Steps to create ext-web-components-react application using create-react-app with react version 16.8.6:

1. Install create-react-app by running the following command *npx create-react-app ext-web-components-react-app*

2. Run the following command : *cd ext-web-components-react-app*

3. To consume ext-web-components, we need to add following packages to dependencies in package.json
   and run *npm install* to install these dependencies in our project.
   ```
     "@sencha/ext": "~6.7.0",
     "@sencha/ext-modern": "~6.7.0",
     "@sencha/ext-modern-theme-material": "~6.7.0",
     "@sencha/ext-modern-treegrid": "~6.7.0",
     "@sencha/ext-web-components": "~7.0.0",
     "@webcomponents/webcomponentsjs": "^2.2.10",
   ```

4. Add following files to your d*evDependencies* in *package.json* file and run *npm install* to install webpack into your project.

   ```
   "webpack": "~4.31.0",
   "webpack-cli": "~3.3.2",
   "webpack-dev-server": "~3.3.1",
   ```

5. Create a file named *webpack.config.js* in the root of your project to add webpack configuration which is used to bundle your application.

6. Add following to *devDependencies* in *package.json* file and run *npm install*.

   ```
    "portfinder": "^1.0.20",
     "css-loader": "^2.1.0",
     "node-sass": "^4.11.0",
     "react-hot-loader": "^4.8.4",
     "eslint": "^5.16.0",
     "eslint-loader": "^2.1.2",
     "html-loader": "^0.5.5",
     "sass-loader": "^7.1.0",
     "style-loader": "^0.23.1",
     "url-loader": "^2.0.0",
   ```

```
 "html-webpack-plugin": "^3.2.0",
```
This plugin will generate an HTML5 file that will include all webpack bundles in the body in script tag.

```
"base-href-webpack-plugin": "~2.0.0",
```
This is an extension for html-webpack-plugin to programmatically insert or update <base href=” ” />.It requires html-webpack-plugin as a dependency.

```
"@sencha/ext-webpack-plugin": "~7.0.0",
```
This plugin is used to watch file changes during developments of sencha extjs applications.

```
"copy-webpack-plugin": "^5.0.3",
```
This plugin copies individual files or entire directories to the build directory.

7. Change the *start* key of *scripts* object in *package.json* file to

   ```
   "start": "webpack-dev-server --env.browser=true --env.emit=yes --env.verbose=no",
   ```

8. Add the following code snippet in your *webpack.config.js* file to use the above mentioned plugins into your project along with the Ext-Webpack-Plugin.

   ```
   const path = require('path');
   const HtmlWebpackPlugin = require('html-webpack-plugin');
   const { BaseHrefWebpackPlugin } = require('base-href-webpack-plugin');
   const ExtWebpackPlugin = require('@sencha/ext-webpack-plugin');
   const CopyWebpackPlugin = require('copy-webpack-plugin');
   const portfinder = require('portfinder');
   const webpack = require('webpack');

   module.exports = function(env) {
     function get(it, val) {if(env == undefined) {return val;} else if(env[it] == undefined)
   {return val;} else {return env[it];}}

     var profile = get('profile', '');
     var emit = get('emit', 'yes');
     var environment = get('environment', 'development');
     var treeshake = get('treeshake', 'no');
     var browser = get('browser', 'yes');
     var watch = get('watch', 'yes');
     var verbose = get('verbose', 'no');
     var basehref = get('basehref', '/');
   ```

```
    var build_v = get('build_v', '7.0.0.0');

    const isProd = environment === 'production';
    const outputFolder = 'build';
    portfinder.basePort = (env && env.port) || 1962;

    return portfinder.getPortPromise().then(port => {
        const plugins = [
            new HtmlWebpackPlugin({template: 'index.html', hash: true, inject: 'body'}),
            new BaseHrefWebpackPlugin({ baseHref: basehref }),
            new ExtWebpackPlugin({
                framework: 'webcomponents',
                toolkit: 'modern',
                theme: 'theme-material',
                emit: emit,
                port: port,
                packages: [],
                profile: profile,
                environment: environment,
                treeshake: treeshake,
                browser: browser,
                watch: watch,
                verbose: verbose
            }),
            new CopyWebpackPlugin([{
                from:
'../node_modules/@webcomponents/webcomponentsjs/webcomponents-bundle.js',
                to: './webcomponents-bundle.js'
            }]),
            // Debug purposes only, injected via script: npm run-script buildexample --
--env.build_v=<full version here in format maj.min.patch.build>
            new webpack.DefinePlugin({
                BUILD_VERSION: JSON.stringify(build_v)
            })
        ];
        return {
            mode: environment,
            devtool: (environment === 'development') ? 'inline-source-map' : false,
            context: path.join(__dirname, './src'),
            entry: './index.js',
            output: {
                path: path.join(__dirname, outputFolder),
                filename: '[name].js'
},
plugins: plugins,
module: {
    rules: [
        { test: /\.(js|jsx)$/, exclude: /node_modules/,
            use: [
                'babel-loader',
                // 'eslint-loader'
            ]
        },
        { test: /\.(html)$/, use: { loader: 'html-loader' } },
        {
            test: /\.(css|scss)$/,
            use: [
                { loader: 'style-loader' },
                { loader: 'css-loader' },
                { loader: 'sass-loader' }
            ]
        },
        {
             test: /\.(svg|png|jpe?g|gif)(\?\S*)?$/,
             use: [
                  { loader: 'url-loader'  }
            ],
       },

    ]
},
performance: { hints: false },
stats: 'none',
optimization: { noEmitOnErrors: true },
node: false,
devServer: {
    contentBase: outputFolder,
    hot: !isProd,
    historyApiFallback: true,
    host: '0.0.0.0',
    port: port,
    disableHostCheck: false,
    compress: isProd,
    inline:!isProd,
    stats: 'none'
}
```

```
        };
    });
};
```

9. Add the following files to your *devDependencies* in *package.json* file and run *npm install* to install Babel and its plugins

```
    "@babel/core": "^7.3.4",
    "@babel/plugin-proposal-class-properties": "^7.3.4",
    "@babel/plugin-proposal-decorators": "^7.3.0",
    "@babel/plugin-proposal-export-namespace-from": "^7.2.0",
    "@babel/plugin-proposal-function-sent": "^7.2.0",
    "@babel/plugin-proposal-json-strings": "^7.2.0",
    "@babel/plugin-proposal-numeric-separator": "^7.2.0",
    "@babel/plugin-proposal-throw-expressions": "^7.2.0",
    "@babel/plugin-syntax-dynamic-import": "^7.2.0",
    "@babel/plugin-syntax-import-meta": "^7.2.0",
    "@babel/plugin-transform-runtime": "^7.3.4",
    "@babel/preset-env": "^7.3.4",
    "@babel/preset-react": "^7.0.0",
    "@babel/runtime": "^7.3.4",
```

10. Create a new file named *.babelrc* in the root of your project to add babel configuration which is mainly used to convert ES6+ code into backwards compatible version of Javascript in old browsers or environments.

11. Add the following code snippet to your *.babelrc* file to include the plugins installed.

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "modules": false
      }
    ],
    "@babel/preset-react"
  ],
  "plugins": [
    "react-hot-loader/babel",
    "@babel/plugin-transform-runtime",
    "@babel/plugin-syntax-dynamic-import",
    "@babel/plugin-syntax-import-meta",
    "@babel/plugin-proposal-class-properties",
      "@babel/plugin-proposal-json-strings",
      [
        "@babel/plugin-proposal-decorators",
        {
          "legacy": true
        }
      ],
      "@babel/plugin-proposal-function-sent",
      "@babel/plugin-proposal-export-namespace-from",
      "@babel/plugin-proposal-numeric-separator",
      "@babel/plugin-proposal-throw-expressions"
    ],
    "ignore": [
      "build"
    ],
    "env": {
      "test": {
        "presets": [
          "@babel/preset-env",
          "@babel/preset-react"
        ],
        "plugins": [
          "@babel/plugin-syntax-dynamic-import",
          "@babel/plugin-syntax-import-meta",
          "@babel/plugin-proposal-class-properties",
          "@babel/plugin-proposal-json-strings",
          [
            "@babel/plugin-proposal-decorators",
            {
              "legacy": true
            }
          ],
          "@babel/plugin-proposal-function-sent",
          "@babel/plugin-proposal-export-namespace-from",
          "@babel/plugin-proposal-numeric-separator",
          "@babel/plugin-proposal-throw-expressions"
        ]
      }
    }
  }
```

12. Add a new file named *index.html* in the *src* folder and add following code snippet to it

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <script src="webcomponents-bundle.js"></script>
    <title>Sencha Web Components React Boilerplate</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

13. To consume any ext-web-component, we need to import it into the index.js file.

```
import '@sencha/ext-web-components/dist/ext-panel.component';
```

14. Ext-web-components can be now inserted into App.js file.

    Example: Add the following code snippet in the return body of App.js file

```
<ext-panel
     height="100%"
     shadow="true"
     bodyPadding="20px"
   >
     <h1>I am an Ext Panel</h1>
     <p>
       I am HTML inside Ext Panel.
     </p>
   </ext-panel>
```

15. Now run npm start in the root of your directory.

If you don’t want to follow the above mentioned steps, then you can consume the boilerplate code ext-web-components-boilerplate-react and start creating your application by consuming ext-web-components directly.