Rendering dot code blocks

By adding gatsby-remark-graphviz to your Gatsby site, you can create graphs powered by Viz.js by adding `dot` code blocks in your Markdown files:

````
```dot
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```
````

Will render as:

```
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```

A code block without a `dot` or `circo` will not be processed:

```
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```

Adding custom attributes

You can add custom attributes to the resulting SVG:

````
```dot id="small-digraph" style="border: solid 3px tomato; box-shadow: 5px 5px 5px;
padding: 15px; box-sizing: content-box" class="graphviz-figure" data-mydata123
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```
````

Will render as:

```
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```

Don't be shy, go ahead and inspect that SVG and see all the attributes added to it.

Width, height and responsiveness

You can control the layout, spacing and size of the rendered SVG by using Graphviz attributes like this:

````
```dot
digraph graphname {
  graph [size="1.5,1.5"];
  a -> b;
  b -> c;
  a -> c;
}
```
````

This will give you a slightly smaller SVG:

```
digraph graphname {
  graph [size="1.5,1.5"];
  a -> b;
  b -> c;
  a -> c;
}
```

Alternatively, you can overwrite those values by passing custom SVG attributes like this:

````
```dot width="178pt" height="auto"
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```
````

Whoa!

```
digraph graphname {
  a -> b;
  b -> c;
  a -> c;
}
```

By default, gatsby-remark-graphviz is adding the following inline style to every rendered SVG:

```
max-width: 100%;
height: auto;
```

This will make graphs work as expected most of the time - small graphs will remain small and big ones will shrink to fit the parent's box. Graphs can get really big (from Gatsby the docs):

```
digraph graphname {

  node [ style = filled, fillcolor = white ];

  ## Legend

  subgraph cluster_legend {
    label = "Legend";
    gatsby [ label = "Gatsby", width=1 ];
    redux [ label = "redux namespace", shape = box, fillcolor = skyblue, width=1 ];
    cache [ label = "site/.cache/", shape = cylinder, fillcolor = moccasin, width=1
];
    public [ label ="site/public/", shape = cylinder, fillcolor = palegreen, width=1
];
    siteData [ label = "site/external data", shape = cylinder, fillcolor = gray,
width=1 ];

    siteData -> gatsby [ style = invis ];
    gatsby -> redux [ style = invis ] ;
    redux -> cache [ style = invis ];
    cache -> public [ style = invis ];
  }

  ## Source Nodes

  dataSource [ label = "data sources. e.g. file, contentful", shape = cylinder,
fillcolor = gray ];
  sourceNodes [ label = "source nodes" URL = "/docs/node-creation/" ];
  nodes [ label = "nodes", shape = box, fillcolor = skyblue, URL = "/docs/node-
creation/" ];
  nodesTouched [ label = "touchedNodes", shape = box, fillcolor = skyblue, URL =
"/docs/node-creation/#freshstale-nodes" ];
  rootNodeMap [ label = "rootNodeMap", shape = box, fillcolor = skyblue, URL =
"/docs/node-tracking/" ];

  dataSource -> sourceNodes;
  sourceNodes -> nodes;
  sourceNodes -> nodesTouched;
  sourceNodes -> rootNodeMap;

  ## Schema

  pluginResolvers [ label = "plugin resolvers", shape = cylinder, fillcolor = gray,
URL = "/docs/schema-input-gql/#inferring-input-filters-from-plugin-fields" ];
  generateSchema [ label = "generate schema", URL = "/docs/schema-generation/" ];
  schema [ label = "schema\l (inc resolvers)", shape = box, fillcolor = skyblue ];

  nodes -> generateSchema;
  nodes -> schema;
  pluginResolvers -> generateSchema;
  rootNodeMap -> generateSchema;
```

```
  generateSchema -> schema;

  ## Pages

  componentFiles [ label = "React components\l (src/template.js)", shape = cylinder,
fillcolor = gray ];
  createPages [ label = "site.createPages", URL = "/docs/page-creation/" ];
  pages [ label = "pages", shape = box, fillcolor = skyblue ];
  components [ label = "components", shape = box, fillcolor = skyblue ];

  schema -> createPages;
  componentFiles -> createPages;
  createPages -> pages;
  createPages -> components;

  ## Query

  fragments [ label = "query fragments *.js", shape = cylinder, fillcolor = gray ];
  runQueries [ label = "extract and run queries", URL = "/docs/query-behind-the-
scenes/" ];
  componentsWithQueries [ label = "components\l (with queries)", shape = box,
fillcolor = skyblue ];
  queryResults [ label = "JSON result\l /public/static/d/dataPath", shape =
cylinder, fillcolor = palegreen, URL = "/docs/query-execution/#save-query-results-
to-redux-and-disk" ];
  dataPaths [ label = "jsonDataPaths", shape = box, fillcolor = skyblue ];

  fragments -> runQueries;
  schema -> runQueries;
  pages -> runQueries;
  components -> runQueries;
  runQueries -> componentsWithQueries;
  runQueries -> queryResults;
  runQueries -> dataPaths;

  ## Write Pages

  writePages [ label = "writePages", URL = "/docs/write-pages/" ];
  dataJson [ label = "data.json", shape = cylinder, fillcolor = moccasin ];
  asyncRequires [ label = "async-requires.js", shape = cylinder, fillcolor =
moccasin ];
  syncRequires [ label = "sync-requires.js", shape = cylinder, fillcolor = moccasin
];
  pagesJson [ label = "pages.json", shape = cylinder, fillcolor = moccasin ];

  dataPaths -> writePages;
  components -> writePages;
  pages -> writePages;
  writePages -> dataJson;
  writePages -> asyncRequires;
  writePages -> syncRequires;
  writePages -> pagesJson;
```

```
  ## App.js

  appWebpack [ label = "configure webpack\l (`build-javascript`)", URL =
"/docs/production-app/#webpack-config" ];
  productionApp [ label = "production-app.js", shape = cylinder, fillcolor =
moccasin, URL = "/docs/production-app/#production-appjs" ];
  buildJavascript [ label = "build-javascript.js", URL = "/docs/production-app/" ];
  componentChunks [ label = "component chunks\l component---src-blog-[hash].js",
shape = cylinder, fillcolor = palegreen, URL = "/docs/how-code-splitting-works/" ];
  appChunk [ label = "app-[hash].js", shape = cylinder, fillcolor = palegreen ];
  webpackStats [ label = "webpack.stats.json", shape = cylinder, fillcolor =
palegreen, URL = "/docs/how-code-splitting-works/#webpackstatsjson" ];
  chunkMap [ label = "chunk-map.json", shape = cylinder, fillcolor = palegreen, URL
= "/docs/how-code-splitting-works/#chunk-mapjson" ];

  appWebpack -> buildJavascript;
  asyncRequires -> productionApp;
  dataJson -> productionApp;
  productionApp -> buildJavascript;
  buildJavascript -> componentChunks;
  buildJavascript -> appChunk;
  buildJavascript -> webpackStats;
  buildJavascript -> chunkMap;

  queryResults -> componentChunks;

  ## Generate html

  htmlWebpack [ label = "configure webpack\l (`build-html`)", URL = "/docs/html-
generation/#webpack" ];
  staticEntry [ label = "static-entry.js", shape = cylinder, fillcolor = moccasin,
URL = "/docs/html-generation/#static-entryjs" ];
  buildHtml [ label = "build-html.js", URL = "/docs/html-generation/" ];
  pageRenderer [ label = "page-renderer.js", shape = cylinder, fillcolor = palegreen
];
  htmlFiles [ label = "html files\l (index.html)", shape = cylinder, fillcolor =
palegreen ];

  htmlWebpack -> buildHtml;
  syncRequires -> staticEntry;
  dataJson -> staticEntry;
  webpackStats -> staticEntry;
  chunkMap -> staticEntry;
  staticEntry -> buildHtml;
  buildHtml -> pageRenderer;
  pages -> buildHtml;
  pageRenderer -> buildHtml;
  buildHtml -> htmlFiles;
}
```

You can overwrite the `style` attribute if you don't like that behaviour:

````
```dot style=""
digraph graphname {

  node [ style = filled, fillcolor = white ];

  ## Legend

  subgraph cluster_legend {
    ...
```
````

There:

```
digraph graphname {

  node [ style = filled, fillcolor = white ];

  ## Legend

  subgraph cluster_legend {
    label = "Legend";
    gatsby [ label = "Gatsby", width=1 ];
    redux [ label = "redux namespace", shape = box, fillcolor = skyblue, width=1 ];
    cache [ label = "site/.cache/", shape = cylinder, fillcolor = moccasin, width=1
];
    public [ label ="site/public/", shape = cylinder, fillcolor = palegreen, width=1
];
    siteData [ label = "site/external data", shape = cylinder, fillcolor = gray,
width=1 ];

    siteData -> gatsby [ style = invis ];
    gatsby -> redux [ style = invis ] ;
    redux -> cache [ style = invis ];
    cache -> public [ style = invis ];
  }

  ## Source Nodes

  dataSource [ label = "data sources. e.g. file, contentful", shape = cylinder,
fillcolor = gray ];
  sourceNodes [ label = "source nodes" URL = "/docs/node-creation/" ];
  nodes [ label = "nodes", shape = box, fillcolor = skyblue, URL = "/docs/node-
creation/" ];
  nodesTouched [ label = "touchedNodes", shape = box, fillcolor = skyblue, URL =
"/docs/node-creation/#freshstale-nodes" ];
  rootNodeMap [ label = "rootNodeMap", shape = box, fillcolor = skyblue, URL =
"/docs/node-tracking/" ];

  dataSource -> sourceNodes;
```

```
  sourceNodes -> nodes;
  sourceNodes -> nodesTouched;
  sourceNodes -> rootNodeMap;

  ## Schema

  pluginResolvers [ label = "plugin resolvers", shape = cylinder, fillcolor = gray,
URL = "/docs/schema-input-gql/#inferring-input-filters-from-plugin-fields" ];
  generateSchema [ label = "generate schema", URL = "/docs/schema-generation/" ];
  schema [ label = "schema\l (inc resolvers)", shape = box, fillcolor = skyblue ];

  nodes -> generateSchema;
  nodes -> schema;
  pluginResolvers -> generateSchema;
  rootNodeMap -> generateSchema;
  generateSchema -> schema;

  ## Pages

  componentFiles [ label = "React components\l (src/template.js)", shape = cylinder,
fillcolor = gray ];
  createPages [ label = "site.createPages", URL = "/docs/page-creation/" ];
  pages [ label = "pages", shape = box, fillcolor = skyblue ];
  components [ label = "components", shape = box, fillcolor = skyblue ];

  schema -> createPages;
  componentFiles -> createPages;
  createPages -> pages;
  createPages -> components;

  ## Query

  fragments [ label = "query fragments *.js", shape = cylinder, fillcolor = gray ];
  runQueries [ label = "extract and run queries", URL = "/docs/query-behind-the-
scenes/" ];
  componentsWithQueries [ label = "components\l (with queries)", shape = box,
fillcolor = skyblue ];
  queryResults [ label = "JSON result\l /public/static/d/dataPath", shape =
cylinder, fillcolor = palegreen, URL = "/docs/query-execution/#save-query-results-
to-redux-and-disk" ];
  dataPaths [ label = "jsonDataPaths", shape = box, fillcolor = skyblue ];

  fragments -> runQueries;
  schema -> runQueries;
  pages -> runQueries;
  components -> runQueries;
  runQueries -> componentsWithQueries;
  runQueries -> queryResults;
  runQueries -> dataPaths;

  ## Write Pages
```

```
  writePages [ label = "writePages", URL = "/docs/write-pages/" ];
  dataJson [ label = "data.json", shape = cylinder, fillcolor = moccasin ];
  asyncRequires [ label = "async-requires.js", shape = cylinder, fillcolor =
moccasin ];
  syncRequires [ label = "sync-requires.js", shape = cylinder, fillcolor = moccasin
];
  pagesJson [ label = "pages.json", shape = cylinder, fillcolor = moccasin ];

  dataPaths -> writePages;
  components -> writePages;
  pages -> writePages;
  writePages -> dataJson;
  writePages -> asyncRequires;
  writePages -> syncRequires;
  writePages -> pagesJson;

  ## App.js

  appWebpack [ label = "configure webpack\l (`build-javascript`)", URL =
"/docs/production-app/#webpack-config" ];
  productionApp [ label = "production-app.js", shape = cylinder, fillcolor =
moccasin, URL = "/docs/production-app/#production-appjs" ];
  buildJavascript [ label = "build-javascript.js", URL = "/docs/production-app/" ];
  componentChunks [ label = "component chunks\l component---src-blog-[hash].js",
shape = cylinder, fillcolor = palegreen, URL = "/docs/how-code-splitting-works/" ];
  appChunk [ label = "app-[hash].js", shape = cylinder, fillcolor = palegreen ];
  webpackStats [ label = "webpack.stats.json", shape = cylinder, fillcolor =
palegreen, URL = "/docs/how-code-splitting-works/#webpackstatsjson" ];
  chunkMap [ label = "chunk-map.json", shape = cylinder, fillcolor = palegreen, URL
= "/docs/how-code-splitting-works/#chunk-mapjson" ];

  appWebpack -> buildJavascript;
  asyncRequires -> productionApp;
  dataJson -> productionApp;
  productionApp -> buildJavascript;
  buildJavascript -> componentChunks;
  buildJavascript -> appChunk;
  buildJavascript -> webpackStats;
  buildJavascript -> chunkMap;

  queryResults -> componentChunks;

  ## Generate html

  htmlWebpack [ label = "configure webpack\l (`build-html`)", URL = "/docs/html-
generation/#webpack" ];
  staticEntry [ label = "static-entry.js", shape = cylinder, fillcolor = moccasin,
URL = "/docs/html-generation/#static-entryjs" ];
  buildHtml [ label = "build-html.js", URL = "/docs/html-generation/" ];
  pageRenderer [ label = "page-renderer.js", shape = cylinder, fillcolor = palegreen
];
  htmlFiles [ label = "html files\l (index.html)", shape = cylinder, fillcolor =
```

```
palegreen ];

  htmlWebpack -> buildHtml;
  syncRequires -> staticEntry;
  dataJson -> staticEntry;
  webpackStats -> staticEntry;
  chunkMap -> staticEntry;
  staticEntry -> buildHtml;
  buildHtml -> pageRenderer;
  pages -> buildHtml;
  pageRenderer -> buildHtml;
  buildHtml -> htmlFiles;
}
```