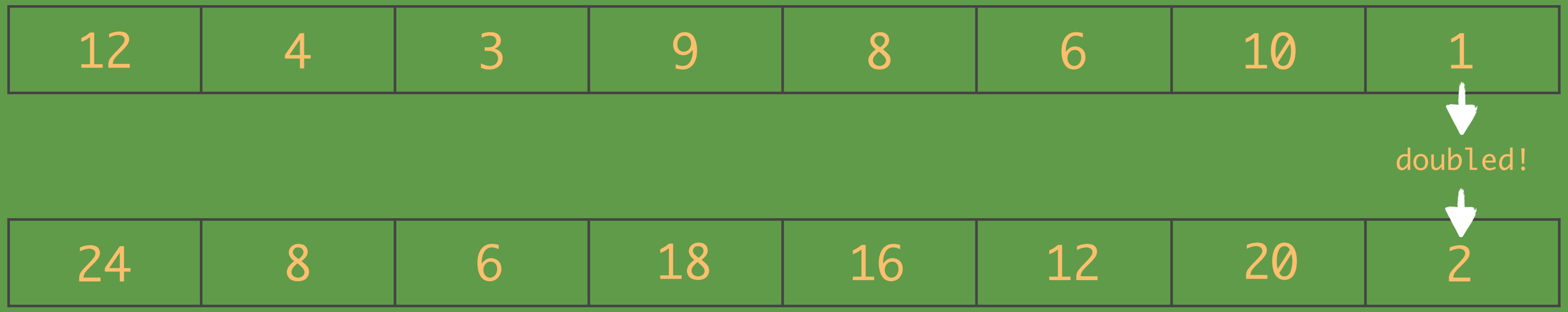
USING FE'S WITH ARRAYS AND MAP()

Let's pass in function that will double each cell's value in our numbers array.

```
var numbers = [12, 4, 3, 9, 8, 6, 10, 1];
```

```
var results = numbers.map( function (arrayCell) {
                              return arrayCell * 2;
                           }
);
```



USING FE'S WITH ARRAYS AND MAP()

Let's pass in function that will double each cell's value in our numbers array.

```
var numbers = [12, 4, 3, 9, 8, 6, 10, 1];
```

```
var results = numbers.map( function (arrayCell) {
                              return arrayCell * 2;
                           }
);
```

```
console.log(results);
```

→ `[24, 8, 6, 18, 16, 12, 20, 2]`

USING FE'S WITH ARRAYS AND MAP()

Let's pass in function that will double each cell's value in our numbers array.

```
var numbers = [12, 4, 3, 9, 8, 6, 10, 1];
```

```
var results = numbers.map( function (arrayCell) { return arrayCell * 2; } );
```

Short functions are often built in one line for clarity and simplicity.

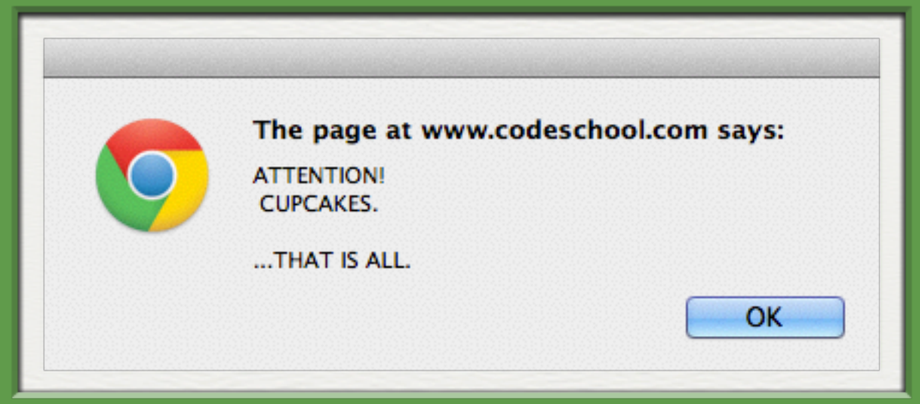
```
console.log(results);
```

→ [24, 8, 6, 18, 16, 12, 20, 2]

Visit the Wondrous

FOREST OF FUNCTION EXPRESSIONS

```
var sweetAnnouncement = function () { alert("ATTENTION!\n CUPCAKES.\n\n...THAT IS ALL.") };
sweetAnnouncement();
```



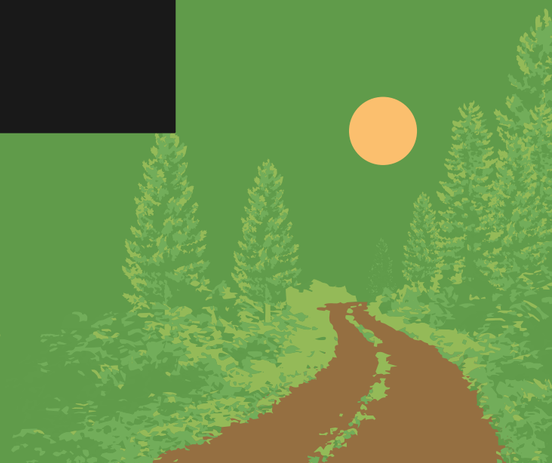

```
var fruits = [ "Apple", "Orange", "Pineapple", "Cranberry", "Pomegranate" ];
var fruitJuice = fruits.map( function (fruit) { return "\n" + fruit + " juice"; } );
alert(fruitJuice);
```

```
function mystery () {

    *some mystery code...ooh, spooky.*

    return *a function expression*

}
```

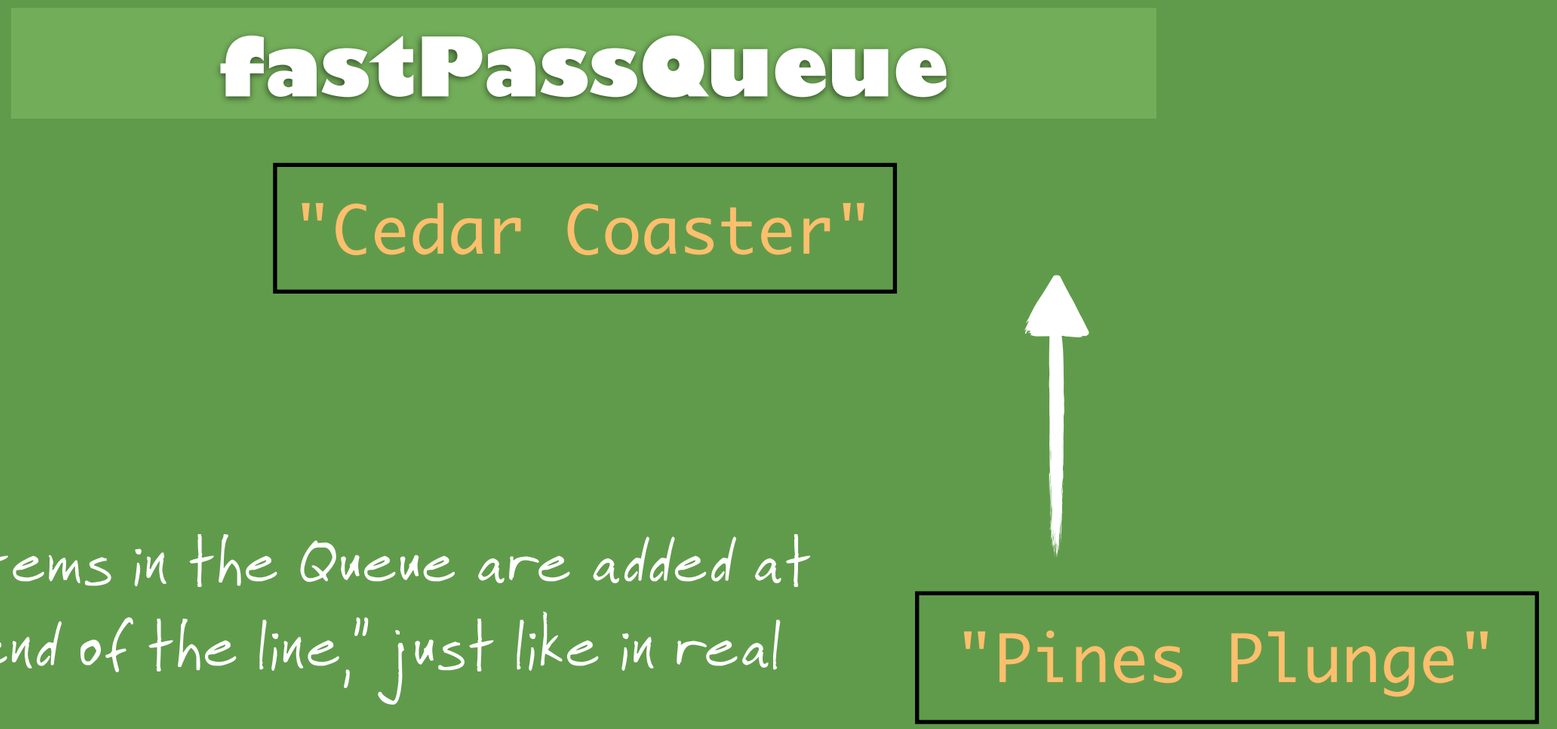


A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```



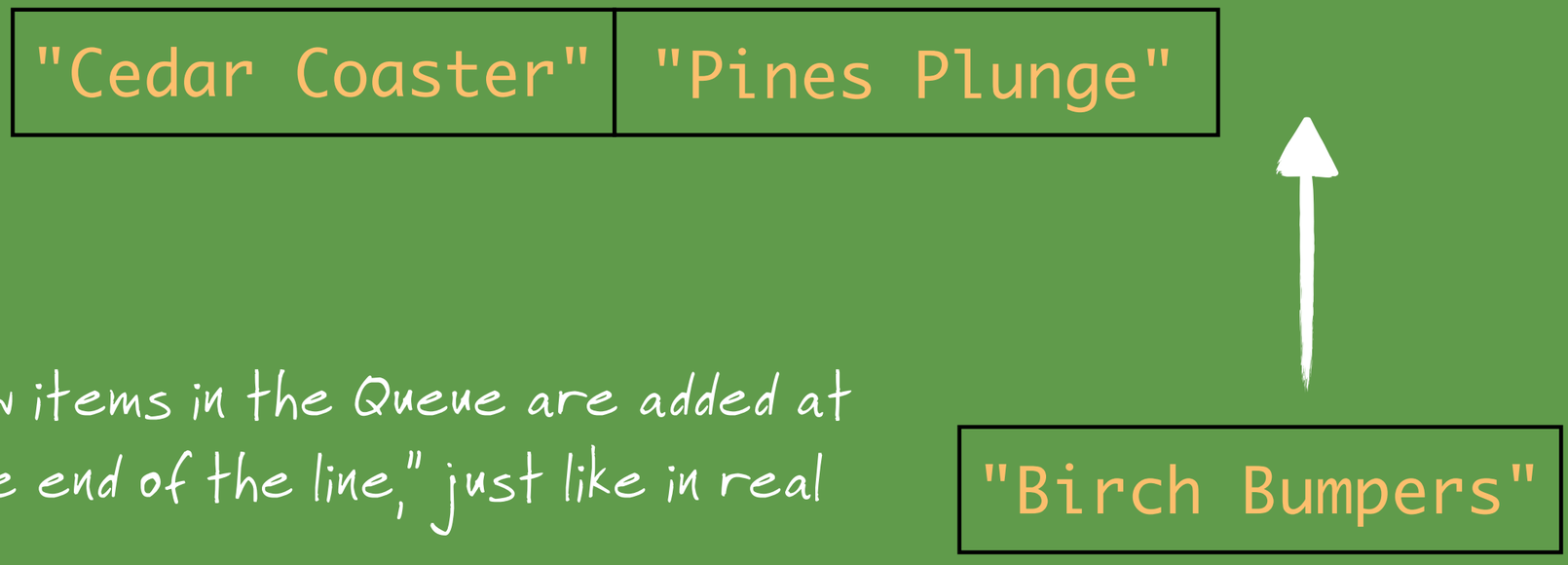
A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue



New items in the Queue are added at "the end of the line," just like in real life.

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

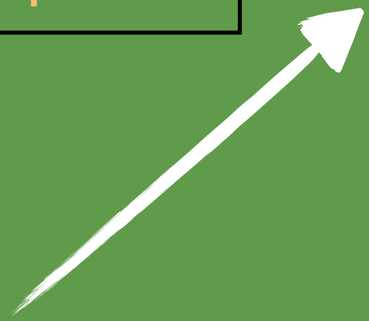
```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Cedar Coaster"	"Pines Plunge"	"Birch Bumpers"

We know a method that adds cells to the back of arrays, right??



```
fastPassQueue.push("Pines Plunge");
```

"Pines Plunge"

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Cedar Coaster"	"Pines Plunge"	"Birch Bumpers"	"Pines Plunge"

We know a method that adds cells to the back of arrays, right??

```
fastPassQueue.push("Pines Plunge");
```

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Cedar Coaster"	"Pines Plunge"	"Birch Bumpers"	"Pines Plunge"

To empty the queue, however, we need a new method that removes cells from the front of the array!

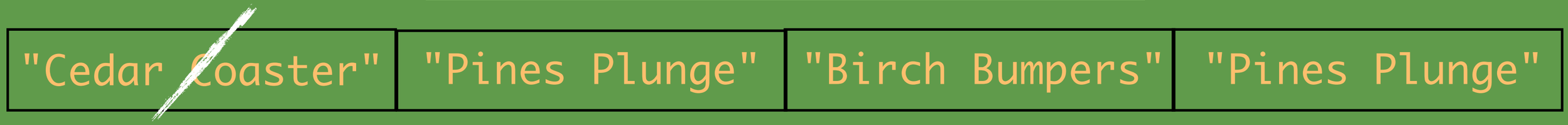
A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue




```
fastPassQueue.shift();
```

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Pines Plunge"	"Birch Bumpers"	"Pines Plunge"

```
fastPassQueue.shift();
```

→ "Cedar Coaster"

The shift method will return the cell that it removes from the front of the array.

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Pines Plunge"	"Birch Bumpers"	"Pines Plunge"

```
fastPassQueue.shift();
```

→ "Cedar Coaster"

The shift method will return the cell that it removes from the front of the array.

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Pines Plunge"	"Birch Bumpers"	"Pines Plunge"

```
fastPassQueue.shift();
```

→ "Cedar Coaster"

```
fastPassQueue.length;
```

→ 3

Like pop and push, shift will automatically modify the array's length.

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue


```
fastPassQueue.shift();
```

→ "Cedar Coaster"

```
var firstFastPass = fastPassQueue.shift();
```

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Birch Bumpers"	"Pines Plunge"

```
fastPassQueue.shift();
```

→ "Cedar Coaster"

```
var firstFastPass = fastPassQueue.shift();
```

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue

"Birch Bumpers"	"Pines Plunge"

```
fastPassQueue.shift();
```

→ "Cedar Coaster"

```
var firstFastPass = fastPassQueue.shift();
console.log(firstFastPass);
```

→ "Pines Plunge"

shift always returns the first cell, whether you use it in an expression, store it in a variable, or don't even use it at all.

A TICKET SYSTEM FOR THE FOREST THEME PARK

Using an array as a “queue” for Fast Pass delivery

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

fastPassQueue


```
fastPassQueue.shift();
```

→ `"Cedar Coaster"`

```
var firstFastPass = fastPassQueue.shift();
console.log(firstFastPass);
```

→ `"Pines Plunge"`

Now, we'll build a simple ticket system using our queue!

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {



}
```

This parameter will be the array of the rides and their wait times.

...and this will be the array of the next available Fast Pass rides.

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {




}
```

This will be the actual ride for which our customer would like a ticket!

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){


  }


}
```

If the next available Fast Pass is for the ride that the customer seeks, we'll give them that pass.

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();


  }



}
```

Here's our shift! It takes the front cell off the array and stores it in the pass variable.

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();
     return function ( ) { alert("Quick! You've got a Fast Pass to " + pass + "!");
           };
  }



}
```

Notice we are treating the function as an expression and returning it directly. No extra storage variable needed!

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();
     return function ( ) { alert("Quick! You've got a Fast Pass to " + pass + "!");
            };
  } else {
     for(var i = 0; i<allRides.length; i++){



     }
  }
}
```

To search for the ride the customer wants, we'll loop over the entire array of rides.

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();
     return function ( ) { alert("Quick! You've got a Fast Pass to " + pass + "!");
            };
  } else {
     for(var i = 0; i<allRides.length; i++){
        if(allRides[i][0] == pick){

        }
     }
  }
}
```

The ride names are contained within the first index of each subarray, or [0].

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();
     return function ( ) { alert("Quick! You've got a Fast Pass to " + pass + "!");
            };
  } else {
     for(var i = 0; i<allRides.length; i++){
        if(allRides[i][0] == pick){
           return function () { alert("A ticket is printing for " + pick + "!\n" +
                                      "Your wait time is about " + allRides[i][1] + " minutes.");
           };
        }
     }
  }
}
```

The wait times are in the second index of the subarrays, or [1].

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
function buildTicket ( allRides, passRides, pick ) {
  if(passRides[0] == pick){
     var pass = passRides.shift();
     return function ( ) { alert("Quick! You've got a Fast Pass to " + pass + "!");
            };
  } else {
     for(var i = 0; i<allRides.length; i++){
        if(allRides[i][0] == pick){
           return function () { alert("A ticket is printing for " + pick + "!\n" +
                                      "Your wait time is about " + allRides[i][1] + " minutes.");
           };

        }
     }
  }
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Birch Bumpers";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

When `buildTicket` returns the correct ticket function, we'll store it in a `ticket` variable for later use!

```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Birch Bumpers";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
ticket();
```

To call the function contained in the `ticket` variable, we need a set of parentheses and a semicolon.

```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

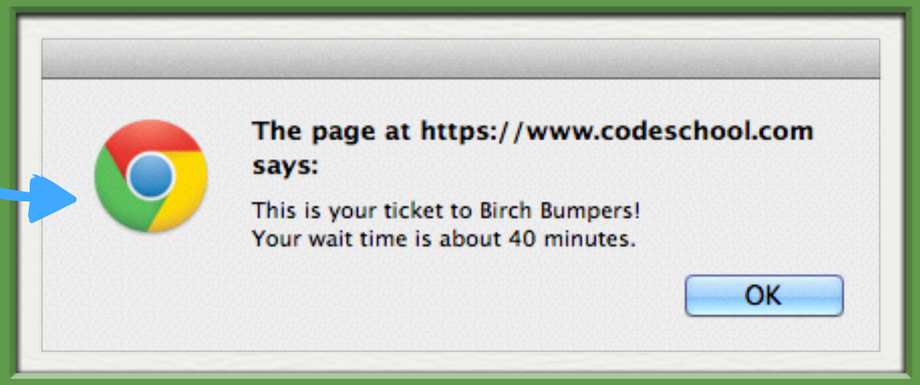
```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Birch Bumpers";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
ticket();
```




```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

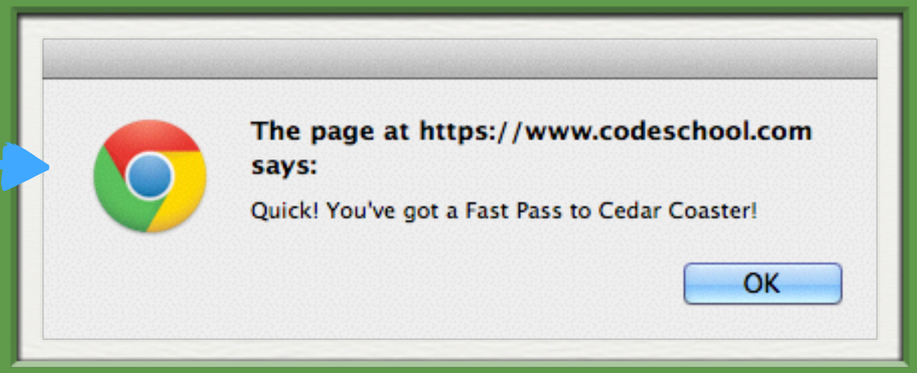
```
var fastPassQueue = [ "Cedar Coaster", "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Cedar Coaster";
```

Now, the desired ride matches the first Fast Pass!

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
ticket();
```




```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

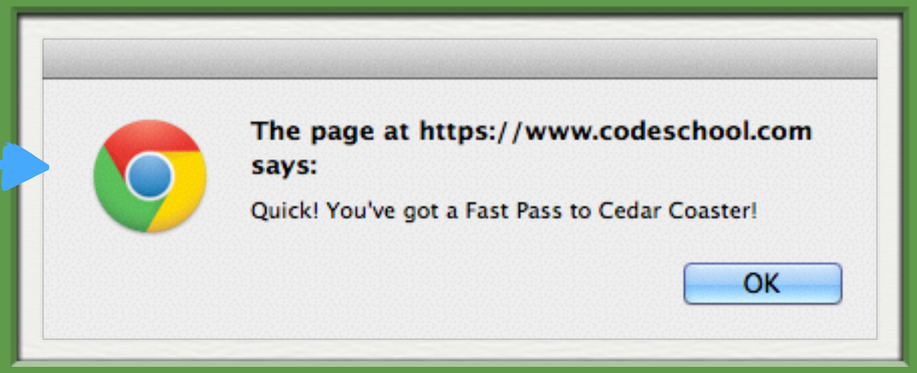
```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [                "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Cedar Coaster";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
ticket();
```




```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

LET'S MAKE SOME TICKETS!

Since functions can be treated as expressions, they can also be returned like values!

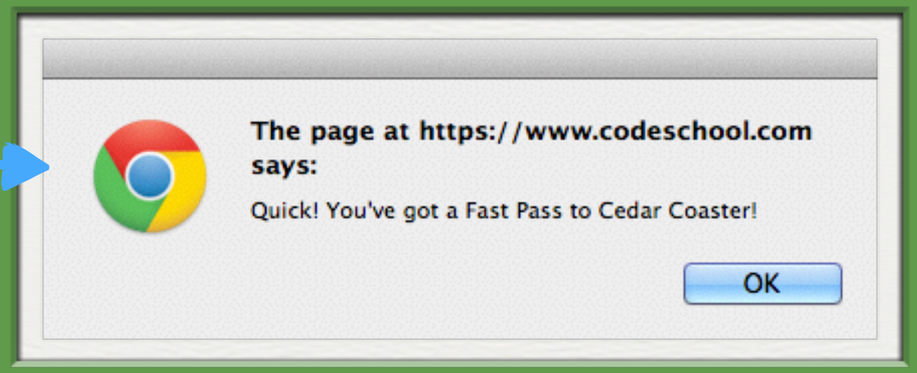
```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Cedar Coaster";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
ticket();
```




```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
var ticket = buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
function buildTicket ( allRides, passRides, pick ) {
    ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
( function ( ) {
    alert("Quick! You've got a Fast Pass to " + pass + "!");
} )
```

So far, all we get back is a function expression. Need more in order to call it!

The contents of pass are saved in a process called "closure," which we'll explore in the next level. For now, we know it's filled with the first available Fast Pass.

```
function buildTicket ( allRides, passRides, pick ) {
   ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide );
```

```
( function ( ) {
    alert("Quick! You've got a Fast Pass to " + pass + "!");
} )
```

```
function buildTicket ( allRides, passRides, pick ) {
   ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide )()                ;
```

```
( function ( ) {
    alert("Quick! You've got a Fast Pass to " + pass + "!");
} )()
```

Okay, now we've given our expression some parameter parentheses. We're on the right track!

```
function buildTicket ( allRides, passRides, pick ) {
   ...
}
```

USING AN IMMEDIATELY-INVOKED FUNCTION

Calling returned functions instantly instead of variable storage

```
var parkRides = [ ["Birch Bumpers", 40] , ["Pines Plunge", 55]
                  ["Cedar Coaster", 20] , ["Ferris Wheel of Firs", 90]  ];
```

```
var fastPassQueue = [ "Pines Plunge", "Birch Bumpers", "Pines Plunge" ];
```

```
var wantsRide = "Pines Plunge";
```

```
buildTicket( parkRides, fastPassQueue, wantsRide )()                ;
```

```
( function ( ) {
    alert("Quick! You've got a Fast Pass to " + pass + "!");
} )()
```

```
function buildTicket ( allRides, passRides, pick ) {
   ...
}
```