WHAT CAN WE DO TO ENSURE THE CORRECT VALUE?

Several options exist for timing closures correctly

```
function assignTorpedo ( name, passengerArray ){

        for (var i = 0; i<passengerArray.length; i++) {
            if (passengerArray[i] == name) {
                return function ( ) {
                alert("Ahoy, " + name + "!\n" +
                      "Man your post at Torpedo #" + (i+1) + "!");
                };
            }
        }
}
```

Now the function will be immediately returned when the right name is found, locking i in place.

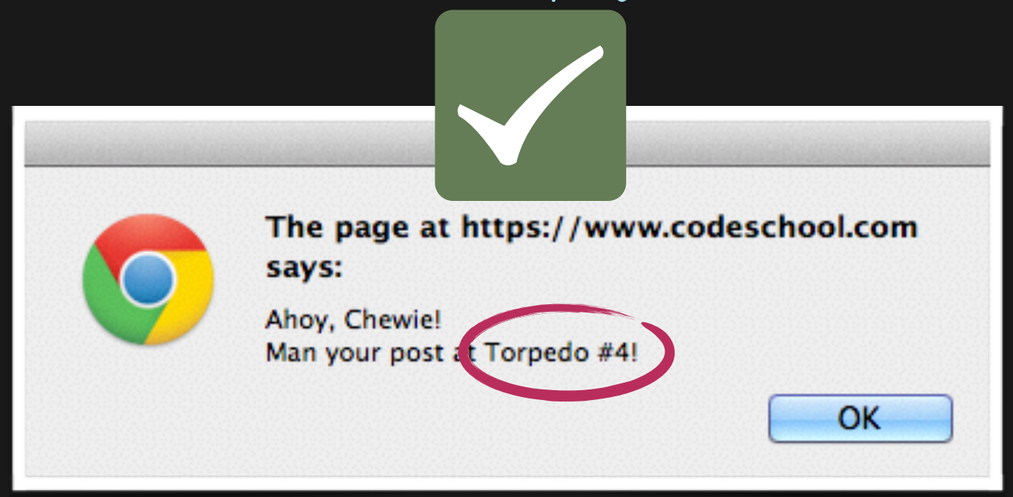
```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

WHAT CAN WE DO TO ENSURE THE CORRECT VALUE?

Several options exist for timing closures correctly

```
function assignTorpedo ( name, passengerArray ){

        for (var i = 0; i<passengerArray.length; i++) {
            if (passengerArray[i] == name) {
                return function ( ) {
                alert("Ahoy, " + name + "!\n" +
                        "Man your post at Torpedo #"
                };
            }
        }

}
```

An immediate return has the expected effect, because `i` is not allowed to progress!




```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

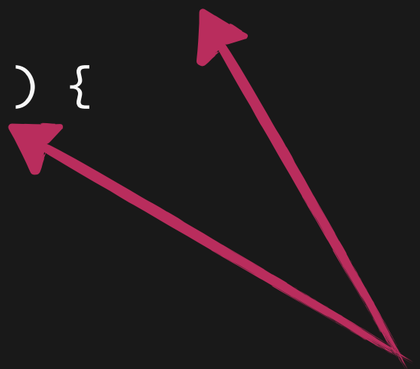
```
var giveAssignment = assignTorpedo("Chewie", subPassengers);
```

```
giveAssignment();
```

WHAT CAN WE DO TO ENSURE THE CORRECT VALUE?

We could also design the torpedo assigners a bit more like our ticket makers

```
function makeTorpedoAssigner ( passengerArray ) {

        return function ( name ) {









        };

}
```



This time our external function will only take in the passengerArray, and we'll let the returned function deal with a specific name.

```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

WHAT CAN WE DO TO ENSURE THE CORRECT VALUE?

We could also design the torpedo assigners a bit more like our ticket makers

```
function makeTorpedoAssigner ( passengerArray ) {

        return function ( name ) {
           for (var i = 0; i<passengerArray.length; i++) {


           }
        };

}
```

At this point, whatever passengerArray got passed in to makeTorpedoAssigner will be bound into the closure. Parameters are part of the environment, too!

```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

WHAT CAN WE DO TO ENSURE THE CORRECT VALUE?

We could also design the torpedo assigners a bit more like our ticket makers

```
function makeTorpedoAssigner ( passengerArray ) {

        return function ( name ) {
           for (var i = 0; i<passengerArray.length; i++) {
              if (passengerArray[i] == name) {
                 alert("Ahoy, " + name + "!\n" +
                       "Man your post at Torpedo #" + (i+1) + "!");
              }
           }
        };

}
```

Since we've put the loop inside the returned function, `i` will come directly from that local scope.

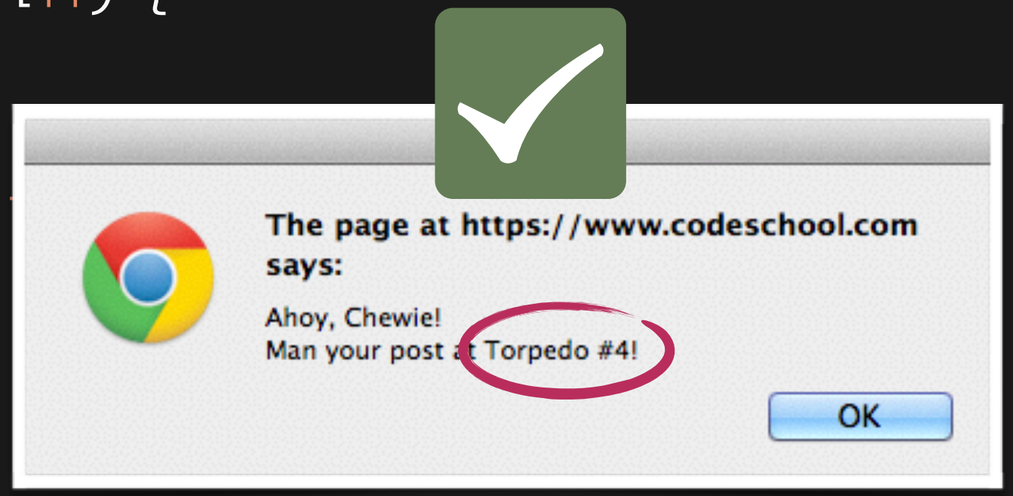
The only closed variable from the external scope is `passengerArray`, which never changes.

```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

NOW WE CAN PASS OUT TORPEDOES LIKE CANDY

TIE Fighter, dead ahead!...Er, underwater...

```
function makeTorpedoAssigner ( passengerArray ) {

        return function ( name ) {
           for (var i = 0; i<passengerArray.length; i++) {
              if (passengerArray[i] == name) {
                 alert("Ahoy, " + name + "!\n" +
                       "Man your post at Torpedo #" +
              }
           }
        };

}
```




```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", "C-3PO", "Boba"];
```

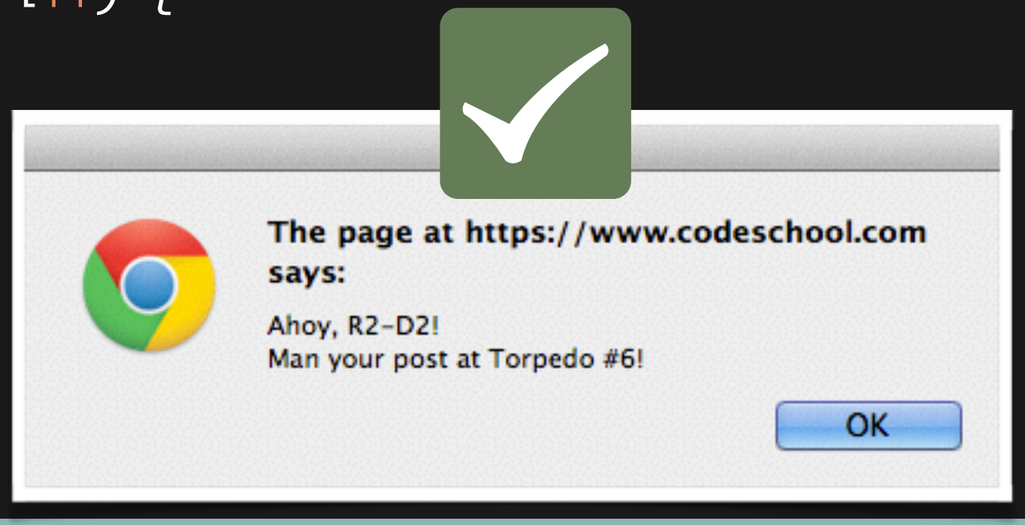
```
var getTorpedoFor = makeTorpedoAssigner(subPassengers);
```

```
getTorpedoFor("Chewie");
```

NOW WE CAN PASS OUT TORPEDOES LIKE CANDY

TIE Fighter, dead ahead!...Er, underwater...

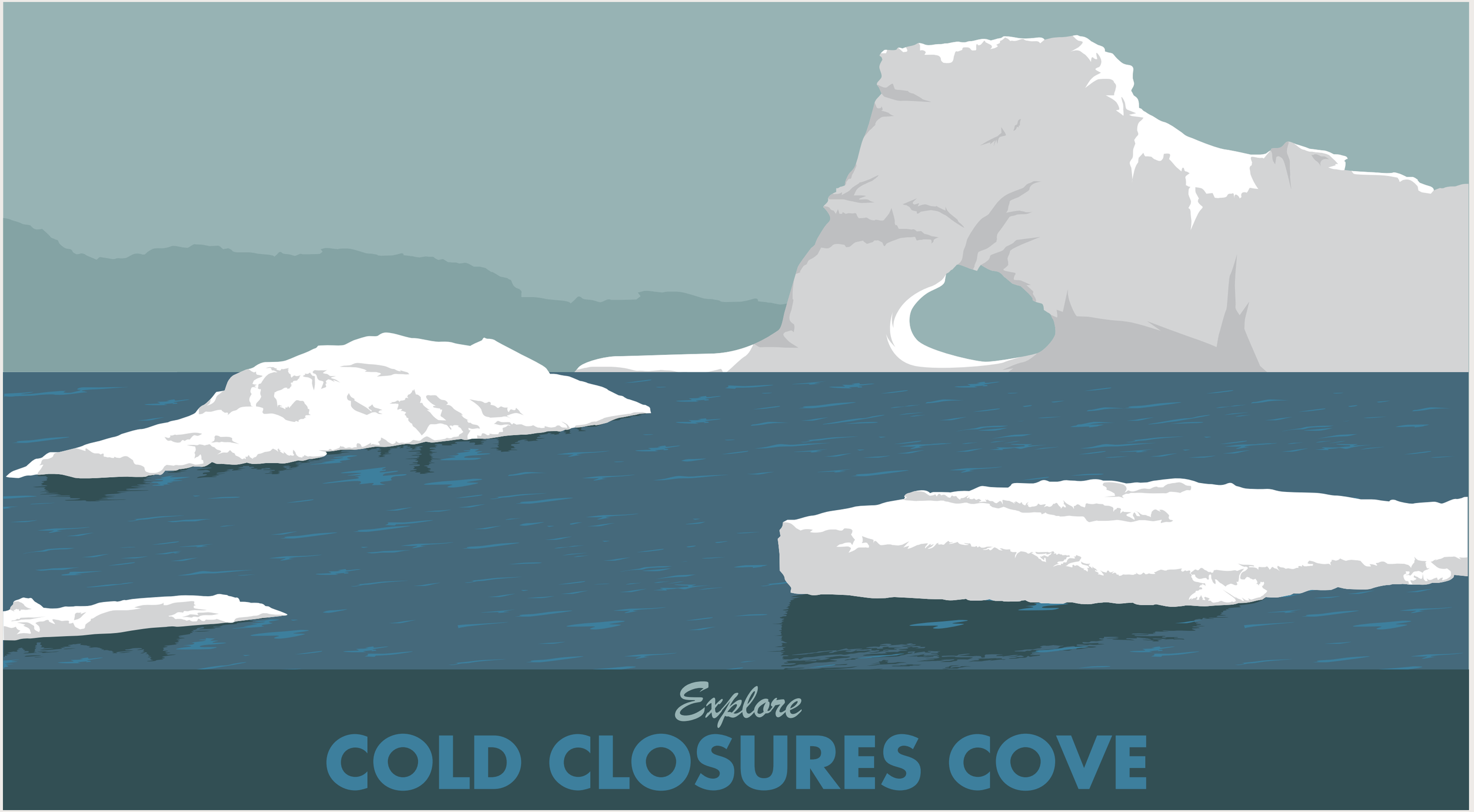
```
function makeTorpedoAssigner ( passengerArray ) {

        return function ( name ) {
           for (var i = 0; i<passengerArray.length; i++) {
              if (passengerArray[i] == name) {
                 alert("Ahoy, " + name + "!\n" +
                       "Man your post at Torpedo #"
              }
           }
        };

}
```




```
var subPassengers = ["Luke", "Leia", "Han", "Chewie", "Yoda", "R2-D2", “C-3PO”, “Boba”];
```

```
var getTorpedoFor = makeTorpedoAssigner(subPassengers);
```

```
getTorpedoFor( "R2-D2" );
```


Explore
COLD CLOSURES COVE

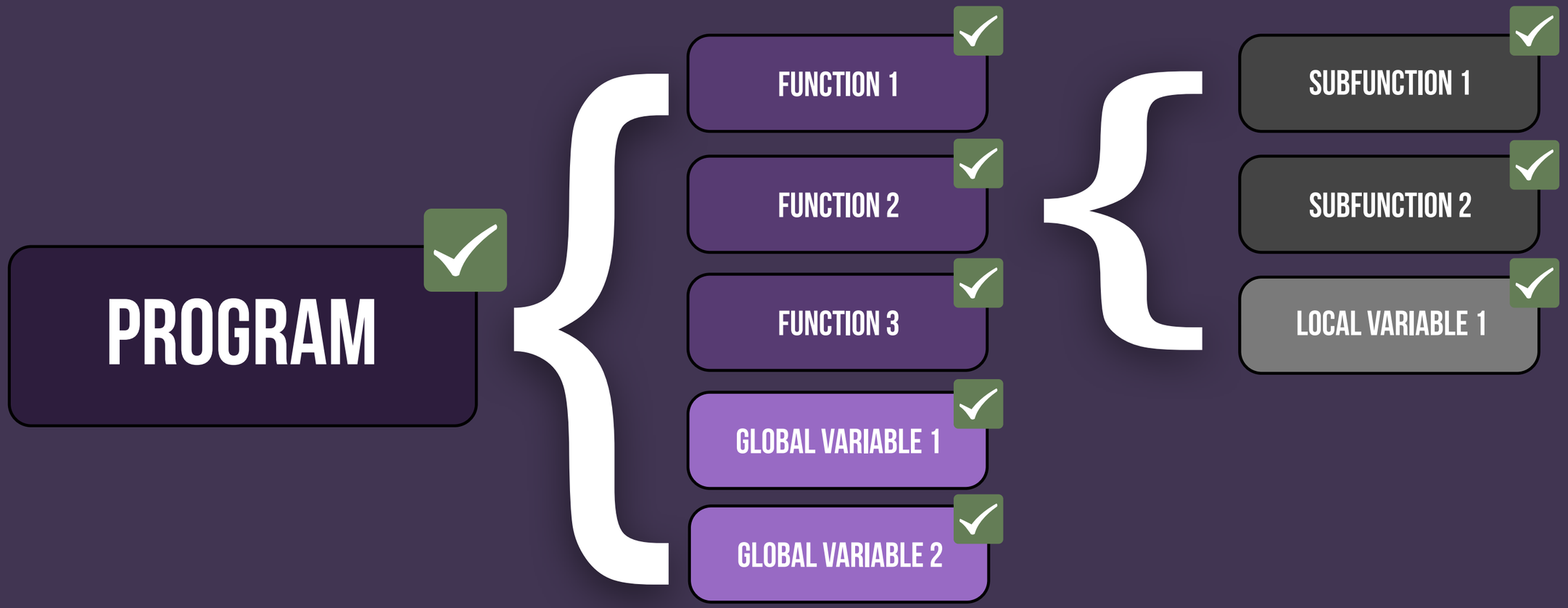

Climb
THE HOISTING HILLS

LEVEL 3

THE HOISTING HILLS

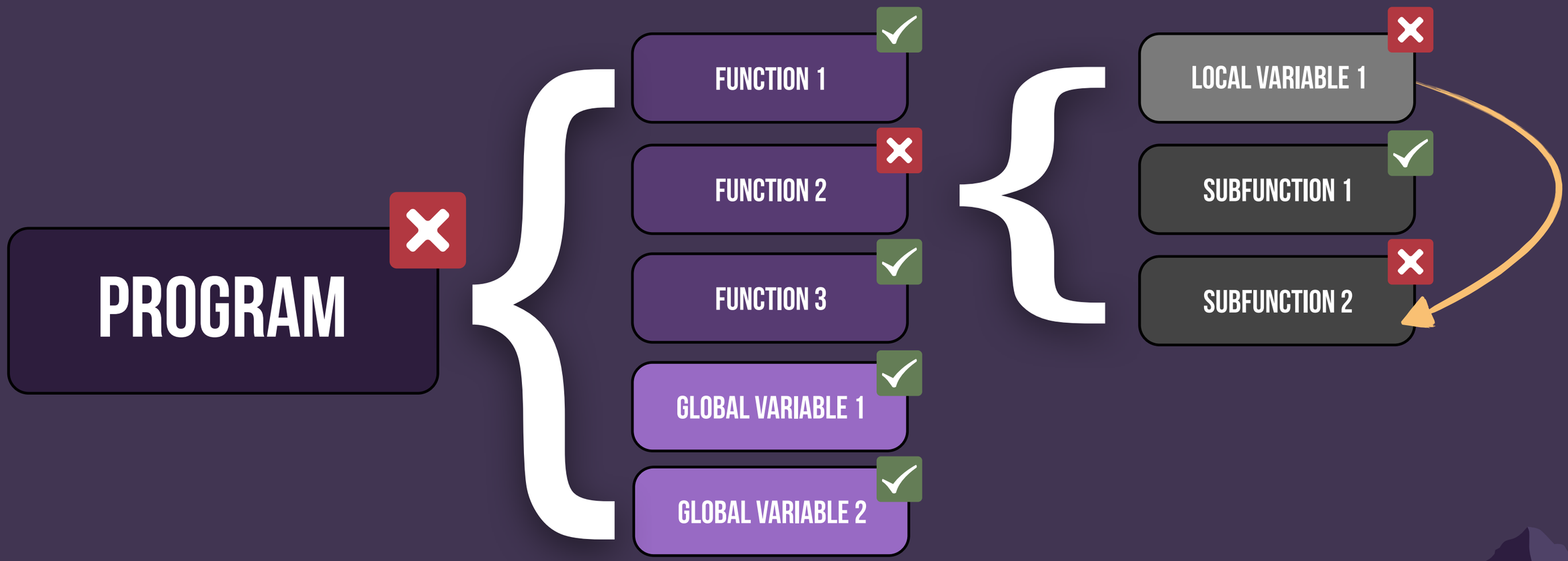
THE IMPORTANCE OF LOAD ORDER

Ensuring that every line of code can execute when it's needed



THE IMPORTANCE OF LOAD ORDER

Ensuring that every line of code can execute when it's needed



“HOISTING” WITHIN A JAVASCRIPT SCOPE

First, memory is set aside for all necessary variables and declared functions.

We build it like this...

```
function sumOfSquares (a, b){

    var x = add(a*a, b*b);
    return x;

    function add (c, d){
        var a = c + d;
        return a;
    }
}
```

...but JavaScript loads it like this.

```
function sumOfSquares (a, b){

    var x = undefined;
    function add (c, d){
        var a = c + d;
        return a;
    }

    x = add(a*a, b*b);
    return x;
}
```

Declared stuff that needs space in memory is first "hoisted" to the top of scope before any operational code is run.

CODING CAREFULLY FOR SMOOTH EXECUTION

Some examples of the impact of hoisting

```
function getMysteryNumber () {

  function chooseMystery() {
          return 12;
  }

  return chooseMystery();

  function chooseMystery() {
          return 7;
  }

}
```

Loads like this

```
function getMysteryNumber () {

  ✗ function chooseMystery() {
          return 12;
    }

  ✓ function chooseMystery() {
          return 7;
    }

    return chooseMystery();

}
```

```
getMysteryNumber( );
```

→ ?

The chooseMystery function is redefined by the time all hoisting is finished!

```
getMysteryNumber( );
```

→ 7

CODING CAREFULLY FOR SMOOTH EXECUTION

Function Expressions are never hoisted! They are treated as assignments.

```
function getMysteryNumber () {

  var chooseMystery = function() {
        return 12;
  }

  return chooseMystery();

  var chooseMystery = function() {
        return 7;
  }

}
```

```
getMysteryNumber( );
```

→ ?

Loads like this

```
function getMysteryNumber () {

  var chooseMystery = undefined;
  var chooseMystery = undefined;
  chooseMystery = function () {
        return 12;
  };

  return chooseMystery();

  chooseMystery = function () {
        return 7;
  }
}
```

Unreachable!

```
getMysteryNumber( );
```

→ 12

CODING CAREFULLY FOR SMOOTH EXECUTION

Function Expressions are never hoisted! They are treated as assignments.

```
function getMysteryNumber () {

  return chooseMystery();


  var chooseMystery = function() {
        return 12;
  }

  var chooseMystery = function() {
        return 7;
  }

}
```

Loads like this

```
getMysteryNumber( );
```

→ ?

```
function getMysteryNumber () {

  var chooseMystery = undefined;
  var chooseMystery = undefined;
  return chooseMystery();

  chooseMystery = function () {
        return 12;
  };

  chooseMystery = function () {
        return 7;
  }
}
```

Unreachable!

```
getMysteryNumber( );
```

→ ERROR

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    *if the train is full*
        execute a function that alerts a message
        that no seats remain and then returns false.


    *if the train is NOT full*
        execute a function that alerts a message with
        how many seats remain, and then returns true.



}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity)
        execute a function that alerts a message
        that no seats remain and then returns false.


   *if the train is NOT full*
        execute a function that alerts a message with
        how many seats remain, and then returns true.




}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    }



    *if the train is NOT full*
        execute a function that alerts a message with
        how many seats remain, and then returns true.




}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
         noSeats();
    } else {

    }


        execute a function that alerts a message with
        how many seats remain, and then returns true.




}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }









}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }



}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

```
capacityStatus(60, 60);
```

?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

```
capacityStatus(60, 60);
```

?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    var noSeats = undefined;
    var seatsAvail = undefined;
    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }
    noSeats = function (){
        alert("No seats left!");
        return false;
    }
    seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

Doesn't exist yet!

```
capacityStatus(60, 60);
```

?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    var noSeats = undefined;
    var seatsAvail = undefined;
    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }
    noSeats = function (){
        alert("No seats left!");
        return false;
    }
    seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

Doesn't exist yet!

```
capacityStatus(60, 60);
```

→ ERROR

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }

    if (numPassengers == capacity) {
         noSeats();
    } else {
         seatsAvail();
    }
}
```

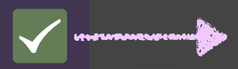
```
capacityStatus(60, 60);
```

?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    var noSeats = undefined;
    var seatsAvail = undefined;
    noSeats = function (){
        alert("No seats left!");
        return false;
    }
    seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }
}
```



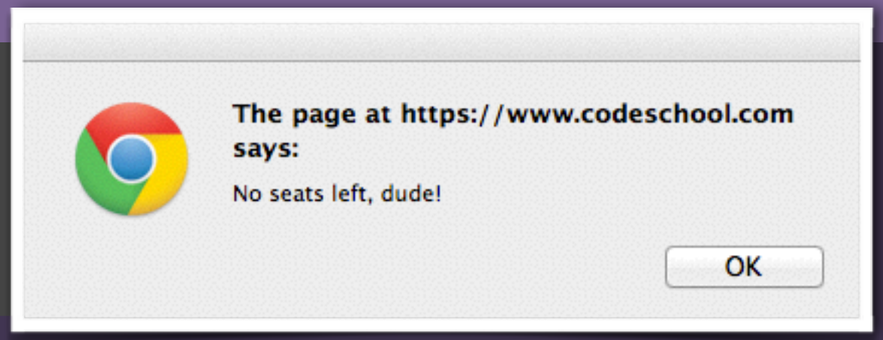
```
capacityStatus(60, 60);
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    var noSeats = undefined;
    var seatsAvail = undefined;
    noSeats = function (){
        alert("No seats left!");
        return false;
    }
    seatsAvail = function(){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }
}
```

```
capacityStatus(60, 60);
```



HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }


    var noSeats = function (){
        alert("No seats left!");
        return false;
    }
    var seatsAvail = function (){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

    function noSeats (){
        alert("No seats left!");
        return false;
    }
    function seatsAvail (){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }
}
```

```
capacityStatus(20, 60);
```

?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    function noSeats (){
        alert("No seats left!");
        return false;
    }
    function seatsAvail (){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

}
```

```
capacityStatus(20, 60);
```

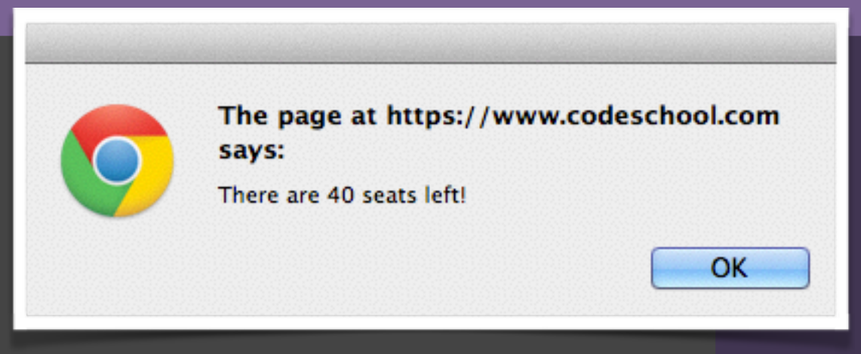
?

HOW MIGHT THIS AFFECT OUR EARLIER TRAIN SYSTEM?

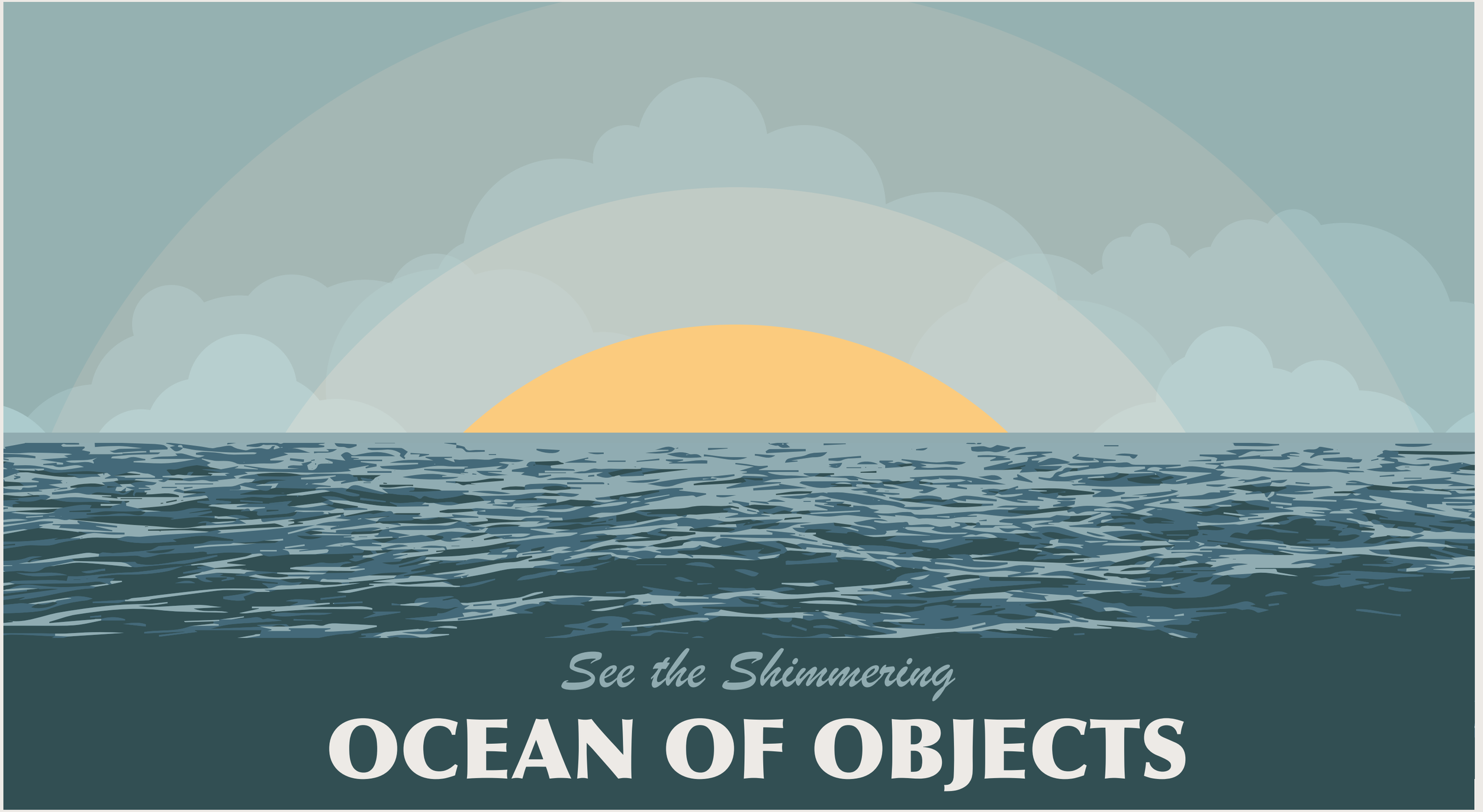
Hoisting in a function that returns a capacity status for the JavaScript Express

```
function capacityStatus (numPassengers, capacity) {
    function noSeats (){
        alert("No seats left!");
        return false;
    }
    function seatsAvail (){
        alert("There are " + (capacity - numPassengers) + " seats left!");
        return true;
    }

    if (numPassengers == capacity) {
        noSeats();
    } else {
        seatsAvail();
    }

}
```

```
capacityStatus(20, 60);
```




Climb
THE HOISTING HILLS


See the Shimmering
OCEAN OF OBJECTS

LEVEL 4

OCEAN OF OBJECTS

OBJECTS ARE "CONTAINERS" OF RELATED INFORMATION

Multiple pieces of data, called properties, are grouped within an Object

OBJECT

property 1
property 2
property 3

property 4
property 5
property 6

All of these properties "belong" to this Object.

WE CAN REPRESENT EVERYDAY STUFF WITH JS OBJECTS

Since common things have “bits” of related info, they make good Object examples

OBJECT

property 1
property 2
property 3

property 4
property 5
property 6

WE CAN REPRESENT EVERYDAY STUFF WITH JS OBJECTS

Since common things have “bits” of related info, they make good Object examples

BOOK

property 1
property 2
property 3

property 4
property 5
property 6