

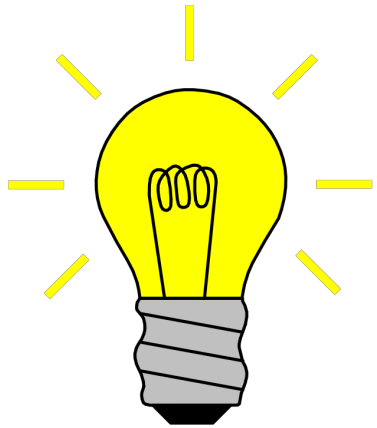
HTML5 Is the Future of Book Authorship

**Digital Book World
January 14, 2014**

Sanders Kleinfeld
O'Reilly Media, Inc.

**The Goal of
Publishing:**

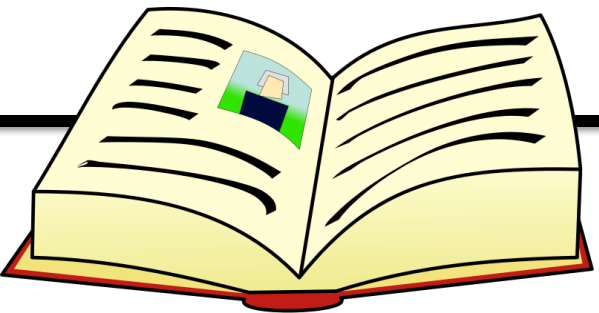
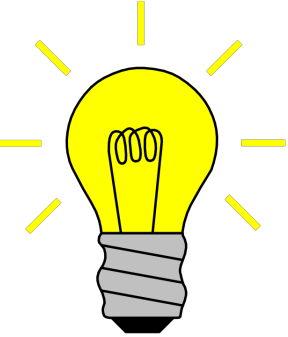
**Packaging and
Distribution of Ideas**



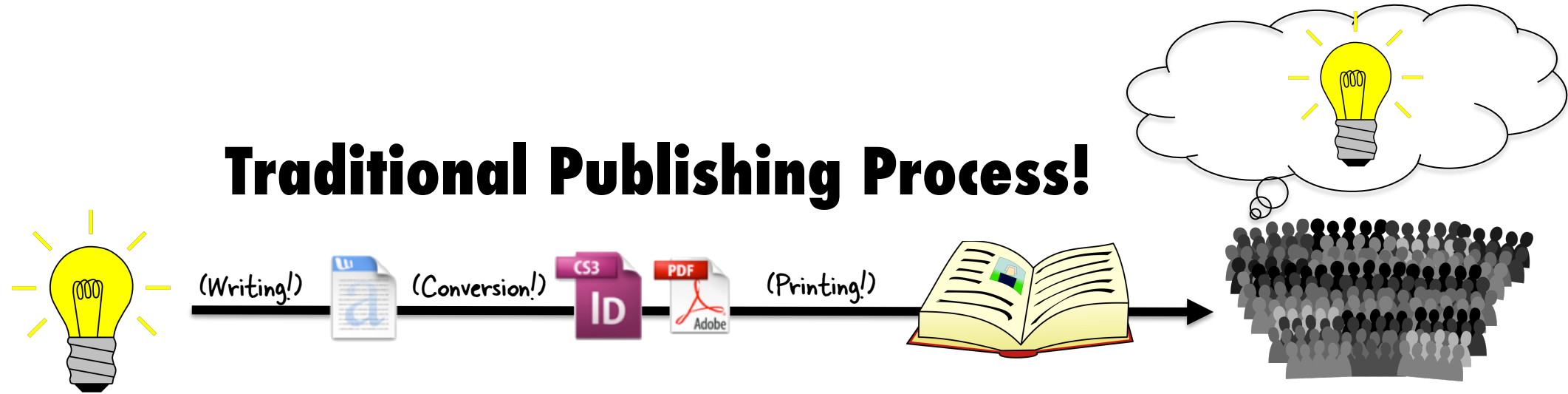
Publishing!



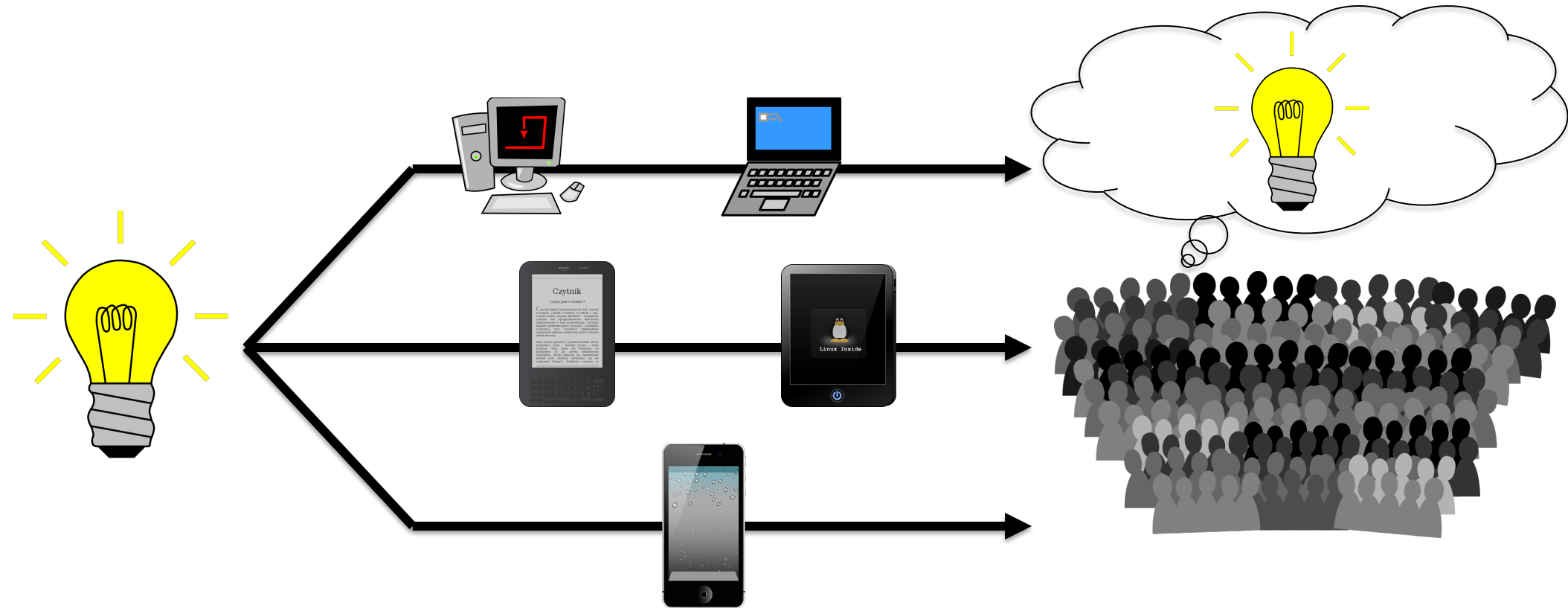
Traditional Publishing!



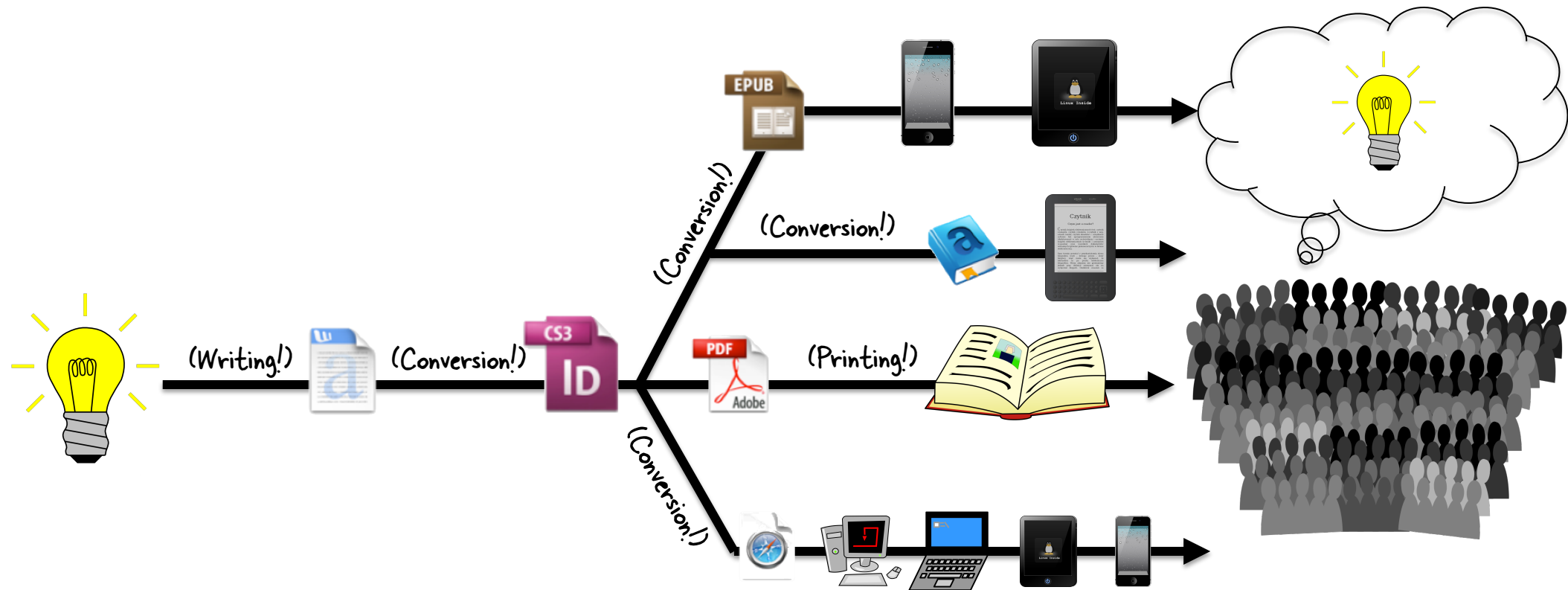
Traditional Publishing Process!



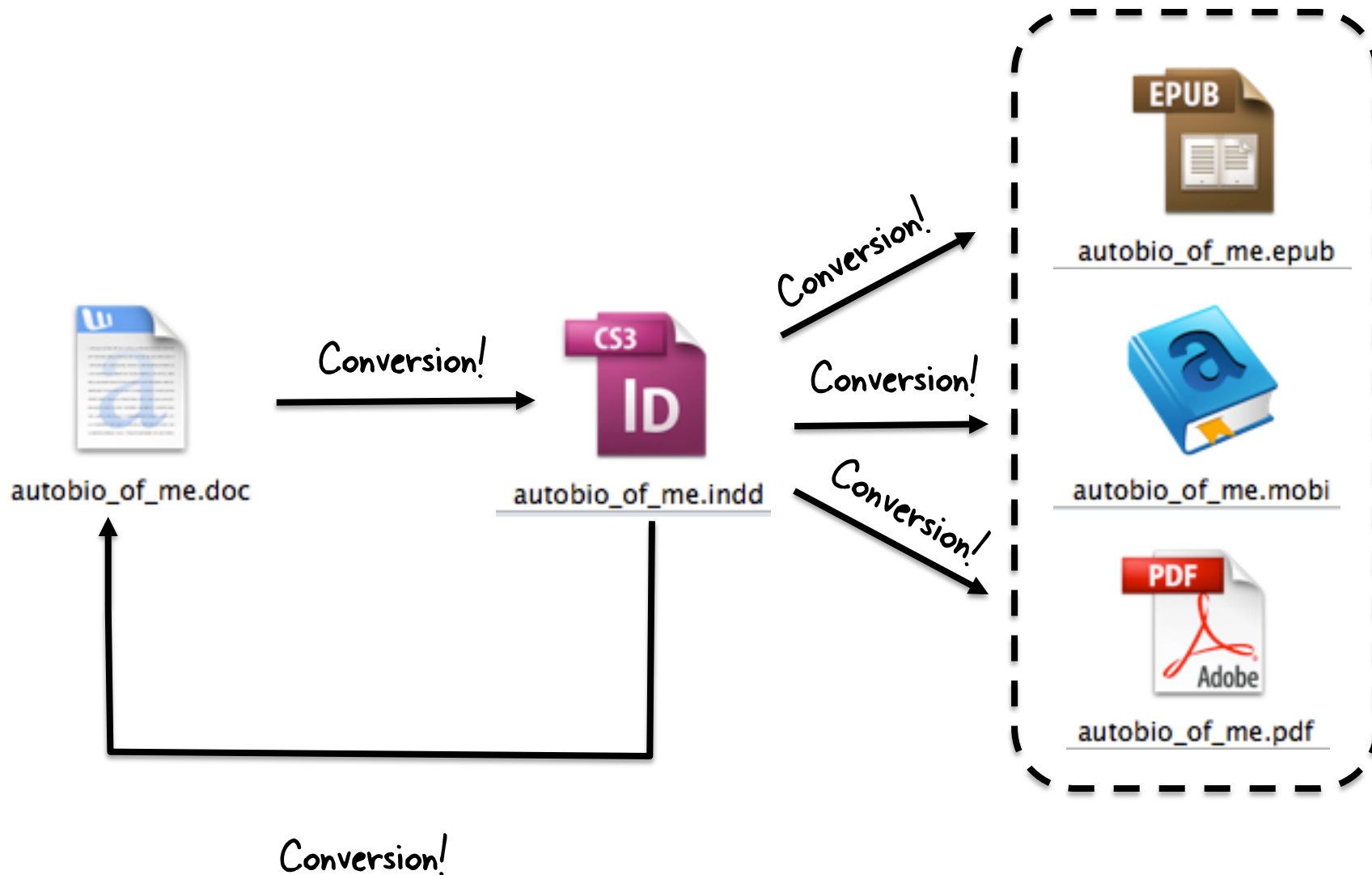
Digital Publishing!



(Post)Modern Publishing Process! (Both Print and Digital)



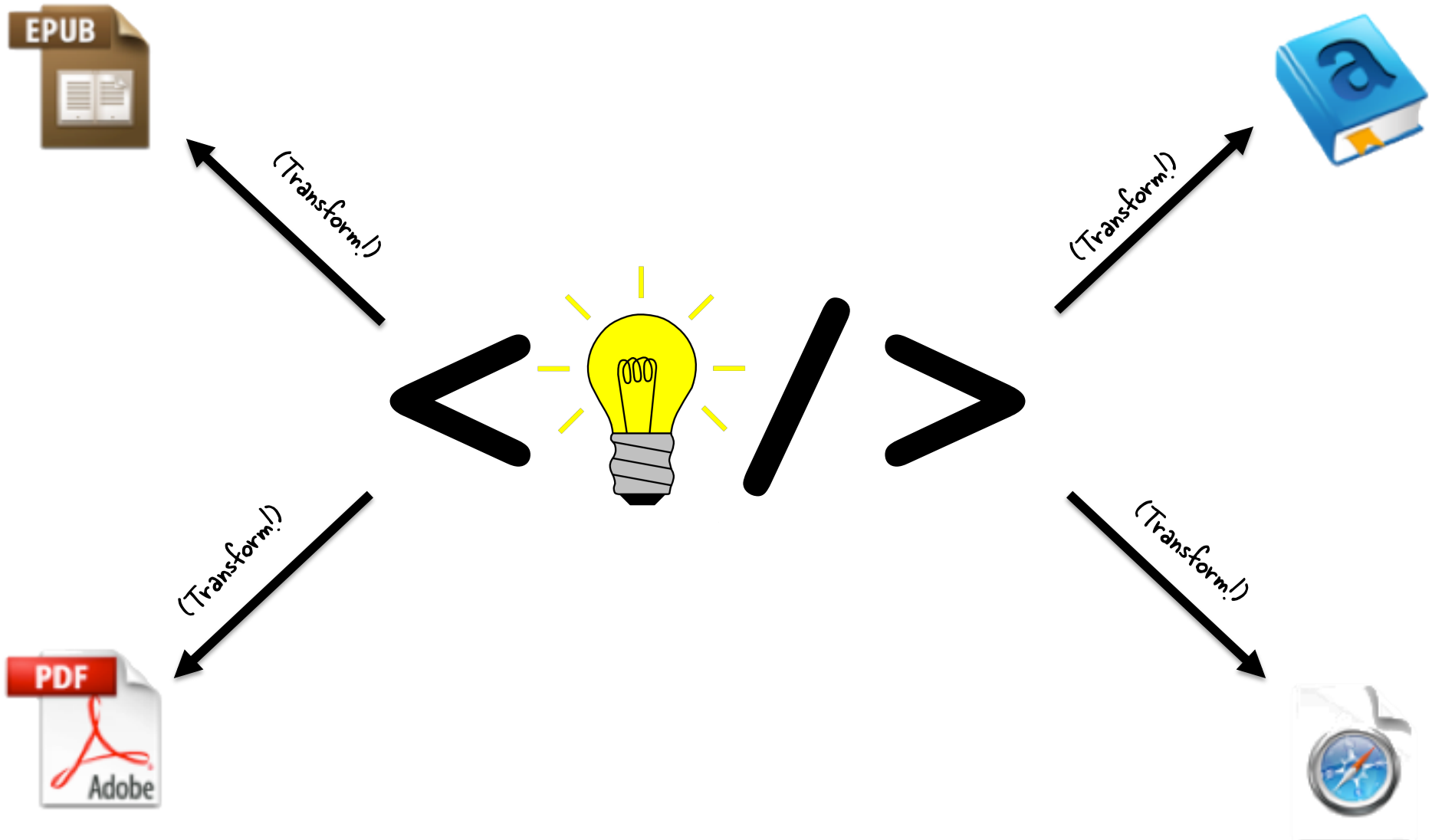
Welcome to Conversion City



Enjoy Your Stay 😊

The Single-Source Solution:

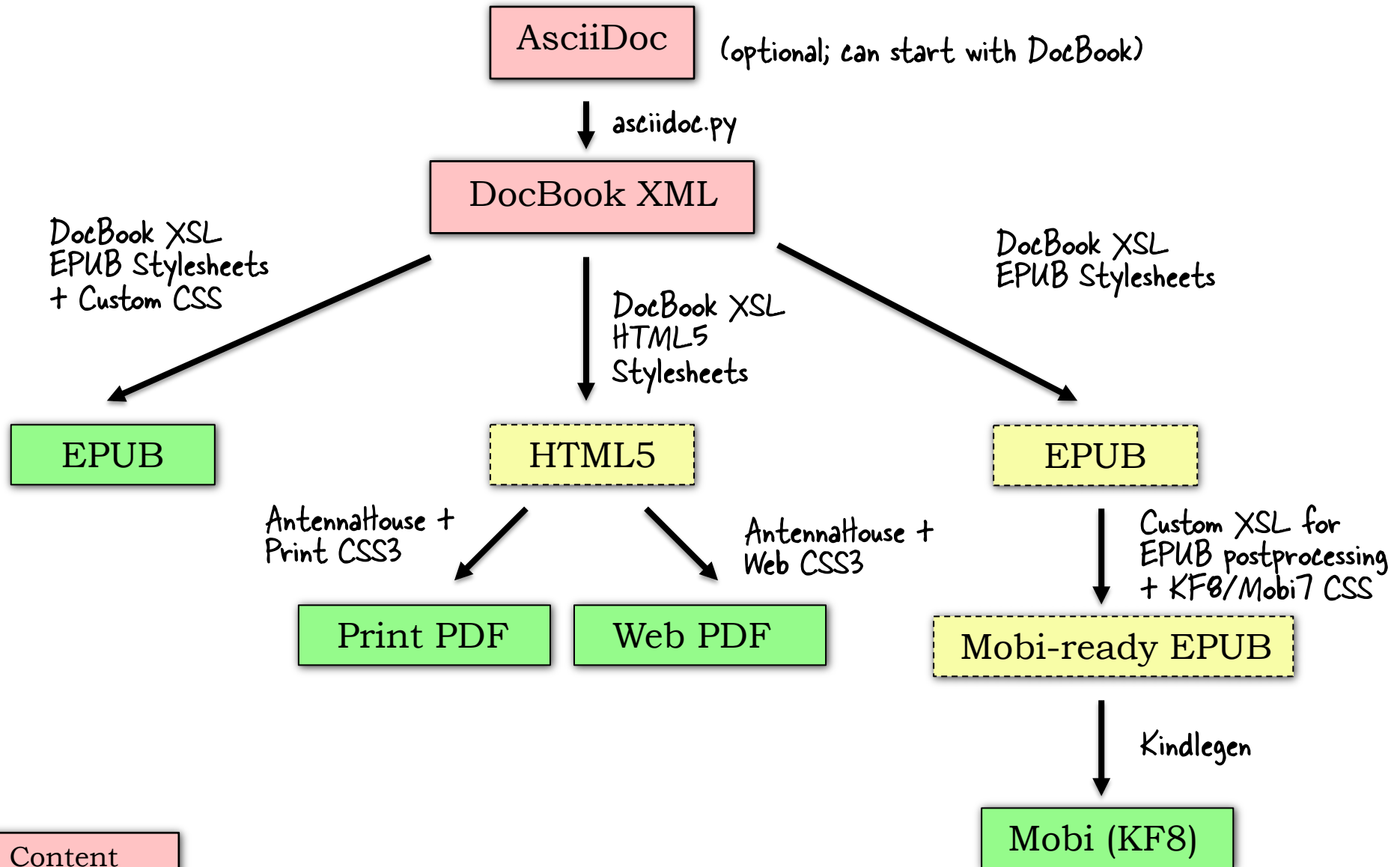
Replace conversions with semantic markup and automated transforms



How We Did It:

- 1. Encourage authors to write in DocBook (heavyweight, *semantic* XML markup) or AsciiDoc (lightweight, *semantic* wiki-like “markdown”)**
- 2. If authors prefer to write in Microsoft Word (☹), let them (☹☹☹), but convert to DocBook when book goes into Production**
- 3. Maintain a customized version of the DocBook project stylesheets for automatically generating print/ebook outputs**

O'Reilly's Single-Source Workflow (2006-2013):



Source Content

Intermediate Output

Final Output For Sale

**Three Slowly
Dawning
Realizations About
Our Workflow**

**Realization #1:
Our toolchain is
rather heavyweight,
complex**

PDF* Toolchain Stats

The DocBook project XHTML5 stylesheets** contain:

- **33,707** lines of HTML-generation code...
- ...which rely on **8,346** lines of common dependencies

Or, in terms of functions, they contain:

- **1,857** `<xsl:template>`s...
- ...which rely on **272** common dependency `<xsl:template>`s

* Separate code base for EPUB/Mobi

** docbook-epub3-addon-b3

**When doing
transforms,
this complexity is a
necessary evil,
emphasis on *evil***

Peril of a Transform- Heavy Workflow:

Troubleshooting is a real
\$!%#@&*

An example!

Let's say your DocBook source is:

```
<chapter>  
  <title>Poodles and Cookies</title>  
  ...  
</chapter>
```

**And your desired/expected HTML
output is:**

```
<div class="chapter">  
  <h1>Poodles and Cookies</h1>  
  ...  
</div>
```

But then you run your transform...

```
<chapter>
  <title>Poodles
and Cookies</title>
  ...
</chapter>
```



**33,707
lines of
code**



```
<p class="sect1">
  <h2>Poodles and
Cookies</h2>
  ...
</p>
```

...And you say, “What the \$!%#@&*?”

What about implementing validation to preemptively catch errors before running the transforms?

**Streamlining Production
Workflows isn't just about
automating conversions
whenever possible...**

**...It's about *eliminating*
conversions whenever
possible!**

Rather than build a toolchain around:



Or:



Or:







Why not seriously consider:



**Realization #2:
HTML5 Is Ideal for
Digital-First
Content**

Digital-First Content Development

When doing digital-first (ebook/web) content development, these are the key output formats:

- **EPUB (2.0 and 3.0)** The icon shows a brown document with a white page and a small tab at the top labeled 'EPUB'.
- **Amazon Kindle Mobi (Mobi7/KF8)** The icon is a blue 3D book with a white spine and a yellow bookmark, featuring a white lowercase 'a' on the cover.
- **PDF** The icon is a white document with a red Adobe logo and the word 'Adobe' at the bottom, with a red 'PDF' label at the top.
- **HTML** The icon is a white document with a blue and white compass rose in the center.

The Common Thread: HTML + CSS



= HTML + CSS + open source packaging



= HTML + CSS + proprietary packaging



= HTML + CSS + PDF processor*



= HTML + CSS (duh!)

* e.g., AntennaHouse Formatter or Prince

Interactivity/Multimedia Is Ultimately All About HTML5

Animation/Games → `<canvas>` or `<svg>`

Music/Narration → `<audio>`

Video Clips → `<video>`

Math Equations → `<math>`

It's way easier to do stuff like this:



If you start with HTML5

It's way easier to do stuff like this:

helpful function that calculates and returns the center point of any arc. We translate each text label element to each arc's centroid, and that's how we get the text labels to float right in the middle of each wedge.

Try It Now!

Try changing the values in dataset and see how this affects the rendering of the pie chart.

[Open in new window](#)

```
JS Bin Save HTML CSS JavaScript Console Output Help
```

```
HTML
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>D3: Pie layout</title>
    <script type="text/javascript"
src="http://d3js.org/d3.v3.min.js">
    </script>
    <style type="text/css">

      text {
        font-family: sans-serif;
        font-size: 12px;
        fill: white;
      }

    </style>
  </head>
  <body>
    <script type="text/javascript">
```

Segment Color	Value
Red	45
Brown	25
Green	20
Orange	10
Purple	6
Blue	5

Bonus tip: Remember how `arc()` required an `innerRadius` value? We can expand that to anything greater than zero, and our pie chart becomes a *ring* chart like the one shown in [Figure 11-3](#):

```
var innerRadius = w / 3;
```

If you start with HTML5


**Realization #3:
Authors Generally
Don't Want to Deal
with Markup**

**Authors prefer *visual*
authoring platforms
because...**

**“Nobody’s going to
learn your markup
language”**

Books in Browsers 2012: Liza Daly & Keith Fahlgren,
“The self-publishing book”
<http://www.youtube.com/watch?v=UWftLHopWQ0#t=5m25s>

Non-Technical Authors Don't Like This...



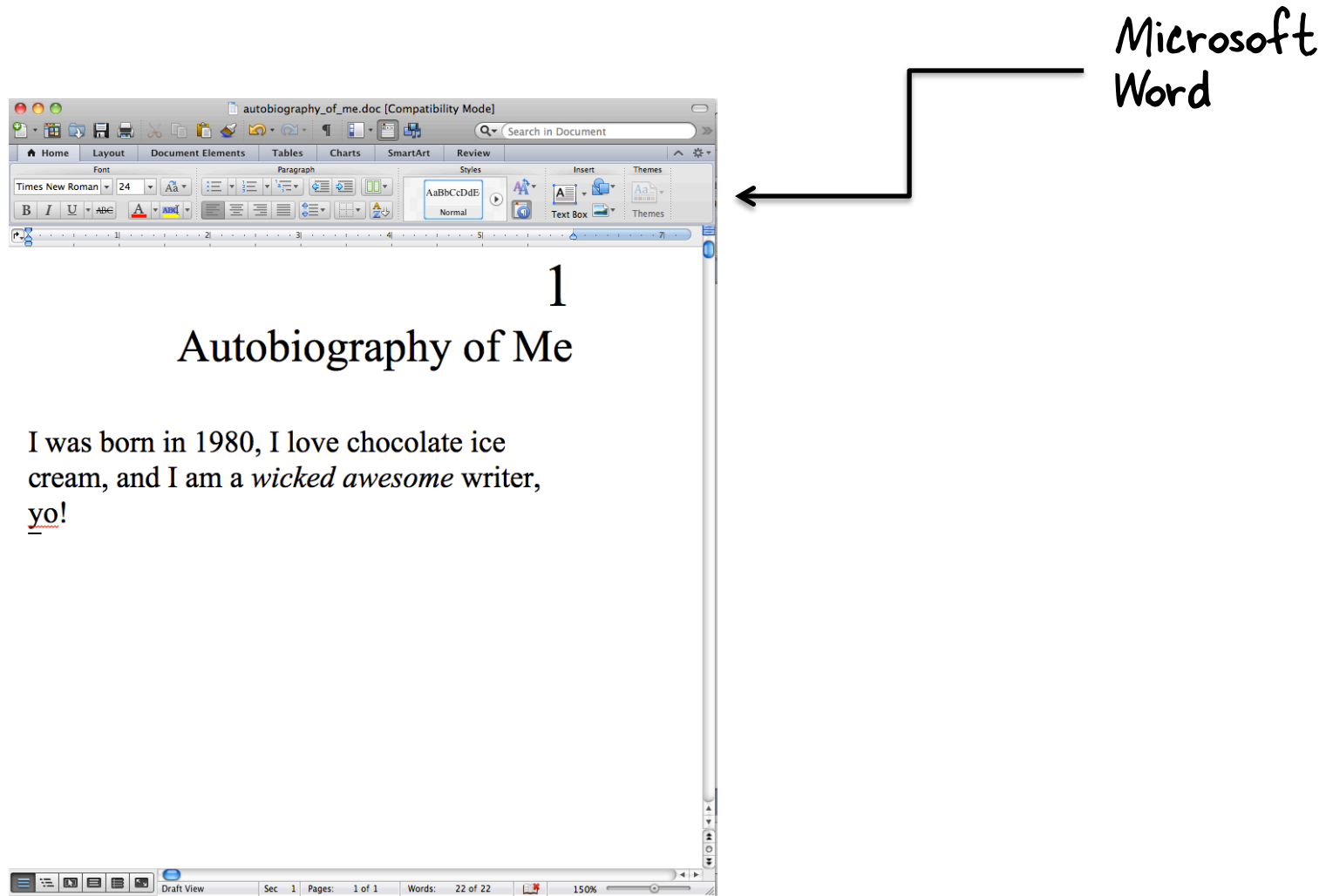
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"
"http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">
<chapter>
  <title>Autobiography of Me</title>
  <para>I was born in 1980, I love chocolate ice cream, and I
am a <emphasis>wicked awesome</emphasis> writer, yo!</para>
</chapter>
```

Non-Technical Authors Will Sometimes Tolerate This...

`== Autobiography of Me`  `AsciiDoc`

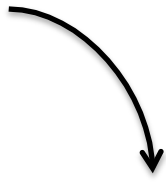
I was born in 1980, I love chocolate ice cream,
and I am a wicked awesome writer, yo!

Non-Technical Authors Really Want This...



**But This is the
Future of Digital
Content Creation:**

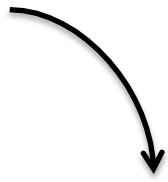
Medium



(Short-Form Web Publishing)

The screenshot shows a web browser window with the title 'Editing Autobiography of Me'. The address bar contains 'https://medium.com/...' and the search bar has 'Google'. The browser's bookmark bar includes 'Apple', 'Yahoo!', 'Google Maps', 'YouTube', 'Wikipedia', 'News (529)', and 'Popular'. On the page, there is a Medium logo (a black square with a white 'M') and the word 'SAVED'. To the right are three buttons: 'Delete', 'Share Draft', and 'Publish'. Below these is a placeholder image icon. The main heading is 'Autobiography of Me' in a large, bold font, followed by the subtitle 'Type your subtitle (optional)'. The main text of the post reads: 'I was born in 1980, I love chocolate ice cream, and I am a wicked awesome writer, yo!'.

O'Reilly
Atlas



(Short and Long-Form Print,
Digital, and Web Publishing)

The screenshot shows a web browser window with the address bar displaying <https://atlas.oreilly.com/sandersk/Autobioofme/editor/master/autobio.html>. The browser tab is titled 'autobio.html at master - A x'. The editor interface includes a left-hand sidebar with a 'Files' panel containing a list of files: atlas.json, autobio.html (highlighted), graphing_calculator_ipad.png, and svg_coloring_book_final.png. Below the file list are buttons for 'New File' and 'New Folder', and a 'Drag a file here to upload' area. The main editing area features a rich text editor toolbar with icons for bold (B), italic (I), underline (U), link, unlink, list, ordered list, quote, table, code, and a 'Paste From...' dropdown. The text content is enclosed in a dashed border and includes a 'Chapter' label, the title 'Autobiography of me', and the paragraph 'I was born in 1980, I love chocolate ice cream, and I am a *wicked awesome* writer, yo!'. On the right side of the editor, there are three buttons: 'Save', 'File History', and 'Builds'.

Next-Generation Content Authoring =

- **Visual Editing**
- **Web-Based (Responsive Design)**
- **Version-Controlled**
- **Seamless**

Visual Editing

The screenshot shows a web browser window with the URL `https://atlas.oreilly.com/oreilly-media-content/razzpisampler/editor/master/Connecting_an_LED.html`. The browser's address bar and navigation buttons are visible at the top. Below the browser, there is a document editor interface. On the left side, there are three buttons: 'Back', 'Code Editor', and 'Files'. The main editing area contains a rich text editor with a toolbar at the top featuring bold, italic, underline, link, unlink, list, ordered list, quote, table, code, and paste options. The document content is structured as follows:

- Chapter**
 - Connecting an LED**
 - A video player showing a video titled "9.1 Connecting an LED" from the "Raspberry Pi Cookbook" by Simon Monk. The video player includes a play button, a progress bar at 0:00 / 6:19, and a YouTube logo.
- Section 1**
 - Problem**

You want to know how to connect an LED to the Raspberry Pi.
 - Solution**

Connect an LED (see "Opto-Electronics") to one of the GPIO pins using a 470Ω or 1kΩ series resistor (see "Resistors and Capacitors") to limit the current. To make this

On the right side of the editor, there is a 'Builds' sidebar. It contains a grid of build options: PDF, EPUB, MOBI, and HTML. Below this grid is a 'Build' button. The sidebar lists several build entries, each with a user profile icon, a timestamp, a status icon (refresh or checkmark), a format type (HTML or PDF), and a 'Download' or 'Build Log' link. The entries are:

- 10 minutes ago: HTML (refresh icon), Build Log
- 11 minutes ago: PDF (refresh icon), Build Log
- 17 days ago: HTML (checkmark icon), Download, Build Log
- 17 days ago: HTML (checkmark icon), Download, Build Log
- 20 days ago: HTML (refresh icon), Download

Responsive Design

The screenshot shows a mobile browser interface on an iPad. The address bar displays 'atlas.oreilly.com' and the page title is 'Connecting_an_LED.html at master - razzpisampler | O'Reilly Atlas'. The page content includes a video player for 'Connecting an LED' from the 'Raspberry Pi Cookbook' by Simon Monk. Below the video, the page is divided into sections: 'Section 1 Problem' and 'Section 1 Solution'. The 'Problem' section states: 'You want to know how to connect an LED to the Raspberry Pi.' The 'Solution' section provides instructions on connecting an LED to a GPIO pin using a resistor, and lists the required components: a breadboard and jumper wires, a 1kΩ resistor, and an LED. A diagram at the bottom shows the physical connection of an LED to a breadboard and a Raspberry Pi GPIO pin.

Section 1

Problem

You want to know how to connect an LED to the Raspberry Pi.

Section 1

Solution

Connect an LED (see “Opto-Electronics”) to one of the GPIO pins using a 470Ω or 1kΩ series resistor (see “Resistors and Capacitors”) to limit the current. To make this recipe, you will need:

- Breadboard and jumper wires (see “Prototyping Equipment”)
- 1kΩ resistor (see “Resistors and Capacitors”)
- LED (see “Opto-Electronics”)

shows how you can wire this using solderless breadboard and male-to-female jumper leads.

Connecting an LED to a Raspberry Pi

The diagram illustrates the physical connection of an LED to a breadboard and a Raspberry Pi. A red LED is connected to a breadboard. One lead of the LED is connected to a male-to-female jumper lead, which is plugged into a Raspberry Pi GPIO pin. The other lead of the LED is connected to another male-to-female jumper lead, which is plugged into a different GPIO pin. The breadboard is shown with its standard layout of holes and power rails.

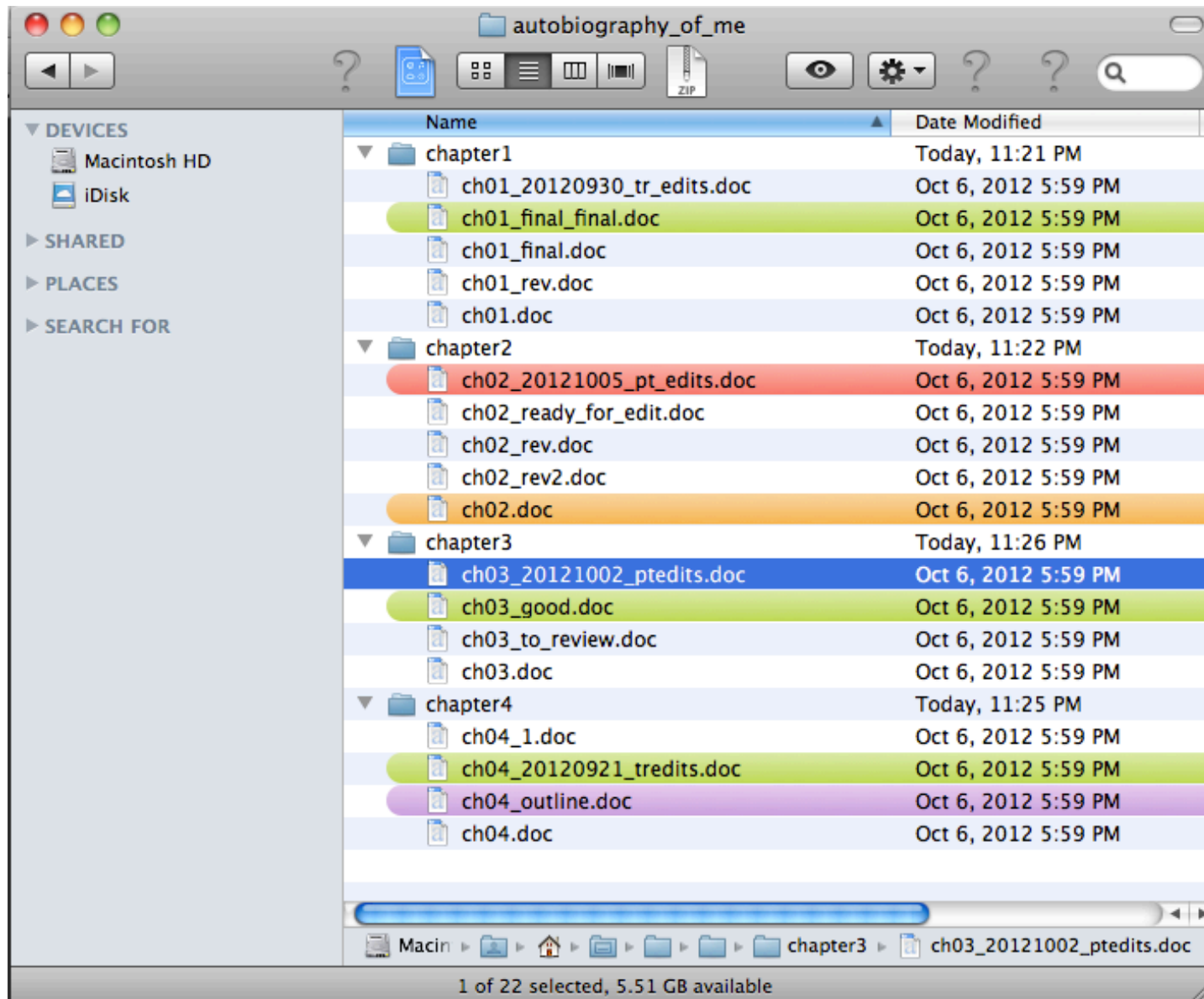
On Version Control...

Two Questions About Your (e)Book's Editorial Lifecycle

- 1. Will more than one person be working on the manuscript files?**
- 2. Will there be more than one draft of the manuscript?**

**If you answered *yes*
to either question,
you *need* a version-
control system.**

Key Feature #1 of Version Control: Revision Snapshots



Key Feature #2 of Version Control: *Diffing*

The screenshot displays a Microsoft Word window titled "tale_of_two_cities_corrected.doc [Compatibility Mode]". The ribbon includes Home, Layout, Document Elements, Tables, Charts, SmartArt, and Review. The Review tab is active, showing the Track Changes pane on the right. The document text is as follows:

I. The Period

It was the best of times,
~~it~~ was the worst of times,
~~it~~ was the age of wisdom,
~~it~~ was the age of foolishness,
~~it~~ was the epoch of belief,
~~it~~ was the epoch of incredulity,
~~it~~ was the season of Light,
~~it~~ was the season of Darkness,
~~it~~ was the spring of hope,
~~it~~ was the winter of despair,
~~We~~ had everything before us,
~~We~~ had nothing before us,
~~We~~ were all going direct~~ly~~ to Heaven,
~~We~~ were all going direct~~ly~~ the other way.

In short, the period was so far like the present period, that some of
~~its~~ noisiest authorities insisted on its being received, for good or for
~~evil~~, in the superlative degree of comparison only.

There were a king with a large jaw and a queen with a plain face, on the
throne of England; there were a king with a large jaw and a queen with
a fair face, on the throne of France. In both countries, it was clearer
~~than~~ crystal to the lords of the State preserves of loaves and fish,
~~that~~ things in general were settled forever.











The right-hand pane shows several comments and deletions:

- Comment 1: Deleted: , (Sanders Klei..., 10/8/12 11:33 PM)
- Comment 2: Deleted: -- (Sanders Klei..., 10/8/12 11:35 PM)
- Comment 3: Deleted: i (Sanders Klei..., 10/8/12 11:35 PM)
- Comment 4: Deleted: , (Sanders Klei..., 10/8/12 11:36 PM)
- Comment 5: Deleted: es (Sanders Klei..., 10/8/12 11:36 PM)
- Comment 6: Deleted: (Sanders Klei..., 10/8/12 11:36 PM)

The status bar at the bottom indicates "Print Layout View", "Sec 1", "Pages: 1 of 1", "Words: 154 of 188", and "150%".

**What if we
versioned
manuscripts like
software developers
version code?**

Revision snapshots

Dec 11, 2013		
	Fix formatting. benjaminwei authored a month ago	b30cb966a0 + Browse code →
Dec 10, 2013		
	Fixed formatting and typo. ... benjaminwei authored a month ago	6bbabb9fb7 + Browse code →
	Fix typo. benjaminwei authored a month ago	7f7fbb22f1 + Browse code →
Dec 03, 2013		
	Merge pull request #6 from benjaminwei/patch-5 ... jdeisenberg authored a month ago	ee5b282061 + Browse code →
	Merge pull request #13 from mikulely/patch-3 ... jdeisenberg authored a month ago	8eca76d706 + Browse code →
	Update wiki link mikulely authored a month ago	0a7b0af4b3 + Browse code →
Dec 01, 2013		
	IO.readline has been deprecated, IO.read(device, :line) works ... mikulely authored a month ago	9faae9a4f8 + Browse code →
	IO.readline has been deprecated, IO.read(device, :line) works ... mikulely authored a month ago	8535c0e78b + Browse code →
Nov 17, 2013		
	Merge pull request #8 from petehuang/patch-1 ... jdeisenberg authored 2 months ago	add7c38837 + Browse code →
	Merge pull request #9 from petehuang/master ... jdeisenberg authored 2 months ago	aa7d521334 + Browse code →

Diffing


PUBLIC oreillymedia / etudes-for-elixir

Unwatch 20 Star 66 Fork 18

Changed grammar error in comments for solution ch07-06.

[Browse code](#)

master

 jdeisenberg authored a day ago

1 parent [47a70c1](#) commit [5933a2355ad9c371c5679b60f545901d7613771](#)

Showing 1 changed file with 1 addition and 1 deletion.

[Show Diff Stats](#)

code/ch07-06/cards.ex [View file @ 5933a23](#)

@@ -35,7 +35,7 @@ defmodule Cards do		
35	35	# That first card goes onto a new pile ([h acc]).
36	36	# Now put together the part above the split and the
37	37	# part below the split (leading ++ t) and go through
38	-	# the process with the deck (which is now has one less card).
	38	+ # the process with the deck (which now has one less card).
39	39	# This keeps going until you run out of cards to shuffle;
40	40	# at that point, all the cards will have gotten to the
41	41	# new pile, and that's your shuffled deck.

0 notes on commit 5933a23

Show line notes below

Études for Elixir: <https://github.com/oreillymedia/etudes-for-elixir>

Seamless Authoring & Production...

Step #1: Author!

The screenshot shows a web browser window with the URL `https://atlas.oreilly.com/oreilly-media-content/razzpisampler/editor/master/Connecting_an_LED.html`. The browser's address bar and navigation buttons are visible at the top. Below the browser, there is a rich text editor interface. On the left side of the editor, there are three buttons: "Back", "Code Editor", and "Files". At the top of the editor, there is a toolbar with various icons for text formatting (bold, italic, underline), linking, bulleted and numbered lists, indentation, quote, table, code, and paste. On the right side, there are three buttons: "Save", "File History", and "Builds".

The main content area of the editor is enclosed in a dashed border and contains the following sections:

- Chapter**
 - Connecting an LED**
 - A video player showing a video titled "9.1 Connecting an LED" from the "Raspberry Pi Cookbook" by Simon Monk. The video player includes a play button, a progress bar at 0:00 / 6:19, and the YouTube logo.
- Section 1**
 - Problem**
 - You want to know how to connect an LED to the Raspberry Pi.
- Section 1**
 - Solution**
 - Connect an LED (see "[Opto-Electronics](#)") to one of the GPIO pins using a 470Ω or 1kΩ series resistor (see "[Resistors and Capacitors](#)") to limit the current. To make this

Step #2: Build!

The screenshot shows a web browser window with the URL `https://atlas.oreilly.com/oreilly-media-content/razzpisampler/editor/master/Connecting_an_LED.html`. The page is a document editor for a chapter titled "Connecting an LED" from the "Raspberry Pi Cookbook" by Simon Monk. The editor includes a toolbar with various text and media tools, a left sidebar with "Back", "Code Editor", and "Files" buttons, and a right sidebar with a "Builds" panel. The main content area is divided into sections: "Chapter", "Section 1 Problem", and "Section 1 Solution". A video player is embedded in the "Section 1 Problem" section, showing a falcon and the text "9.1 Connecting an LED". The "Builds" panel on the right shows a list of builds with checkboxes for PDF, EPUB, MOBI, and HTML, and a "Build" button. The builds are listed with their status (checked or unchecked) and a "Download" link. The builds are grouped by user and time, with the most recent builds being "a few seconds ago" and the oldest being "a day ago".

Connecting an LED

Raspberry Pi Cookbook
by Simon Monk

9.1 Connecting an LED

Section 1
Problem

You want to know how to connect an LED to the Raspberry Pi.

Section 1
Solution

Connect an LED (see "Opto-Electronics") to one of the GPIO pins using a 470Ω or 1kΩ

Builds

- PDF EPUB
- MOBI HTML

Build

a few seconds ago

- EPUB [Download](#)
- MOBI [Download](#)
- HTML [Download](#)
- PDF [Download](#)

Build Log

a few seconds ago

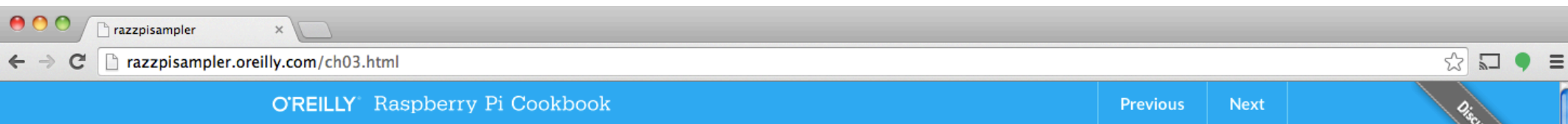
- EPUB [Download](#)
- MOBI [Download](#)
- PDF [Download](#)
- HTML [Download](#)

Build Log

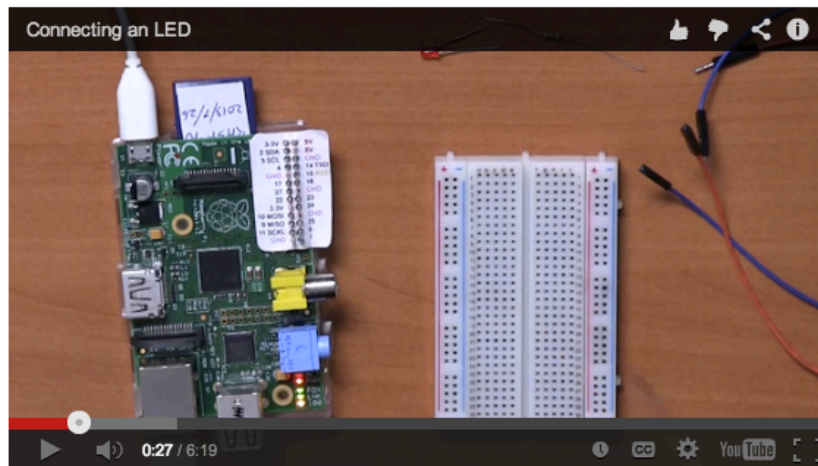
a day ago

- HTML [Download](#)

Step #3: Review Results!



Connecting an LED



Sampler Contents:

1. Raspberry Pi Radio Transmitter
2. Finding Your Way Around the GPIO Connector
3. Connecting an LED
4. Using Lots of LEDs (Charlieplexing)
5. Controlling Servo Motors
6. Using a RaspiRobot Board to Drive a Bipolar Stepper Motor
7. Connecting a Push Switch
8. Using Resistive Sensors
9. Using a Four-Digit LED Display
10. Programming an Arduino from Raspberry Pi

Problem

You want to know how to connect an LED to the Raspberry Pi.

Solution

Connect an LED (see "Opto-Electronics") to one of the GPIO pins using a 470 Ω or 1k Ω series resistor (see "Resistors and Capacitors") to limit the current. To make this recipe, you will need:

- Breadboard and jumper wires (see "Prototyping Equipment")
- 1k Ω resistor (see "Resistors and Capacitors")
- LED (see "Opto-Electronics")

Figure 3-1 shows how you can wire this using solderless breadboard and male-to-female jumper



Buy the book on oreilly.com.

**All Roads Lead to
HTML5...**

**...But Is It Semantic
Enough for Book
Publishing?**



This repository

Search or type a command

Explore Gist Blog Help

sandersk



oreillymedia / HTMLBook

Unwatch 15

Star 9

Fork 5

Let's write books in HTML! — Edit

429 commits

12 branches

0 releases

4 contributors



branch: master

HTMLBook

Merge pull request #60 from oreillymedia/HTMLBook

sandersk authored 5 hours ago

htmlbook-xsl Actually generation order does not matter here, as logic is now written. 5 hours ago

samples Updated files for HTML5. 23 days ago

schema Tweaked schema input for XML. Resolved issue to make only happen. 3 months ago

stylesheets a base stylesheet 3 months ago

LICENSE adding MIT license 4 months ago

README.asciidoc Update README.asciidoc 3 months ago

poster.png Added poster.png 23 days ago

specification.asciidoc Updated specification to reflect the use of @data-type for formal sem... a month ago

README.asciidoc

HTMLBook

Let's write books in HTML! HTMLBook is an open, HTML5-based standard for the authoring and production of both print and digital books. HTMLBook is built on the following premises:

Introducing HTMLBook

(github.com/oreillymedia/htmlbook)

Code

Issues 3

Pull Requests 0

Wiki

Pulse

Graphs

Network

Settings

SSH clone URL

git@github.com:or

You can clone with HTTPS, SSH, Subversion, and other methods.

Clone in Desktop

Download ZIP

HTMLBook =

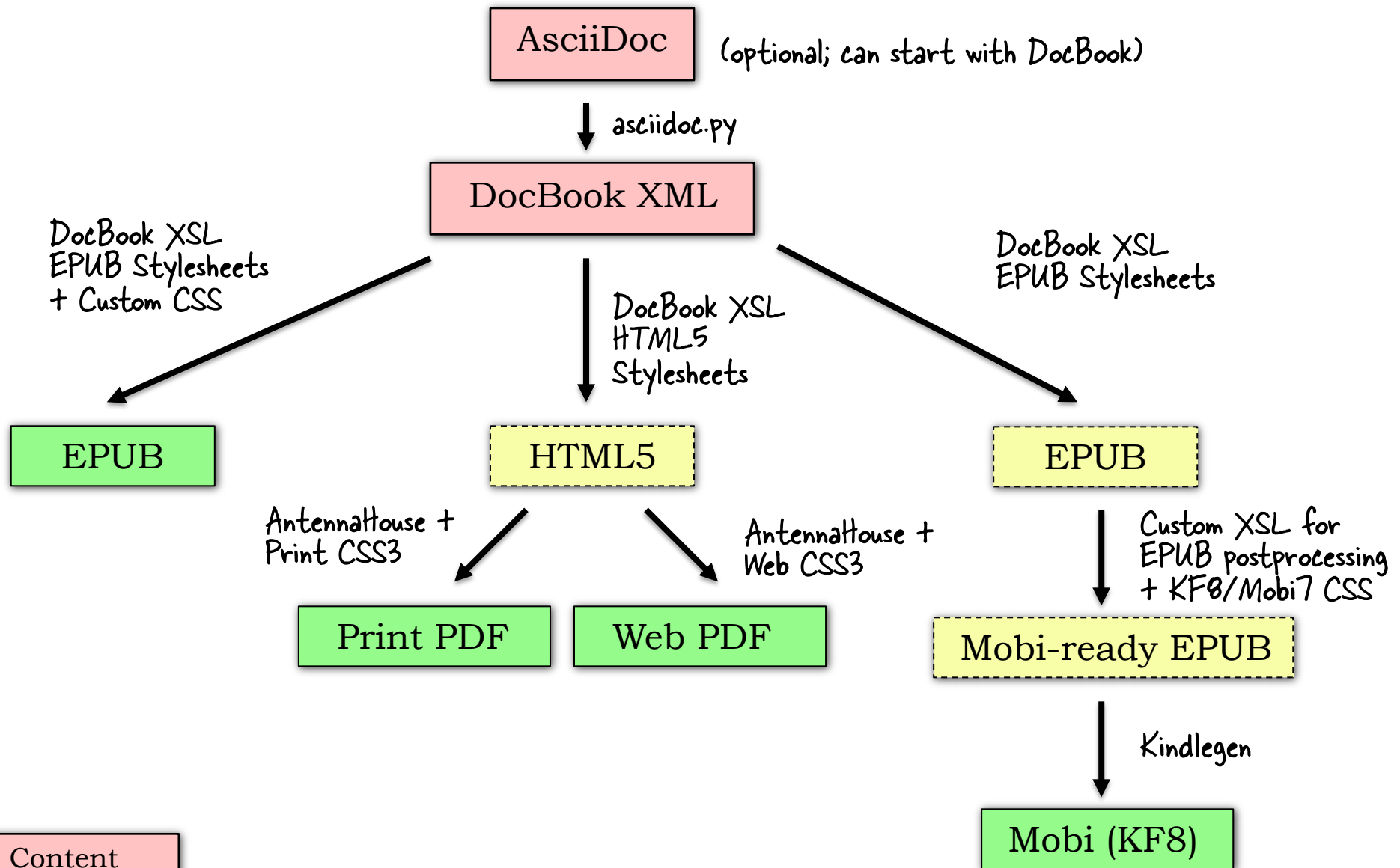
- **Open Spec for Book Authoring**
- **Subsets XHTML5 Vocabulary and Content Model**
- **Adds Book-Specific Semantics (e.g., `<section data-type="chapter">`)**
- **Open Source Tooling for Producing Ebook Outputs**

HTMLBook Sample

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>HTMLBook Sample</title>
  </head>
  <body data-type="book" id="htmlbook">
    <section data-type="chapter" id="chapter01">
      <h1>Chapter 1. HTMLBook Markup</h1>
      <p>This chapter describes and demonstrates the
types of markup<a data-type="indexterm" data-
primary="markup" data-secondary="types of"></a> that might
appear in a chapter. See <em>mappings.asciidoc</em> for
more information. HTMLBook borrows much of its
semantics from the EPUB 3 specification, as applied via the
<a href="http://idpf.org/accessibility/guidelines/content/
semantics/epub-type.php"><code>epub:type</code></a>
attribute.</p>
    </section>
  </body>
</html>
```

(github.com/oreillymedia/HTMLBook/blob/master/samples/htmlbook.html)

O'Reilly's Single-Source Workflow (2006-2013):

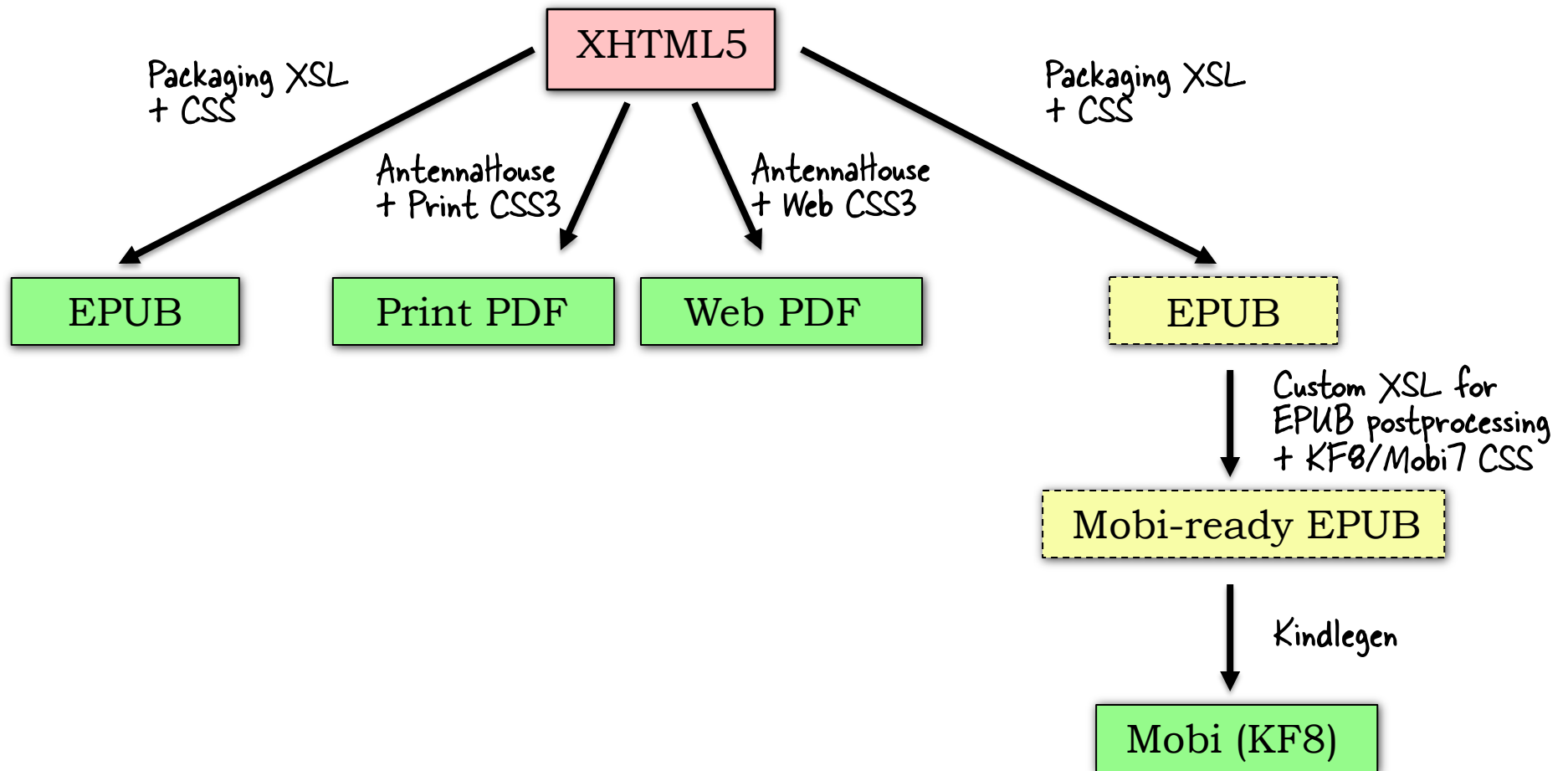


Source Content

Intermediate Output

Final Output For Sale

O'Reilly's Single-Source Workflow (2014!):



Source Content

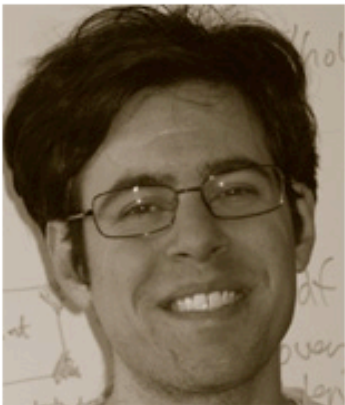
Intermediate Output

Final Output For Sale

Contact Me!

Email: sanders@oreilly.com

Twitter: [@sandersk](https://twitter.com/sandersk)



Sanders Kleinfeld

sanders@oreilly.com

<http://twitter.com/sandersk>

Cambridge, Massachusetts

Ebook Alchemist

Areas of Expertise:

- DocBook XML
- XSLT
- XPath
- XQuery
- RDF
- SPARQL
- EPUB
- Mobi
- HTML5
- speaking
- writing

Biography

Books

Blog

Multimedia

Sanders Kleinfeld has been employed at O'Reilly Media since 2004 and has held a variety of positions, including roles on O'Reilly's Production, Editorial, and Tools teams. Currently, he works as a Publishing Technologies Specialist, maintaining O'Reilly's XML-based toolchain for generating EPUB and Mobi formats of both frontlist and backlist titles. He also helps coordinate O'Reilly's digital distribution efforts to electronic sales channels, and is currently assisting in R&D efforts surrounding HTML5

Hire Sanders Kleinfeld

For Inquiries Contact
talent@oreilly.com

Press Inquiries
press@oreilly.com

Find Other Authors

Consultants

Also find Sanders on:



Recent Twitter posts:

- [@nbousquet](#) Most EPUBs out now use XHTML 1.1, so EPUB is already pretty Web-compliant. Maybe HTML5 ebooks will be more attractive to pirates? [2 days ago](#)

Author Events

Webcast: HTML5 for Publishers