ARROW FUNCTIONS

Andrew Sheehan

Boston University/MET College
Computer Science Department

BROWSER SUPPORT
SINCE 2015

Chrome	Edge *	Safari	Firefox	Opera
4-44		3.1-9.1	2-21	10-31
45-119	12-119	10-17.1	22-120	32-103
120	120	17.2	121	104
121-123		17.3-TP	122-124	

WHAT ARE ARROW FUNCTIONS?

```
( parameter1, parameter2)  =>  {
    // this is a standard one-line
    // comment within the body of
    // this function.

    /* This is a multi-line comment.
       Notice that this function example
       does not have a name.    (Meaning,
       this function does not have a name)
     */
}
```

A notation that is a shorter form of the standard use of the function declaration or expression variants.

EXAMPLES

```
( param1, param2, …paramN )  =>  {  expressions;  }

 oneParam     =>  expression;

(oneParam)   =>  expression;

(oneParam = 200)  =>  { exp1 (); exp2; exp3…; }

( )  =>   { expressions; }
```

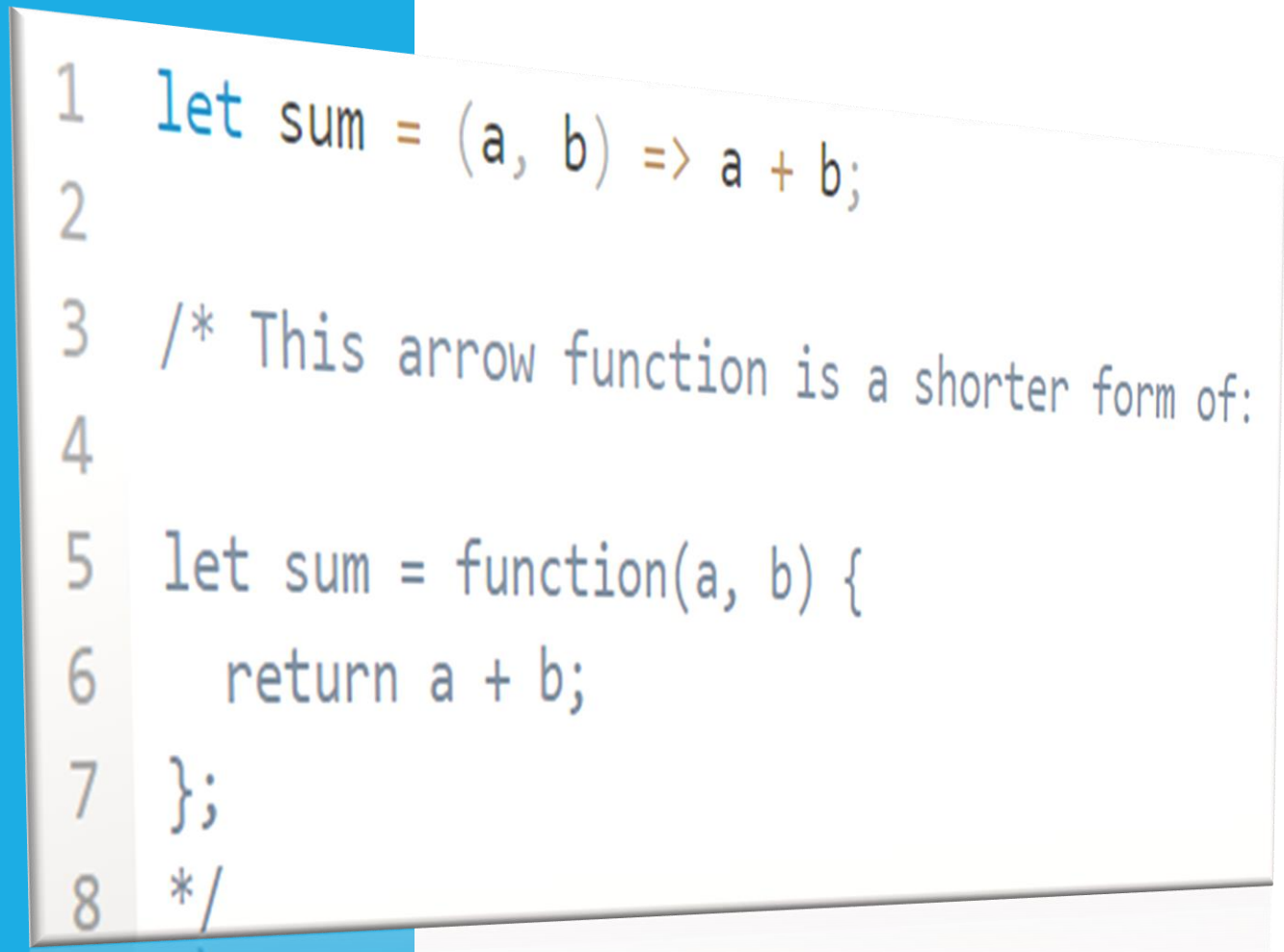
```
const func = (p1, p2…) =>  { expressions; }
```

```
param => expression; // notice there are no
                     // parentheses used here...

(param1, param2) => { /* ... */ }
```

When you have multiple arguments or no arguments, you'll need to re-introduce parentheses around the arguments.

EITHER WAY..

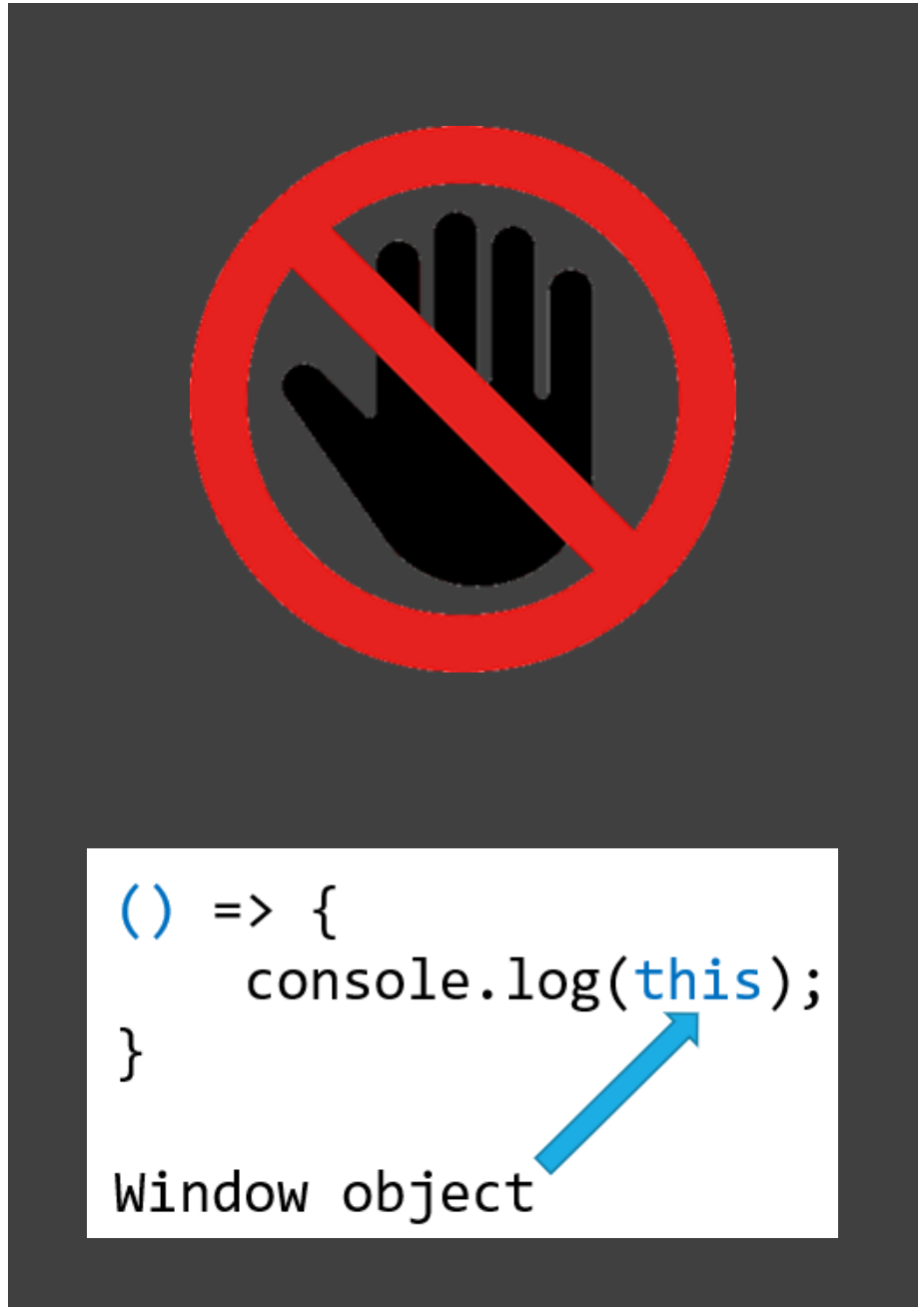

1 let sum = (a, b) => a + b;
2
3 /* This arrow function is a shorter form of:
4
5 let sum = function(a, b) {
6 return a + b;
7 };
8 */

Arrow functions do not have immediate access/binding to **'this'**.

They inherit it from the parent scope (window object)

Remember, 'this' refers to the object that is using it.

THIS



```
let test = {                    // Creating an Object Literal does not create block scope.
    name: 'Andrew',
    broken: () => `hi, ${this.name}`,
    works: function() { `hi, ${this.name}` }
};

console.log( test.broken() );  // undefined..
console.log( test.works() );   // hi, Andrew
```

HOW TO CAPTURE | 'THIS'

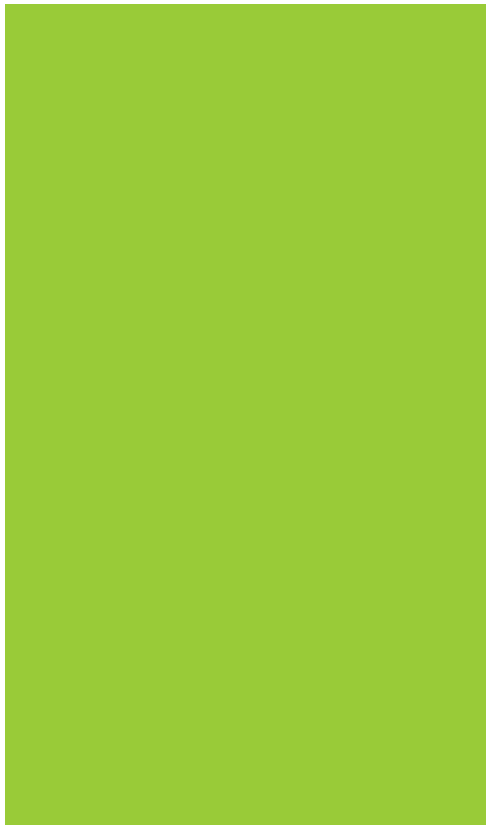
NOT USABLE AS CONSTRUCTORS

Arrow functions cannot be used as **function constructors**.
You will get an error.

```
// Creates an error.   Do not code this:
let AuditComponent = () => {
    this.createDate = new Date();
    this.logger = LoggerFactory.default();
}
```

One statement?
No explicit return needed

```
let getBreakingNews =
    () => BreakingService.getNews();
```



returns
undefined

```
() => { }
```

EMPTY ARROW FUNCTIONS

EXAMPLES

```
let callback;

callback = callback || function() {}; // ok

callback = callback || () => {};
// SyntaxError: invalid arrow-function arguments

callback = callback || (() => {});     // ok
```

A JavaScript immediately invoked function expression (“IIFE”) is a function defined as an expression and executed immediately after creation

PROTOTYPE

Arrow functions do not have access to the prototype property. They were not created with ‘function’.

```
let MenuComponent = () => {
    console.log(MenuComponent.prototype);
                                    //undefined
}
```