Modules – Part I

Level 5 – Section 2

Polluting the Global Namespace

The common solution for modularizing code relies on using **global variables**. This increases the chances of **unexpected side effects** and potential **naming conflicts**.

index.html

```
<!DOCTYPE html>
<body>
  <script src="./jquery.js"></script>
  <script src="./underscore.js"></script>
  <script src="./flash-message.js"></script>
</body>
```

Libraries add to the global namespace

```
let element = $("...").find(...);
let filtered = _.each(...);
flashMessage("Hello");
```

Global variables can cause naming conflicts

Creating Modules

ES2015

Let's create a new JavaScript module for displaying flash messages.

flash-message.js

flash-message.js

```
export default function(message){
  alert(message);
}
```

The export keyword exposes this function to the module system

The default type export is the simplest way to export a function

Importing Modules With Default Export

ES2015

To import modules we use the *import* keyword, specify a new local variable to hold its content, and use the *from* keyword to tell the JavaScript engine where the module can be found.

- flash-message
- app.js

app.js

```
import flashMessage from './flash-message';

flashMessage("Hello");
```

Can be named anything because it's default export

File name with a .js extension must be on the same folder

flash-message.js

```
export default function(message){
  alert(message);
}
```

Running Code From Modules

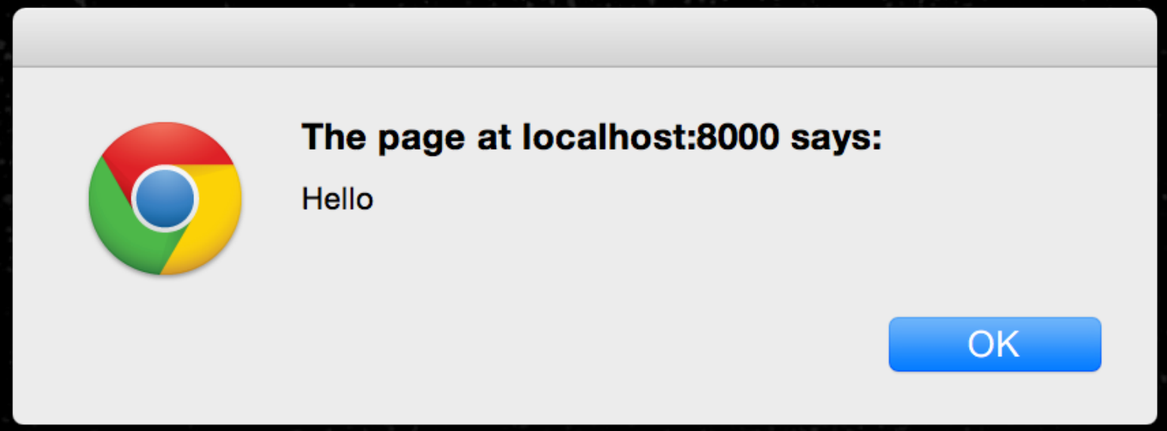
Modules still need to be imported via *<script>*, but **no longer pollute the global namespace**.

- flash-message
- app.
- index.html

index.html

```
<!DOCTYPE html>
<body>
  <script src="./flash-message.js"></script>
  <script src="./app.js"></script>
</body>
```

Not adding to the global namespace



Can’t Default Export Multiple Functions

The *default* type export **limits** the number of functions we can export from a module.

- flash-message.js
- app.
- index.html

flash-message.js

```
export default function(message){
  alert(message);
}

function logMessage(message){
  console.log(message);
}
```

Not available outside this module

Using Named Exports

In order to **export multiple functions** from a single module, we can use the **named** export.

- flash-message.js
- app.
- index.html

No longer using default type export

flash-message.js

```
export function alertMessage(message){
  alert(message);
}

export function logMessage(message){
  console.log(message);
}
```

Importing Named Exports

Functions from **named** exports must be assigned to variables with **the same name** enclosed in curly braces.

- flash-message.js
- app.js
- index.html

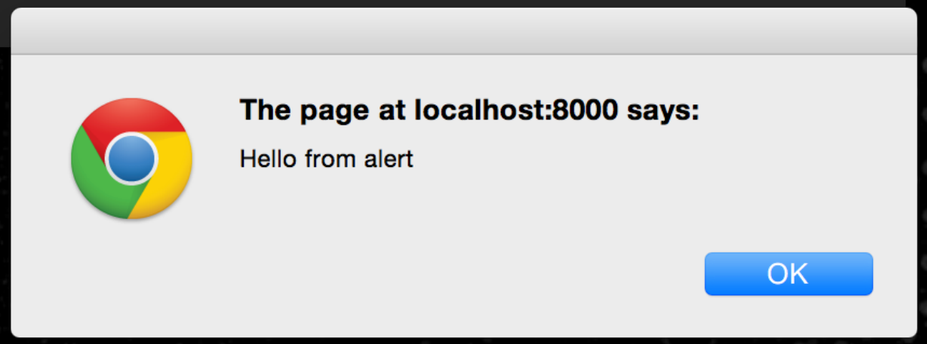
app.js

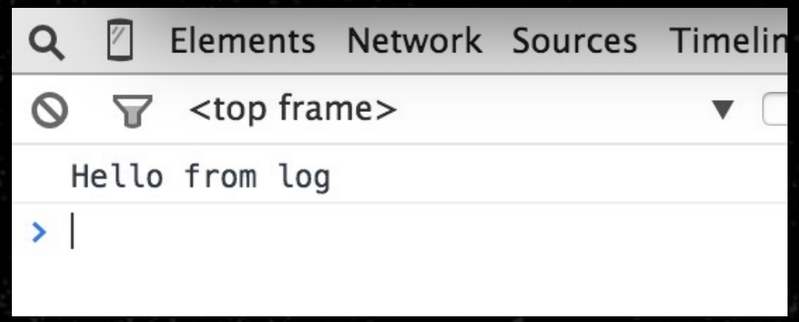
```
import { alertMessage, logMessage } from './flash-message';

alertMessage('Hello from alert');
logMessage('Hello from log');
```

Names must match

flash-message.js

```
export function alertMessage(message){
  alert(message);
}

export function logMessage(message){
  console.log(message);
}
```





Importing a Module as an Object

We can also **import the entire module** as an object and call each function as a **property** from this object.

- flash-message.js
- app.js
- index.html

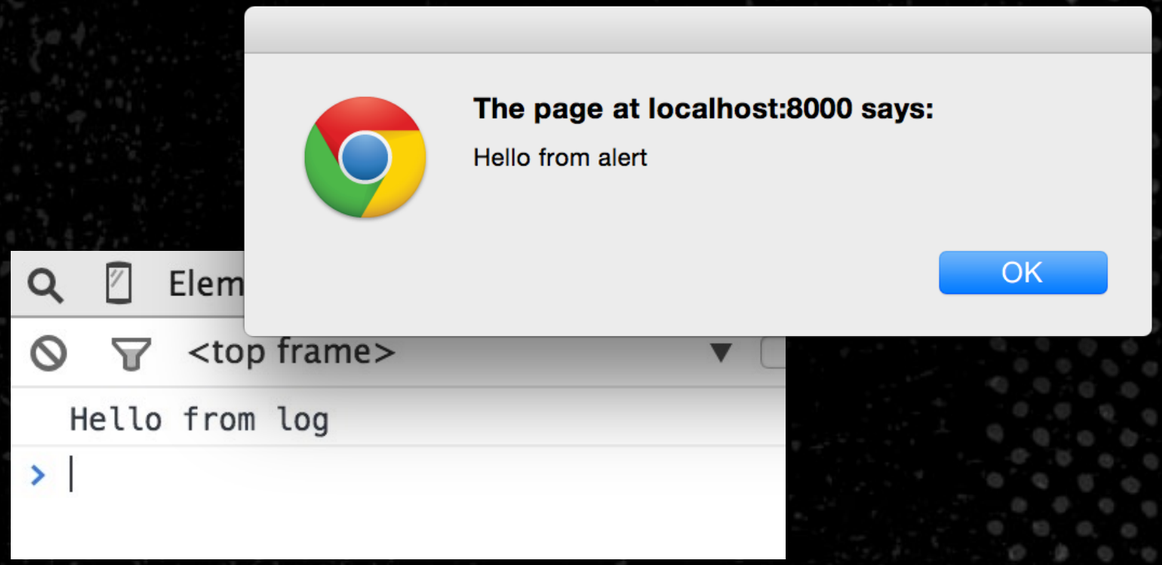
app.js

```
import * as flash from './flash-message';

flash.alertMessage('Hello from alert');
flash.logMessage('Hello from log');
```

functions become object properties

flash-message.js

```
export function alertMessage(message){
  alert(message);
}

export function logMessage(message){
  console.log(message);
}
```



Removing Repeated Export Statements

We are currently calling export statements every time we want to export a function.

- flash-message.js
- app.
- index.html

flash-message.js

```
export function alertMessage(message){
  alert(message);
}

export function logMessage(message){
  console.log(message);
}
```

One export call for each function that we want to expose from our module

Exporting Multiple Functions at Once

We can export multiple functions at once by passing them to *export* inside curly braces.

- flash-message.js
- app.js

export can take multiple function names between curly braces

Imported just like before

flash-message.js

```
function alertMessage(message){
  alert(message);
}

function logMessage(message){
  console.log(message);
}

export { alertMessage, logMessage }
```

app.js

```
import { alertMessage, logMessage } from './flash-message';

alertMessage('Hello from alert');
logMessage('Hello from log');
```