Airbnb CSS-in-JavaScript Style Guide

> Una aproximación razonable a CSS-en-JavaScript

Tabla de Contenidos Contents

1. Nombres

> Usa camelCase para objetos claves (por ejemplo. "selectores").

- ¿Por qué? Tenemos acceso a estas claves como propiedades en el objeto styles del componente, por lo que es más conveniente usar camelCase.

```
// mal
{
    'bermuda-triangle': {
    display: 'none',
    },
}

// bien
{
    bermudaTriangle: {
    display: 'none',
    },
}
```

> Usa un guión bajo para los modificadores de otros estilos.

- ¿Por qué? Similar a BEM, esta convención de nomenclatura deja claro que los estilos están destinados a modificar el elemento precedido por el subrayado. Los subrayados no necesitan ser citados, por lo que se prefieren sobre otros caracteres, como guiones.

```
// mal
{
  bruceBanner: {
    color: 'pink',
    transition: 'color 10s',
  },

  bruceBannerTheHulk: {
    color: 'green',
  },
}

// bien
{
  bruceBanner: {
    color: 'pink',
    transition: 'color 10s',
  },

  bruceBanner_theHulk: {
    color: 'green',
```

```
  },
}
```

> Utiliza selectorName_fallback para conjuntos de estilos de fallback.

- ¿Por qué? Al igual que los modificadores, mantener el nombre consistente ayuda a revelar la relación de estos estilos con los estilos que los sustituyen en navegadores más adecuados.

```
// mal
{
  muscles: {
    display: 'flex',
  },

  muscles_sadBears: {
    width: '100%',
  },
}

// bien
{
  muscles: {
    display: 'flex',
  },

  muscles_fallback: {
    width: '100%',
  },
}
```

> Utiliza un selector independiente para conjuntos de estilos de los fallback.

- ¿Por qué? Mantener los estilos de los fallbacks contenidos en un objeto separado aclara su propósito, lo que mejora la legibilidad.

```
// mal
{
  muscles: {
    display: 'flex',
  },

  left: {
    flexGrow: 1,
    display: 'inline-block',
  },

  right: {
    display: 'inline-block',
  },
}

// bien
{
  muscles: {
    display: 'flex',
  },

  left: {
    flexGrow: 1,
  },

  left_fallback: {
```

```
    display: 'inline-block',
  },

  right_fallback: {
    display: 'inline-block',
  },
}
```

> Usa nombres agnósticos de dispositivos (por ejemplo "small", "medium", and "large") para nombrar puntos de ruptura en las 'media query'.

- ¿Por qué? Los nombres comúnmente usados como "movil", "tablet" y "desktop" no coinciden con las características de los dispositivos en el mundo real. El uso de estos nombres establece expectativas equivocadas.

```
// mal
const breakpoints = {
  mobile: '@media (max-width: 639px)',
  tablet: '@media (max-width: 1047px)',
  desktop: '@media (min-width: 1048px)',
};

// bien
const breakpoints = {
  small: '@media (max-width: 639px)',
  medium: '@media (max-width: 1047px)',
  large: '@media (min-width: 1048px)',
};
```

2. Orden

> Define estilos después del componente.

- ¿Por qué? Utilizamos un componente de orden superior para el tema de nuestros estilos, que se utiliza naturalmente después de la definición del componente. Pasar el objeto de estilos directamente a esta función reduce indirección.

```
// mal
const styles = {
  container: {
    display: 'inline-block',
  },
};

function MyComponent({ styles }) {
  return (
    <div {...css(styles.container)}>
      Never doubt that a small group of thoughtful, committed citizens can
      change the world. Indeed, it’s the only thing that ever has.
    </div>
  );
}

export default withStyles(() => styles)(MyComponent);


// bien
function MyComponent({ styles }) {
  return (
    <div {...css(styles.container)}>
      Never doubt that a small group of thoughtful, committed citizens can
      change the world. Indeed, it’s the only thing that ever has.
```

```
    </div>
  );
}

export default withStyles(() => ({
  container: {
    display: 'inline-block',
  },
}))(MyComponent);
```

3. Anidamiento

> Deja una línea en blanco entre bloques adyacentes en el mismo nivel de sangría.

- ¿Por qué? El espacio en blanco mejora la legibilidad y reduce la probabilidad de unir conflictos.

```
// mal
{
  bigBang: {
    display: 'inline-block',
    '::before': {
      content: "''",
    },
  },
  universe: {
    border: 'none',
  },
}

// bien
{
  bigBang: {
    display: 'inline-block',

    '::before': {
      content: "''",
    },
  },

  universe: {
    border: 'none',
  },
}
```

4. En linea

> Utiliza estilos en línea para los estilos que tienen una alta cardinalidad (por ejemplo, utiliza el valor de una propiedad) y no para los estilos que tienen una baja cardinalidad.

- ¿Por qué? Generar hojas de estilo temáticas puede ser costoso, por lo que son mejores para conjuntos discretos de estilos.

```
// mal
export default function MyComponent({ spacing }) {
  return (
    <div style={{ display: 'table', margin: spacing }} />
  );
}

// bien
function MyComponent({ styles, spacing }) {
```

```
    return (
      <div {...css(styles.periodic, { margin: spacing })} />
    );
  }
  export default withStyles(() => ({
    periodic: {
      display: 'table',
    },
  }))(MyComponent);
```

5. Temas

> Use an abstraction layer such as react-with-styles that enables theming. *react-with-styles gives us things like withStyles(), ThemedStyleSheet, and css() which are used in some of the examples in this document.*

> Utiliza una capa de abstracción como [reaccionar con estilos] (https://github.com/airbnb/react-with-styles) que permite el tema. * Reaccionar con estilos nos da cosas como withStyles (), ThemedStyleSheet ycss ()que se utilizan en algunos de los ejemplos de este documento. *

- ¿Por qué? Es útil tener un conjunto de variables compartidas para diseñar sus componentes. El uso de una capa de abstracción lo hace más conveniente. Además, esto puede ayudar a prevenir que sus componentes estén estrechamente acoplados a cualquier implementación subyacente en particular, lo que le da más libertad.

> Define colores solo en temas.

```
```js
// mal
export default withStyles(() => ({
  chuckNorris: {
    color: '#bada55',
  },
}))(MyComponent);

// bien
export default withStyles(({ color }) => ({
  chuckNorris: {
    color: color.badass,
  },
}))(MyComponent);
```
```

> Define fonts solo en temas.

```
```js
// mal
export default withStyles(() => ({
  towerOfPisa: {
    fontStyle: 'italic',
  },
}))(MyComponent);

// bien
export default withStyles(({ font }) => ({
  towerOfPisa: {
    fontStyle: font.italic,
  },
}))(MyComponent);
```
```

Define fonts como conjuntos de estilos relacionados.

```
```js
// mal
export default withStyles(() => ({
  towerOfPisa: {
    fontFamily: 'Italiana, "Times New Roman", serif',
    fontSize: '2em',
    fontStyle: 'italic',
    lineHeight: 1.5,
  },
}))(MyComponent);

// bien
export default withStyles(({ font }) => ({
  towerOfPisa: {
    ...font.italian,
  },
}))(MyComponent);
```
```

Define unidades base de la grid en el tema (ya sea como un valor o una función que tiene un multiplicador). ```js // mal export default withStyles(() => ({ rip: { bottom: '-6912px', // 6 feet }, }))(MyComponent);

```
// bien
export default withStyles(({ units }) => ({
  rip: {
    bottom: units(864), // 6 feet, assuming our unit is 8px
  },
}))(MyComponent);

// good
export default withStyles(({ unit }) => ({
  rip: {
    bottom: 864 * unit, // 6 feet, assuming our unit is 8px
  },
}))(MyComponent);
```
```

Define 'media queries' solo en temas.

```
```js
// mal
export default withStyles(() => ({
  container: {
    width: '100%',

    '@media (max-width: 1047px)': {
      width: '50%',
    },
  },
}))(MyComponent);

// bien
export default withStyles(({ breakpoint }) => ({
  container: {
    width: '100%',

    [breakpoint.medium]: {
```

```
      width: '50%',
    },
  },
}))(MyComponent);
```

Define las propiedades complicadas de fallback en temas.

- ¿Por qué? Muchas implementaciones CSS-in-JavaScript combinan objetos de estilo que hacen que la especificación de fallback para la misma propiedad (por ejemplo, display) sea un poco complicada. Para mantener el enfoque unificado, poner estos fallbacks en el tema.

```
// mal
export default withStyles(() => ({
  .muscles {
    display: 'flex',
  },

  .muscles_fallback {
    'display ': 'table',
  },
}))(MyComponent);

// bien
export default withStyles(({ fallbacks }) => ({
  .muscles {
    display: 'flex',
  },

  .muscles_fallback {
    [fallbacks.display]: 'table',
  },
}))(MyComponent);

// bien
export default withStyles(({ fallback }) => ({
  .muscles {
    display: 'flex',
  },

  .muscles_fallback {
    [fallback('display')]: 'table',
  },
}))(MyComponent);
```

Crea tan pocos temas personalizados como te sea posible. Muchas aplicaciones sólo pueden tener un tema.

Configura el tema personalizado del espacio de nombres en un objeto anidado con una clave única y descriptiva.

```
```js
// mal
ThemedStyleSheet.registerTheme('mySection', {
  mySectionPrimaryColor: 'green',
});

// bien
ThemedStyleSheet.registerTheme('mySection', {
  mySection: {
    primaryColor: 'green',
  },
});
```
```

CSS puns adapted from Saijo George.