Assignment one

Arrow Operator

Core structure:

(argument1, argument2, ... argumentN) **=>** {

*// function body*

}

Provide a sample

Sample 1: Basic syntax with one parameter

//ES5

var phraseSplitterEs5 = function phraseSplitter(phrase) {

return phrase.split(' ');

};

//ES6

const phraseSplitterEs6 = phrase => phrase.split(" ");

console.log(phraseSplitterEs6("ES6 Awesomeness")); // ["ES6", "Awesomeness"]

Sample 2: Basic syntax with multiple parameters

// (param1, param2, paramN) => expression

// ES5

var multiplyES5 = function(x, y) {

return x \* y;

};

// ES6

const multiplyES6 = (x, y) => { return x \* y };

Sample 3: No parameters

//ES5

var docLogEs5 = function docLog() {

console.log(document);

};

//ES6

var docLogEs6 = () => { console.log(document); };

docLogEs6(); // #document... <html> ….

Sample 4: Object Literal Syntax

//ES5

var setNameIdsEs5 = function setNameIds(id, name) {

return {

id: id,

name: name

};

};

// ES6

var setNameIdsEs6 = (id, name) => ({ id: id, name: name });

console.log(setNameIdsEs6 (4, "Kyle")); // Object {id: 4, name: "Kyle"}

Explain in a bullet list the key features

* New syntax for writing functions
* No need for writing the word function or return
* The value of ‘this’ keyword is used in lexical scope

Explain the pros and cons

pros

* Reduce code size
* No need to define return statement in one line functions
* Save time
* Simplify function scope
* No curly brackets necessary for single line functions
* Best for callbacks or methods like map, reduce, or forEach

Cons

* Arrow functions are unusable for object methods
* They cant be constructors, no prototype property, cant be used with ‘new’
* Can’t be used as generators
* No arguments object

When to use the arrow function

* Use function in the global scope and for Object.prototype properties.
* Use class for object constructors.
* Use => everywhere else.