Intro to functions

CODE CHRYSALIS コードクリザリス

By the end of this lesson, you should be able to...

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions
3. *Invoke* functions

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions
3. *Invoke* functions
4. *Return* a value from a function using the `return` keyword

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions
3. *Invoke* functions
4. *Return* a value from a function using the `return` keyword
5. Use `console.log` to *print* values in the *console*

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions
3. *Invoke* functions
4. *Return* a value from a function using the `return` keyword
5. Use `console.log` to *print* values in the *console*
6. Identify inputs and outputs in functions

CODE CHRYSALIS コードクリサリス

By the end of this lesson, you should be able to...

1. Know what functions are and why we use them
2. *Declare* functions
3. *Invoke* functions
4. *Return* a value from a function using the `return` keyword
5. Use `console.log` to *print* values in the *console*
6. Identify inputs and outputs in functions
7. Know the difference between a `return` statement and `console.log`

CODE CHRYSALIS コードクリサリス

“Go get me a coffee!”

CODE CHRYSALIS コードクリサリス

“Go get me a coffee!”

CODE CHRYSALIS コードクリサリス

“Go get me a coffee!”

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyMart
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

CODE CHRYSALIS コードクリサリス

Why do we have functions?

To summarise a series of instructions!

Why do we have functions?

To summarise a series of instructions!

Why do we have functions?

To summarise a series of instructions!

Instead of reciting the entire list of instructions every time you want someone to get you a coffee, you teach them to associate those steps with a much shorter phrase: “Go get me a coffee!”

Why do we have functions?

To summarise a series of instructions!

Instead of reciting the entire list of instructions every time you want someone to get you a coffee, you teach them to associate those steps with a much shorter phrase: “Go get me a coffee!”

Writing programs is the same. We group instructions together and associate them with a label. This grouping of instructions is a *function*.

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg
- Put down your right leg

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg
- Put down your right leg
- Pick up your left leg

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg
- Put down your right leg
- Pick up your left leg
- Move it forward, in front of your left leg

CODE CHRYSALIS コードクリサリス

“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg
- Put down your right leg
- Pick up your left leg
- Move it forward, in front of your left leg
- Put down your left leg

CODE CHRYSALIS コードクリサリス

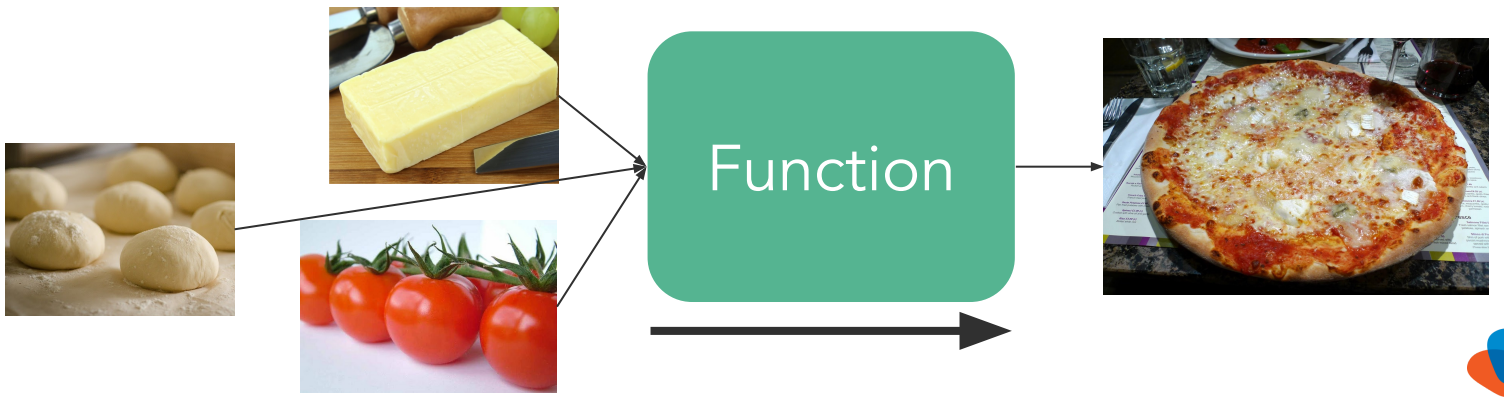
“Go Upstairs” Function

- Go upstairs
- Go onto the street
- Turn to the right
- Walk until you see the FamilyM
- Go inside
- Go to refrigerator in the back
- Grab a coffee
- Take coffee to cash register
- Give cashier correct amount
- Walk outside
- Turn to the right
- Walk until you reach our building
- Go to B2
- Give me the coffee

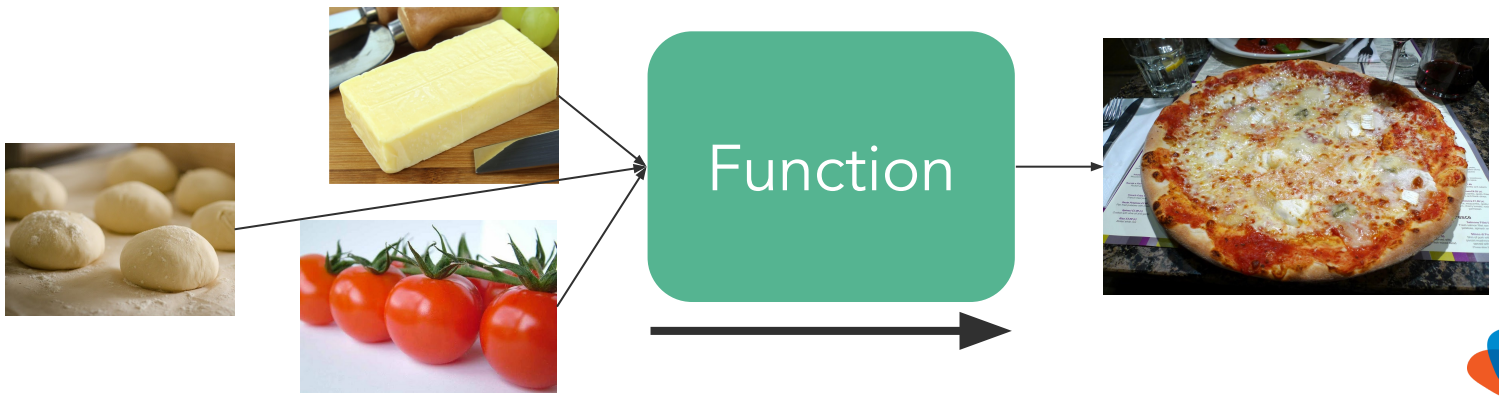
```
goUpstairs();
```

- Stand with both legs straight
- Pick up your right leg
- Move it forward, in front of your left leg
- Put down your right leg
- Pick up your left leg
- Move it forward, in front of your left leg
- Put down your left leg
- Repeat

Functions are input and output machines

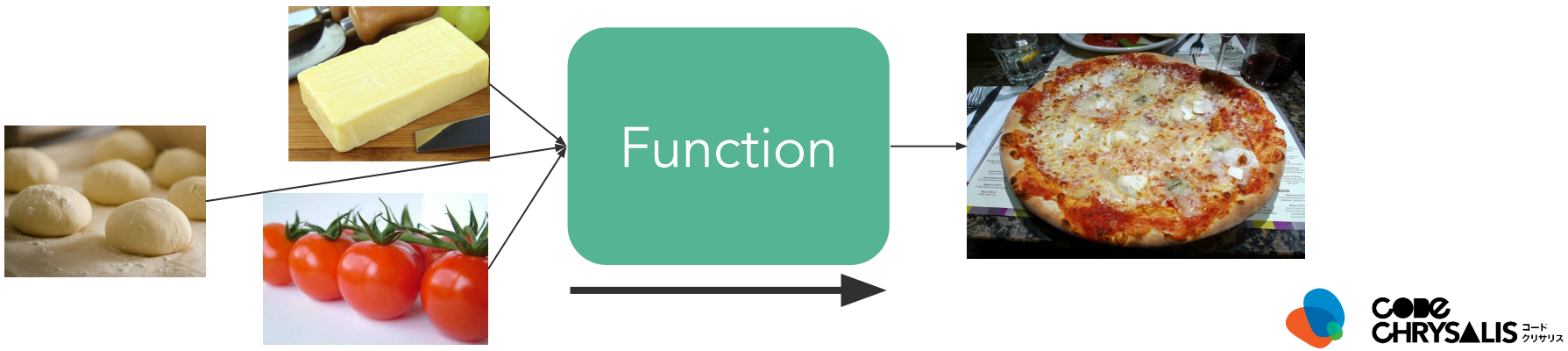


Functions are input and output machines



Functions are input and output machines

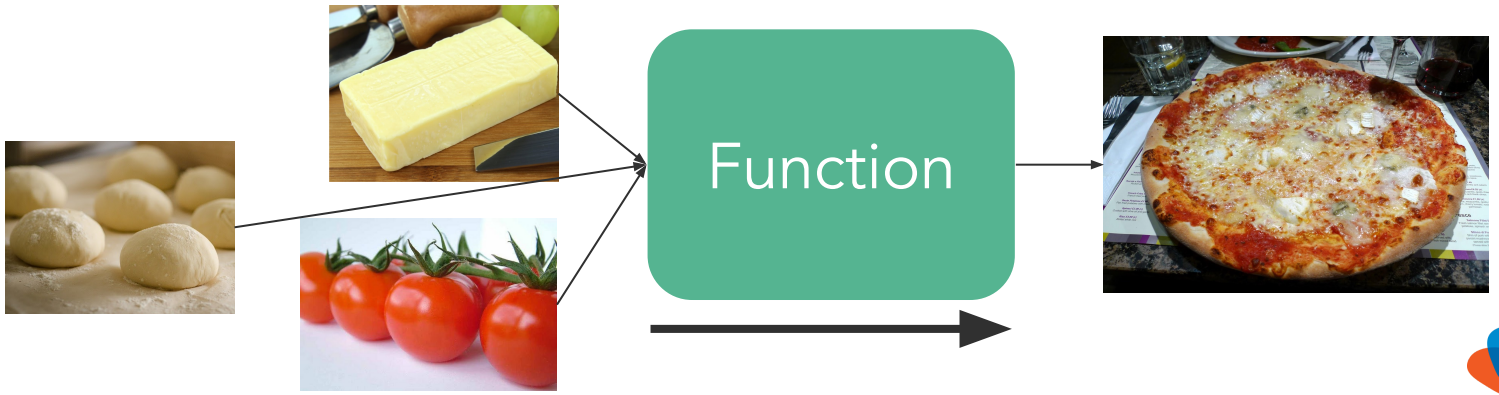
Let's say you have a pizza machine.



Functions are input and output machines

Let's say you have a pizza machine.

`pizzaMachine` is a function that takes 3 parameters (dough, tomatoes, and cheese) and returns a pizza.



A Simple Function

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function

Let's write a function called `getGreeting`

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function

Let's write a function called `getGreeting`

`getGreeting` is a function that takes 1 parameter (a `string`) and returns a `string`.

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - The **`return`** keyword

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - The **`return`** keyword

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - The **`return`** keyword

`getGreeting` is able to give an output because it uses the `return` keyword to create a return statement.

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - Vocabulary

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - Vocabulary

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - Vocabulary

What is the function declaration?

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

A Simple Function - Vocabulary

What is the function declaration?

What is the function invocation?

```
function getGreeting(name) {
  return "Hello, " + name;
}

getGreeting("Genki");
  => "Hello, Genki"
```

CODE CHRYSALIS コードクリサリス

Inputs are optional!

```
function meaningOfLife() {
  return 42;
}

meaningOfLife();
```

CODE CHRYSALIS コードクリサリス

Inputs are optional!

```
function meaningOfLife() {
  return 42;
}

meaningOfLife();
```

CODE CHRYSALIS コードクリサリス

Inputs are optional!

Inputs can be optional, too! Here's an example.

```
function meaningOfLife() {
  return 42;
}

meaningOfLife();
```

CODE CHRYSALIS コードクリサリス

Inputs are optional!

Inputs can be optional, too! Here's an example.

`meaningOfLife` is a function that takes 0 parameters and returns a `number`.

```
function meaningOfLife() {
  return 42;
}

meaningOfLife();
```

Inputs are optional!

Inputs can be optional, too! Here's an example.

`meaningOfLife` is a function that takes 0 parameters and returns a `number`.

```
function meaningOfLife() {
  return 42;
}

meaningOfLife("is sleep?");
```

What happens if you invoke `meaningOfLife` with an input?

CODE CHRYSALIS コードクリサリス

Your Turn!

Your Turn!

Your Turn!

We have a function called **addTen** that takes a number and returns the number plus 10.

Your Turn!

We have a function called **addTen** that takes a number and returns the number plus 10.

How can we write this? Here is some starter code for you.

Your Turn!

We have a function called **addTen** that takes a number and returns the number plus 10.

How can we write this? Here is some starter code for you.

```
function addTen(number) {
  // What goes in here?
}

addTen(5);
  => 15
```

CODE CHRYSALIS コードクリサリス

More on functions

More on functions

More on functions

What if `addTen` does NOT have a `return` keyword?

All functions return *something*. If a function doesn't specify a return value, it returns `undefined`.

More on functions

What if `addTen` does NOT have a `return` keyword?

All functions return *something*. If a function doesn't specify a return value, it returns `undefined`.

```
function addTen(number) {
  number + 10;
}

addTen(5);
  => undefined
```

More on functions

More on functions

More on functions

We can use `console.log` inside of functions, too!

We can use it to figure out what our function is doing.

More on functions

We can use `console.log` inside of functions, too!

We can use it to figure out what our function is doing.

```
function addTen(number) {
  console.log("The argument is", number);
  return number + 10;
}

addTen(5);
  => 15
```

CODE CHRYSALIS コードクリサリス

More on functions

More on functions

More on functions

What if we write more code below the `return` statement?

What will be printed to the console?

CODE CHRYSALIS コードクリサリス

More on functions

What if we write more code below the `return` statement?

What will be printed to the console?

```
function addTen(number) {
  console.log("The argument is", number);
  return number + 10;
  console.log("This is the last line of the function!");
};

addTen(5);
  => 15
```

CODE CHRYSALIS コードクリサリス

More on functions

When the computer reaches the `return` statement, the function will return the value and exit.
No code written *after* the return statement will be executed.

```
function addTen(number) {
  console.log("The argument is", number);
  return number + 10;
  console.log("This is the last line of the function!");
}

addTen(5);
  => 15
```

CODE CHRYSALIS コードクリサリス

More on functions: the difference between return and console.log

`console.log` does NOT return a value or stop execution. It only prints a value to the console.

`return` returns the value of the expression in the *return statement*.

```
function addTen(number) {
  return number + 10;
}

addTen(5);
  => 15
```

```
function addTen(number) {
  console.log(number + 10);
}

addTen(5);
  => undefined
```

CODE CHRYSALIS コードクリサリス

More on functions: the difference between return and console.log

If you want to see what a function returns, you can use `console.log` to see the value *after* you invoke the function.

```
function addTen(number) {
  return number + 10;
}

const answer = addTen(5);

console.log(answer); // should print 15
console.log(addTen(5)); // should also print 15
```

CODE CHRYSALIS コードクリサリス

Breaking Down Functions

```
function addTen(parameters) {
  // instructions go here!
}

addTen(someInput);
```

CODE CHRYSALIS コードクリサリス

Breaking Down Functions

```
function addTen(parameters) {
  // instructions go here!
}

addTen(someInput);
```

CODE CHRYSALIS コードクリサリス

Breaking Down Functions

```
function addTen(parameters) {
    instructions go here!

Input);
```

The *function* keyword is a special word that lets the JavaScript engine know that we are creating a function.

Breaking Down Functions

```
function addTen(parameters) {
  // instructions go here!
}

addTen
```

addTen is the name of our function.

If we want to invoke this function in the future, we need to refer to it by its given name (addTen).

Breaking Down Functions

```
function addTen(parameters) {
  // instructions go here
}

addTen(someInput
```

The parentheses contain our function's *parameters*. These are (optional) inputs that our function takes when we use it.

CODE CHRYSALIS コードクリサリス

Breaking Down Functions

```
function addTen(parameters) {
  // instructions go here!
}
```

The curly brackets contain the *body* of our function. These are the instructions that we want our computer to follow.

(Invoking) Running Functions

```
function addTen(parameters) {
  // instructions go here!
}

addTen(someInput);
```

To invoke a function, use the function's name.

Note that there is NO function keyword used here.

CODE CHRYSALIS コードクリサリス

(Invoking) Running Functions

```
function addTen(parameters) {
  // instructions go here!
}

addTen(someInput);
```

Parentheses follow the function name.

The parentheses will hold the arguments (inputs) and will instruct the computer to invoke our function.

CODE CHRYSALIS コードクリサリス

Function Signature

```
function addTen(number) {
  return number + 10;
}
```

A *function signature* is used to describe a function. For our class, a *function signature* describes the following:

- parameters and types
- return type

For example:
`addTen` is a function that takes 1 parameter (a `number`) and returns a `number`.

CODE CHRYSALIS コードクリサリス

Function Signature

```
function addTen(number) {
  return number + 10;
}
```

When describing a *function signature*, use this pattern:

`functionName` is a function that takes n parameters (a TYPE and a TYPE) and returns a TYPE.

Function Signature (more examples)

```
function divide(dividend, divisor) {
  return dividend / divisor;
}
```

```
function getGreeting(greeting, name) {
  return greeting + " " + name;
}
```

```
function printHello(name) {
  console.log("Hello, " + name);
}
```

Function Signature (more examples)

```
function divide(dividend, divisor) {
  return dividend / divisor;
}
```

```
function getGreeting(greeting, name) {
  return greeting + " " + name;
}
```

```
function printHello(name) {
  console.log("Hello, " + name);
}
```

Function Signature (more examples)

```
function divide(dividend, divisor) {
  return dividend / divisor;
}
```

`divide` is a function that takes two parameters (a number and a number) and returns a number.

```
function getGreeting(greeting, name) {
  return greeting + " " + name;
}
```

```
function printHello(name) {
  console.log("Hello, " + name);
}
```

Function Signature (more examples)

```
function divide(dividend, divisor) {
  return dividend / divisor;
}
```

`divide` is a function that takes two parameters (a number and a number) and returns a number.

```
function getGreeting(greeting, name) {
  return greeting + " " + name;
}
```

`getGreeting` is a function that takes two parameters (a string and a string) and returns a string.

```
function printHello(name) {
  console.log("Hello, " + name);
}
```

Function Signature (more examples)

```
function divide(dividend, divisor) {
  return dividend / divisor;
}
```

`divide` is a function that takes two parameters (a number and a number) and returns a number.

```
function getGreeting(greeting, name) {
  return greeting + " " + name;
}
```

`getGreeting` is a function that takes two parameters (a string and a string) and returns a string.

```
function printHello(name) {
  console.log("Hello, " + name);
}
```

`printHello` is a function that takes one parameter (a string) and returns undefined.

```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

REVIEW

```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

REVIEW

1. What is this function's name?

```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

REVIEW

1. What is this function's name?
2. What are this function's parameters?

```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

REVIEW

1. What is this function's name?
2. What are this function's parameters?
3. What does this function return?

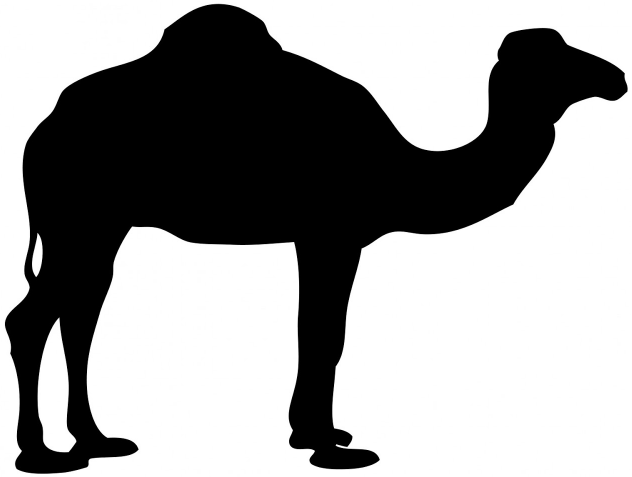
```
function addTen(number) {
  return number + 10;
}
```

CODE CHRYSALIS コードクリサリス

Things Developers Say:

Make your function names meaningful

JavaScript uses camelCase because you can't use spaces.



CODE CHRYSALIS コードクリサリス

What's wrong with this function?

```
function AddTEN(f,a){
return f+a}
```

What's wrong with this function?

```
function AddTEN(f,a){
return f+a}
```

What's wrong with this `function`?

```
function AddTEN(f,a){
return f+a}
```

Remember - you aren't just writing your code just for the machine… you are writing it for OTHER HUMANS. It should be readable.

CODE CHRYSALIS コードクリサリス

Remember to make it readable!

```
function AddTEN(f,a){
return f+a}
```

Not readable.

```
function sum(num1, num2) {
  return num1 + num2;
}
```

Readable.

What will this function return?

REVIEW

```
function subtract(num1, num2) {
  return num1 - num2;
}

subtract(2, 4);
```

CODE CHRYSALIS コードクリサリス

REVIEW TIME

1. What is a function?
2. How do you invoke a function?
3. What is the difference between a function *declaration* and a function *invocation*?
4. What is the keyword you need to output a value from a function?
5. What does a function return if it does NOT have a return statement?

CODE CHRYSALIS コードクリサリス

Activity

Follow the exercises for
Intro to Functions!

CODE CHRYSALIS コードクリサリス