# Object and Strings

Level 3 – Section 1

# Removing Repetition From Creating Objects

The buildUser function returns an object with the first, last, and fullName properties.

```
function buildUser(first, last){
  let fullName = first + " " + last;

  return { first: first, last: last, fullName: fullName };
}
```

Calling the buildUser function:

```
let user = buildUser("Sam", "Williams");
console.log( user.first );
console.log( user.last );
console.log( user.fullName );

> Sam
> Williams
> Sam Williams
> Sam Williams
```

Returning objects with keys and variables with the same name looks repetitive

# The Object Initializer Shorthand



We can remove **duplicate** variable names from object properties when those properties have the **same name** as the variables being assigned to them.

```
function buildUser(first, last){
  let fullName = first + " " + last;

return { first, last, fullName };
}
Way cleaner! {}
```

Yields the same result:

# Assigning With Object Initializer Shorthand

The object initializer shorthand works **anywhere** a new object is returned, not just from functions.

```
let name = "Sam";
     let age = 45;
     let friends = ["Brook","Tyler"];
     let user = { name, age, friends };
     console.log( user.name );
                                                 > Sam
                                                 > 45
     console.log( user.age );
                                                   ["Brook", "Tyler"]
     console.log( user.friends );
Same thing let user = { name: name, age: age, friends: friends };
```



## **Object Destructuring**

We can use shorthand to assign **properties** from objects to **local variables** with the **same name**.

```
(\mathbf{x})
let user = buildUser("Sam", "Williams");
                                                       Unnecessary repetition
let first = user.first;
let last = user.last;
let fullName = user.fullName;
                                        This function returns { first, last, fullName }
Same names as properties
from return object
let { first, last, fullName } = buildUser("Sam", "Williams");
console.log( first );
                                          > Sam
console.log( last );
                                            Williams
console.log( fullName );
                                            Sam Williams
```

# **Destructuring Selected Elements**

Not all properties have to be destructured all the time. We can explicitly select the ones we want.

# Recap Object Initializer vs. Destructuring

#### **Object Initializer Shorthand Syntax**

```
let name = "Sam";
                              From variables to object properties
let age = 45;
let user = { name, age };
console.log( user.name );
console.log( user.age );
```

#### **Object Destructuring**

```
From object properties to variables
let { first, last, fullName } = buildUser("Sam", "Williams");
console.log( first );
console.log( last );
                                           Returns { first, last, fullName }
console.log( fullName );
```

# Adding a Function to an Object

In previous versions of JavaScript, adding a function to an object required specifying the **property name** and then the **full function definition** (including the *function* keyword).

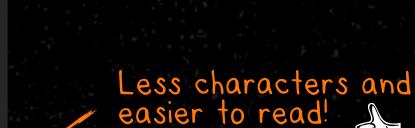
```
function buildUser(first, last, postCount){
 let fullName = first + " " + last;
  const ACTIVE_POST_COUNT = 10;
  return {
    first,
    last,
    fullName,
    isActive: function(){
      return postCount >= ACTIVE_POST_COUNT;
```

```
An anonymous function is assigned to an object property
```

# Using the Method Initializer Shorthand

A new shorthand notation is available for adding a method to an object where the keyword *function* is no longer necessary.

```
function buildUser(first, last, postCount){
 let fullName = first + " " + last;
  const ACTIVE_POST_COUNT = 10;
  return {
    first,
    last,
    fullName,
    isActive(){
      return postCount >= ACTIVE_POST_COUNT;
```



### **Template Strings**

Template strings are **string literals** allowing embedded expressions. This allows for a much better way to do **string interpolation**.

```
function buildUser(first, last, postCount){
  let fullName = first + " " + last;
  const ACTIVE_POST_COUNT = 10;
  //...
}
```

```
function buildUser(first, last, postCount){

let fullName = `${first} ${last}`;

const ACTIVE_POST_COUNT = 10;

//...

Enclosed by back-ticks, NOT single quotes, and code is wrapped inside dollar sign and curly braces
```

# Writing Multi-line Strings

Template strings offer a new — and much better — way to write **multi-line strings**.

```
let userName = "Sam";
let veryLongText = `Hi ${userName},
this is a very
                                             Hi Sam,
very
                                             this is a very
veeeery
                                             very
long text.
                                             veeeery
Regards,
                                             long text.
 ${admin.fullName}
                               Newline characters
                                             Regards,
                               are preserved
                                             Alex Williams
console.log( veryLongText );
```