




JavaScript Functions



In design we often use the
phrase **form over function**,
what does that **imply**?





functions are one of the
fundamental **building**
blocks in JavaScript

(a procedure in the code, must be defined in scope)



We write a **function** when:

- There is a **block of code** that together performs a **single task**
- We have to **call the code more than once** (don't repeat code)
- We **define an object** that will act as a **template** for new things (instances)

```
1 //A function is a piece of code that is called by name
2 function newSlide() {
3   //add new slide to slideshow
4 }
5
6 //We use the function by writing
7 newSlide();
```

Writing a function

```
1  //number is the parameter passed in
2  function square(number) {
3      result = number * number;
4  }
5
6  //call it by using
7  square(4);
8
9  //the result would be 16 (4x4)
```

It can be
passed
data to
operate on
(parameters)
.

1 //More parameters can be passed by adding a comma inside the brackets

2 function multiplier(*number1*, *number2*) {

3 //code

4 }

5 //Functions can optionally return data (the return value)

6 function multiplier(*number1*, *number2*) {

7 return *number1* * *number2*;

8 }

9
10 let x = multiplier(3, 4)

11 console.log(x);

12

```
1 function multiplier(number1 = 3, number2 = 4) {  
2   return number1 * number2;  
3 }  
4  
5 let x = multiplier(2)  
6 console.log(x);  
7  
8 //Notice the default values assigned to each parameter
```



Parameters
can have a
default value


```
1 //this calls the above createDiv and the defaults will be applied
2 multiplier();
3
4 //now this calls again, but with new parameters:
5 multiplier(5, 2);
6
7 //and again, leaving default width and height:
8 createDiv(undefined, 7);
```

```
1  var square = function(number) {  
2    return = number * number;  
3  }  
4  
5  var x = square(3);  
6  
7  /* This initially looks like more code for the  
8  same result, there are times such as in objects  
9  that this makes more sense or as in a later slide: */
```

Functions
can also be
written as
**a function
expression**

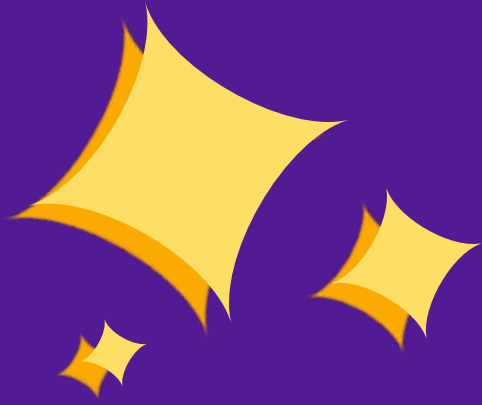
```
1  /* This looks a bit odd at first, but the basic rule is,
2  substitute the word 'function' for '=>' (equals and greater
3  than with no space). Here's the old way: */
4
5  var add = function(x, y) {
6    return x + y;
7  }
8  console.log(add(10, 20));
9
10 /* The new way, notice the arrow moves to the other side of
11 the parameters brackets: */
12 let addUp = (x, y) => {
13   return x + y;
14 }
15 console.log(addUp(10, 20));
```



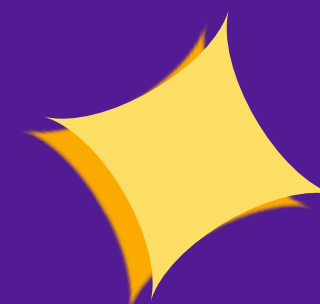
Arrow Functions

```
1 //Here's what was on the last slide
2 let addUp = (x, y) => {
3   return x + y;
4 }
5 console.log(addUp(10, 20));
6
7 //No curly braces as it's all on one line and no return keyword:
8 let addUp = (x, y) => x + y;
9 console.log(addUp(10, 20));
```

Arrow Functions
can **write the
same code even
smaller**



*Let's look at some
examples*



Functions as **objects** :

- This is a brief intro, we'll cover this in **more detail next week**
- Functions are **also objects** in their own right and can be used to **make new copies** (instances) of those objects
 - *Sounds confusing?!*
 - *I can't make a new coin from a coin!*
 - *But if I had the mould (and hot metal) I could!*

```
1  //Consider this:
2  function Make_person_object(firstname, lastname, age) {
3      this.firstname = firstname;
4      this.lastname = lastname;
5      this.age = age;
6  }
7
8  //We make new people (instances) out of the object
9  var john = new Make_person_object('John', 'Smith', 45);
10 //This is known as the constructor
11
12 //We can access individual properties of our John person
13 console.log(john.age);
14 //we grabbed John's age by calling the age of object John (we can add more)
15
16 var janet = new Make_person_object('Janet', 'Patel', 22);
17 console.log(janet.age);
```

```
1  /* We can also create nested functions that do something useful for us,
2  add this inside the function: */
3  function Make_person_object(firstname, lastname, age) {
4      this.firstname = firstname;
5      this.lastname = lastname;
6      this.age = age;
7
8      this.myName = function () {
9          var fullname = firstname + ' ' + lastname;
10         return fullname;
11     }
12 }
13
14 //Now we are using the function expression because it allows it to be called as follows
15 console.log(janet.myName());
```



```
1  janet.myName();
2
3  /*
4   * When accessing a function like this it is known as a method
5
6   * If a function is inside an object we are accessing a method
7   * of the object, otherwise a function is just a function.
8
9   * Method meaning “a way of using” the object.
10 */
```

Practice Quiz!

Give one example of when you write a function?

How do I pass in two parameters to a function?

```
1  function square(number) {  
2    result = number1 * number2;  
3  }
```

Can you write an arrow function using function seen here?