Creating a Design System Using Glamorous

What’s on tap

1. Why should I use a design system?
2. Why use `glamorous`?
3. How can I do this myself?

Design Systems

- save team members from confusion over style details
- reduce decision making when building enhancements and features, stay high level
- maintain a consistent appearance across the interface, consistent expectations for users

A design system is as an API for your UI


glamorous

Why not bootstrap

- made for starting out new projects
- left with a lot of bloat you no longer need
- similar to whipping up a boilerplate API

A super small group of developers and I got together to design and build a new internal tool…Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company.

— Mark Otto

What is `glamorous-native`?

- a CSS in JS solution for React Native
- Dynamic styles
- Theming
- Deeper component composition
- Smarter property forwarding

Why glamorous?

- flexible api
- support from PayPal, good community
- Combines styles + components to make for a smaller presentational piece
- the glamorous factory!


Campbell's
CONDENSED
TOMATO
SOUP

Important note: The following content is a work in progress draft of guidance around how to use IBM Plex, IBM's new corporate typeface. This typeface is open sourced. It can be used by IBM employees and by the public.

We welcome all feedback and improvements through our GitHub issues. All written copy and examples are subject to change. Anyone who uses the font files and code may experience breaking changes at this time.

Typography is the appearance and arrangement of type. Each word or paragraph lives in a grid system—creating helpful alignments, harmonious relationships and clear organizations. Typography should reflect careful choices in size, placement, and weight to create useful hierarchies to guide users through the content.

With our new bespoke corporate typeface, IBM Plex™, comes a new set of guidance and best practices. IBM typography is international and modern to reflect our brand and our design principles.

Important note: The following content is a work in progress draft of guidance around how to use IBM Plex, IBM's new corporate typeface. This typeface is open sourced. It can be used by IBM employees and by the public.

We welcome all feedback and improvements through our GitHub issues. All written copy and examples are subject to change. Anyone who uses the font files and code may experience breaking changes at this time.

IBM PLEX SYSTEM

Typography is the appearance and arrangement of type. Each word or paragraph lives in a grid system—creating helpful alignments, harmonious relationships and clear organizations. Typography should reflect careful choices in size, placement, and weight to create useful hierarchies to guide users through the content.

With our new bespoke corporate typeface, IBM Plex™, comes a new set of guidance and best practices. IBM typography is international and modern to reflect our brand and our design principles.

IBM Plex System

Defined Typescale

The Plex typescale roughly follows the formula below when determining font size:

$$y_0 = 12$$
$$y_0 = y_{n-1} + \lfloor (n-2) \div 4 \rfloor \cdot 2$$

IBM Plex System

Line-height

For line-height, Plex is a bit more flexible.

Font Size	Line-height
12	Font-size + 4
14-24	Font-size + 6
28-48	Font-size + 8
54-112	Font-size + 10
112-180	Font-size + 8

Creating a schema

Define your "API"

Font size

```
const BASE_EMS_SCALE = [.75, .875, 1, 1.125,
1.25, 1.5, 1.75, 2, 2.25, 2.625, 3, 3.375, 3.75,
4.25, 4.75, 5.25, 5.75, 6.375,
7, 7.625, 7.25, 9, 9.75, 10.5];

const getFontSizeFromScale = (scale) => {
  return BASE_EMS_SCALE[scale] * 16;
}
```

Line-height

```
const getLineHeight = (size) => {
  let lineHeight;
  switch (size) {
    case size === 12:
      lineHeight = size + 4;
      break;
    case (size >= 14 && size <= 24):
      lineHeight = size + 6;
      break;
    case (size >= 54 && size <= 112):
      lineHeight = size + 8;
      break;
    case (size >= 54 && size <= 112):
      lineHeight = size + 10;
      break;
    case (size >= 112 && size <= 180):
      lineHeight = size + 8;
      break;
  }
  return lineHeight
}
```

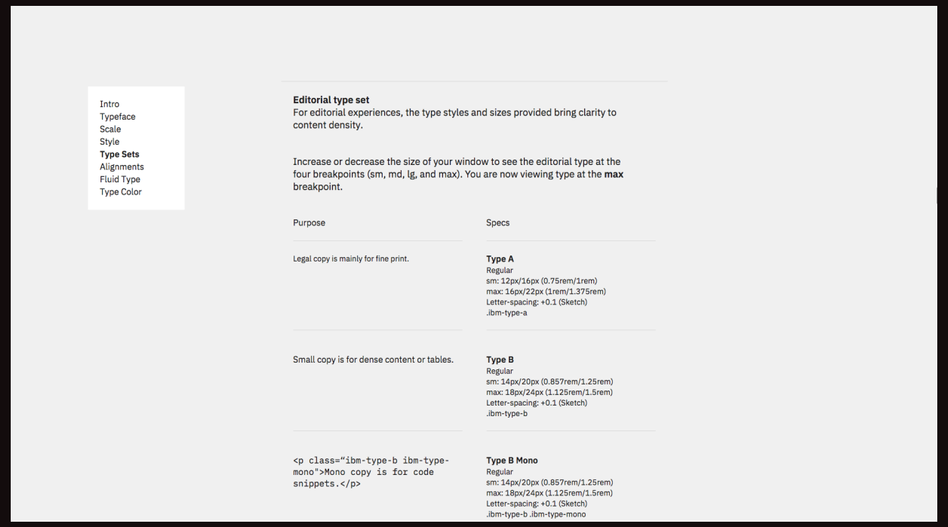
Letterspacing

```
const getLetterSpacing = (size) => {
  let letterSpacing;
  switch (size) {
    case size === 12:
      letterSpacing = 1;
    case size === 14:
      letterSpacing = 1;
    case (size >= 16 && size <= 84):
      letterSpacing = 0;
    case (size >= 92 && size <= 132):
      letterSpacing = -0.05;
    case (size >= 144 && size <= 168):
      letterSpacing = -0.1;
    default:
      letterSpacing = 0;
  }

  return letterSpacing
}
```

Plex Editorial

The designers at IBM advise using a "curated" type set.

For editorial experiences, the type styles and sizes provided bring clarity to content density.



Regular
sm: 15px/21px (0.9375rem/1.375rem)
max: 19px/25px (1.1875rem/1.625rem)
.ibm-type-c .ibm-type-serif

Small headings are for short section headlines.

Type D
SemiBold
sm: 16px/22px (1rem/1.375rem)
max: 20px/26px (1.25rem/1.625rem)
.ibm-type-d

This style is for section headings or high level paragraphs.

Type F
Regular
sm: 20px/26px (1.25rem/1.625rem)
md: 24px/30px (1.5rem/1.875rem)
max: 32px/40px (2rem/2.5rem)
.ibm-type-f

Heading style

Type I
Regular
sm: 28px/36px (1.75rem/2.25rem)
md: 36px/44px (2.25rem/2.75rem)
lg: 42px/50px (2.625rem/3.125rem)
max: 60px/70px (3.75rem/4.375rem)
.ibm-type-i

“Quote.”

Type I
Regular
sm: 27px/36px (1.6875rem/2.25rem)
md: 35px/44px (2.1875rem/2.75rem)
lg: 41px/50px (2.5625rem/3.125rem)
max: 59px/70px (3.6875rem/4.375rem)
.ibm-type-i

The Schema

```
const EditorialTypeSet = {
  A: {
    weight: getWeight(getFontSizeFromScale(0)),
    fontSize: getFontSizeFromScale(0),
    letterSpacing: getLetterSpacing(getFontSizeFromScale(0)),
    lineHeight: getLineHeight(getFontSizeFromScale(0)),
  },
  B: {
    weight: getWeight(getFontSizeFromScale(1)),
    fontSize: getFontSizeFromScale(1),
    letterSpacing: getLetterSpacing(getFontSizeFromScale(1)),
    lineHeight: getLineHeight(getFontSizeFromScale(1)),
  },
  C: {
    weight: getWeight(getFontSizeFromScale(2)),
    fontSize: getFontSizeFromScale(2),
    letterSpacing: getLetterSpacing(getFontSizeFromScale(2)),
    lineHeight: getLineHeight(getFontSizeFromScale(2)),
  },
  ... // everything else
}
```

Goal for our components

- follow our schema
- write once use anywhere
- extensible within guidelines, but not too flexible

Base component

```
import Colors from '@system/colors';

const Base = glamorous.text(
  {
    color: Colors.UI.Text,
    fontFamily: 'Plex Sans'
  },
  props => props.weight && {weight: props.weight},
  props => props.color && {color: props.color},
  props => props.fontSize && {fontSize: props.fontSize},
  props => props.lineHeight && {lineHeight: props.lineHeight},

  // not cross platform
  props => props.letterSpacing && {letterSpacing: props.letterSpacing},
);
```

The Glam Factory

With the `glamorous` component factory we can create our own `glamorous`-like components!

```
const myGlamorousComponentFactory = glamorous(
  MyComponent,
  {displayName: 'MyGlamorousComponent'}
)
```

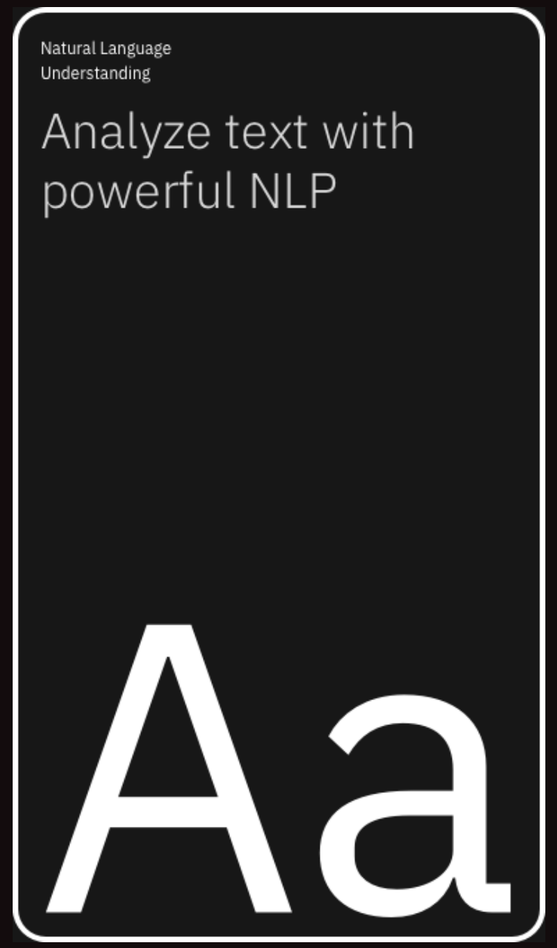
The Plex Factory

```
import {EditorialTypeSet} from './plexRules'
import Base from ‘./text’

// PlexType Factory
// Exports an object of items that can
// be accessed like <PlexType.A>I am text</PlexType.A>

const PlexType = Object.keys(EditorialTypeSet).reduce((plex, key) => {
  plex[key] = glamorous(Base)({
    fontSize: EditorialTypeSet[key];
    weight: EditorialTypeSet[key];
    lineHeight: EditorialTypeSet[key];
    letterSpacing: EditorialTypeSet[key];
  });
  return plex
}, {})

export default PlexType;
```

In Practice

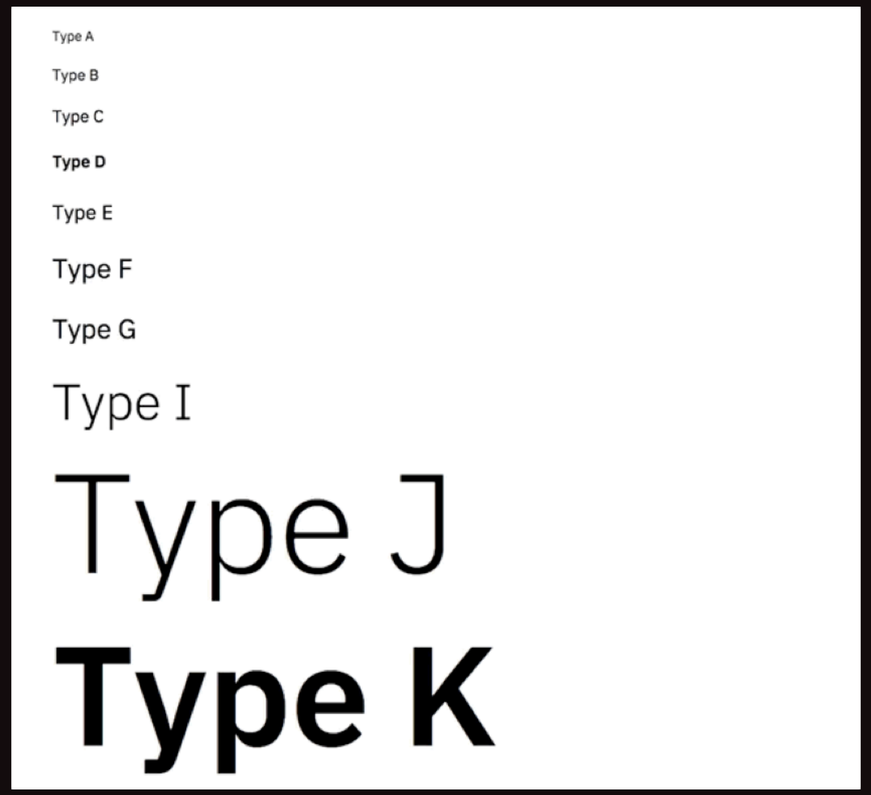
```
export default class App extends Component {
  render() {
    return (
      <Container color={"black"}>
        <PlexType.A>Natural Language Understanding</PlexType.A>
         <PlexType.J>Analyze text with powerful NLP</PlexType.J>
        <PlexType.F>Aa</PlexType.F>
      </Container>
    );
  }
}
```

Output


Natural Language
Understanding
Analyze text with
powerful NLP
Aa

Extending This

IBM has fluid type declarations as well.



Theme Provider

```
class RobinThemeProvider extends React.Component  {
  render() {
    const {theme, layout} = this.props;
    return (
      <ThemeProvider theme={{theme, layout}}>
        {React.Children.only(this.props.children)}
      </ThemeProvider>
    );
  }
}
```

Resources

- https://airbnb.design/building-a-visual-language/
- https://airbnb.design/the-way-we-build/
- http://atomicdesign.bradfrost.com/chapter-1/
- https://medium.com/salesforce-ux/the-lightning-design-system-is-the-next-generation-of-living-style-guides-9addc769c317
- https://uxdesign.cc/designing-design-system-for-complex-

Questions?