Atoms

Cory House

@housecor | reactjsconsulting.com

The Plan

What's an atom?

Core decisions

- Wrap HTML primitives
- Folder structure

Design tips

Let's code!

- ProgressBar
- Label
- Icon

This is an atom.
You can’t break it down any farther.

Smallest building block

The foundation of a component library

Typically composed together

Demonstrate your base styles

Key Decisions

Decision:
Wrap HTML primitives?

Potential HTML Primitive Enhancements

- Integrated label
- Unique element ID
- Required field marker
- Default value / placeholder
- Horizontal and vertical layout
- Error messaging
- Touched, dirty meta data
- Polyfill older browsers

Wrap HTML Primitives?

<input type="text">	TextInput
<input type="checkbox">	Checkbox
<select>	Dropdown
<input type="radio">	RadioList
<select> or <input type="radio">	PickOneFromList

We'll wrap <input> and <label> in this module.

Decision:
Component folder structure

What Belongs in /components?

React Bootstrap

Button.js	Component

Blueprint

Button.tsx	Component
Button.scss	Styles

Material-UI

Button.js	Component
Button.spec.js	Tests
index.js	Export Component

React Toolbox

Button.js	Component
config.css	Configure css vars
index.js	Export component, theme
index.d.ts	Intellisense support
readme.md	Docs
theme.css	Default theme

Ant Design

Button.tsx	Component
test	Tests
demo	Demo
index.tsx	Export component
markdown	Docs in .md format

What Belongs in /Components?

	React Toolbox	Blueprint	Ant Design	Material-UI	React Bootstrap
Folder per component	x	x	x	x	
Style	x	x	x	Inline	
Tests			x	x	
Theme	x				
Docs	x		x		
Demo			x		
index.js	x		x	x	

Our Plan

/Button

Button.js

Button.spec.js

index.js

Handled elsewhere:

- Docs
- Demos

Atom Design Tips

Tip 1:

Honor the native API

```
const passwordInput({value, maxLength}) => {
  return (
      <input
        type=“password”
        value={value}
        maxLength={maxLength} />
}
```

Honor the underlying element’s API

- value
- maxLength
- events (onFocus, onBlur...)

```
// Avoid
userList

// Instead
users
```

Tip 2:

Pass Props Via Spread

Pass Props via Spread

{...props}

Use spread operator to transfer props

```
return (
  <div>
    <input type=“password” {...this.props}></input>
  </div>
)
```

Tip 3:

Use Spread with Destructuring

```
const Hello = ({ name, ...rest}) =>
  <div {...rest}>Hi {name}!</div>
```

Object Spread = Stage 3

Not officially part of JS spec yet

Some editors may lack support

TC39 Stages

0 Strawman: just an idea, possible Babel plugin.

1 Proposal: this is worth working on.

2 Draft: initial spec.

3 Candidate: complete spec and initial browser implementations.

4 Finished: will be added to the next yearly release.

COMPAT ES ECMAScript | 5 | 6 | 2016+ | next | intl | non-standard | compatibility table

Flattr 56 by kangax Gratipay & webbedspace & zloirock Fork 419

Sort by Engine types | Show obsolete platforms | Show unstable platforms

V8 | SpiderMonkey | JavaScriptCore | Chakra | Carakan | KJS | Other

Minor difference (1 point) | Small feature (2 points) | Medium feature (4 points) | Large feature (8 points)

		Compilers/polyfills					Desktop browsers										
	100%	56%	71%	48%	59%	18%	5%	11%	83%	93%	95%	86%	94%	94%	97%	97%	
Feature name ▶	Current browser	Traceur	Babel + core-js[2]	Closure	Type-Script + core-js	es6-shim	KQ 4.14[3]	IE 11	Edge 13[4]	Edge 14[4]	Edge 15[4]	FF 45 ESR	FF 51	FF 52 Beta	FF 53 Aurora	FF 54 Nightly	C OF
Optimisation																	
proper tail calls (tail call optimisation) ▶	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	
Syntax																	
default function parameters ▶	7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	7/7	4/7	6/7	6/7	7/7	7/7	
rest parameters ▶	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	
spread (...) operator ▶	15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	1
object literal extensions ▶	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	
for..of loops ▶	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	7/9	9/9	7/9	7/9	7/9	9/9	9/9	
octal and binary literals ▶	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	
template literals ▶	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	
RegExp "y" and "u" flags ▶	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	2/5	5/5	5/5	5/5	5/5	
destructuring, declarations ▶	22/22	20/22	21/22	19/22	15/22	0/22	0/22	0/22	0/22	21/22	22/22	19/22	21/22	21/22	22/22	22/22	
destructuring, assignment ▶	24/24	23/24	24/24	17/24	19/24	0/24	0/24	0/24	0/24	23/24	24/24	21/24	23/24	23/24	24/24	24/24	
destructuring, parameters ▶	23/23	19/23	20/23	18/23	15/23	0/23	0/23	0/23	0/23	22/23	23/23	18/23	20/23	20/23	23/23	23/23	

Tip 4:

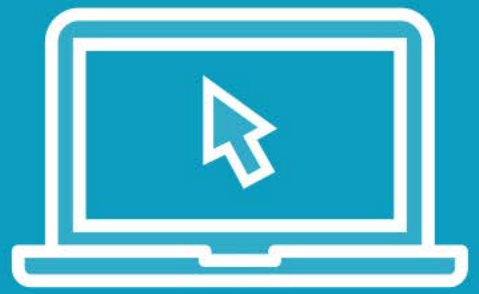
Create Formatting Components

Formatting Component Example

`<Cash>6</Cash>` → `$6.00`

`<Cash showDecimal={false}>6.42</Cash>` → `$6`

Demo



Create Atoms

- ProgressBar
- Label
- Icon

Summary

Atoms

- Smallest piece
- Building blocks
- Foundation
- Often composed
- Encapsulate opinions

Tips

- Honor the native API
- Pass props via spread
- Use spread with destructuring
- Consider formatting components

Next up: Molecules