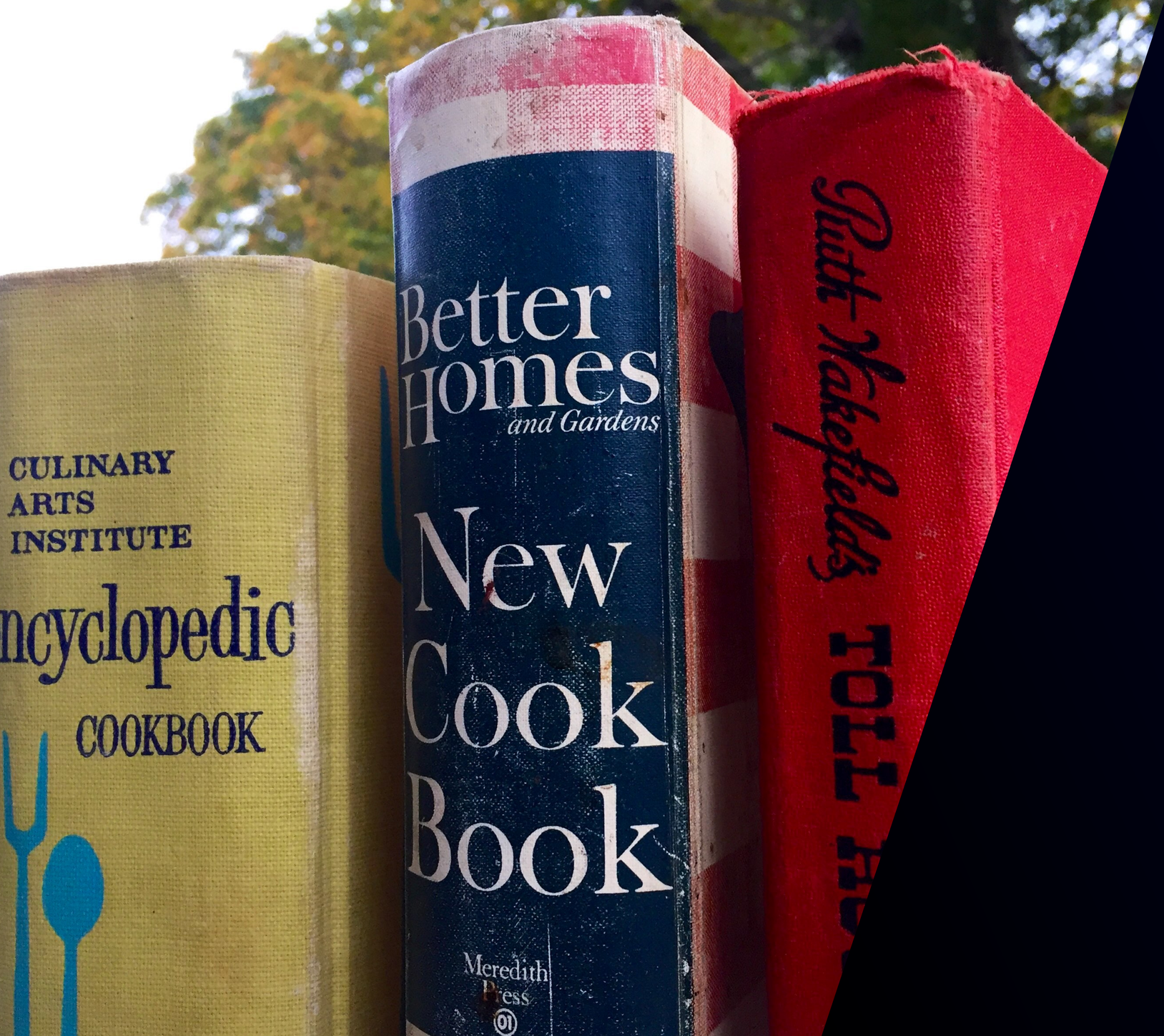


Accessibility-flavored

React components

make your design system delicious

Kathleen McMahon
@resource11



@resource11

https://noti.st/resource11

Slide deck posted after the talk



@resource11

@resource11

Twitter | Instagram | GitHub

Good grief, there's an agenda...

Why accessibility first?

Design systems are a cookbook

Design systems and React

Icons

Buttons

Inputs

Documentation

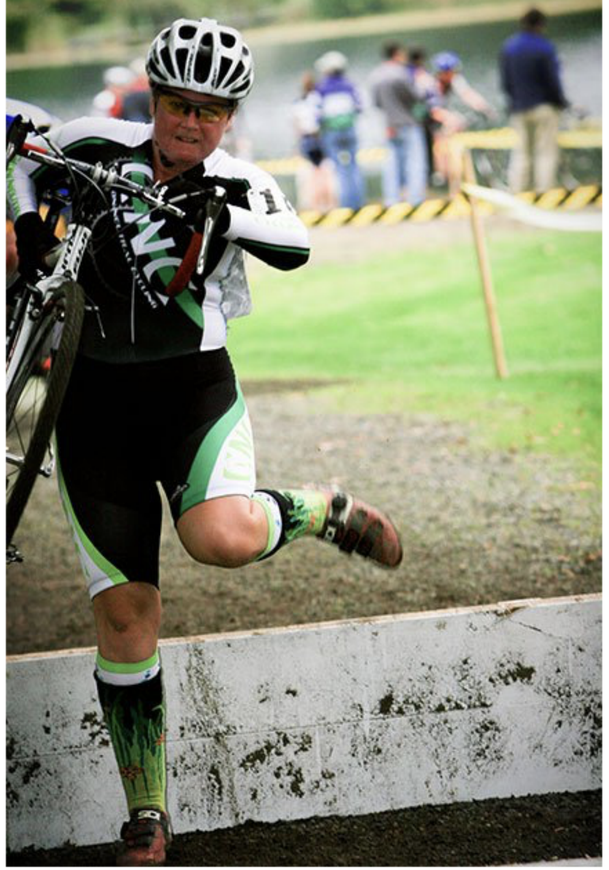


@resource11

Who are you, again?


CarGurus®

Cyclocross!








SPAM
Hormel


200
RALEIGH
STALKER


70




SPAM
315
Hormel




SPAM
Hormel

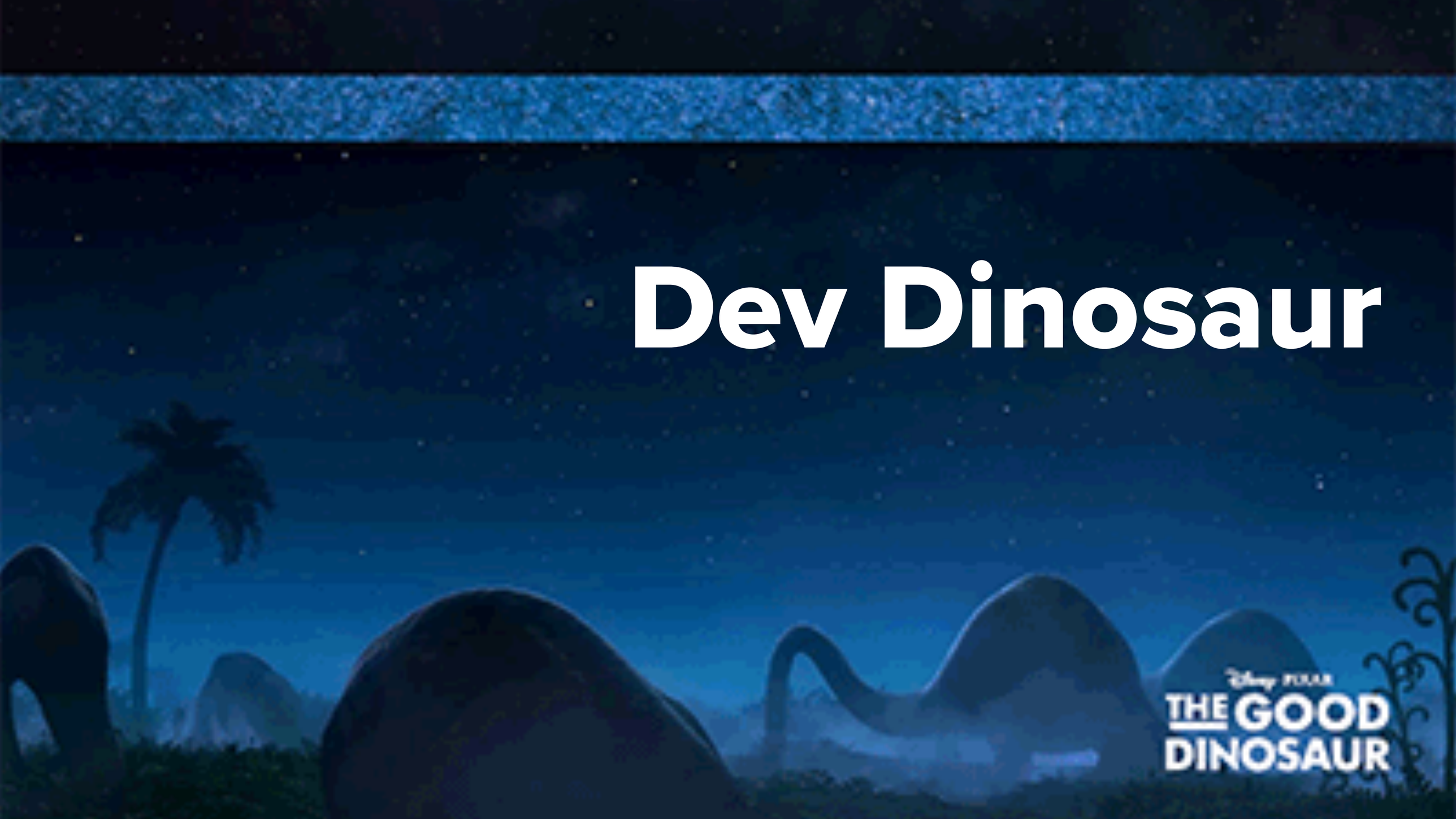



200
RALEIGH
STALKER


70


SPAM
315
Hormel
@resource11

Dev Dinosaur

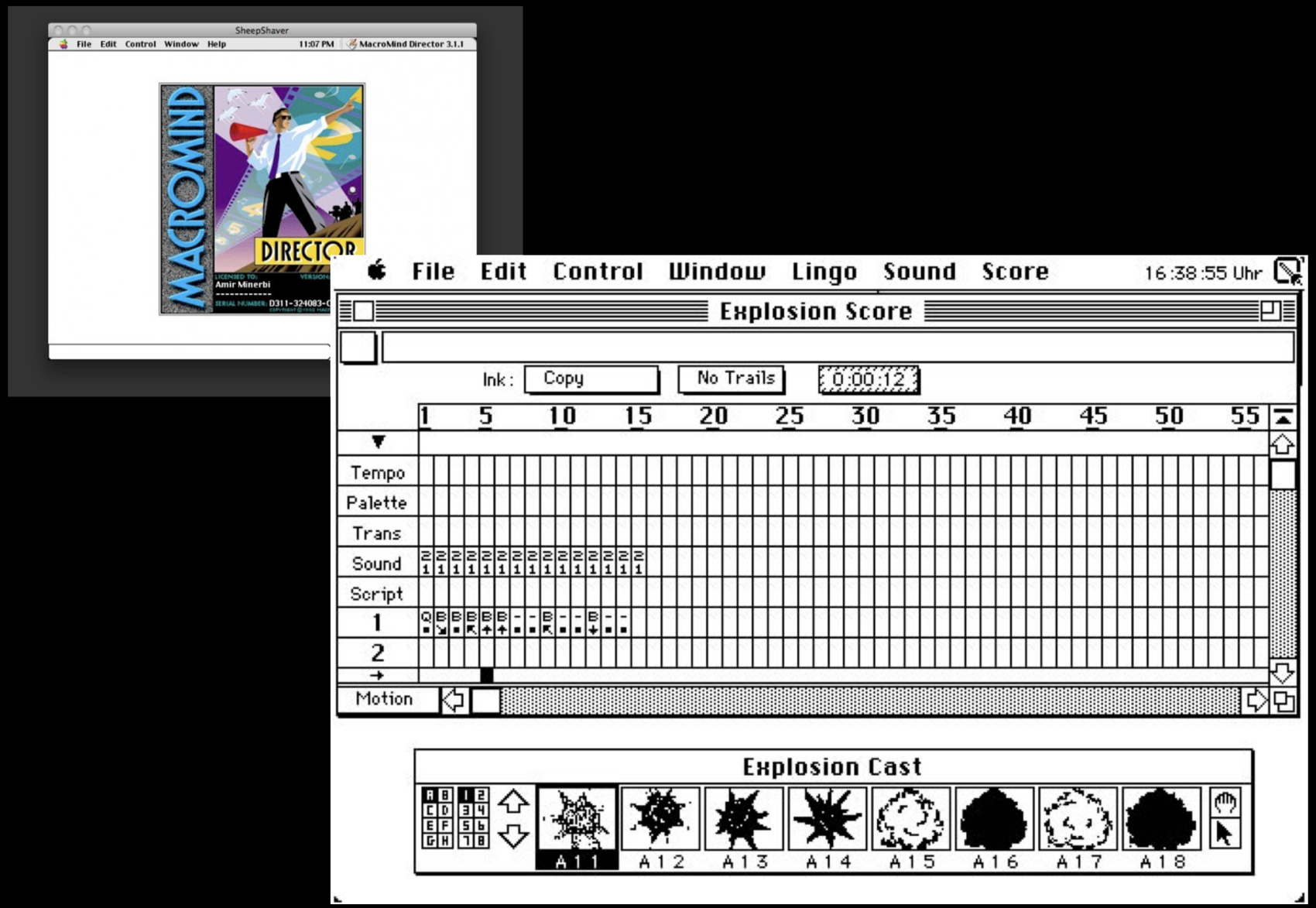




@resource11


File Edit Mode Image Filter Select Window Help
Wed 11:15 AM
S7300917.JPG (RGB, 1:4)
cintosh HD
Brushes
Normal
35
45
65
100
Trash

@resource11


SheepShaver
File Edit Control Window Help
11:07 PM
MacroMind Director 3.1.1
MACROMIND
DIRECTOR
Amir Minerbi
File Edit Control Window Lingo Sound Score
16:38:55 Uhr
Explosion Score
Ink: Copy
No Trails
0:00:12
Tempo
Palette
Trans
Sound
Script
1
2
Motion
Explosion Cast
A11
A12
A13
A14
A15
A16
A17
A18

@resource11


Welcome to
Zip™ drive
Inside is all the stuff you
to get start
maxell
MF2-DD
zip
iomega
SyQuest
88MB
BERNOULLI
Multidisk
@resource11


@resource11
The Photoshop 3 Wow! Book Macintosh Edition
THE Photoshop 7 Wow! BOOK Jack Davis
Photoshop cs Down & Dirty Tricks
Adobe ImageReady User Guide Windows/Macintosh
DeBABELIZER
THE OFFICIAL BBEDIT BOOK
Dynamic HTML
The Definitive Reference
Essential ActionScript 3.0
O'REILLY
CSS
THE MISSING MANUAL
O'REILLY
Eric Meyer on CSS
How to Wow with Flash
ADOBE FLASH CS4 PROFESSIONAL
ADOBE FLASH CS3 PROFESSIONAL CLASSROOM IN A BOOK
Adobe Flash CS3 Professional Video
ADOBE AFTER EFFECTS CS3 PROFESSIONAL CLASSROOM IN A BOOK
FLASH 5 MAGIC
The Flash 2 Web Animation
MACROMEDIA DIRECTOR MX
MACROMEDIA DIRECTOR FOR WINDOWS & MACINTOSH
NI-D
NI-D
MANUAL



NETSCAPE

@resource11

HTML + CSS + JS

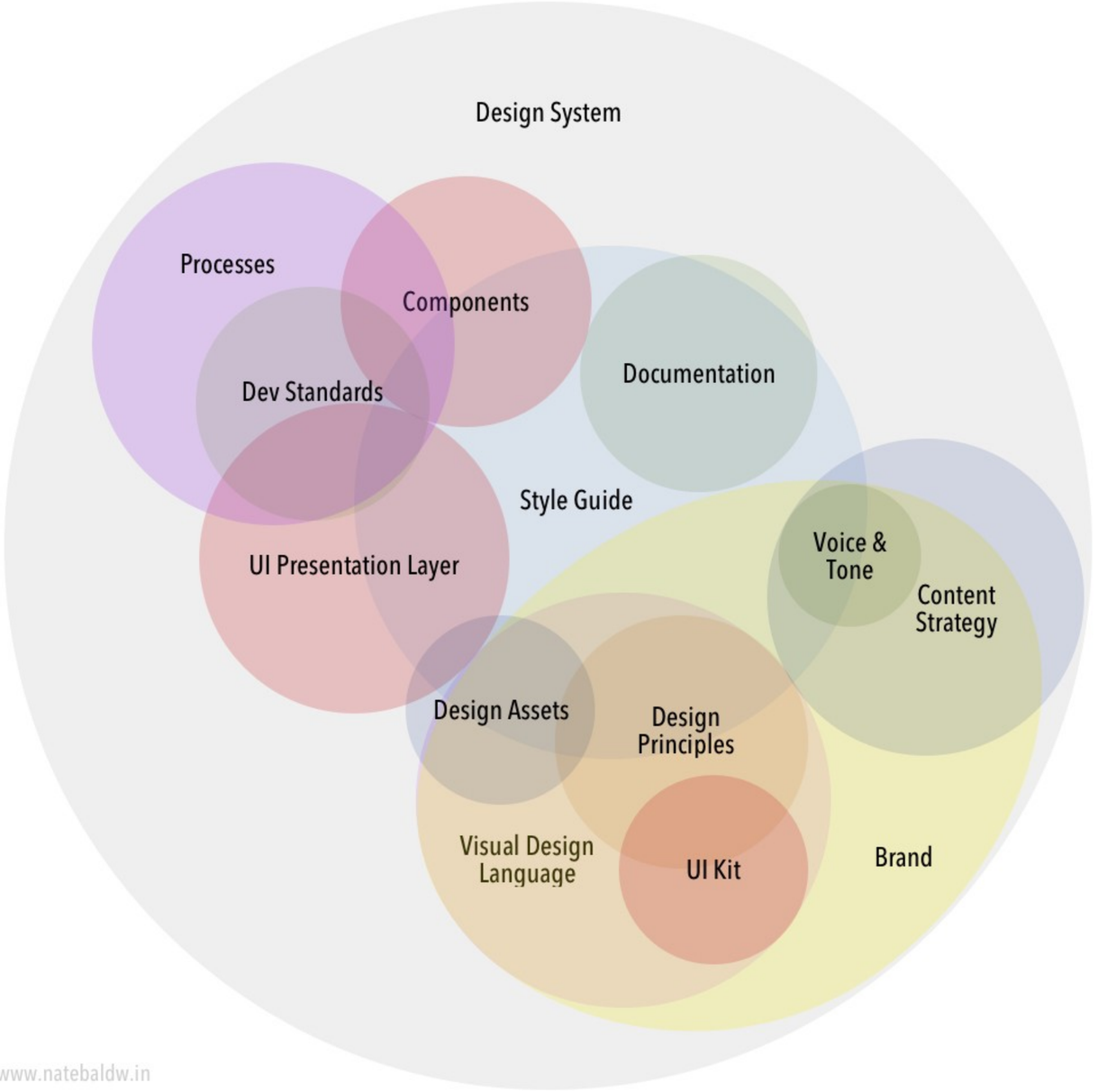
@resource11

JavaScript Framework of the week


OPEN WIDE!!


Dinosaurs are
ALWAYS the
hotness
@resource11


@resource11
O'REILLY®


Design System
Processes
Components
Dev Standards
Documentation
Style Guide
UI Presentation Layer
Voice &
Tone
Content
Strategy
Design Assets
Design
Principles
Visual Design
Language
UI Kit
Brand
www.natebaldw.in

Great design systems combine team **AND** consumer experience



First Priority

Business logic out, accessibility in

Why accessibility first?

Our users are diverse

Vision Hearing Motor Cognitive



Sensory Language Low bandwidth

Inclusive Design Persona Spectrum (CC BY-NC-SA 2.0)

@resource11

February 2019

WebAIM Million

An accessibility analysis of the top 1,000,000 home pages

@resource11

97.8%

of home pages had detectable WCAG 2 failures.

59%

of 3.4 million form inputs
were unlabeled

60.1%

of the 1 million pages had ARIA present

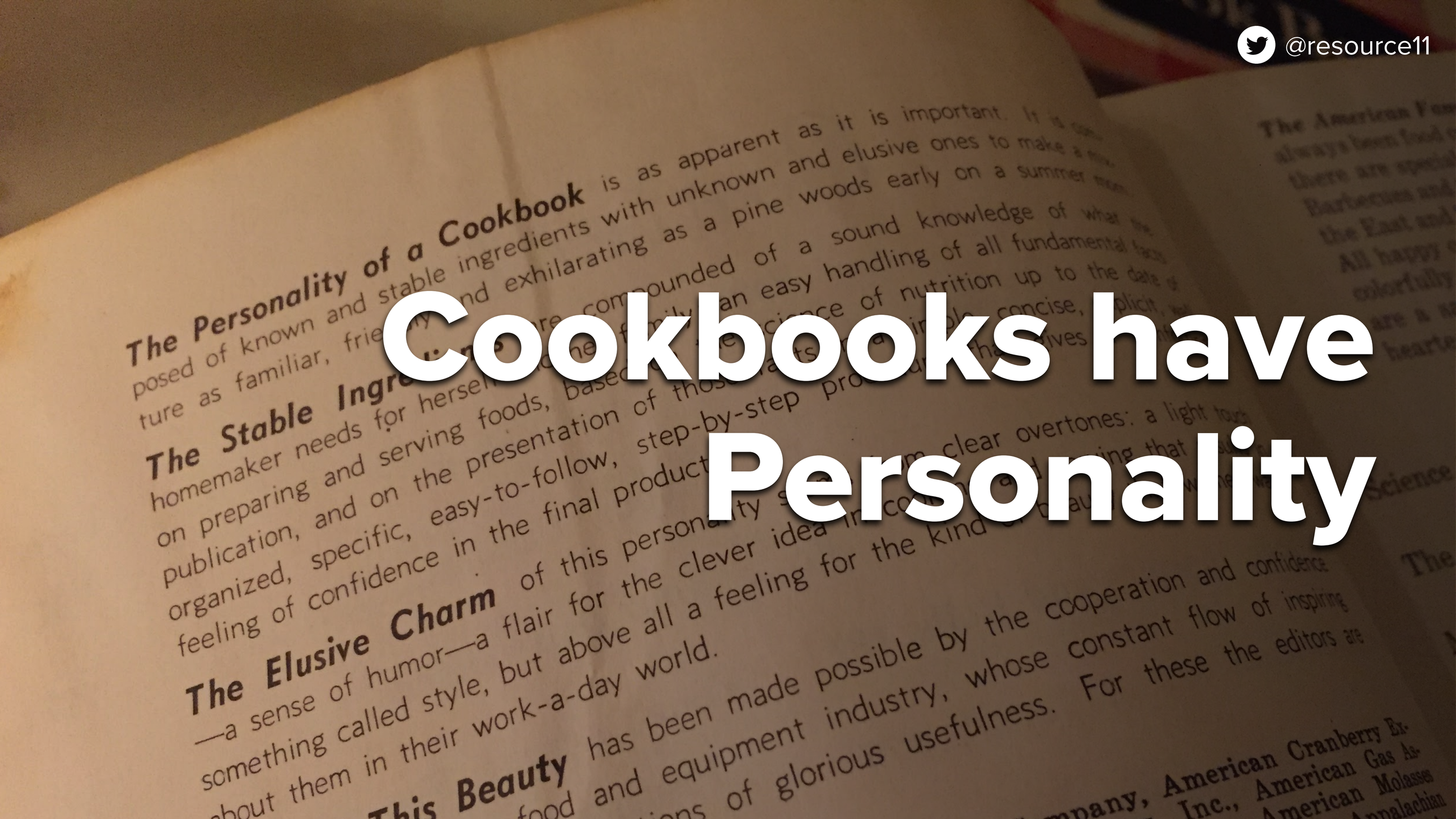
Web Content Accessibility Guidelines (WCAG)

Imagine... accessibility baked into components.



Your Design system is a cookbook

@resource11


@resource11
Cookbooks have Personality

@resource11

BAKING

Yeast breads	Temp.	Time in minutes
Loaves (8½ x 4½ x 2½)	375°–400°	30–45
Rolls		
Plain	400°	10–20
Sweet	375°–400°	10–25
Quick breads		
Biscuits	450°	10–15
Muffins	375° 425°	15–25
Corn bread	425°	12–15
Popovers	475°	15
then	350°	25–30
Nut bread	350°	40–60
Coffeecake	350° 375°	25–40
Pastry		
Pastry shell	450°	10–12
Fruit pies	400° 425°	35–50
or	450°	10
then	350°	30
Custard pie	400°	25–30
Pumpkin pie	400°	50
Meringue	350°	12–15
Cakes		
Cakes with shortening		
Cupcakes	375°	20
Layers	350° 375°	20–35
Squares, oblongs, upside-down cakes	350°	25–40
Loaf cakes	300° 350°	45–85
Angel (tube)	375°	30–40
Chiffon (tube)	325°	55
then	350°	10–15
Sponge (tube)	325°	55–60
Fruitcake	275°	2½–3½ hrs.
Cookies		
Drop	350°–400°	8–15
Rolled	350°–375°	5–12
Refrigerator	350°–400°	8–15
Filled	350°–375°	8–12
Bars	325°–375°	10–35

DEEP-FAT FRYING

	Temp. of fat	Time in minutes
Croquettes	375°	till golden 3–4
Fritters	360°	
French fr[illegible]s	375°	till golden 3–5
then	375°	3–4
[illegible]uliflowerets	375°	2–5
Eggplant	375°	
Onion rings	375°	till golden
Sea foods	375°	till golden 2



Top: Perfect Divinity, Toffee Bars, Caesar Salad, Spice Layer Cake; Bottom: Ov

Why nearly 9,000,000 women cherish this cook book

You serve nutritious me This cook book lists cal ratings of many foods; sug gests plans for delicious meals.

You save time, effort, fuss Here are techniques for fast efficient cooking; see the com plete chapter on jiffy meals.

You reduce food bills. Rec ipes call for even cans and pack ages of ingredients—no waste

You're a good hostess! Se the section on entertaining— luncheons, brunches, buffets; table settings, centerpieces.

Printed in the United States of America.



Chicken, Toasty Garlic Bread, Carrot Bake, Winter Fruit Compote, Barbecu

Every recipe in the new Cook Book is Test-Kitchen endorsed!

The checked seal means that each recipe has been tested over and over till it rates high in practicality, family appeal, and downright deliciousness! This red and white symbol assures you a perfect product every time! A recipe must meet our high standards and merit the faith of our readers. These recipes are so practical—fast to fix, use on-hand ingredients, no special equipment. Even company dishes take minimum bother.

Better Homes TEST KITCHEN

The new Cook Boo



Better Homes & Gardens New Cook Book


Cook Bo
@resource11


@resource11

@resource11

chapter 13

pages 235 through 278

Meats, poultry, and fish

Meat

Methods of cooking 236-237

Beef

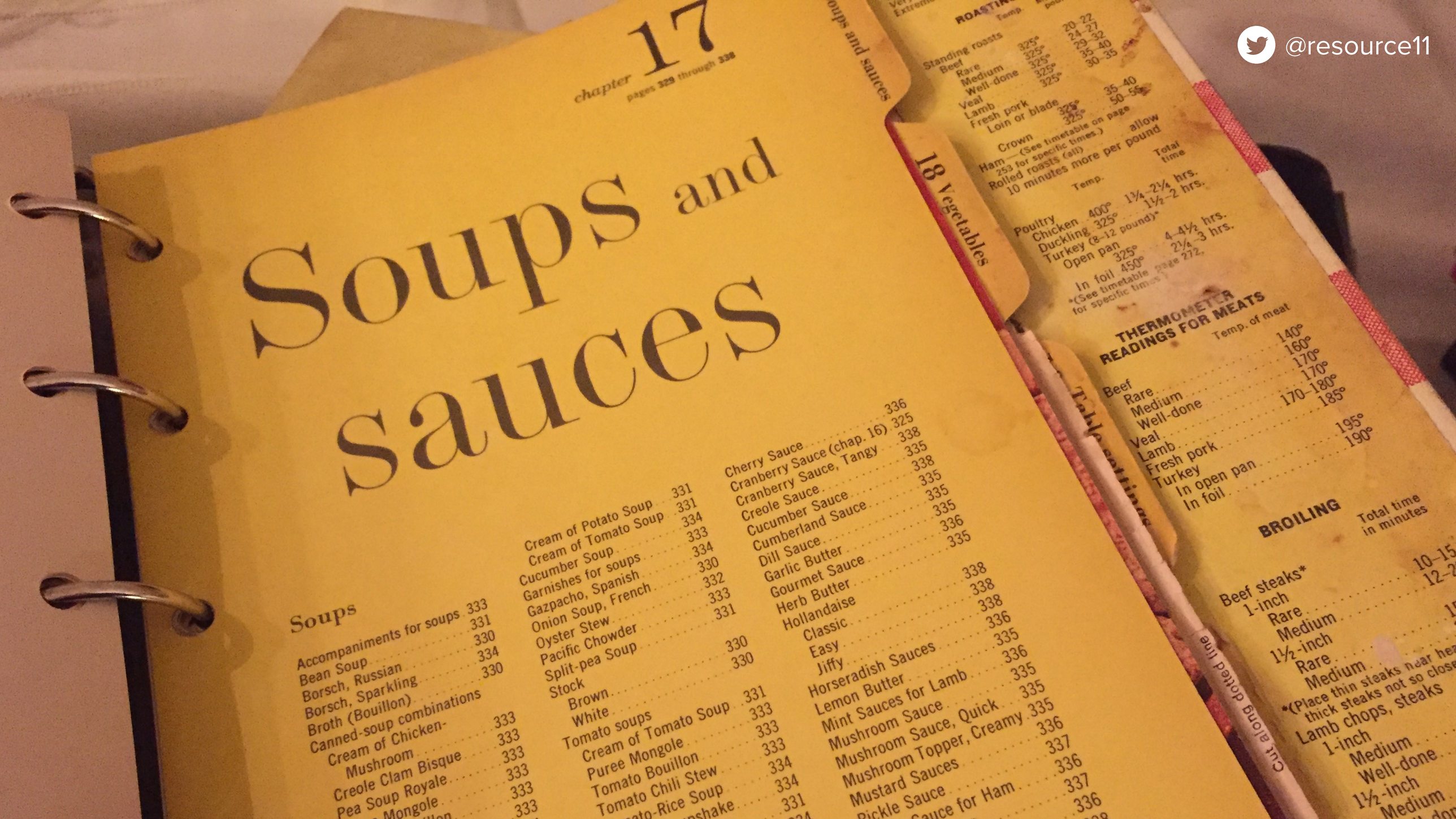
Beefsteak and Kidney Pie 265
Corned-beef Dinner 245
Hamburger Patties, Saucy 264
Meat Loaves 238-239
Pot Roast 241
Steak 242
Broiled 242
Flank 242
Minute 242
Pan-broiled 243
Round 243
Swiss 244
Stew, Old-fashioned 245
Stroganoff 250

Pork

Chops 249
Crown Roast 246-247
Ham 251
Rotisserie-roasted Ham 251
Sliced Buffet Ham 252
Picnic 251, 252
Roasts 248
Crown 248
Loin 248
Stuffed Shoulder 248
Smoked Shoulder Butt 251
Spareribs 250, 251
Steaks with Apple 249
Sweet-sour Pork 250
Tenderloin 250

Veal

Baked Veal-and-Ham Birds 256
Breast, Roast Stuffed 256
Chops 256
Crown Roast 256
Cutlets 257



chapter 17
pages 329 through 338

Soups and sauces

Soups

Accompaniments for soups 333
Bean Soup 331
Borsch, Russian 330
Borsch, Sparkling 334
Broth (Bouillon) 330
Canned-soup combinations
Cream of Chicken- 333
Mushroom 333
Creole Clam Bisque 333
Pea Soup Royale 333
Pea Soup Mongole 333
Cream of Potato Soup 331
Cream of Tomato Soup 331
Cucumber Soup 334
Garnishes for soups 333
Gazpacho, Spanish 334
Onion Soup, French 330
Oyster Stew 332
Pacific Chowder 333
Split-pea Soup 331
Stock 330
Brown 330
White
Tomato soups
Cream of Tomato Soup 331
Puree Mongole 333
Tomato Bouillon 333
Tomato Chili Stew 333
Tomato-Rice Soup 334
Tomato Shake 331

Sauces

Cherry Sauce 336
Cranberry Sauce (chap. 16) 325
Cranberry Sauce, Tangy 338
Creole Sauce 335
Cucumber Sauce 338
Cumberland Sauce 335
Dill Sauce 335
Garlic Butter 336
Gourmet Sauce 335
Herb Butter 338
Hollandaise 338
Classic 338
Easy 336
Jiffy 335
Horseradish Sauces 336
Lemon Butter 335
Mint Sauces for Lamb 335
Mushroom Sauce 335
Mushroom Sauce, Quick 335
Mushroom Topper, Creamy 336
Mustard Sauces 337
Pickle Sauce 336
Sauce for Ham 337

soups and sauces

18 Vegetables

Cut along dotted line

ROASTING

	Temp.	Time per pound
Standing roasts		
Beef		
Rare	325°	20–22
Medium	325°	24–27
Well-done	325°	29–32
Veal	325°	35–40
Lamb	325°	30–35
Fresh pork		35–40
Loin or blade	325°	50–55
Crown	325°	
Ham—(See timetable on page 253 for specific times.)	325°	
Rolled roasts (all)		allow 10 minutes more per pound

	Temp.	Total time
Poultry		
Chicken	400°	1¼–2¼ hrs.
Duckling	325°	1½–2 hrs.
Turkey (8–12 pounds)*		
Open pan	325°	4–4½ hrs.
	325°	2¼–3 hrs.
In foil	450°	

*(See timetable on page 272 for specific times.)

THERMOMETER READINGS FOR MEATS

	Temp. of meat
Beef	
Rare	140°
Medium	160°
Well-done	170°
Veal	170°
Lamb	170–180°
Fresh pork	185°
Turkey	
In open pan	195°
In foil	190°

BROILING

	Total time in minutes
Beef steaks*	
1-inch	10–15
Rare	12–2
Medium	1
1½-inch	
Rare	
Medium	

*(Place thin steaks near hea thick steaks not so close

Lamb chops, steaks
1-inch
Medium
Well-done
1½-inch
Medium

@resource11

@resource11

chapter 18
pages 339 through 358

Vegetables

Cooking chart 341-343
Cooking methods 340
Creamed Vegetables 340
Scalloped Vegetables 340
Vegetables au Gratin 340
Vegetables Cooked in Oil 340

Artichokes 341, 344
Artichokes with Butter 344
Crab-stuffed Artichokes 344

Asparagus 341, 345
Asparagus with Croutons 344
Asparagus dress-ups 344

Beans 341
..... 346
..... 346

Cabbage 341
Cabbage with Cheese 346
Cabbage with Golden Sauce 347
Savory Cabbage 347
Pennsylvania Red Cabbage 347

Carrots 342
Carrot Bake with Basil 348
Carrots Chinese 348
Glazed Carrots 348
Herbed Carrots 348
Minted Carrots 348

Cauliflower 342
Cheese-frosted Cauliflower with Shrimp Sauce 348
..... 348

Celeriac 342

Celery 342
Celery Luncheon Bake 349
Celery Victor 349
N-M Celery Oriental 349

Corn 342
..... the Cob 349
..... 350
..... 349

Greens ..... see cooking chart

Kohlrabi 342

Leeks 342

Mushrooms 342
Creamed Mushrooms 350
Fresh Mushroom Saute 350

Okra 342

Onions 343
Creamed Onions 350
Dutch Glazed Onions 351
French-fried Onion Rings 351
Onions Gourmet 350

Parsnips 343, 351

Peas 343, 351
Company Peas 351
Garden Peas in Cream 351
Peas Cooked in Oil 351

Potatoes 343
Baked Potatoes 354
Boiled Potatoes 353
..... Potatoes in Foil 355
..... 353
..... New 353

Cut along dotted line

ROASTING	Temp.	Min. per pound
Standing roasts		
Beef		
Rare	325°	20–22
Medium	325°	24–27
Well-done	325°	29–32
Veal	325°	35–40
Lamb		30–35
Fresh pork		
Loin or blade	325°	35–40
Crown	325°	50–55

Ham — (See timetable on page 253 for specific times.)
Rolled roasts (all) allow 10 minutes more per pound

	Temp.	Total time
Poultry		
Chicken	400°	1¼–2¼ hrs.
Duckling	325°	1½–2 hrs.
Turkey (8–12 pound)*		
Open pan	325°	4–4½ hrs.
In foil	450°	2¼–3 hrs.

*(See timetable on page 272, for specific times.)

THERMOMETER READINGS FOR MEATS	Temp. of meat
Beef	
Rare	140°
Medium	160°
Well-done	170°
Veal	170°
Lamb	170–180°
Fresh pork	185°
Turkey	
In open pan	195°
In foil	190°

BROILING

Total tim in minut

Beef steaks*
1-inch
Rare
Medium
1½-inch
Rare
Medium
*(Place thin steaks n thick steaks not
Lamb chops, ste
1-inch
Medium
Well-d



chapter 10
pages 187 through 208

Desserts

10 Desserts

Alaskas 199
Baked Alaska 199
Melba Alaska 199
Ribbon Alaska Pie 199

Cake desserts
Almond-Brittle Cake 203
Chocolate-fleck Cake 203
Chocolate Fluff Layers 203
Coffee Ribbon Loaf 202
Daffodil Layers 203
Fruited Angel Surprise 202
Lemonade Angel Dessert 203
Orange Angel Ring 203
Pineapple Dessert 202
Stirred Custard 191
Strawberry Floating Island 191

Eclairs 200

Fruit desserts
Ambrosia 205
Apple Betty Pie 204
Apple Cobbler 207
Apple Dumplings 204
Apple-Raisin Rolls 206
Apples, Baked 204
Applesauce 204
Blueberry Crumb Pudding 191
Blueberry Rolls 206
Cherry Puff 205
Cherry Cobbler 207
Winter Fruit 204
Fruit Cobbler 205
Glorified Rice 189

Ice cream
Chocolate-Almond Velvet 197
Chocolate-chip Cups 198
Chocolate Ice Cream 196
Coffee Ice Cream 196
French Chocolate Ice Cream 196
Holiday Ice Cream 198
How to freeze ice cream 196
Orange Ice Cream 197
Peach-tree Ice Cream 197
Peppermint Ice Cream 197
Pineapple Ice Cream 196
Pistachio Ice Cream 197
Strawberry Ice Cream 198
Tutti-frutti Tortoni 196
Vanilla Custard Ice Cream 196
Vanilla Ice Cream 196

Meringues 200

Pastries 197

@resource11

chapter 19

Pages 359 through 378

Table settings and entertaining

Table settings

Centerpieces 363, 368, 369

Tables 361-362

Special meals

Buffets and casual dinners

Before-the-Game Buffet 375
Card-table Buffet 375
Chopstick Dinner 375
Fireside Supper for 4 375
Happy Birthday Dinner 375
Potluck with a Plan 375
...hetti Supper 375

...holidays 378

"Queen-of-the-Day" Party 373
Tray Brunch 373

Outdoor feasts

Bring Your Own Steaks 377
Eat-Anywhere Picnic 377
Garden Fiesta 377
Swimming Pool Special 377

Snacks and refreshments

"Come on over for dessert" 376
Men's Choice Snack Bar 376
Planned Refrigerator Raid 376
Platter Party 376
Special-occasion Coffee 376
...mer Refreshment Plate 376
...In Between 376

ROASTING

	Temp.	Min. per pound
Standing roasts		
Beef		
Rare	325°	20-22
Medium	325°	24-27
Well-done	325°	29-32
Veal	325°	35-40
Lamb		30-35
Fresh pork		
Loin or blade	325°	35-40
	325°	50-55
Crown		
Ham—(See timetable on page 253 for specific times.)		
Rolled roasts (all)		allow 10 minutes more per pound

	Temp.	Total time
Poultry		
Chicken	400°	1¼-2¼ hrs.
Duckling	325°	1½-2 hrs.
Turkey (8-12 pounds)*		
Open pan	325°	4-4½ hrs.
In foil	450°	2½-3 hrs.

*(See timetable on page 272 for specific times.)

THERMOMETER READINGS FOR MEATS

	Temp. of meat
Beef	
Rare	140°
Medium	160°
Well-done	170°
Veal	170°
Lamb	170-180°
Fresh pork	185°
Turkey	
In open pan	
In foil	

BROILING

Beef steaks*
1-inch
Rare
Me
1¼

@resource11

@resource11

Design Systems and React

React

A JavaScript library for building user interfaces

Get Started Take the Tutorial

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node

Accessibility

Why Accessibility?

Web accessibility (also referred to as **a11y**) is the design and creation of websites that can be used by everyone. Accessibility support is necessary to allow assistive technology to interpret web pages.

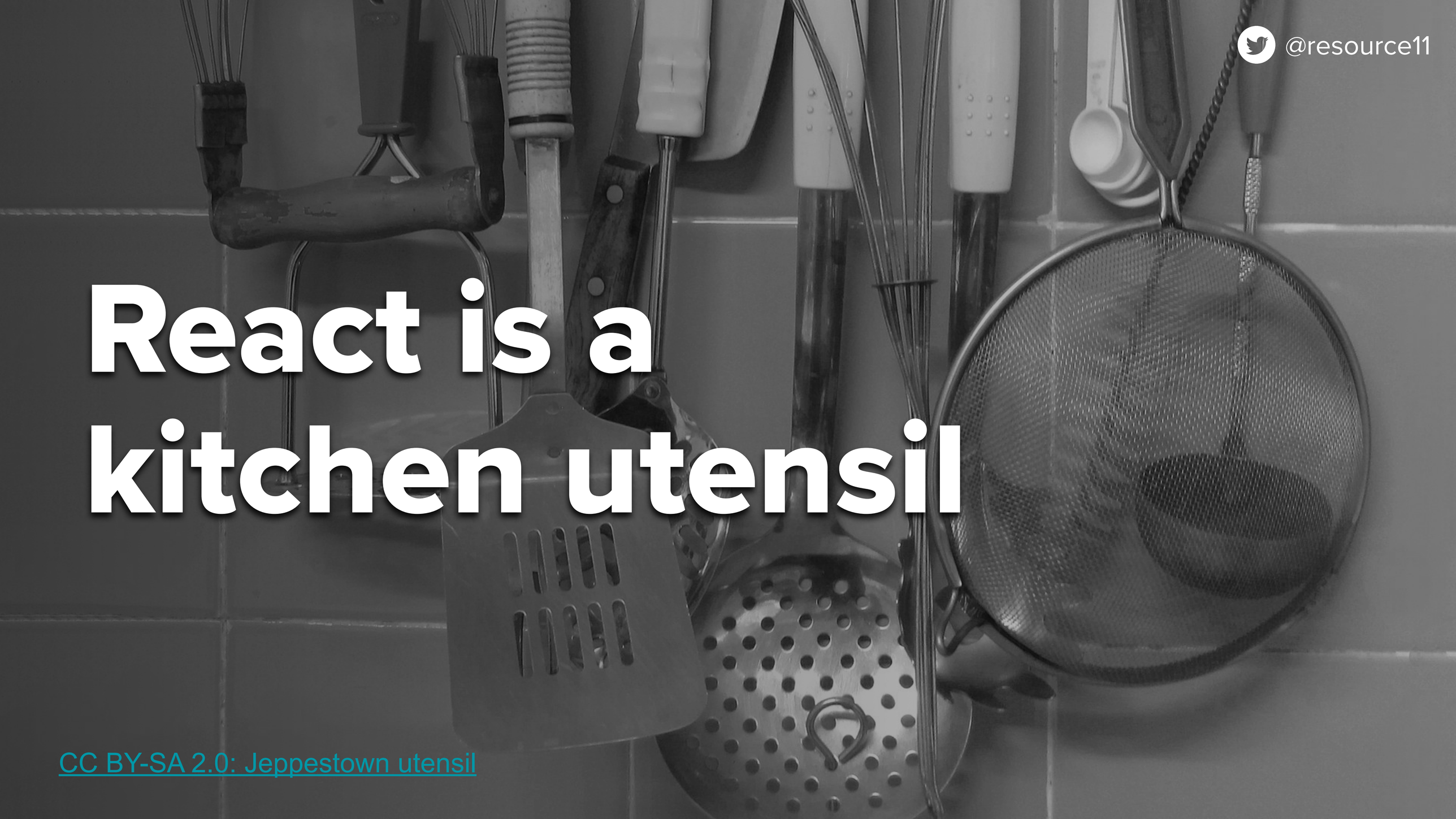
React fully supports building accessible websites, often by using standard HTML techniques.

Standards and Guidelines

WCAG

The Web Content Accessibility Guidelines provides guidelines for creating accessible web sites.

INSTALLATION
MAIN CONCEPTS
ADVANCED GUIDES
Accessibility
Code-Splitting
Context
Error Boundaries
Forwarding Refs
Fragments
Higher-Order Components
Integrating with Other Libraries
JSX In Depth
Optimizing Performance
Portals
React Without ES6
React Without JSX
Reconciliation
Refs and the DOM
Render Props



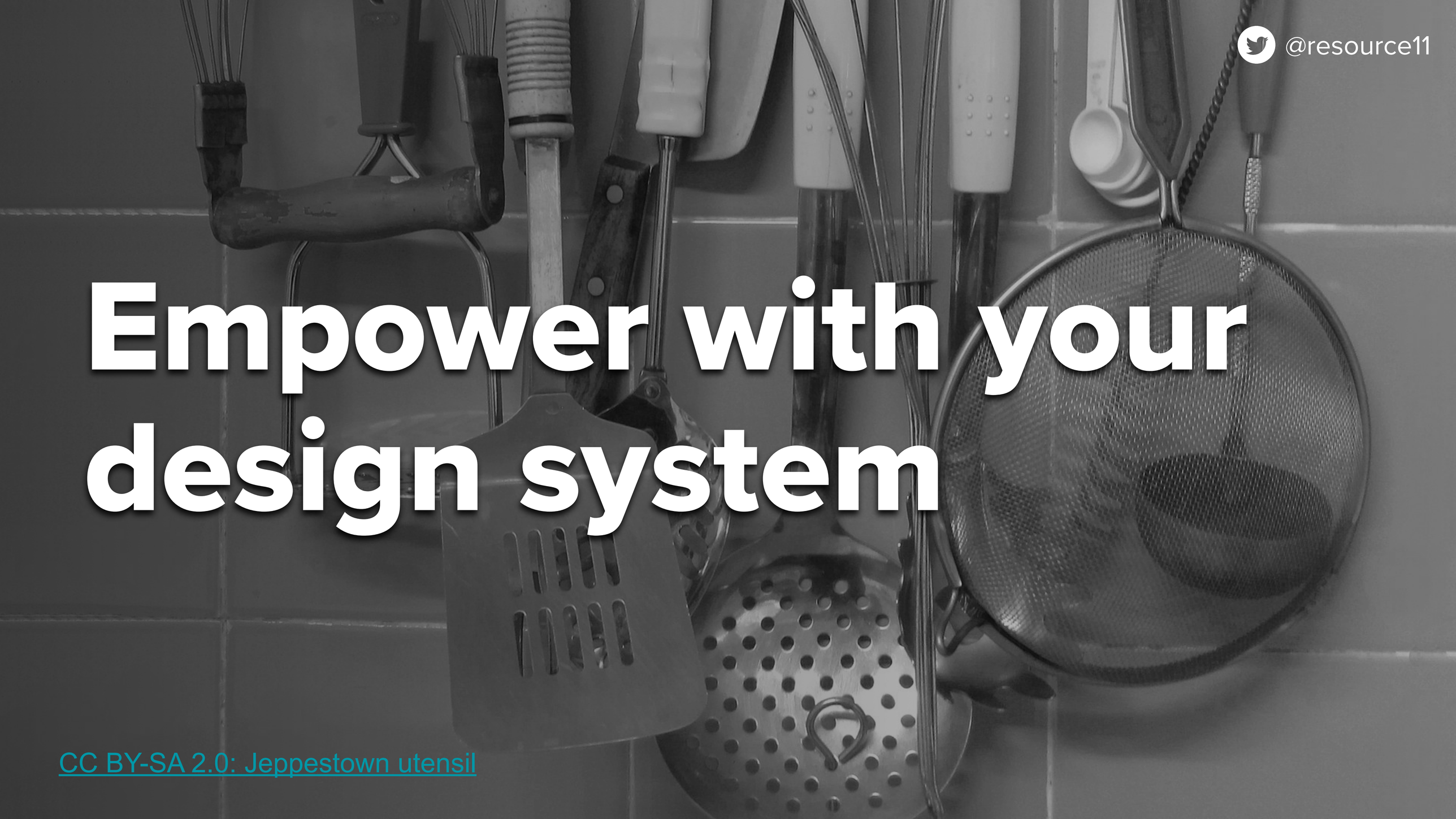
@resource11

React is a kitchen utensil

CC BY-SA 2.0: Jeppestown utensil



CC BY-SA 2.0: Jeppestown utensil



@resource11

Empower with your design system

CC BY-SA 2.0: Jeppestown utensil



Matt May
@mattmay

Getting a hard-fought accessibility win is nice. But even nicer is seeing a co-worker doing good accessibility work without you having to ask (much less fight) for it.

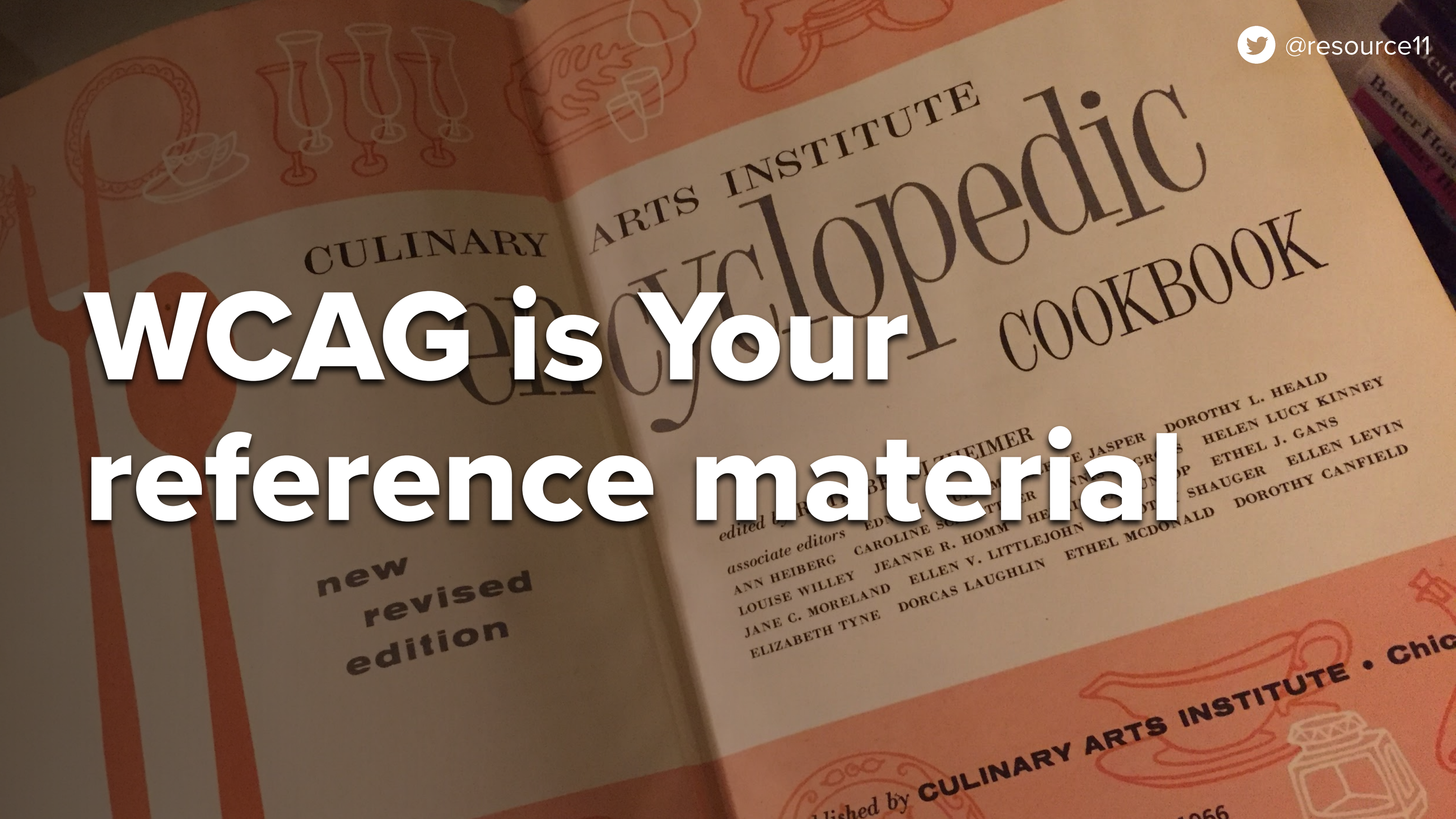
6:43 PM · Nov 6, 2019 · TweetDeck

32 Retweets **162** Likes



Components are your tried and true recipes

@resource11



WCAG is Your reference material



Component recipe

@resource11

Luncheon or dinner guests are sure to enjoy colorful Citrus Lobster Salad.

CITRUS LOBSTER SALAD

@resource11

FRESH SPRINGTIME SALAD

1 pound fresh asparagus spears
4 cups torn fresh spinach
2 cups torn red leaf lettuce
2 cups torn leaf lettuce
Warm Seasoned Dressing

Snap off tough e
scales with a knife
sired. Cut asparagus
asparagus in a vege
water. Cover and ste
crisp-tender.

Combine asparagu
a large bowl, tossing gently. Pour dressing over salad; toss gently. Serve immediately. Yield: 9 servings (30 calories per 1-cup serving).

Warm Seasoned Dressing:

¼ cup plus 3 tablespoons canned low-sodium chicken broth, undiluted

Remove stems from spinach; discard stems. Wash leaves thoroughly, and pat dry with paper towels. Set spinach aside.

Combine chutney and next 5 ingredients in container of an electric blender or food processor; top with cover, and process 30 seconds or until combi

ons,
owl.
ntly.
ories

ROL 0 /

Use high-quality ingredients

semantic HTML



SAVOR THE FLAVOR

When using fresh herbs, keep in mind that fresh herbs are milder in flavor than dried herbs. As a rule of thumb, add three times the amount of a fresh herb to a recipe to match the intensity of flavor of the dried variety. For instance, if

@resource11

FRESH SPRINGTIME SALAD

1 pound fresh asparagus spears
4 cups torn fresh spinach
2 cups torn red leaf lettuce
2 cups torn leaf lettuce
Warm Seasoned Dressing

Snap off tough ends of
scales with a knife or vege
sired. Cut asparagus into 2-i
asparagus in a vegetable st
water. Cover and steam 4 t
crisp-tender.

Combine asparagus and n
a large bowl, tossing gently. Pour dre
salad; toss gently. Serve immediat
servings (30 calories per 1-cup se

Mix in seasonings

ARIA ...just a touch

Remove stems from spinach; discard stems. Wash leaves thoroughly, and pat dry with paper towels. Set spinach aside.

Combine chutney and next 5 ingredients in container of an electric blender or food processor; top with cover, and process 30 seconds or until combined

Warm Seasoned Dressing:

¼ cup plus 3 tablespoons canned low-sodium chicken broth, undiluted



SAVOR THE FLAVOR

When using fresh herbs, keep in mind that fresh herbs are milder in flavor than dried herbs. As a rule of thumb, add three times the amount of a fresh herb to a recipe to match the intensity of flavor of the dried variety. For instance, if

@resource11

FRESH SPRINGTIME SALAD

1 pound fresh asparagus spears
4 cups torn fresh spinach
2 cups torn red leaf lettuce
2 cups torn leaf lettuce
Warm Seasoned Dressing

off tough ends of asparagus. Re

large bowl, tossing gently. Pour dressing over salad; toss gently. Serve immediately. Yield: 9 servings (30 calories per 1-cup serving).

Follow the directions

documentation

Warm Seasoned Dressing:

¼ cup plus 3 tablespoons canned low-sodium chicken broth, undiluted

Remove stems from spinach; discard stems. Wash leaves thoroughly, and pat dry with paper towels. Set spinach aside.

Combine chutney and next 5 ingredients in container of an ele ic blender or food processor; top wit , and process 30 seconds or until scraping sides of container.

orange sections, in a large bowl. alad; toss gently. ervings (90 calories

E 14.9 / CHOLESTEROL 0 /



SAVOR THE FLAVOR

When using fresh herbs, keep in mind that fresh herbs are milder in flavor than dried herbs. As a rule of thumb, add three times the amount of a fresh herb to a recipe to match the intensity of flavor of the dried variety. For instance, if

@resource11

FRESH SPRINGTIME SALAD

1 pound fresh asparagus spears
4 cups torn fresh spinach
2 cups torn red leaf lettuce
2 cup… leaf lettuce
Warm…

Sna…
scales…
sired.
aspara…
water.
crisp-tender.

Combine asparagus and next 3 ingredients in a large bowl, tossing gently. Pour dressing over salad; toss gently. Serve immediately. Yield: 9 servings (30 calories per 1-cup serving).

Warm Seasoned Dressing:

¼ cup plus 3 tablespoons canned low-sodium chicken broth, undiluted

Remove stems from spinach; discard stems. Wash leaves thoroughly, and pat dry with paper towels. Set spinach aside.

Combine chutney and next 5 ingredients in container of … blender or food proces- … process 30 seconds or … sides of container. …pple, orange sections, …monds in a large bowl. … over salad; toss gently. …ld: 8 servings (90 calories …

…FAT 3.5 / CARBOHYDRATE 14.9 / CHOLESTEROL 0 / …A 101 / CALCIUM 37

Read helpful hints

best practices



SAVOR THE FLAVOR

When using fresh herbs, keep in mind that fresh herbs are milder in flavor than dried herbs. As a rule of thumb, add three times the amount of a fresh herb to a recipe to match the intensity of flavor of the dried variety. For instance, if

Icons

 checkmark-box

 checkmark-circle

 checkmark

 chevron-down

 chevron-left

 chevron-right

 chevron-up

 close-x

 code

 collection

Icons

Informative or decorative

Informative icons need descriptive text

Decorative icons need to be hidden from assistive technology

SVGs vs. Icon Fonts

VoiceOver bugs make SVG sadness



Icon fonts to the rescue

Icon font pattern

```
<span class="icon icon-email"></span>
```

Accessible icon pattern

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Icon font

@resource11

```
.icon {
    color: currentColor;
    font-family: "ORM Icons";
}
```

```
<span class="root
    <span class="icon icon-email"
          aria-hidden="true"><
    <span class="visuallyHidde
</span>
```

```
.icon-email::before {
    content: "\f12f";
}
```

Hide icon font from screen readers

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidde
</span>
```

```
.icon-email::before {
    content: "\f12f";
}
```

Descriptive icon name

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Visually-hidden text

```
.visuallyHidden {
    position: absolute;
    overflow: hidden;
    clip: rect(0 0 0 0);
    height: 1px;
    width: 1px;
    margin: -1px;
    padding: 0;
    border: 0;
}
```

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Margin/padding support

```
<span class="root">
    <span class="icon icon-email"
        aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

```
.root {
    display: inline-block;
}
```

Spans all the way down

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Informative or decorative?

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Informative icons should announce

```
<span class="root">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

Hide decorative icons from screen readers

```
<span class=“root" aria-hidden="true">
    <span class="icon icon-email"
          aria-hidden="true"></span>
    <span class="visuallyHidden">email</span>
</span>
```

React functional component

@resource11

HTML syntax

```
export const Icon = props => {

  return (
    <span class="root">
      <span class="icon icon-email" aria-hidden="true" />
      <span class="visuallyHidden">email</span>
    </span>
  );
}

export default Icon;
```

```
export const Icon = props => {
  return (
    <span className="root">
      <span aria-hidden={true} />
      <span class="visuallyHidden">email</span>
    </span>
  );
}

export default Icon;
```

class ⇒ className

“true” ⇒ {true}

Close empty span

JSX syntax

```
export const Icon = props => {

  return (
    <span className="root">
      <span className="icon icon-email" aria-hidden={true} />
      <span className="visuallyHidden">email</span>
    </span>
  );
}

export default Icon;
```

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;
    const icon = Icons[iconName];
    if (!icon) return null;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <span
          className={`icon icon-${iconName}`} aria-hidden={true}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Support incoming props, add guardrails

Guardrails



```
export const Icon = props => {
  const { iconHidden, iconName, iconTitle } = this.props;
  const icon = Icons[iconName];
  if (!icon) return null;

  return (
    <span className="root" aria-hidden={iconHidden ? true : null}>
      <span
        className={`icon icon-${iconName}`} aria-hidden={true}
      />
      <span className="visuallyHidden">
        {iconTitle || iconName}
      </span>
    </span>
  );
}
export default Icon;
```

Icon naming guardrails

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;
    const icon = Icons[iconName];
    if (!icon) return null;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <span
          className={`icon icon-${iconName}`} aria-hidden={true}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Icon description guardrails

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;
    const icon = Icons[iconName];
    if (!icon) return null;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <span
          className={`icon icon-${iconName}`} aria-hidden={true}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Icon Hiding guardrails

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;
    const icon = Icons[iconName];
    if (!icon) return null;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <span
          className={`icon icon-${iconName}`} aria-hidden={true}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Icon Hiding guardrails

...but SVGs!



Now Brewing... Font Awesome 6!

The next generation of our icon library + toolkit is coming with more icons, more styles, more services, and more awesome. Pre-order now to get early access and releases over the next year!

Learn More!

Font Awesome

Get vector icons and social logos on your website with Font Awesome, the web's most popular icon set and toolkit.

Start for Free

Get More with Pro

Version 5.14.0 • 1,598 Free Icons • 7,848 Pro Icons • Download

Now Brewing... Font Awesome 6!
The next generation of our icon library + toolkit is coming with more icons, more styles, more services, and more awesome. Pre-order now to get early access and releases over the next year!
Learn More!

React

Font Awesome now has an official React component that's available for a friction-less way to use our icons in your React applications.

Before we start, review the following and make sure you're...

- wanting to use React and not React Native - that's a different react-native-fontawesome component.
- okay with using SVG + JS to render icons in your project - this react component relies on that technology.
- already set up to use Pro packages if you plan on using Pro-only icons.

Get Started



It's time to take control of your online reputation. Get more reviews on the sites that matter most, for free.
ads via Carbon

Set Up
Getting Started
Upgrading from Version 4
Hosting Font Awesome Yourself
Using a Package Manager

Font Awesome works how and where you need it to

Web Fonts + CSS

SVG + JS

Sketch

Adobe Apps

Vue.js

Angular

React

Ember

Accessibility-Minded

Font Awesome loves screen readers! With our auto-accessibility feature, it's super simple to do the right thing for your users.

Desktop-Friendly

Font Awesome is fantastic to work with on the desktop, especially with all-new ligatures. Try it in your next design or

Tried & Tested

We've stress-tested them, so that your icons and styling work and display perfectly in all modern browsers.

Pixel-Perfect Rendering

Our icons are meticulously designed on a grid that helps them render perfectly and

```
<FontAwesomeIcon
    icon="coffee"
/>
```

```
<svg
   data-icon="coffee" aria-hidden="true"
   focusable="false"
/>
      <path>…</path>
</svg>
```

```
<FontAwesomeIcon
    icon="coffee"
    ariaLabel="beverage"
/>
```

```
<FontAwesomeIcon
    icon="coffee"
    ariaLabel="beverage"
/>
```

```
<svg
   data-icon="coffee" aria-hidden="true"
   focusable="false" aria-label="beverage"
/>
      <path>…</path>
</svg>
```

@resource11

aria-hidden hides from screen reader

```
<svg
   data-icon="coffee"
   focusable="false"
/>
      <path>…</path>
</svg>
```

```
aria-hidden="true"
aria-label="beverage"
```

aria-label will not announce

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;

    return (
      <span className=“root” aria-hidden={iconHidden ? true : null}>
        <span
          className={`icon icon-${iconName}`} aria-hidden={true}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Swap out font-based className

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <FontAwesomeIcon
          icon={iconName}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

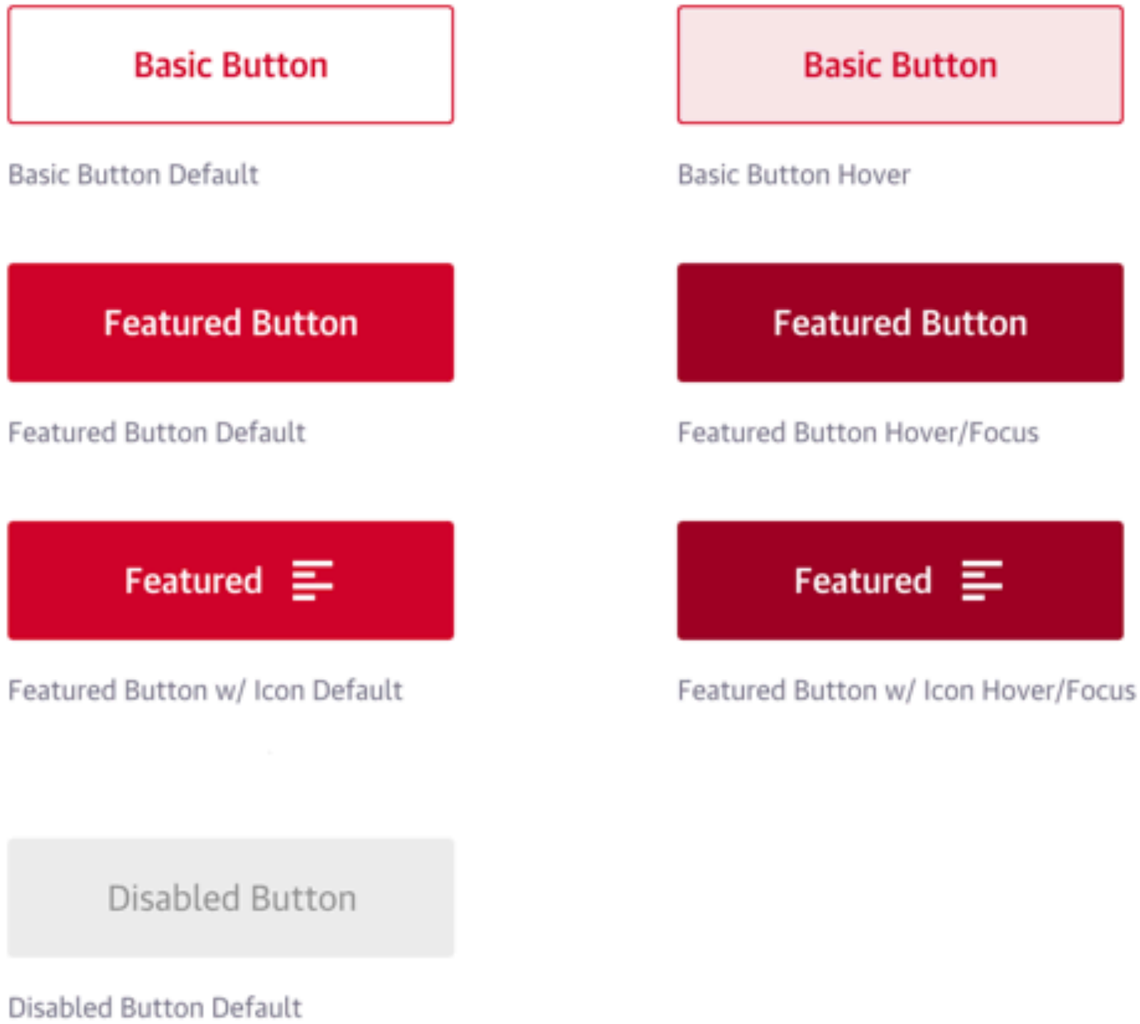
Swap in FontAwesomeIcon

```
export const Icon = props => {
    const { iconHidden, iconName, iconTitle } = this.props;

    return (
      <span className="root" aria-hidden={iconHidden ? true : null}>
        <FontAwesomeIcon
          icon={iconName}
        />
        <span className="visuallyHidden">
          {iconTitle || iconName}
        </span>
      </span>
    );
}
export default Icon;
```

Informative icon naming supported

Buttons


Basic Button
Basic Button Default
Basic Button
Basic Button Hover
Featured Button
Featured Button Default
Featured Button
Featured Button Hover/Focus
Featured
Featured Button w/ Icon Default
Featured
Featured Button w/ Icon Hover/Focus
Disabled Button
Disabled Button Default

Buttons

Perform an action on the page

Should look and act like a button

Screen reader and keyboard functionality for free

Button

```
<button class="root">
    Send an email
</button>
```

Button

Sprinkle some ARIA

```
<button class="root" aria-label="Read more about Dinosaurs">
    Read more
</button>
```

Button

Sprinkle some ARIA

```
<button class="root" aria-label=“Read more about dinosaurs">
    Read more
</button>
```

Button text and aria label start with same words

Button

Sprinkle some ARIA

```
<button class="root" aria-label=“Read more about dinosaurs">
    Read more
</button>
```

Button text and aria label start with same words

Button

```
<button className="root" aria-label="Read about Dinosaurs">
    Read More
</button>
```

```
<button className="root" aria-label="Read about Dinosaurs>
        {children}
            <Icon
                iconName={iconName}
                iconHidden={true}
            />
</button>
```

Add icon support

```
<button className="root" aria-label="Read about Dinosaurs
        onClick={onClick} disabled={disabled ? true : null}>
        <span className="btnContentWrap">
            {children}
             <Icon
                 iconName={iconName}
                 iconHidden={true}
             />
        </span>
</button>
```

Wrap contents in span

```
export const Button = props => {
  const { ariaLabel, children, disabled, iconName, onClick } = this.props;

    return (
      <button className="root" aria-label={ariaLabel}
        onClick={onClick} disabled={disabled ? true : null}>
        <span className="btnContentWrap">
          {children}
          {iconName && (
            <Icon
              iconName={iconName}
              iconHidden={true}
            />
          )}
        </span>
      </button>
  );
}
```

Add props, click handler & disabled button support

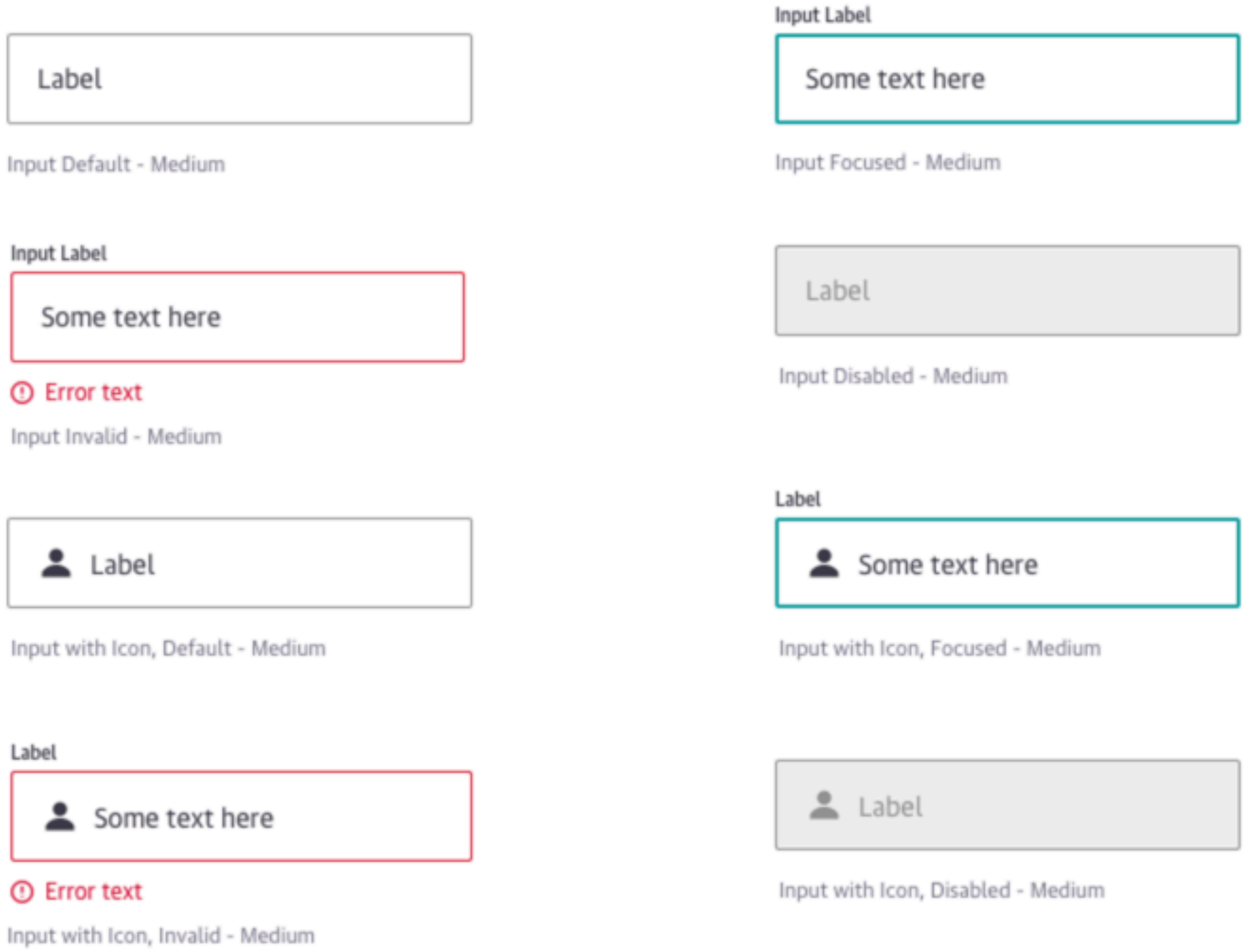
```
export const Button = props => {
  const { ariaLabel, children, disabled, iconName, onClick.}.= this.props;

    return (
      <button className="root" aria-label={ariaLabel}
        onClick={onClick} disabled={disabled ? true : null}>
        <span className="btnContentWrap">
          {children}
          {iconName && (
            <Icon
              iconName={iconName}
              iconHidden={true}
            />
          )}
        </span>
      </button>
  );
}
```

Add guardrails

No iconName?
No icon rendered

Inputs


Label
Input Default - Medium
Input Label
Some text here
Input Focused - Medium
Input Label
Some text here
Error text
Input Invalid - Medium
Label
Input Disabled - Medium
Label
Input with Icon, Default - Medium
Label
Some text here
Input with Icon, Focused - Medium
Label
Some text here
Error text
Input with Icon, Invalid - Medium
Label
Input with Icon, Disabled - Medium

Inputs need labels and error messages

Labeled inputs give all users more context

Placeholders are NOT labels

Avoid using placeholders instead of labels, users will lose context

Hard to style across browsers

Placeholders aren’t auto translated

Avoid Horizontal Scrolling

Max input width:
80 characters

Keep labels stacked vertically in close proximity

Labels above input, errors below input

```
<div className="root">
    {label && (
        <label htmlFor={id} className="inputLabel">{label}</label>
        {iconName && (
            <Icon
                iconName={iconName}
                iconHidden={true}
            />
        )}
        <input
            className={inputClasses}
            id={id}
            name={id}
            type={type}
            value={value}
            disabled={disabled ? true : null}
            onChange={onChange}
            onKeyPress={onKeyPress}
            aria-describedby={invalid && error ? `error-${id}` : null}
            aria-invalid={invalid ? true : null}
            aria-required={required ? true : null}
        />
    }
    {invalid &&
        error &&
            <div className="errorWrapper">
                <Icon iconName="warning-fill"
                    iconHidden={true} />
                <span
                    id={`error-${id}`}
                    className="errorTxt"
                    aria-live="polite"
                >
                    {error}
                </span>
            </div>
     }
</div>
```

```
<div class="root">
    <label for="inputFoo" class="inputLabel">
        First name
    </label>
    <input
        class="inputClasses"
        id="inputFoo"
        name="inputFoo"
        type="text"
    />
    <span id="errorFoo" class="errorTxt">
        Whoops! Error.
    </span>
</div>
```

Semantic HTML

```
<div class="root">
  <label for="inputFoo" class="inputLabel">
    First name
  </label>
  <input
    class="inputClasses"
    id="inputFoo"
    name="inputFoo"
    type="text"
  />
  <span id="errorFoo" class="errorTxt">
    Whoops! Error.
  </span>
</div>
```

Labels and error messages

```
<div className="root">
  <label htmlFor="inputFoo" className="inputLabel">
    First name
  </label>
  <input
    className="inputClasses"
    id="inputFoo"
    name="inputFoo"
    type="text"
  />
  <span id="errorFoo" className="errorTxt">
    Whoops! Error.
  </span>
</div>
```

Associate label with input

```
<div className="root">
  <label htmlFor="inputFoo" className="inputLabel">
    First name
  </label>
  <input
    className="inputClasses"
    id="inputFoo"
    name="inputFoo"
    type="text"
    aria-describedby="errorFoo"
    aria-invalid="true"
    aria-required="true"
  />
  <span id="errorFoo" className="errorTxt">
    Whoops! Error.
  </span>
</div>
```

Sprinkle some ARIA

```
<div className="root">
  <label htmlFor="inputFoo" className="inputLabel">
    First name
  </label>
  <input
    className="inputClasses"
    id="inputFoo"
    name="inputFoo"
    type="text"
    aria-describedby="errorFoo"
    aria-invalid="true"
    aria-required="true"
  />
  <span id="errorFoo" className="errorTxt" aria-live="polite">
    Whoops! Error.
  </span>
</div>
```

disabled and synthetic event support

```
disabled={disabled ? true : null}
onChange={onChange}
onKeyPress={onKeyPress}
aria-describedby={invalid && error ? `error-${id}` : null}
aria-invalid={invalid ? true : null}
aria-required={required ? true : null}
```

and true/null handling

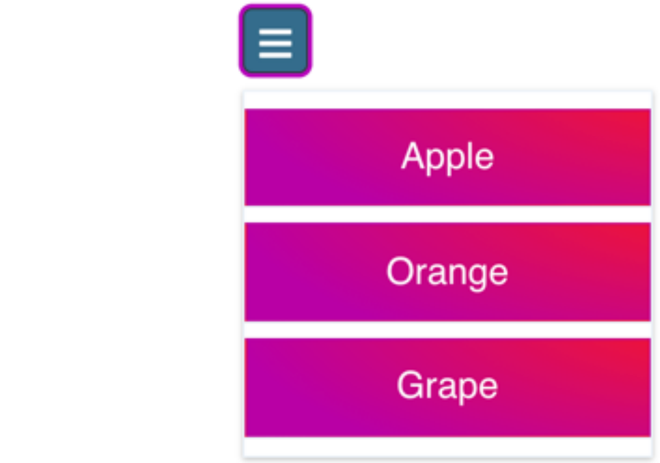
```
{label && (
  <label htmlFor={id} className="inputLabel">
    {label}
  </label>
  {iconName && (
    <Icon
      iconName={iconName}
      iconHidden={iconHidden}
      title={iconTitle}
    />
  )}
  <input
    className={inputClasses}
    id={id}
    name={id}
    type={type}
    value={value}
    disabled={disabled ? true : null}
    onChange={onChange}
    onKeyPress={onKeyPress}
    aria-describedby={invalid && error ? `error-${id}` : null}
    aria-invalid={invalid ? true : null}
    aria-required={required ? true : null}
  />
)}
```

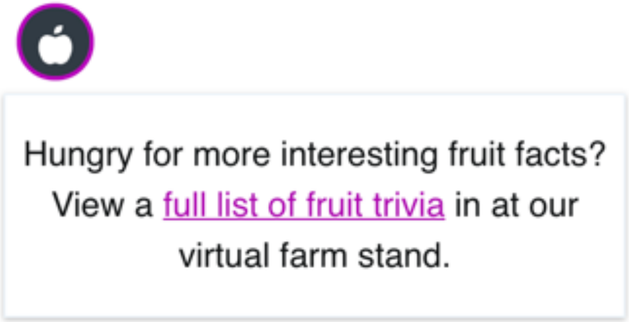
Add guardrails

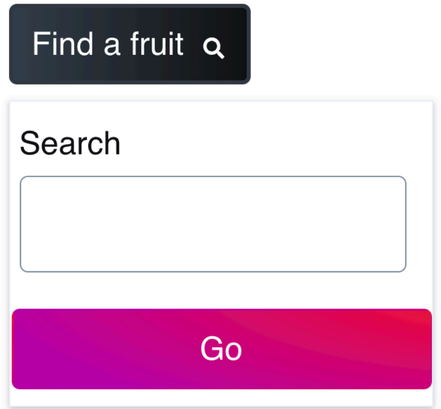
```
{invalid &&
        error && (
            <div className="errorWrapper">
                <Icon iconName="warning-fill"
                    iconHidden={true} />
                <span
                    id={`error-${id}`}
                    className="errorTxt"
                    aria-live="polite"
                >
                    {error}
                </span>
            </div>
)}
```

Add guardrails

Disclosure Widgets







Disclosure Widgets

Flexible pattern

Keyboard and mouse operable

Shared base functionality

Menus & Toggle Tips

Tooltips are NOT Toggle Tips

Tooltips can be problematic

Primary keyboard interactions

Space bar/Enter key

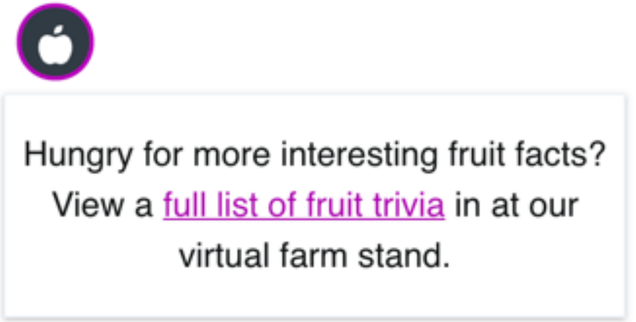
Escape key

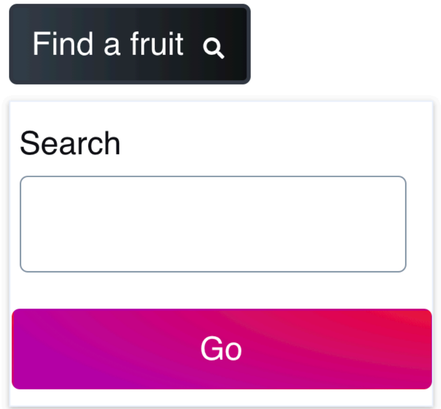
Mouse clicks

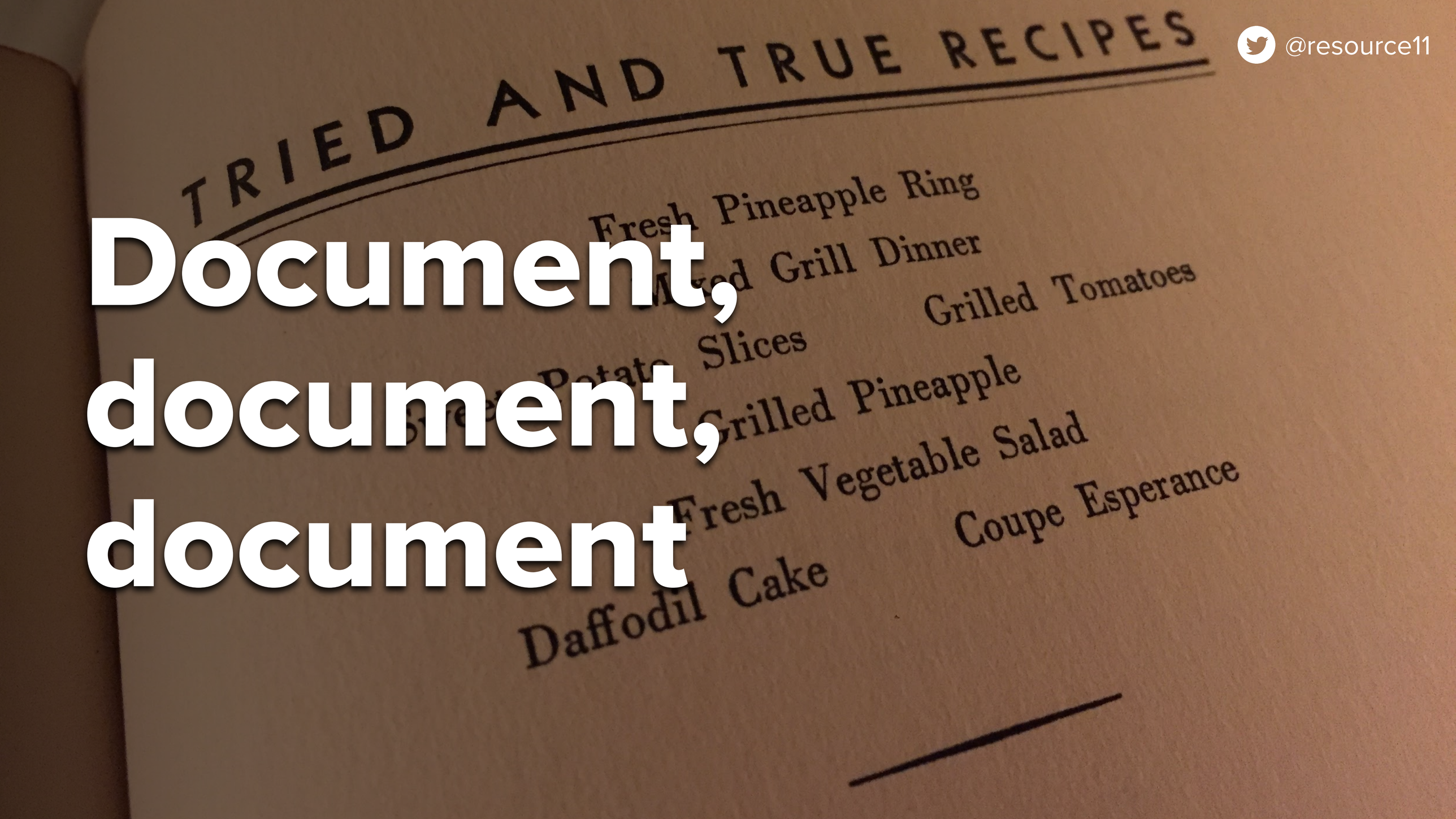
Targeted focus management

Disclosure Widgets Demo



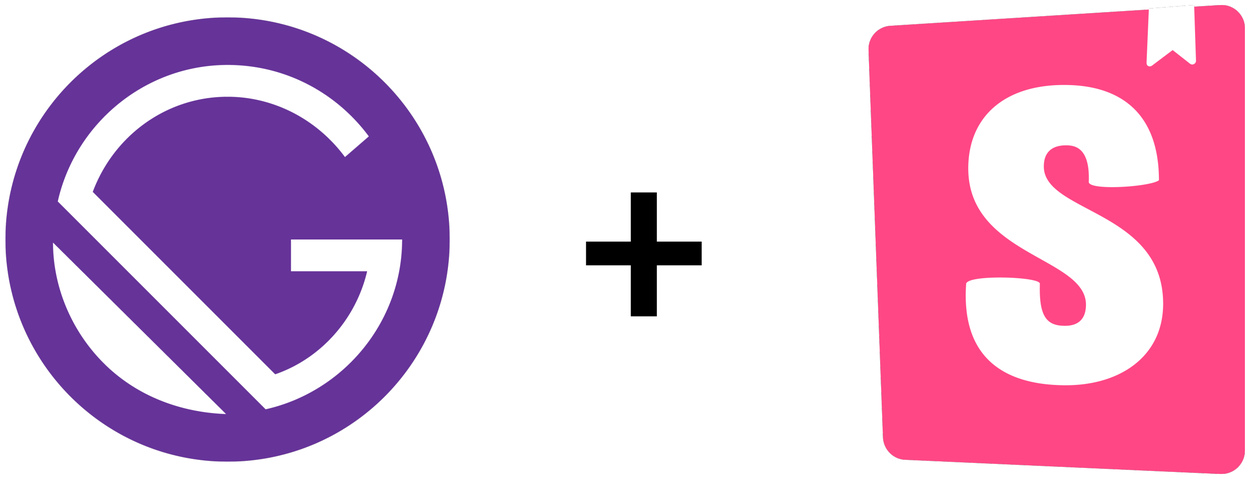


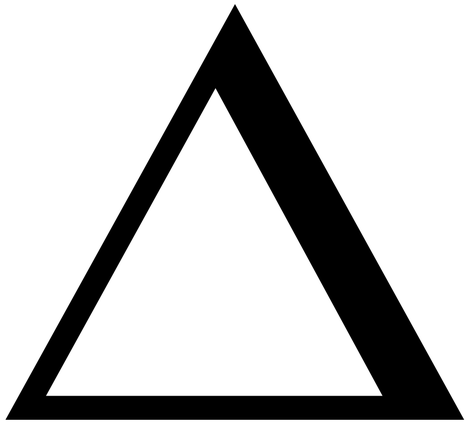




Document, document, document

@resource11






Vercel

@resource11



Storybook

Storybook

Press "/" to search...

EXPERIMENTAL

- Avatar
- BlockContainer
- CardContainer
- Experimental CardGroup Patterns
- Carousel
- Tag
- Toggle

CORE

- Button
- ByLine
- CallToAction
- Card
  - Default using max props, div wrapper, truncated text
  - **Shorter body copy has content tolerance, except for in IE11**
  - max props using CardGroup wrapper
  - full-width horizontal card in CardGroup
  - using columnSizes for all
  - using mixed imnSizes



LEARNING PATH

OCTOBER 26, 1985

essary because it is too long to show the entire thing

Madsen, Martín Bravo, Kathleen McMahon, Pauline Masigla, S. Kimball, James Touhey and Paul

This text describes the contents of the article illustrated in the image above donec consectet...

foo



LEARNING PATH

OCTOBER 26, 1985

essary because it is too long to show the entire thing

Madsen, Martín Bravo, Kathleen McMahon, Pauline Masigla, S. Kimball, James Touhey and Paul

This text describes the contents of the article.

foo



LEARNING PATH

OCTOBER 26, 1985

A super short title

By Kathleen McMahon, Pauline Masigla, S. Kimball and James Touhey

This text describes the contents of the article.

foo

Edit

Share Settings

Duplicate

Delete

Story | Accessibility | Actions

1 Violations | 10 Passes | 1 Incomplete

Highlight results

Ensures buttons have discernible text

Buttons must have discernible text

More info...

1. **article:nth-child(1) > .orm-Card-contentWrapper > footer > div > .orm-Button-root.orm-Button-small.orm-Button-secondary**

Critical

Element does not have inner text that is visible to screen readers

Serious

aria-label attribute does not exist or is empty

Serious

aria-labelledby attribute does not exist, references elements that do not exist or references elements that are empty

Minor

Element's default semantics were not overridden with role="presentation"

Minor

Element's default semantics were not overridden with role="none"

Serious

Element has no title attribut

Rerun tests

google.com

Helpful hints

Informative Icon

Custom icon name

Each icon has a visually-hidden `title` describing it's meaning, and that `title` is announced to screen readers by default. If you want the icon to be announced with a **different name** to screen readers, pass in a different string to the `title` prop.



```
<Icon name="search" title="monocle on a stick" />
```

Helpful hints

Decorative Icon

Hidden from screen readers

Each icon has a visually-hidden title describing its meaning and that title is announced to screen readers by default. If you don't want the title announced to screen readers, pass in `iconHidden={true}` to the component. Otherwise, simply omit the prop.



```
<Icon name="search" iconHidden={true} />
```

Helpful hints

Prop tables

Name	Type	Description
`color`	enum	Icon color.
`extraClasses`	theme	A map of CSS classes to be appended to DOM nodes internally. Valid keys are: See theme.
`iconHidden`	bool	Used to prevent screen readers from announcing the icon. Best practice is to set iconHidden to true or omit the prop.
`name`	enum	**Required.** Icon name.
`onClick`	func	Called when this icon is clicked.
`size`	responsive	Icon size. Receives a value matching `PropTypes.oneOf(IconSizes)` or an object with values assigned to any of these keys: `xsmall`, `xsmall600`, `small`, `small960`, `medium`. See responsive.
`title`	string	Used to override the default visually-hidden Icon title that is announced by screen readers.

Helpful hints

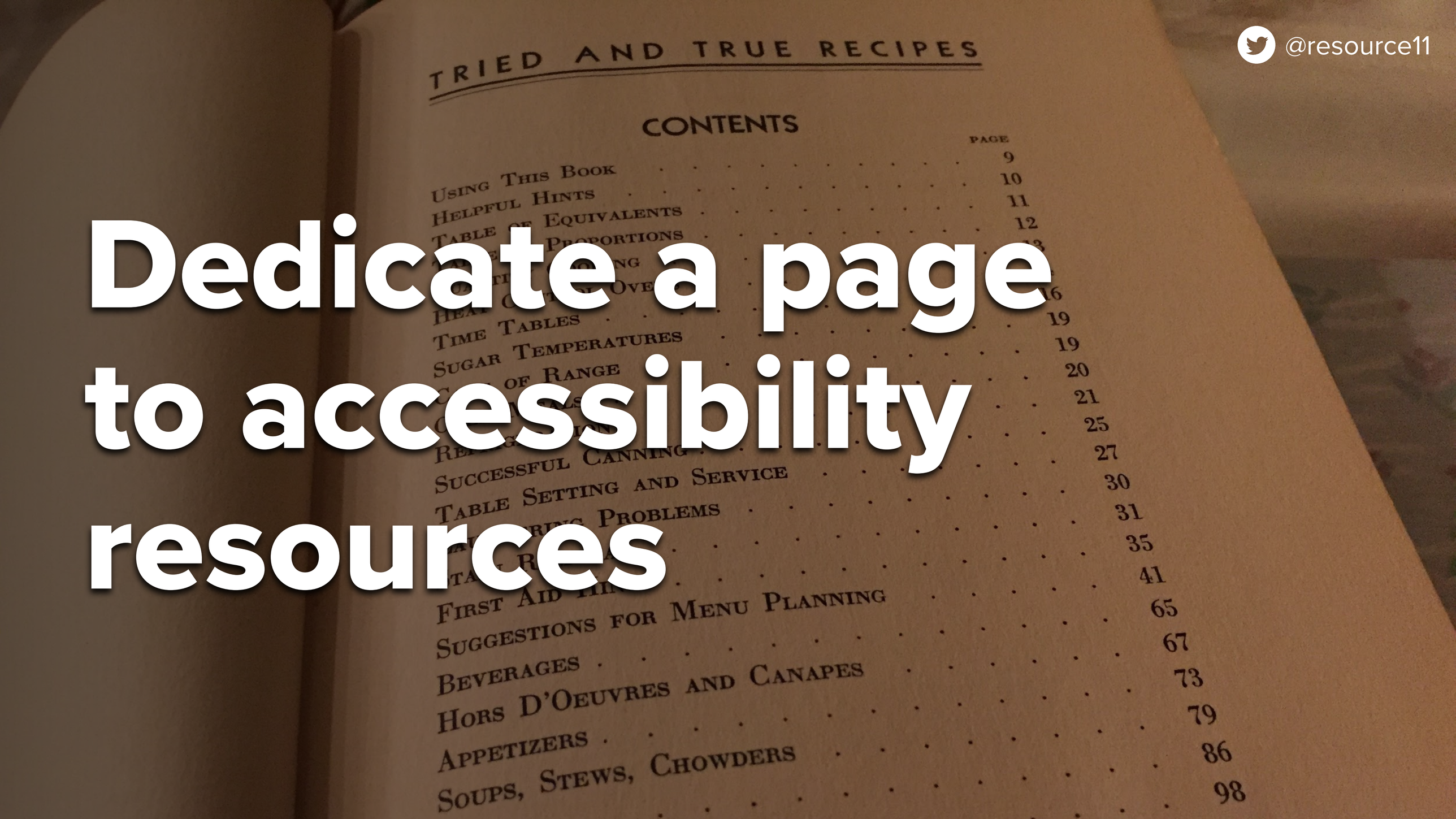
Component dos/don’ts

Do:

- Use a Basic button for a “safe” user action such as ‘Cancel’.
- Use Button with Icon variant either for decorative purposes or to provide extra information to the user.
- Pass the text as your Button's children, unless you're using an icon-only button.
- For icon-only buttons, pass the text through your Button's ariaLabel prop.
- Start your ariaLabel text with same text as your Button's children text when using both props.

Don't:

- Use a Destructive button for ‘Cancel’.
- Use a Feature Button with icon as a visual indicator the user is leaving the page. Use the Link with icon variant instead.
- Pass in block elements or links as your Button's children. Use valid HTML.
- Use an icon-only button without setting the ariaLabel prop



Dedicate a page to accessibility resources

@resource11



Wrap-up

Our users are diverse

Vision Hearing Motor Cognitive



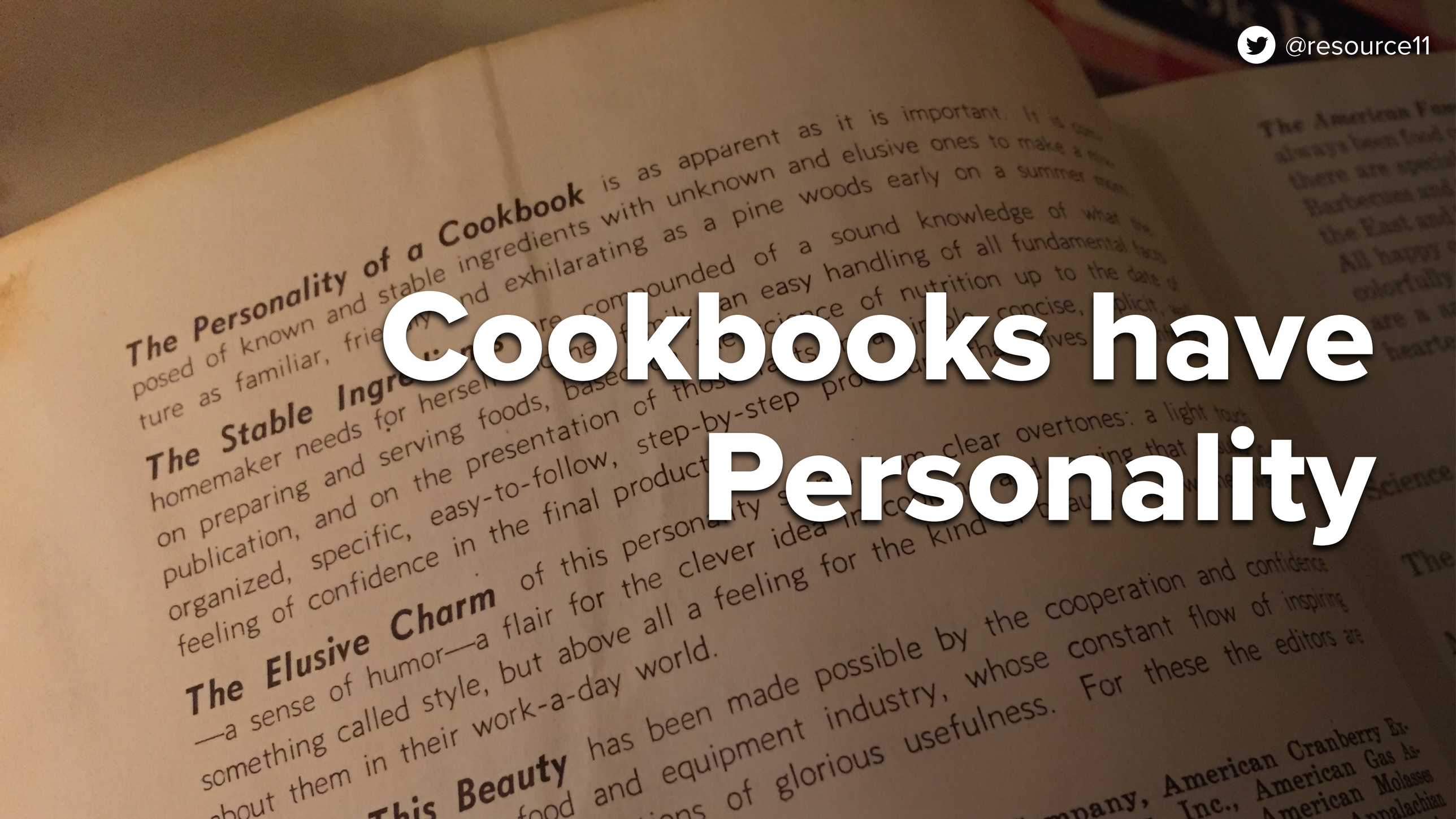
Sensory Language Low bandwidth

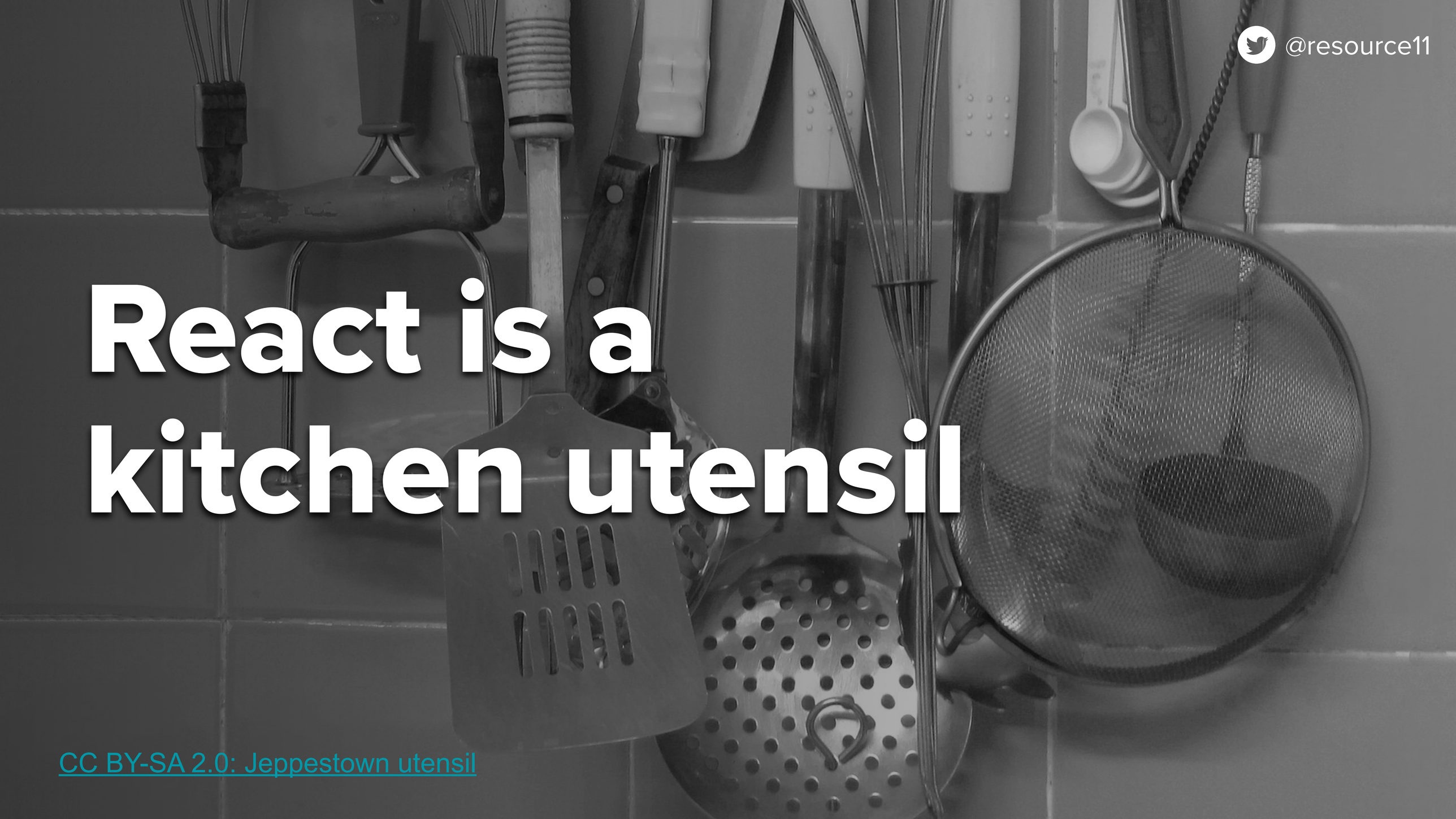
Inclusive Design Persona Spectrum (CC BY-NC-SA 2.0)



Your Design system is a cookbook

@resource11


@resource11
Cookbooks have Personality



@resource11

React is a kitchen utensil

CC BY-SA 2.0: Jeppestown utensil

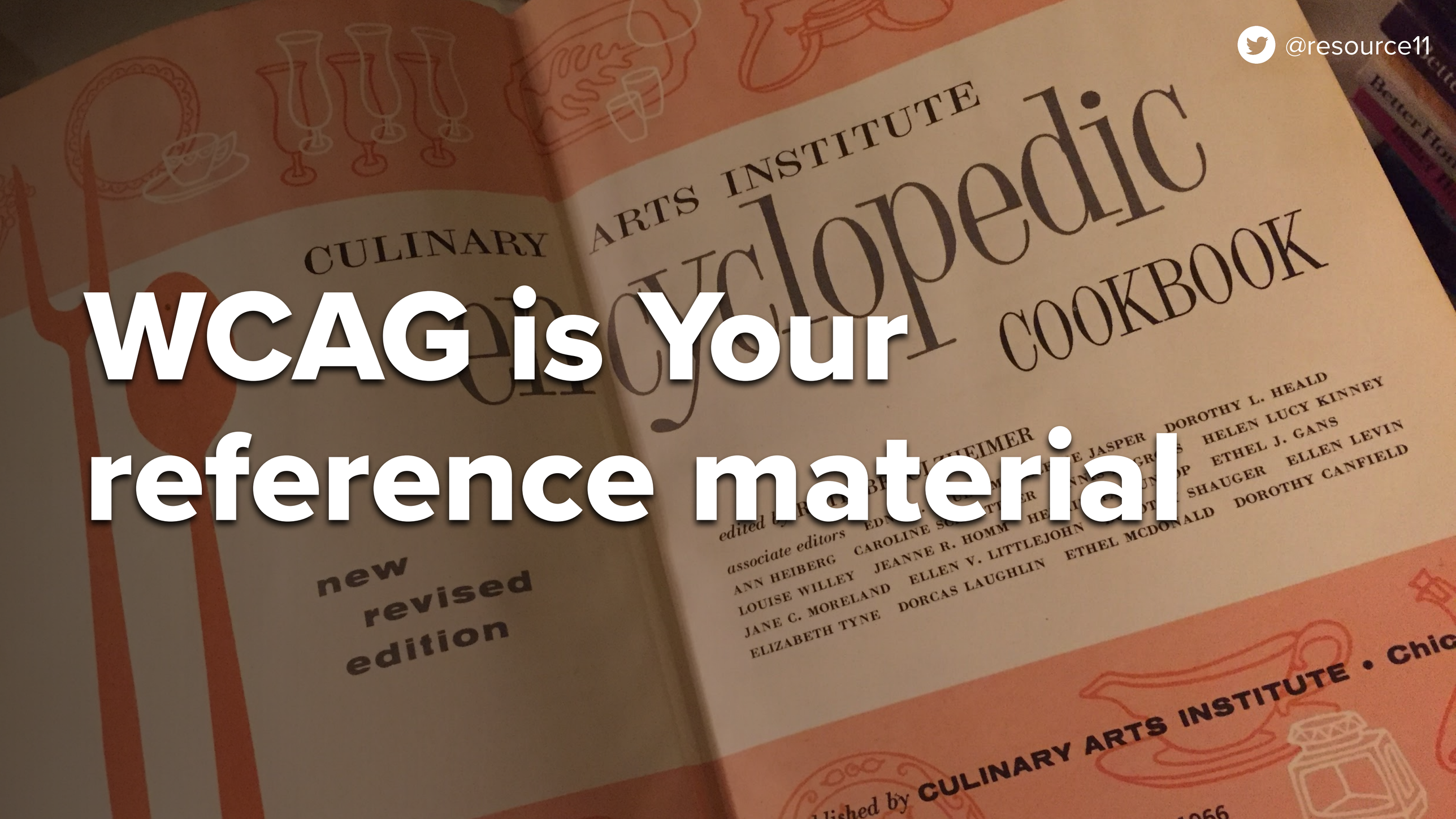


CC BY-SA 2.0: Jeppestown utensil

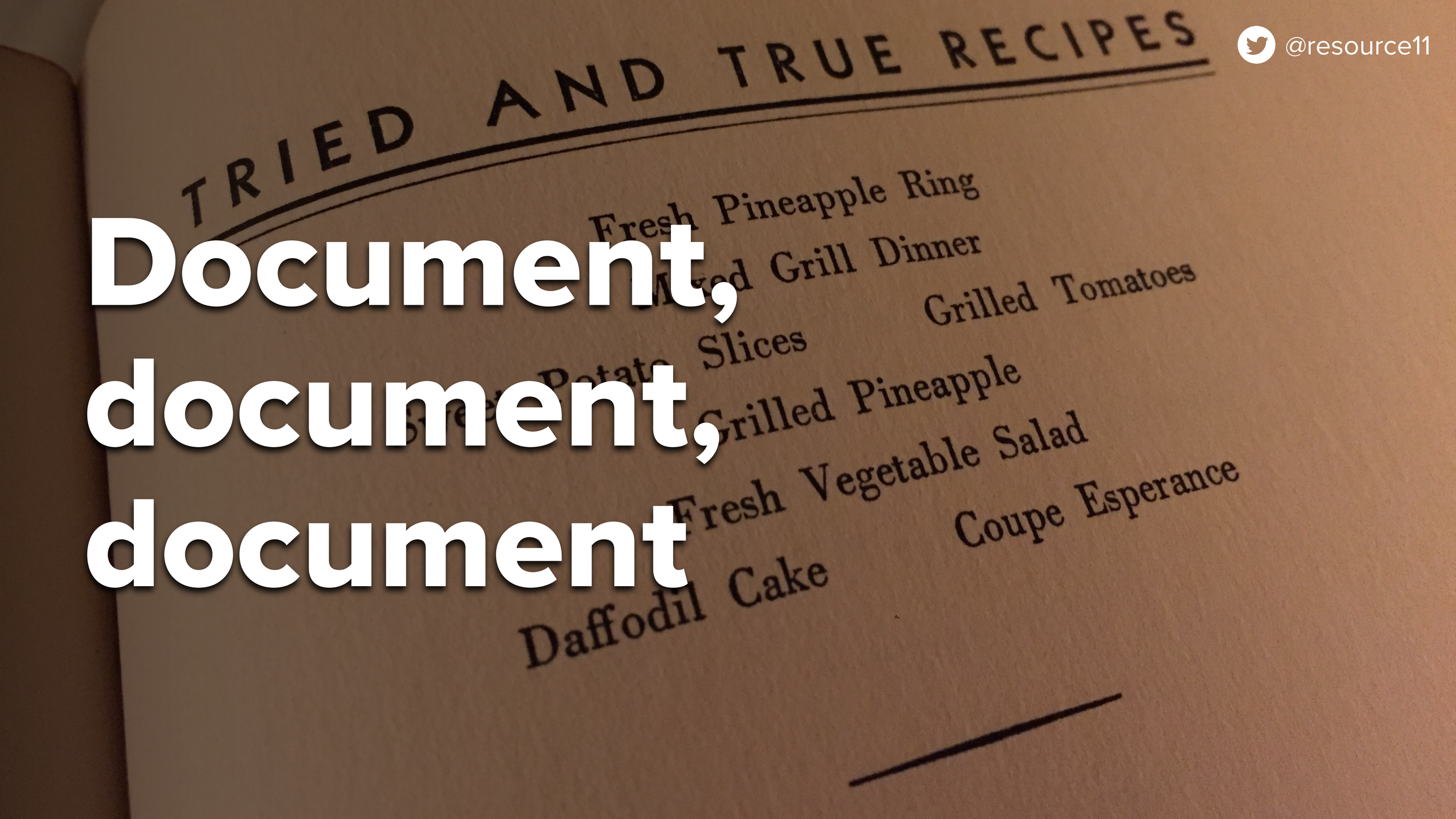


Components are your tried and true recipes

@resource11



WCAG is Your reference material



Document, document, document

@resource11


@resource11
Dinosaurs are ALWAYS the hotness



Thank you.

@resource11



@resource11

https://noti.st/resource11

Slide deck posted after the talk

@resource11

Twitter | Instagram | GitHub