**FUNCTIONS:**

**ARROW FUNCTIONS:**

\_The arrow function does NOT have the RETURN keyword or the semicolon like this:

ex:

const upperizedNames = ['Farrin', 'Kagure', 'Asser'].map(function(name) {

return name.toUpperCase();

});

Instead:

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

name => name.toUpperCase()

);

Q: Function declaration vs Function expression

A: Function Declaration:

function foo() { return 5; }

Function expression:

var foo = function foo() { return 5; }

Difference: Function declarations load before any code is executed.

Function expressions load only when the interpreter reaches that line of code

* Arrow function is function expression

**WEIRD SYNTAX:**

Arrow function is stored in a variable:

Const greet = name => `Hello ${name}!`;

greet(‘Bryan’);

Returns: Hello Bryan

BUT if there are more than 1 items in the parameter list:

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.

Multiple way to write arrow functions:

Each of these is correct:

setTimeout(() => {

console.log('starting the test');

test.start();

}, 2000);

setTimeout( \_ => {

console.log('starting the test');

test.start();

}, 2000);

* **Either () or \_ is fined. Underscore never gets used so it’s undefined**

const vowels = 'aeiou'.split('');

const bigVowels = vowels.map( (letter) => letter.toUpperCase() );

const vowels = 'aeiou'.split('');

const bigVowels = vowels.map( letter => letter.toUpperCase() );

* If there’s only 1 paramter, then without the parenthesis is fine but not wrong

Q: Concise and Block body syntax with Arrow function

A: 1) Concise body syntax:

+) Has no curly braces surrounding the function body

+) Automatically returns the expression

+) NO return statement, no curly braces

ex:

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

name => name.toUpperCase()

);

2)Block body syntax

+) Uses curly braces to wrap the function body

+) And when used with the curly braces, it needs to be used to actually return something from the function

+) Without either the curly braces or the “return” statement, it wouldn’t work

+) Used when there’s more than 1 line of code in the arrow function’s body

Ex:

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

name = name.toUpperCase();

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

});

ex 2:

const squares = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((square) => {

square = square \* square

return square;

});

console.log(...squares);

Q: What do I notice about that?

A: 1) {} always go with “return”. It doesn’t stop the function. Also .map already has loop built inside it

2) When you print out …squares, it’s like you print out everything (spread operator)

Q: Can I use arrow function instead of normal function from now on?

A: 1) When there’s the keyword “this”, can’t

2) Arrow functions are only expression, not declaration

**“THIS” KEYWORD:**

\_ read tmr <https://www.quora.com/What-is-the-difference-between-a-method-and-a-function-in-JavaScript>

* Array.map is a method

<https://classroom.udacity.com/nanodegrees/nd019/parts/290ec447-6555-41bf-ac39-457220a09aae/modules/9c5b7af0-0943-4d6e-b672-520440885aba/lessons/3925704a-be38-4b70-8c8b-a4a812b6a309/concepts/654cbc3e-5081-49e3-a17e-86198de93db6>