

# Declarations with *let*

Level 1 – Section 1

# Forum Web App

The screenshot shows a forum application window titled "The Forum" with the URL "localhost:8000". At the top, there's a logo with the word "FORUM" in a stylized font. Below it, three threads are listed in a vertical stack:

- IN HYBRID MOMENTS, GIVE ME A MOMENT**  
LAST REPLY ON JULY 15, 1997
- SOUND SYSTEM GONNA BRING ME BACK UP**  
LAST REPLY ON JULY 1, 1991
- WHEN I'M OUT WALKIN', I STRUT MY STUFF**  
LAST REPLY ON APRIL 19, 1983

The screenshot shows a forum application window titled "The Forum" with the URL "localhost:8000". At the top, there's a logo with the word "FORUM" in a stylized font. Below it, a single thread is displayed:

### WHAT'S THE BETTER MISFITS ALBUM?

1. STATIC AGE
2. COLLECTION I

Below the list, a user named CARLOS S. has posted a reply:

CARLOS S. 1. COLLECTION I, GOTTA GO WITH THE ORIGINAL SOUND

At the bottom, another user named SAMANTHA A. has a partially visible reply:

SAMANTHA A. 2. AMERICAN PSYCHO IT'S SO CATCHY

The screenshot shows a detailed view of a forum thread titled "IN HYBRID MOMENTS, GIVE ME A MOMENT". The thread has three replies:

- CARLOS S.** AM I FOLLOWING ALL THE RIGHT LEADS? OR AM I ABOUT TO GET LOST IN SPACE? WHEN MY TIME COMES THEY'LL WRITE MY DESTINY. WILL YOU TAKE THIS RIDE?
- SAMANTHA A.** GOOD FEELING, WON'T YOU STAY WITH ME JUST A LITTLE LONGER? IT ALWAYS SEEMS LIKE YOU'RE LEAVING WHEN I NEED YOU HERE JUST A LITTLE LONGER
- JON F.** STATIC PULSE INSIDE OF MUSIC BRINGING US ESCAPE. IT'S ALWAYS TEMPORARY CHANGING

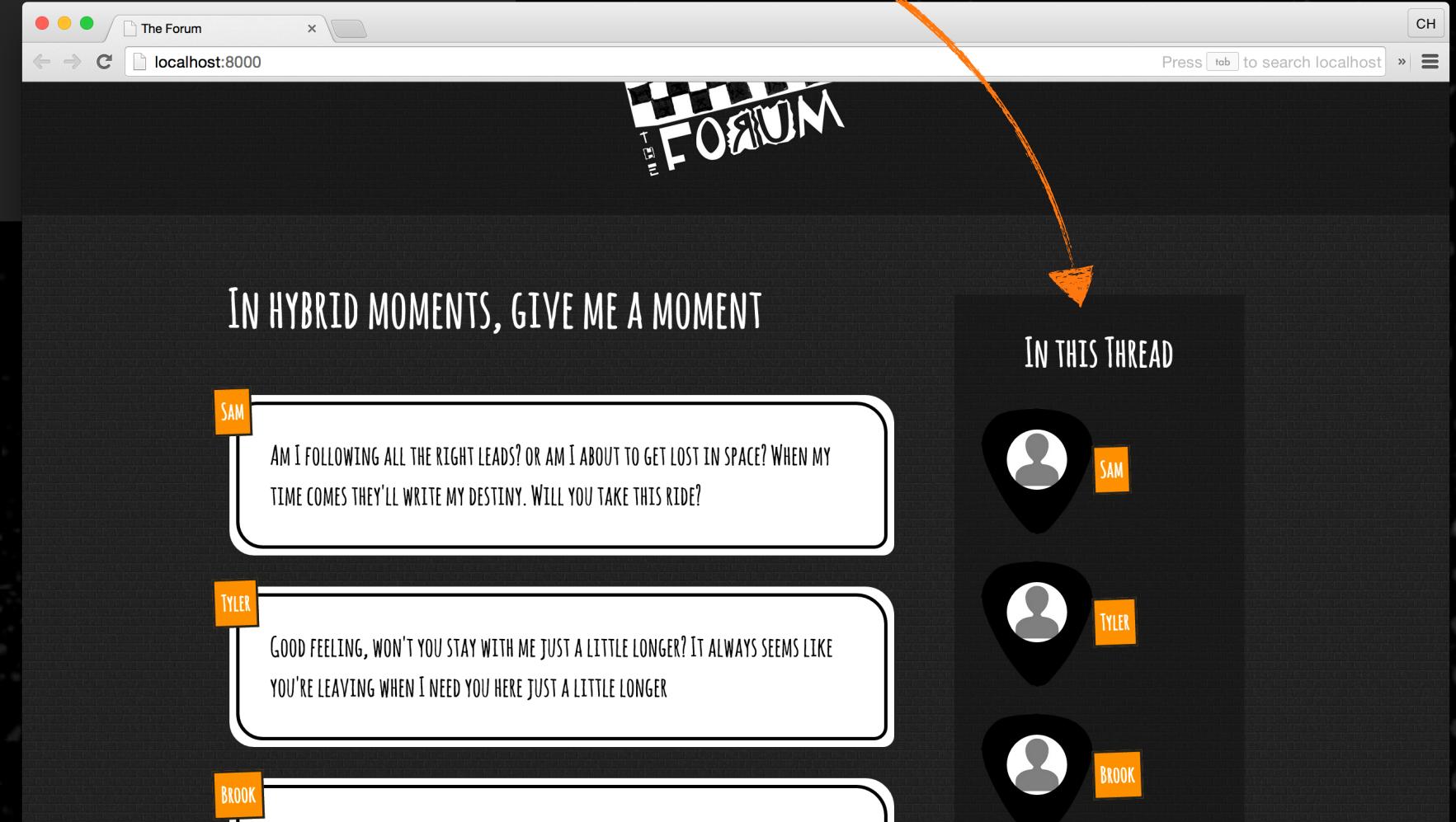
On the right side, there are profile icons for CARLOS S., SAMANTHA A., and JON F. with their names labeled "IN THIS THREAD".

Together, we'll build features that are part of a web forum!

# Loading Profiles

The `loadProfiles` function takes an array of users and adds their profile to the sidebar.

```
<!DOCTYPE html>
//...
<script src="./load-profiles.js">
<script>
  loadProfiles(["Sam", "Tyler", "Brook"]);
</script>
</body>
</html>
```



# The loadProfiles Function

This function displays a different message according to the number of arguments.

```
function loadProfiles(userNames){  
  if(userNames.length > 3){  
    var loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
  }else{  
    var flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
  //... fetch names and build sidebar  
}
```

Might take a while to process,  
so a loading message is displayed

Shouldn't take that long, so  
we display a different message

# How loadProfiles Is Executed

```
loadProfiles(["Sam", "Tyler", "Brook", "Alex"]);
```

1. Executes the if block

```
function loadProfiles(userNames){  
  if(userNames.length > 3){  
    var loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
  } else{  
    var flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
  //...  
}
```

2. The loadingMessage variable is declared and assigned a value

3. The displaySpinner function is called

The else block is not executed

# Unexpected Behavior With var

```
function loadProfiles(userNames){  
  if(userNames.length > 3){  
    var loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
    console.log(flashMessage); → > undefined  
  } else{  
    var flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
  console.log(flashMessage); → > undefined  
}
```

flashMessage is never declared...

This indicates a  
console output

...but outputs undefined and does not generate error (?!)

Why no ReferenceError ?



# Understanding Hoisting

Prior to executing our code, JavaScript moves `var` declarations all the way up to the top of the scope. This is known as **hoisting**.

```
function loadProfiles(userNames){  
  var loadingMessage, flashMessage; // ← Automatically moved here by the JavaScript runtime  
  
  if(userNames.length > 3){  
    var loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
  }else{  
    var flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
  
  //....  
}
```

# Declaring Variables With let

*let* variables are scoped to the nearest block and are NOT hoisted.  
(A block is any code section within **curly braces**, like *if*, *else*, *for*, *while*, etc.)

```
function loadProfiles(userNames){  
  if(userNames.length > 3){  
    let loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
  }else{  
    let flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
  //....  
}
```

Not visible outside if block

Not visible outside else block

# Expected Behavior With `let`

Using `let`, variables are “trapped” inside their respective `if` and `else` blocks.

```
loadProfiles(["Sam", "Tyler", "Brook", "Alex"]);
```

```
function loadProfiles(userNames){  
  if(userNames.length > 3){  
    let loadingMessage = "This might take a while...";  
    _displaySpinner(loadingMessage);  
  }  
  else{  
    let flashMessage = "Loading Profiles";  
    _displayFlash(flashMessage);  
  }  
}
```

```
console.log(flashMessage);
```

→ > ReferenceError: flashMessage is not defined

```
}else{
```

```
  let flashMessage = "Loading Profiles";  
  _displayFlash(flashMessage);
```

Expected behavior, similar to  
other programming languages!



```
}
```

```
console.log(flashMessage);
```

→ > ReferenceError: flashMessage is not defined

```
}
```