

Text is set in Tiempos Headline[1] by Klim Type[2], and code samples are set in Input Mono[3] by David Jonathan Ross[4].

This page was rendered at [Time and Date] with [Browser].

This site was last updated at 3:23 AM on Sunday, February 06, 2022. It is built with Jekyll[5] and hosted on Github Pages[6]. Its source code is available here[7].

1: klim.co.nz/retail-fonts/tiempos-headline
2: klim.co.nz
3: input.fontbureau.com
4: djr.com
5: jekyllrb.com
6: pages.github.com
7: github.com/evnbr/bindery/tree/master/docs