Regular functions can be either **function declarations** or **function expressions**, however arrow functions are *always* **expressions**. In fact, their full name is "arrow function expressions", so they can only be used where an expression is valid. This includes being:

* stored in a variable,
* passed as an argument to a function,
* and stored in an object's property.

One confusing syntax is when an arrow function is stored in a variable.

**const** greet = name => `Hello ${name}!`;

In the code above, the arrow function is stored in the greet variable and you'd call it like this:

greet('Asser');

***Returns:****Hello Asser!*

**Parentheses and arrow function parameteres**

You might have noticed the arrow function from the greet() function looks like this:

name => `Hello ${name}!`

If you recall, the parameter list appears before the arrow function's arrow (i.e. =>). If there's only **one** parameter in the list, then you can write it just like the example above. But, if there are **two or more** items in the parameter list, or if there are **zero** items in the list, then you need to wrap the list in parentheses:

*// empty parameter list requires parentheses*

**const** sayHi = () => console.log('Hello Udacity Student!');

sayHi();

***Prints:****Hello Udacity Student!*

*// multiple parameters requires parentheses*

**const** orderIceCream = (flavor, cone) => console.log(`Here's your ${flavor} ice cream in a ${cone} cone.`);

orderIceCream('chocolate', 'waffle');

***Prints:****Here's your chocolate ice cream in a waffle cone.*

**QUESTION 1 OF 2**

Which of the following choices have correctly formatted arrow functions?

* setTimeout(() => {  
      console.log('starting the test');  
      test.start();  
  }, 2000);
* setTimeout( \_ => {  
      console.log('starting the test');  
      test.start();  
  }, 2000);
* const vowels = 'aeiou'.split('');  
  const bigVowels = vowels.map( (letter) => letter.toUpperCase() );
* const vowels = 'aeiou'.split('');  
  const bigVowels = vowels.map( letter => letter.toUpperCase() );

SUBMIT

**Concise and block body syntax**

All of the arrow functions we've been looking at have only had a single expression as the function body:

**const** upperizedNames = ['Farrin', 'Kagure', 'Asser'].map(

name => name.toUpperCase()

);

This format of the function body is called the *"concise body syntax"*. The concise syntax:

* has no curly braces surrounding the function body
* and automatically returns the expression.

If you need more than just a single line of code in your arrow function's body, then you can use the *"block body syntax"*.

**const** upperizedNames = ['Farrin', 'Kagure', 'Asser'].map( name => {

name = name.toUpperCase();

**return** `${name} has ${name.length} characters in their name`;

});

Important things to keep in mind with the block syntax:

* it uses curly braces to wrap the function body
* and a return statement needs to be used to actually return something from the function.

**QUESTION 2 OF 2**

Using your knowledge of how arrow functions work with automatic returns and curly braces, which of the following choices have correctly formatted arrow functions?

* const colors = ['red', 'blue', 'green', 'yellow', 'orange', 'black'];  
    
  const crazyColors = colors.map( color => {  
      const jumble = color.split('').reverse();  
      return jumble.join('') + '!';  
  });
* 

const colors = ['red', 'blue', 'green', 'yellow', 'orange', 'black'];  
  
const crazyColors = colors.map( color => {  
    color.split('').reverse().join('') + '!';  
});

* 

const colors = ['red', 'blue', 'green', 'yellow', 'orange', 'black'];  
const crazyColors = colors.map( color => return color.split('').reverse().join('') + '!' );

* const colors = ['red', 'blue', 'green', 'yellow', 'orange', 'black'];  
  const crazyColors = colors.map( color => color.split('').reverse().join('') + '!' );

SUBMIT

So arrow functions are awesome!

* The syntax is a lot shorter,
* it's easier to write and read short, single-line functions,
* and they automatically return when using the concise body syntax!

***WARNING:****Everything's not all ponies and rainbows though, and there are definitely times when you might not want to use an arrow function. So before you wipe from your memory how to write a traditional function, check out these implications:*

* *there's a gotcha with the this keyword in arrow functions*
  + *go to the next lesson to find out the details!*
* *arrow functions are only expressions*
  + *there's no such thing as an arrow function declaration*