Introduction to JavaScript Runtime Environments

An introduction to the Node runtime environment and a browser's runtime environment.

What is a Runtime Environment?

A *runtime environment* is where your program will be executed. It determines what global objects your program can access and it can also impact how it runs. This article covers the two JavaScript runtime environments:

1. the runtime environment of a browser (like Chrome, or Firefox)
2. the Node runtime environment

A Browser's Runtime Environment

The most common place where JavaScript code is executed is in a browser. For example, using any text editor, you could create a file on your own computer called **my_website.html** and put the following HTML code inside:

```
<!-- my_website.html -->
<html>
  <body>
    <h1> My Website </h1>
    <script> window.alert('Hello World'); </script>
  </body>
</html>
```

Save your file, then open your favorite browser. Most browsers will allow you to load websites that you have created locally by going to the menu File > Open File > **my_website.html**.

Upon loading, the embedded `<script></script>` will execute and the `window.alert()` method will create a pop-up box in your browser with the text `"Hello World"`. How is this possible? Where did the `window.alert()` method come from and how can it control your browser?

The answer is that you are executing this code in the *browser's runtime environment*. The `window.alert()` method is built into this environment and any program executed in a browser has access to this method. In fact, the `window` object provides access to a huge amount of data and functionality relating to the open browser window beyond just `.alert()`.

> *Try replacing `window.alert()` with `window.prompt()` or `window.confirm()`*

Applications created for and executed in the browser are known as *front-end* applications. For a long time, JavaScript code could only be executed in a browser and was used exclusively for creating front-end applications. In order to create *back-end* applications that could run on a computer WITHOUT a browser, you would need to use other programming languages such as Java or PHP.

The Node Runtime Environment

In 2009, the *Node runtime environment* was created for the purpose of executing JavaScript code without a browser, thus enabling programmers to create *full-stack* (front-end and back-end) applications using only the JavaScript language.

Node is an entirely different runtime environment, meaning that browser-environment data values and functions, like `window.alert()`, can't be used. Instead, the Node runtime environment gives back-end applications access to a variety of features unavailable in a browser, such as access to the server's file system, database, and network.

For example, suppose you created a file called **my-app.js**. We can check to see the directory that this file is located in using the Node runtime environment variable `process`:

```
// my-app.js
console.log(process.env.PWD);
```

> *Notice that we are using `console.log` now instead of `window.alert()` since the `window` object isn't available*

`process` is an object containing data relating to the JavaScript file being executed. `process.env` is an object containing environment variables such as `process.env.PWD` which contains the current working directory (and stands for "**P**rint **W**orking **D**irectory").

To execute the JavaScript code in this file, first make sure that you have set up Node on your computer. Then, open up a terminal and run the following command:

```
$ node my-app.js
/path/to/working/directory
```

The `node` command tells your computer to execute the `my-app.js` file in the Node environment. You can also use the `node` command without a file argument to open up the Node **R**ead-**E**val-**P**rint-**L**oop (REPL):

```
$ node
> process.env.HOME
'/home/ccuser'
```

Summary

A *runtime environment* is where your program will be executed. JavaScript code may be executed in one of two runtime environments:

1. a browser's runtime environment
2. the Node runtime environment

In each of these environments, different data values and functions are available, and these differences help distinguish front-end applications from back-end applications.

- Front-end JavaScript applications are executed in a browser's runtime environment and have access to the `window` object.
- Back-end JavaScript applications are executed in the Node runtime environment and have access to the file system, databases, and networks attached to the server.