**ES6 Syntax**

The Big Picture: ES6 adds new declarative syntax, which shifts our focus away from imperative implementation details and allows us to write higher-level, more interpretable code.

**Template Strings**

Easily embed variables & expressions:

```
'\nThe cactus is named ${name}.\n'
```

You can also use line breaks within backticks to create multi-line strings. No more `\n`!

**Arrow Functions**

A concise syntax for creating functions.

```
let sum = function (a, b) {
  return a + b;
};
```

There are many syntactic variations.

1. Arrow functions do not bind their own `this`; rather they inherit one from their parent `=>` “lexical scoping”.
2. Just because you can use shorthand doesn’t mean you should. Readability first.

**Scoping**

Let vs. Var

To enforce block scoping, rather than relying on stylistic conventions.

```
let Vanya = 1;
```

To declare variables that will be used in a wider context.

```
"Contain this variable within the immediate function scope."
```

Both `let` and `const` are block-scoped variables. Variables only exist within their enclosing block, denoted by `{}`.

**Destructuring**

Extract data from arrays & objects:

```
let [first, second] = myArr;
```

Extract 1st & 2nd elements from `myArr`.

```
let {a, b} = myObj;
```

Extract properties `a` & `b` from `myObj`.

This is just the tip of the iceberg when it comes to destructuring...

**Default Values**

If no value (or `undefined`) is passed:

```
function spam (n = 1) {
  // Default value assigned in function head.
}
```

```
let [a = 42, b, c] = ham;
```

A default value can also be passed when destructuring.

**Rest / Spread” Operator**

The behavior of `...` depends on the context:

1. In an assignment context, `...` transforms an indefinite list of arguments into a proper array:

```
function foo (...args) {
  // is an array of all parameters passed into the function.
}
```

```
let [s, ...m] = [1, 2, 3];
```

```
// it gathers the "rest" of the arguments.
```

2. Otherwise, `...` expands an iterable:

```
foo (...args)
// Pass the elements of an array into a function.
```

```
let a = [1, 2, 3];
let b = [0, ...a];
```

Succinct array creation!

**Concise Properties & Methods**

Eliminate duplicative code in objects.

```
var obj = {
  a, // When the property name = value name.
  b: []
};
```

Shorthand for method definitions.