Modules – Part II

Level 5 – Section 3

Extracting Hardcoded Constants

Redefining constants across our application is **unnecessary repetition** and can lead to **bugs**.

We define our constants here...

load-profiles.js

```
function loadProfiles(userNames){
  const MAX_USERS = 3;
  if(userNames.length > MAX_USERS){
    //...
  }

  const MAX_REPLIES = 3;
  if(someElement > MAX_REPLIES){
    //...
  }
}

export { loadProfiles }
```

...and then we define them again on different places

list-replies.js

```
function listReplies(replies=[]){
  const MAX_REPLIES = 3;
  if(replies.length > MAX_REPLIES){
    //...
  }
}
export { listReplies }
```

display-watchers.js

```
function displayWatchers(watchers=[]){
  const MAX_USERS = 3;
  if(watchers.length > MAX_USERS){
    //...
  }
}
export { displayWatchers }
```

Exporting Constants

Placing constants on their own module allows them to be reused across other modules and **hides implementation details** (a.k.a., **encapsulation**).

constants.js

constants.js ✓

```
export const MAX_USERS = 3;
export const MAX_REPLIES = 3;
```

Either syntax works

constants.js ✓

```
const MAX_USERS = 3;
const MAX_REPLIES = 3;

export { MAX_USERS, MAX_REPLIES };
```

How to Import Constants

To *import* constants, we can use the exact same syntax for importing functions.

Details are encapsulated inside of the constants module

- constants
- load-profiles.js

load-profiles.js ✓

```
import { MAX_REPLIES, MAX_USERS } from './constants';

function loadProfiles(userNames){

  if(userNames.length > MAX_USERS){
    //...
  }

  if(someElement > MAX_REPLIES){
    //...
  }
}
```

Importing Constants

We can now import and use our constants from other places in our application.

- constants
- load-profiles
- list-replies.js
- display-watchers.js

list-replies.js ✓

```
import { MAX_REPLIES } from './constants';

function listReplies(replies = []){
  if(replies.length > MAX_REPLIES){
    //...
  }
}
```

display-watchers.js ✓

```
import { MAX_USERS } from './constants';

function displayWatchers(watchers = []){
  if(watchers.length > MAX_USERS){
    //...
  }
}
```

Exporting Class Modules With Default Export

Classes can also be exported from modules using the same syntax as functions. Instead of 2 individual functions, we now have **2 instance methods** that are part of a class.

flash-message.js

default allows this class to be set to any variable name once it's imported

flash-message.js

```
export default class FlashMessage {

  constructor(message){
    this.message = message;
  }

  renderAlert(){
    alert(`${this.message} from alert`);
  }

  renderLog(){
    console.log(`${this.message} from log`);
  }
}
```

Using Class Modules With Default Export

Imported classes are assigned to a variable using *import* and can then be used to create **new instances**.

- flash-message.js
- app.js

flash-message.js

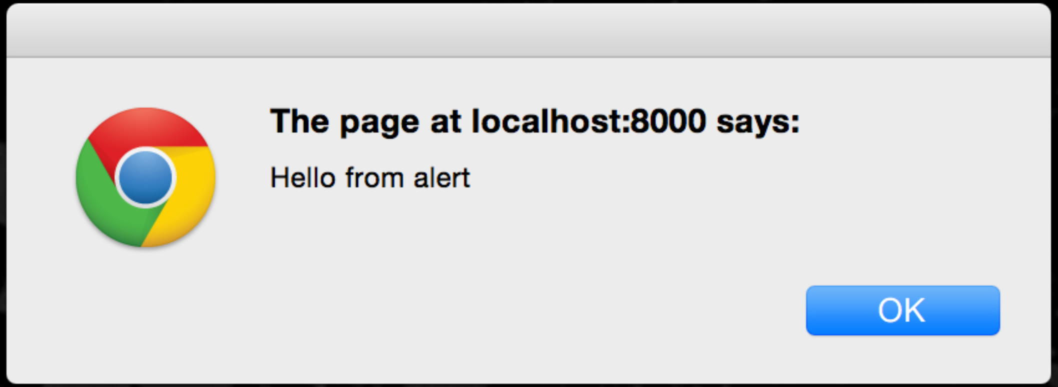
```
export default class FlashMessage {
  // ...
}
```

Exporting a class, so F is capitalized

app.js

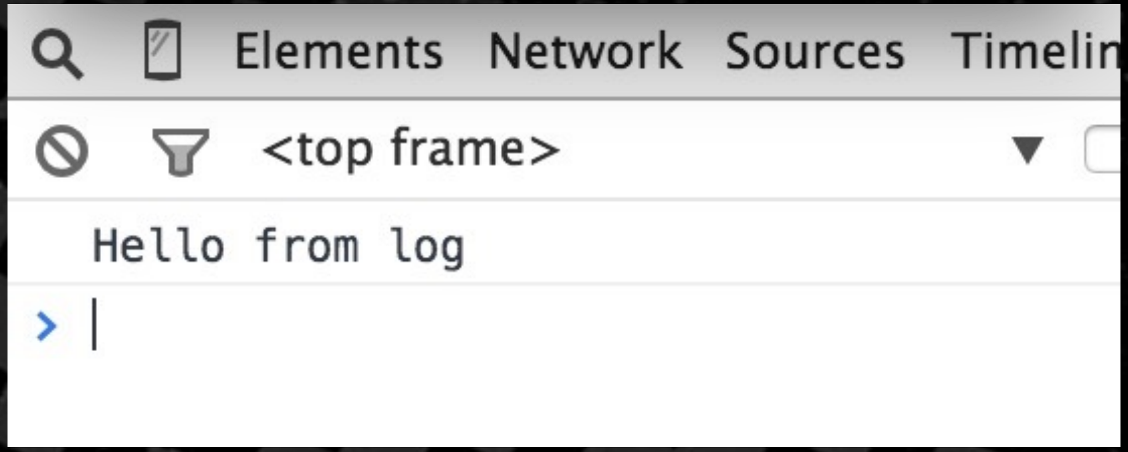
```
import FlashMessage from './flash-message';

let flash = new FlashMessage("Hello");
flash.renderAlert();
flash.renderLog();
```

Creates instance and calls instance methods



The page at localhost:8000 says:

Hello from alert



```
Hello from log
```

Using Class Modules With Named Export

Another way to export classes is to first define them, and then use the *export* statement with the class name inside curly braces.

flash-message.js

Plain old JavaScript class declaration

```
class FlashMessage {

  constructor(message){
    this.message = message;
  }

  renderAlert(){
    alert(`${this.message} from alert`);
  }
  renderLog(){
    console.log(`${this.message} from log`);
  }
}

export { FlashMessage }
```

Exports class to the outside world

Using Class Modules With Named Export

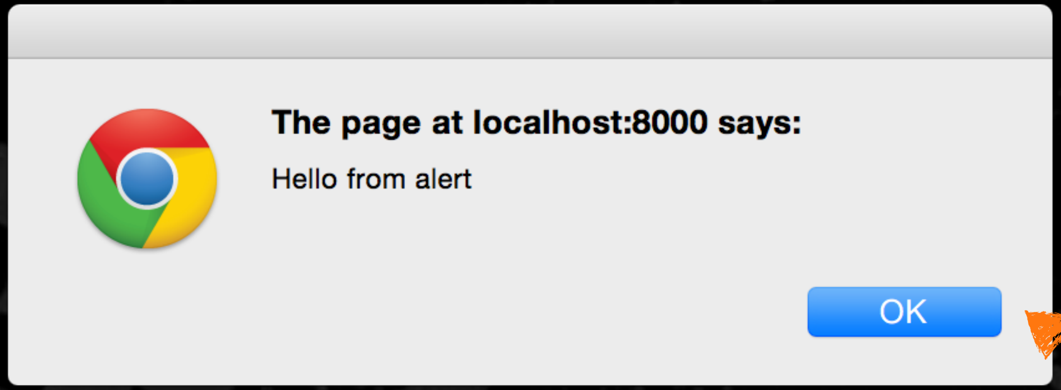
When using **named export**, the script that loads the module needs to assign it to a variable with **the same name as the class**.

- flash-message.js
- app.js

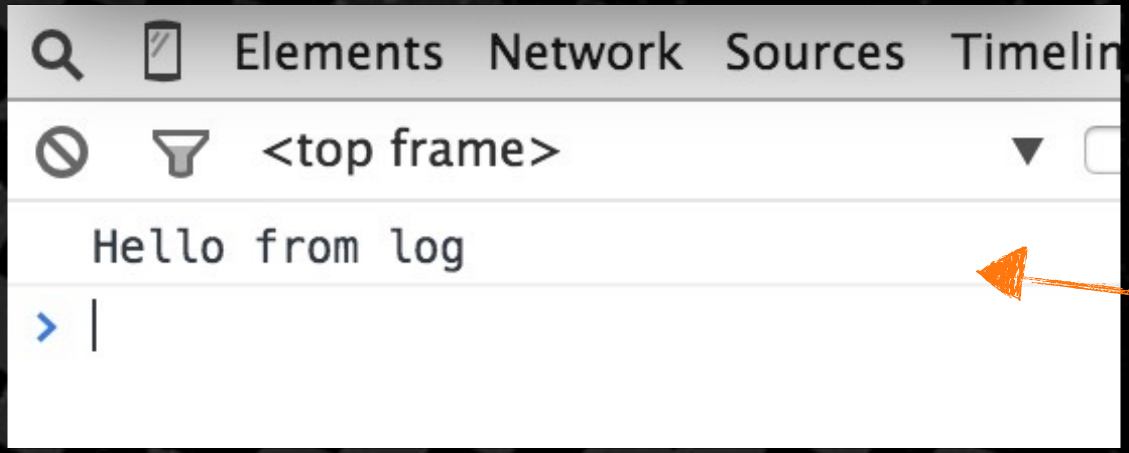
```
class FlashMessage {
  //...
}

export { FlashMessage }
```

Names must match

```
import { FlashMessage } from './flash-message';

let flash = new FlashMessage("Hello");
flash.renderAlert();
flash.renderLog();
```


The page at localhost:8000 says:

Hello from alert



```
Hello from log
> |
```

Same result as before