Lab. Creating Web Components

In this lab we are going to build a web component using vanilla JavaScript.
We are going to create a date-time component and use it using the HTML5 standards.

Excercise 1. Creating the HTML page

Create a new folder and name it 'Labs', navigate into the Lab folder and
Copy the files from the 'starter_files' folder in the '_labs\Lab 2. Web Components' folder

Install the required components by running the following command:

```
npm install
```

Create an HTML file in the root directory with the following contents:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Web Component Tutorial</title>
    <link rel="import" href="components/date-widget.comp.html">
</head>
<body>

</body>
</html>
```

Create a subdirectory named 'components' and add a HTML file named date-widget.comp.html.
This file holds the component we are going to build.

Download the webcomponents.min.js file from webcomponents.org and
add it to the root of your project and then include it into your index.html
file as shown below, ensuring that it is included before our
component is imported:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Web Component Tutorial</title>
    <script src="webcomponents.min.js"></script>
    <link rel="import" href="components/date-widget.comp.html">
</head>
<body>
Test 123
</body>
</html>
```

> If the file is not downloadable, use the version from the starterfiles

Run the command below to start the server

```
npm start
```

and navigate with your favorite
browser to the url: http://localhost:3000

When all went wel we are looking at a served index.html page.
This page does not do anything special at the moment, but more on that later.

Excercise 2. Create the date-widget component

Open up the date-widget.comp.html in the working folder
and add the template markup code like below:

```
<!-- date-widget.comp.html -->
<template>
    <style>
        @import url(http://fonts.googleapis.com/css?
family=Roboto+Condensed:400,300,700);
        .container {
            background-color: #FFF;
            border-radius: 5px;
            box-shadow: 0 0 5px #dadada;
            position: relative;
            min-height: 100px;
            font-family: 'Roboto Condensed', sans-serif;
            margin: 10px 0;
        }
        .container .left {
            position: absolute;
            left: 0;
            top: 0;
            bottom: 0;
            width: 30%;
            color: #FFF;
            border-radius: 5px 0 0 5px;
            text-align: center;
            padding: 18px 0 0 0;
        }
        .container .left .month {
            line-height: 20px;
            font-weight: 300;
        }
        .container .left .day {
            font-size: 40px
        }
        .container .right {
            margin-left: 30%;
            padding: 10px 10px 10px 15px;
            color: #333;
        }
        .container .right .day-long {
```

```
            font-weight: 300;
            font-size: 18px;
            line-height: 35px;
        }
        .container .right .time {
            font-weight: bold;
            font-size: 35px;
            line-height: 40px;
        }
        /* THEME CODE */
        .container.green .left {
            background-color: #37bc9b;
        }
        .container.green .day-long {
            color: #278b70;
        }
        .container.red .left {
            background-color: #bc2751;
        }
        .container.red .day-long {
            color: #922146;
        }
        .container.blue .left {
            background-color: #356dbc;
        }
        .container.blue .day-long {
            color: #2d5ea3;
        }
        .container.gold .left {
            background-color: #bc9600;
        }
        .container.gold .day-long {
            color: #9a7b00;
        }
    </style>
    <div class="container">
        <div class="left">
            <div class="month"></div>
            <div class="day"></div>
        </div>
        <div class="right">
            <div class="day-long"></div>
            <div class="time"></div>
        </div>
    </div>
</template>
```

Note how this code is just HTML and CSS like we already know.

This markup however, does not work on it's own, we need to
add behavior to this component. Add the following script below the template.

```
<script>
    (function() {
        //Get the contents of the template (_currentScript is available with
webcomponents.js, use currentScript if you don't use this Polyfill)
        var template =
document._currentScript.ownerDocument.querySelector('template');
        //Create a prototype for this component
        var proto = Object.create(HTMLElement.prototype);
        //Set some constants
        var months = ["January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"];
        var days =
['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];
        //Register a createdCallback
        proto.createdCallback = function() {
            var clone = document.importNode(template.content, true);
            this.createShadowRoot().appendChild(clone);
            //Grab the elements from the shadow root
            this.$container = this.shadowRoot.querySelector('.container');
            this.$month = this.shadowRoot.querySelector('.month');
            this.$day = this.shadowRoot.querySelector('.day');
            this.$dayLong = this.shadowRoot.querySelector('.day-long');
            this.$time = this.shadowRoot.querySelector('.time');
            this.updateTheme(this.getAttribute('theme'));
            //Call the draw function initially
            this.draw();
            var that = this;
            //Call the draw function every section to update the time
            setInterval(function(){
                that.draw();
            }, 1000);
        };
        proto.draw = function() {
            this.date = new Date();
            this.$month.innerHTML = months[this.date.getMonth()];
            this.$day.innerHTML = this.date.getDate();
            this.$dayLong.innerHTML = days[this.date.getDay()].toUpperCase();
            this.$time.innerHTML = this.date.toLocaleTimeString();
        };
        //When the element is attached to the DOM populate the select
        proto.attachedCallback = function() {
        };
        proto.updateTheme = function(theme) {
            var val = "green";
            if (["green", "red", "blue", "gold"].indexOf(theme) > -1) {
                val = theme;
            }
            this.$container.className = "container " + val;
        };
        //When the options are updated, re-populate the select
        proto.attributeChangedCallback = function(attrName, oldVal, newVal) {
```

```
            switch (attrName) {
                case "theme":
                    this.updateTheme(newVal);
                    break;
            }
        };
        //Register the element with the document
        document.registerElement('date-widget', {prototype: proto});
    }());
</script>
```

The script is wel documented. Try to understand the code line by line. Note that the code is quite verbose.

Save and close the file.

Change the index.html to hold the newly created component. The file should look like:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Web Component Tutorial</title>
    <script src="webcomponents.min.js"></script>
    <link rel="import" href="components/date-widget.comp.html">
</head>
<body>
    <date-widget></date-widget>
</body>
</html>
```

We just declare an instance of the component using it's registered name from the script we wrote earlier.

Save and close the file.

Stop the server, if the server was already started.
Rerun the server with the following command:

```
npm start
```

If all went well, you should see a component rendered showing the current time and updating each second.

Congrats! You have successfully implemented a Web Compnent.