

Functions

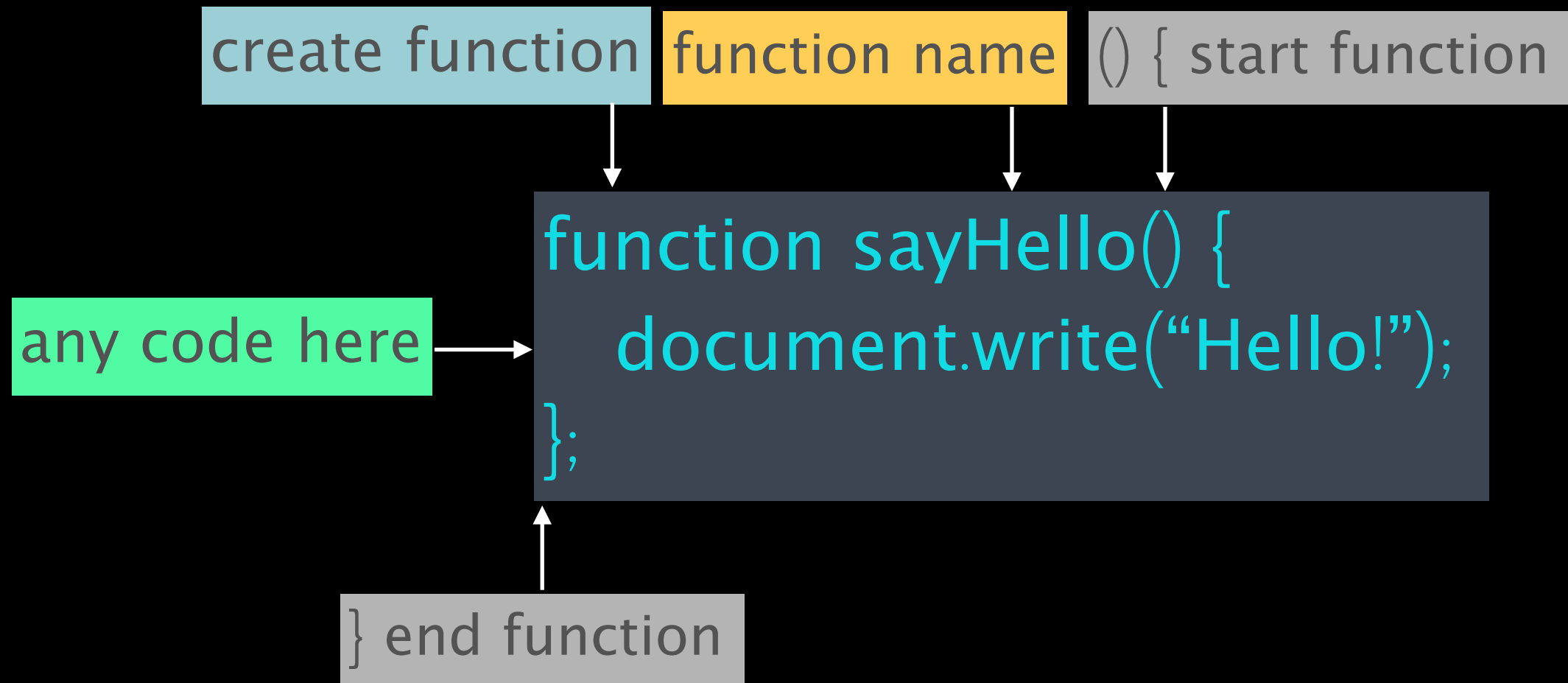
Functions are a way to group code together.
It can be good for:

1. writing reusable code
2. making code easier to read
3. abstracting, or giving a name to a set of instructions

Function rules

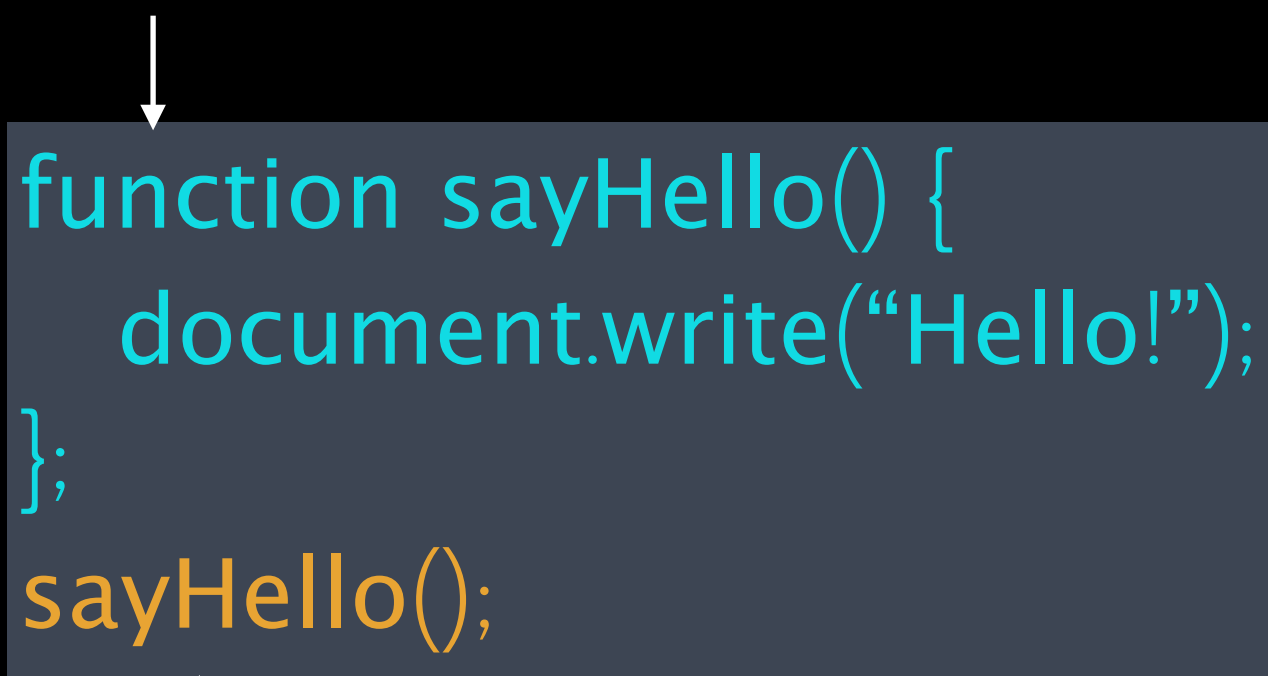
1. Function naming rules are the same as variables
2. Functions must be created before they are used
3. Functions must be called with parentheses ()
4. Function parameters and return statements are optional

Anatomy of a function



Create and call a function

Functions are created with the function keyword



The diagram illustrates the process of creating and calling a function. A white arrow points from the text 'Functions are created with the function keyword' to the 'function' keyword in the code block. Another white arrow points from the text 'Functions are called by writing the name, followed by ()' to the 'sayHello()' call in the code block.

```
function sayHello() {  
    document.write("Hello!");  
};  
sayHello();
```

Functions are called by writing the name, followed by ()

What is the () for?

Functions can also take in parameters. Parameters allow you to customize the behavior of a function.

```
function sayHello(name) {  
    document.write(`Hello, ${name}!`);  
};  
sayHello("John"); //print out 'Hello, John!'  
sayHello("Billy"); //print out 'Hello, Billy!'
```

The string "John" and "Billy" here are stored in variables name and then used in the function

Functions can also return values

When you use return keyword inside a function, the returned value becomes the result/output of that function.

You can store the result/output of a function into a variable.

```
function sum(x, y) {  
    let total = x + y ;  
    return total;  
};  
// save the return value in variable  
let result = sum(5, 10);  
document.write(result); //print out 15
```

Anatomy of a function: return

return keyword

```
function sum(x, y) {  
  let total = x + y ;  
  return total;  
};
```

value to return

Function scope

Variables that are created in functions only live within it. This is called a local scope.

```
function sum(x, y) {  
  let total = x + y ;  
  console.log(total); //15  
};
```

```
sum(5, 10);  
console.log(total); //ReferenceError:total is not defined
```

*variable can not be accessed outside the function

Function scope

Variables created outside of functions are in the **global scope**.

```
let total;  
function sum(x, y) {  
  total = x + y ;  
  console.log(total); //15  
};  
sum(5, 10);  
console.log(total); //15
```

*a function can access an outside variable if it was defined before the function

Arrow function

Arrow functions allow us to write shorter function syntax, and also changes the binding of **this** keyword, but we don't have to get into that now.

```
function sum(x, y) {  
  return x + y  
}
```

// is the same as:

```
const sum = (x, y) => {  
  return x + y  
}
```

// is also equivalent to:

```
const sum = (x, y) => x + y
```

// Without curly braces after => means the x + y will be returned

Exercise :

1. Modify your **FizzBuzz** code to include the usage of functions
2. **Tax Calculator**