Tema 6

React - Foundation

React - Foundation

Foundation es un framework para cualquier dispositivo, medio y accesibilidad. Se basa en un front-end framework que facilita el diseño de sitios web, aplicaciones y correos electrónicos en cualquier dispositivo.

Foundation es semántico, legible, flexible y completamente personalizable. Además, recibe nuevos recursos y actualizaciones continuamente.

Fue diseñado para numerosos dispositivos y navegadores y además probado en ellos. Ha sido construido con Sass/SCSS dando buenas prácticas a diseñadores para el desarrollo rápido.

React - Foundation

Incluye los patrones necesarios más comunes para rápidamente maquetar un sitio. Sus componentes son fácilmente estilizados y sencillos de extender.

Esta herramienta nos proporciona algunos componentes para nuestras aplicaciones con React: botones, grupos de botones, etiquetas, menús, iconos, etc.

Posteriormente veremos algunos de esos componentes y cómo modificarlos para que se visualicen a nuestro gusto.

React - Foundation

Para instalar React-Foundation en nuestro proyecto, deberemos ejecutar el siguiente comando en la línea de órdenes:

```
npm install --save react-foundation-components
```

La forma de usarlo en nuestros ficheros será importándolo donde queramos utilizarlo.

```
import { Button } from 'react-foundation-components/lib/global/button'

//Usar componente importado
```

Ejemplos: Button

La manera de definir botones es la siguiente:

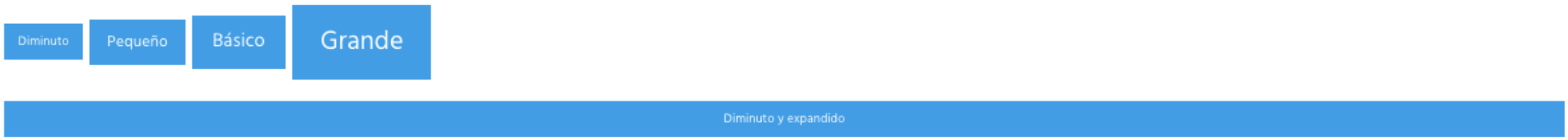
```
<Button color="primary">Boton1</Button>
<Button color="primary">Boton2</Button>
<Button color="success">Boton3</Button>
<Button color="warning">Boton4</Button>
<Button color="alert">Boton5</Button>
```

Mediante este código se mostraría:

Ejemplos: Button

Podemos cambiar el tamaño y el color de los botones añadiendo las correspondientes propiedades.

```
<Button size="tiny">Diminuto</Button>
  <Button size="small">Pequeño</Button>
  <Button>Básico</Button>
  <Button size="large">Grande</Button>
<Button expanded size=“tiny">Diminuto y expandido</Button>
```



Ejemplos: Button

```
<Button color=“primary">Color primario</Button>
<Button color=“secondary">Color secundario</Button>
<Button color=“warning">Color de alerta</Button>
<Button color="warning" disabled>Botón desactivado</Button>
```


```
<Button color="primary" hollow>Color primario</Button>
<Button color="secondary" hollow>Color secundario</Button>
<Button color="alert" hollow>Color de alerta</Button>
```

Ejemplos: Button

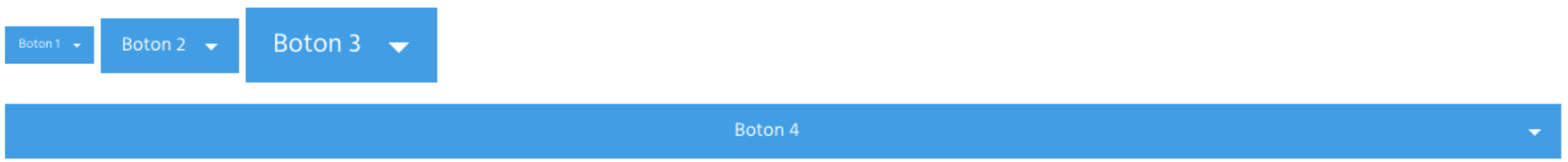
Se puede poner la propiedad ***disabled*** para dar al botón la apariencia de desactivado.

```
<Button color="primary" disabled>Color1</Button>
<Button color="secondary" disabled>Color2</Button>
<Button color="success" disabled>Color3</Button>
<Button color="alert" disabled>Color4</Button>
<Button color="warning" disabled>Color5</Button>
```

Ejemplos: Button

Mediante la propiedad ***dropdown*** se añade una flecha de despliegue al botón.

```
<Button dropdown size=“tiny">Boton 1</Button>
<Button dropdown size=“small">Boton 2</Button>
<Button dropdown>Boton 3</Button>
<Button dropdown expanded>Boton 4</Button>
```



Ejemplos: Button Group

También tenemos predefinidos los grupos de botones.

```
<ButtonGroup>
<Button color="primary">Uno</Button>
<Button color="primary">Dos</Button>
<Button color="primary">Tres</Button>
</ButtonGroup>
```


```
<ButtonGroup>
<ButtonGroup color="alert">
<Button>Harder</Button>
<Button>Better</Button>
<Button>Faster</Button>
<Button>Stronger</Button>
</ButtonGroup>
```

Ejemplos: Switch

Mediante el uso de ***Switchs*** el usuario puede marcar algo como seleccionado o no. Puede haber tanto un único switch como un grupo de ellos.

```
<Switch />
<Switch defaultChecked />
```

```
<RadioSwitch defaultActiveKey="2">
  <Switch eventKey="1" />
  <Switch eventKey="2" />
  <Switch eventKey="3" />
```

Ejemplos: Switch

Para controlar el evento al pulsar sobre el componente se utilizan las propiedades ***checked*** y ***onToggle***.

```
state = { checked: false };

handleToggle = () => this.setState({ checked: !this.state.checked
});

render() {
  return <Switch checked={this.state.checked}
onToggle={this.handleToggle} />;
}
```

Ejemplos: Forms

Sencillas, potentes y versátiles formas para un sistema de layouts. Hay diversos ejemplos de estas formas, como los cuadros de texto o las casillas checkbox. Se pueden importar mediante:

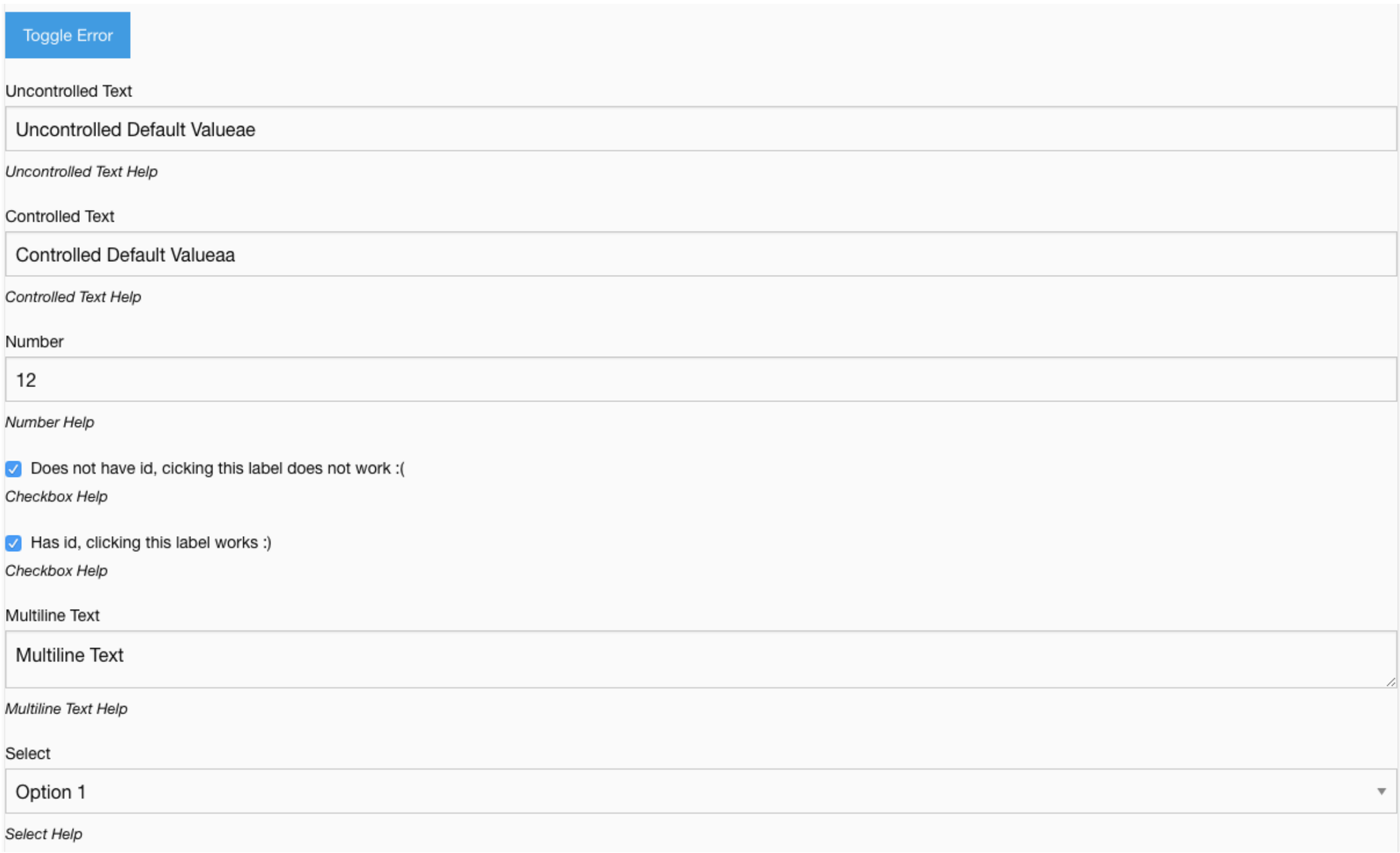
```
import {
  FormField,
  FormFieldInput,
  FormFieldLabel,
  FormFieldError,
  FormFieldInline,
  FormFieldButton,
} from 'react-foundation-components/lib/global/forms';
```

Ejemplos: Forms

El componente **FormField** es un contenedor para todos los demás componentes que forman un campo en una forma. Los componentes básicos son:

- **FormFieldLabel**: tiene el contenido del label. Puede alterar la posición del contenido interno.
- **FormFieldInput**: está envuelto alrededor de etiquetas básicas de HTML.
- **FormFieldError**: contiene información de errores.
- **FormFieldHelp**: contiene información de ayuda.

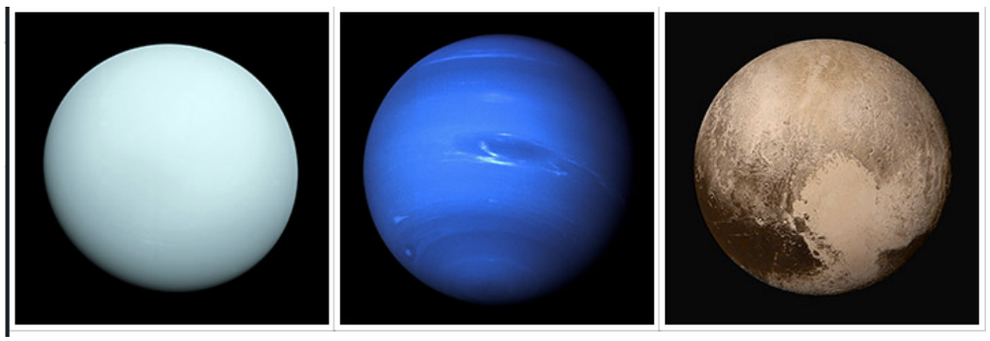
Ejemplos: Forms



Ejemplos: Multimedia

Foundation también nos ofrece la posibilidad de incluir archivos de imagen o vídeo mediante urls.

```
<div>
      <Thumbnail
      alt="Photo of Uranus."
      src="http://foundation.zurb.com/sites/docs/assets/img/thumbnail/01.jpg"
      />
      <Thumbnail
      alt="Photo of Neptune."
      src="http://foundation.zurb.com/sites/docs/assets/img/thumbnail/02.jpg"
      />
      <Thumbnail
      alt="Photo of Pluto."
      src="http://foundation.zurb.com/sites/docs/assets/img/thumbnail/03.jpg"
      />
</div>
```



Ejemplos: Multimedia

```
<FlexVideo
allowFullScreen
frameBorder="0"
height="315"
src=“https://player.vimeo.com/video/168648012"
width="420"
/>
```

Ejemplos: Label

Las etiquetas son útiles para remarcar ciertas secciones o añadir metadatos. Además son personalizables. Para importar este componente, se hace de la manera:

```
import { Label } from 'react-foundation-components/lib/global/label';
```

Y para utilizarlas en el código:


```
<Label color="primary">Primary Label</Label>
<Label color="secondary">Secondary Label</Label>
<Label color="success">Success Label</Label>
<Label color="alert">Alert Label</Label>
<Label color="warning">Warning Label</Label>
```

Ejemplos: Barra de progreso

Las barras de progreso muestran el progreso actual de algún proceso. Pueden utilizarse para indicarle al usuario una estimación de lo que puede tardar una acción. El código es estático, pero se puede manipular para que la barra vaya cambiando según quede mucho o poco por terminar el proceso.

```
import { ProgressBar } from 'react-foundation-components/lib/
global/progress-bar';

<ProgressBar/>
<ProgressBar value={25} />
<ProgressBar value={50} />
<ProgressBar color="success" value={75} />
<ProgressBar value={100} />
<ProgressBar max={200} min={50} value={100} />
```

Ejemplos: Barra de progreso

Se puede añadir texto dentro de la barra proporcionándole una función **labelFormatter**. La función coge cuatro argumentos: el porcentaje calculado, el valor actual, el valor mínimo y el máximo.

```
function complexFormatter(percent, value, min, max) {
  return `percent = ${percent}, value = ${value}, min = ${min}, max = ${max}`;
}

function percentFormatter(percent) {
  const rounded = Math.round(percent * 100);
  return `${rounded}%`;
}

<ProgressBar labelFormatter={complexFormatter} value={50}/>
<ProgressBar labelFormatter={percentFormatter} max={200} min={50} value={150}/
>
```

CSS personalizado

Foundation trae sus propios CSS cuando es instalado. No obstante, si queremos modificar el estilo de nuestros componentes, es recomendable que no se sobrescriba ningún archivo por defecto.

Todos los componentes a los que queramos añadirles un estilo propio y personalizo deben ser escritos en el archivo app.css. Por ejemplo, para añadir un estilo personalizado a un botón habrá que escribir:

```
button, .button {
  background-color: slategray;
  border-color: slategray;
  color: whitesmoke;
}
```