JS FUNCTIONS

A First Course On Functions

OBJECTIVES

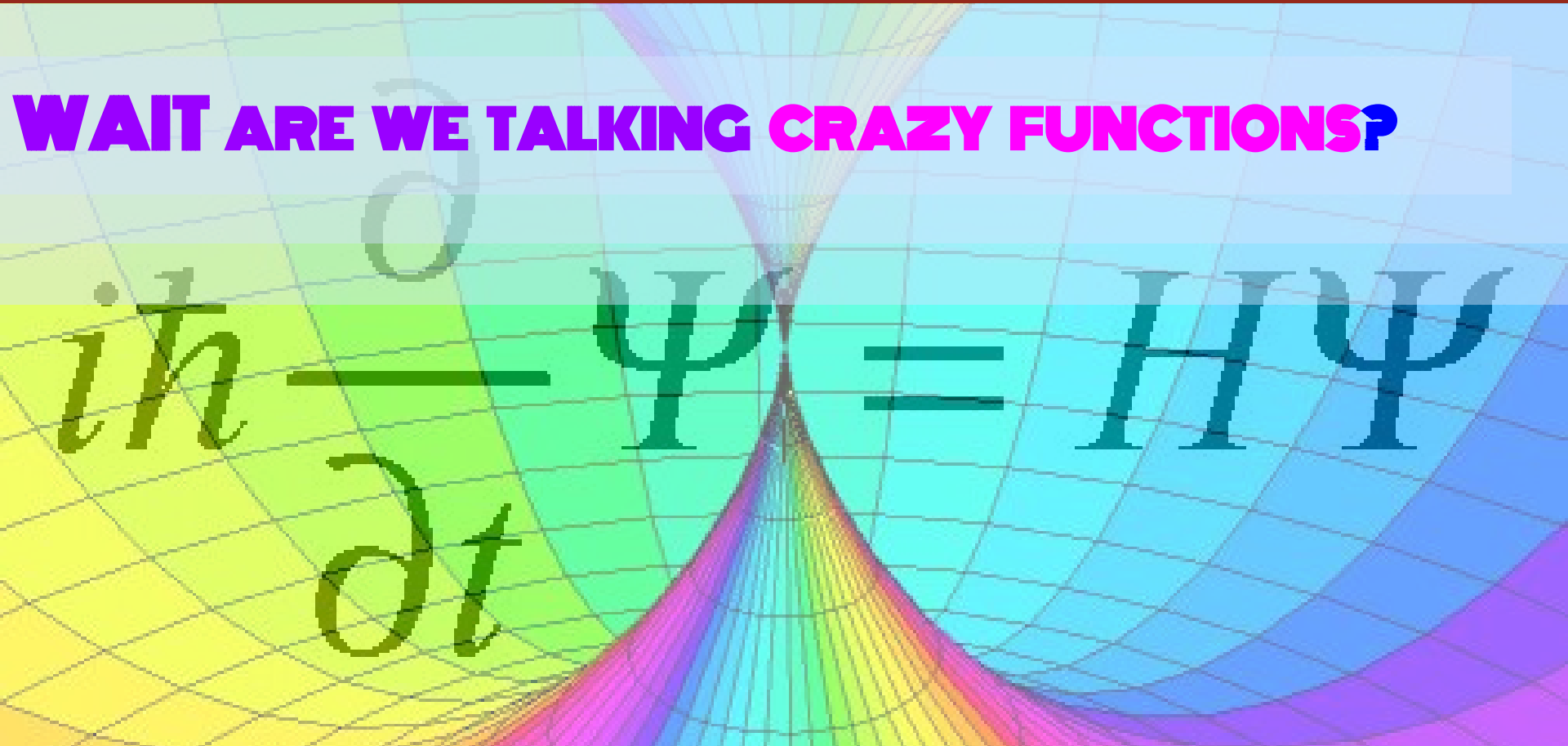
- Explain the role of functions in applications.
- Discuss and use small functions
- Identify and use the parts of a function: *name, arguments, parameters and returns*.

galvanize

WHY FUNCTIONS?

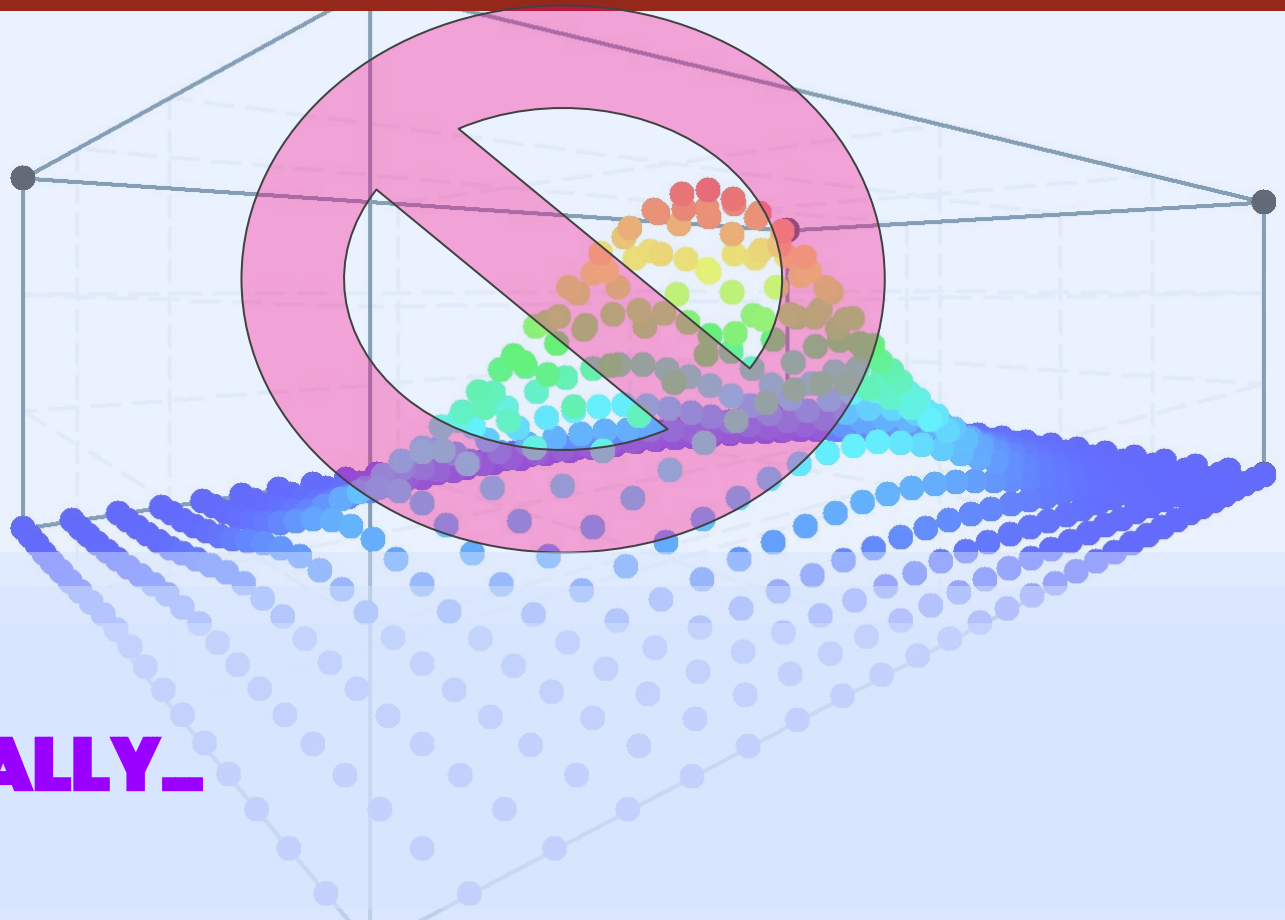
galvanize

WAIT ARE WE TALKING CRAZY FUNCTIONS?



WHY FUNCTIONS?

galvanize



NOT REALLY...

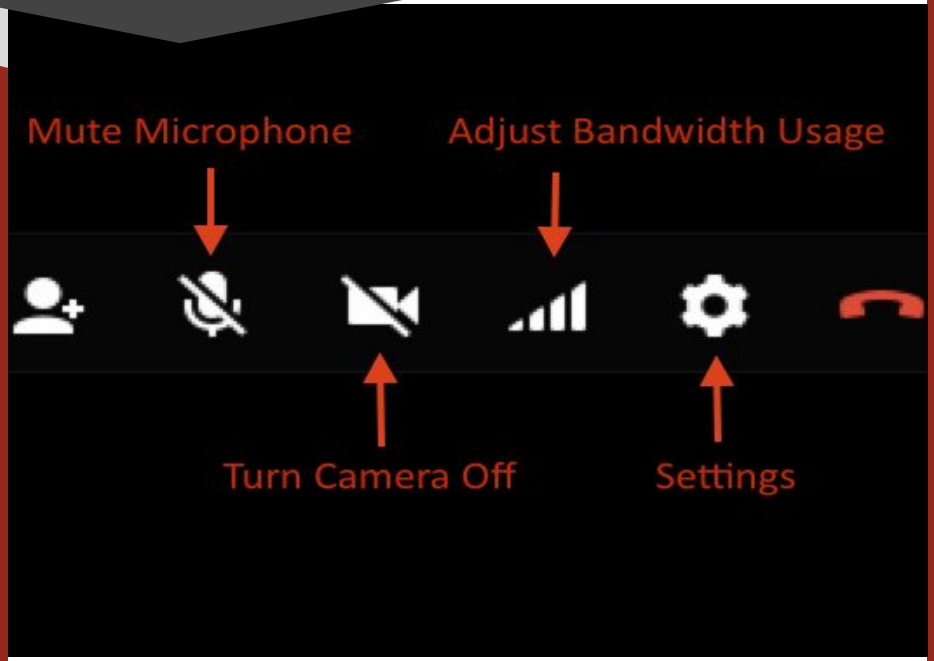
WHY FUNCTIONS?

All the things you do on the web require functions to describe the interactions.

WHY FUNCTIONS?

galvanize

Functions help us control all our favorite apps… like **Google Hangout**.



WHY FUNCTIONS?

galvanize

Grabbing or sending data

SEARCHUSERS

SHOWACTIVITY

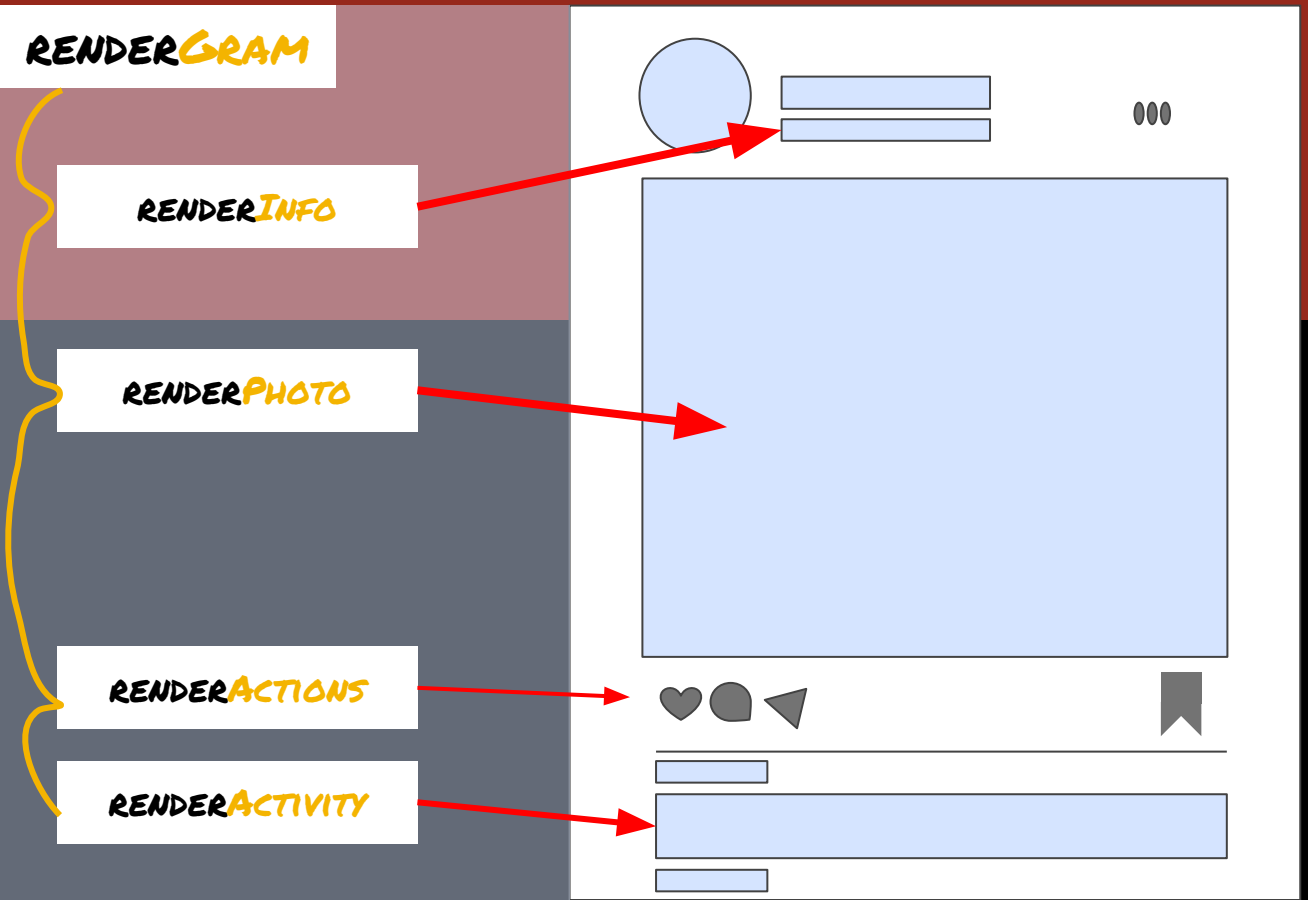
SHOWHOME

ADDGRAM

SHOWFEED

WHY FUNCTIONS?

galvanize



A typical gram on IG requires many functions to render the various parts of a single gram.

WHY FUNCTIONS?

galvanize

Functions help us do many of the things we need for our apps

- Displaying data
- Liking a gram or ending a Google Hangout
- Loading activity and feed data

FUNCTIONS IN THE WILD

galvanize

- When you start writing an email there is a function that creates a new draft and saves it to gmail's servers
  - `createDraftResponse(sendees, previousEmail)`
- As you write an email there might be a function that updates your changes to a draft.
  - `updateDraft(draftInfo, newText);`

FUNCTIONS IN THE WILD

galvanize

- As you scroll through your twitter timeline and reach the bottom the current tweets there might be function that grabs the next page of tweets.
  - **fetchNext(currentPage)**
- As you write a new tweet there might be function check the count of the number of characters you've written.
  - **countCharacters(tweetText);**

FUNCTIONS IN THE WILD

galvanize

IN THE WILD functions allow us to control the general behavior and experience of our application.

USING FUNCTIONS

galvanize

BEFORE WE KNOW FUNCTIONS LET'S TRY TO PLAY WITH SOME BY USING COPY/PASTE.

USING FUNCTIONS

galvanize

Copy and paste the following into your console.

```
function fullName(first, last) {
   return first + ", " + last;
}
```

USING FUNCTIONS

galvanize

Try running the following code to use the fullName function.

```
fullName("Jane", "Doe");
// => "Doe, Jane";

// You can save this to a variable for use later.
var bestFriend = fullName("Jane", "Doe");

bestFriend
// => "Doe, Jane";
```

USING FUNCTIONS

galvanize

Copy and paste the following into your console.

```
function capitalize(word) {
  var letters     = word.split("");
  var firstLetter = letters[0];

  letters[0] = firstLetter.toUpperCase();

  return letters.join("");
}
```

USING FUNCTIONS

galvanize

Try running the following code to use the capitalize function.

```
capitalize("delmer");
// => "Delmer"

capitalize("sam")
// => "Sam"
```

USING FUNCTIONS

galvanize

Copy and paste the following into your console.

```
function isOdd(num) {
  return num % 2 === 1;
}
```

USING FUNCTIONS

galvanize

Try running the following code to use the isOdd function.

```
isOdd(2);
// => false

isOdd(4)
// => false
```

Exercises

galvanize

- Run fullName with your first and last name.
- Verify that the isOdd return true for odd numbers and false otherwise.
- What happens when you run
  - `capitalize("i went to the store.")`

FUNCTION PARTS

galvanize

Now that we've played with some functions let's learn their anatomy.

FUNCTION PARTS

FUNCTION NAME

```
function fullName(first, last) {
  // the function body
  return last + ", " + first;
}
```

To create a function we start with a `function` keyword. This signals the start of a **function declaration**.

It's immediately followed by the **function name** before the parens, ( ).

FUNCTION PARTS

galvanize

```
function fullName(first, last) {
  // the function body
  return last + ", " + first;
}
```

PARAMETERS

PARAMETERS

The parens after the name list out the names for the values being provided to a function. Here we have `first` and `last`.

You decide both what to call them and how many there will be.

Depending on how much you do in a function you may have many or even none at all.

FUNCTION PARTS

galvanize

```
function fullName(first, last) {
  // the function body
  return last + ", " + first;
}
```

Everything between curly's
is part of the body

FUNCTION BODY

All code in the curly braces after the function parameters makes up the function body. You can think of this as a separate little world.

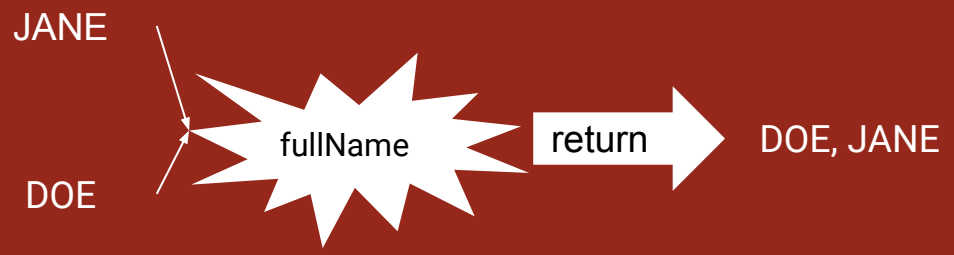
FUNCTION PARTS

galvanize

```
function fullName(first, last) {
  // the function body
  return last + ", " + first;
}
```

The return statement gives back a value

THE **RETURN** STATEMENT



FUNCTION PARTS

galvanize

FUNCTION NAME

```
function fullName(first, last) {
  // the function body
  return last + ", " + first;
}
```

RETURN VALUE

PARAMETERS

Functions

```
function greet() {
  alert("Welcome!");
  alert("Nice to meet you");
}
```

Our first function declaration

- It has *no parameters*
- It has two alerts in *its body*

What is our function's name?

Calling Functions

galvanize

```
// This runs greet!
greet();
```

AFTER WE DEFINE a function it can be run using its name followed by parens.

Calling Functions

```
greet();
// ?
```

Note how this function doesn't have a return in its body. It just alerts twice and doesn't return anything. It doesn't even need inputs!

Function Params

```
function greetUser(userName) {
  alert("Welcome, " + userName);
  alert("Nice to meet you, " + userName);
}
```

```
greetUser("Jane");
greetUser("John");
```

Lets modify our greet function take a name input.

Now we can greet specific people.

QUICK RECAP

There are a few things we learned

- Define a function before we can use it.
- Call it using its name followed by parens
- Any values the function needs should be provided in the parens
  - `greet("Jane")` provides the “Jane” to our function

Function Scope

```
var salutation = "Welcome, ";

function greetUser(userName) {
  alert(salutation + userName);
  alert("Nice to meet you, " + userName);
}
```

```
greetUser("Jane");
greetUser("John");
```

Let's move our salutation into a variable outside our function

Try it out

Function Scope

```
var salutation = "Welcome, ";

function greetUser(userName) {
  alert(salutation + userName);
  alert("Nice to meet you, " + userName);
}
```

```
greetUser("Jane");
greetUser("John");
```

This is a cool feature you might try to use if you have a value used in multiple functions.

Exercises

```
var salutation = "Welcome, ";

function greetUser(userName) {
  alert(salutation + userName);
  alert("Nice to meet you, " + userName);
}
```

- Run the greetUser function with your name.
- Change the salutation to "Hello".
- Run the greetUser function with your name.

What did you notice?

BONUS: Exercise

```
var salutation = "Welcome, ";

function greetUser(userName) {
  alert(salutation + userName);
  alert("Nice to meet you, " + userName);
}
```

- Move the "`Nice to meet you, `" to a variable called `compliment` outside the function
- Redefine greetUser to use the `compliment`.
- Run greetUser with your name
- Change the `complement` to "`you're awesome`"
- Run it with your name

FUNCTION RETURNS

galvanize

```
function fullName(first, last) {
   return first + " " + last;
}
```

- Let’s write a function that computes something and returns the value.
- This function takes the first name and last name adds them together

USING RETURNS

galvanize

```
function fullName(first, last) {
   return first + " " + last;
}
```

```
var myName     = fullName("Delmer", "Reed");
var friendName = fullName("Jane", "Doe");
```

We can save the return value into a variable like myName or friendName.

Try it out for yourself.

FUNCTION PARAMS VS ARGUMENTS

galvanize

When we define a function the **NAMES OF THE PROVIDED VALUES** are called **PARAMETERS.**

```
function fullName(first, last) {
   return first + " " + last;
}
```

```
var myName     = fullName("Delmer", "Reed");
var friendName = fullName("Jane", "Doe");
```

FUNCTION
PARAMS VS ARGUMENTS

galvanize

When we define a function the **NAMES OF THE PROVIDED VALUES** are called **PARAMETERS.**

```
function fullName(first, last) {
   return first + " " + last;
}
```

```
var myName     = fullName("Delmer", "Reed");
var friendName = fullName("Jane", "Doe");
```

When we call the function the values provided are called arguments

QUICK RECAP

There are a few things we learned

- You can use variables defined outside your in your function.
- You can return a value from a function and save it in a variable for later.

THE NEXT SECTION IS BONUS...

Function Inputs

Typically parameters are just specified positionally (the order you chose in the definition)

```
function fullName(first, last) {
   return first + " " + last;
}
```

First comes the first name

Second comes last name

Function Inputs

- When we define a function with two to three parameters this is fine to use positional parameters. *It's not too hard to remember which order they go in.*

```
function fullName(first, last) {
   return first + " " + last;
}
```

```
fullName("Jane", "Doe");
```

Not

~~fullName("Doe", "Jane");~~

Function Inputs

However, If our function has more than 3 params you might want to consider rewriting the function to use one **parameter object** with the expected attributes.

```
function fullName(user) {
   return user.first + " " + user.middle +  " "  + user.last;
}
```

Function Inputs

Using a parameter object allows us to avoid worrying about the order.

```
function fullName(user) {
  return user.first + " " + user.middle +  " "  + user.last;
}


fullName({ first: "John", middle: "Joe", last: "Doe" });
// => "John Joe Doe"
```

END OF BONUS SECTION...

Assessment

```
var greeting = "Hello";

function greet(userName) {
  alert(greeting + " " + userName);
}
```

Assessment

- Use the `greet` function to do the following:
  - `greet("Jane");`
  - `greet("John");`
  - `greet("Jane", "John");`
  - `greet();`

Assessment

- Identify the following in the function below: name, parameters, body, and return values.
- Describe what the function does.

```
function fullName(first, last) {
  alert(first + " " + last);
};
```

Assessment

- Identify the following in the function below: name, parameters, body, and return values.
- Describe what the function does.

```
function add(a, b) {
  return a + b;
}
```

Assessment

- Use the `add` function to evaluate the following:
  - What is the value of `numOne`
  - What is the value of `numTwo`

```
var numOne = add(6, 5);
var numTwo = add(9, 10);
```

Assessment

- Identify the following in the function below: name, parameters, body, and return values.
- Describe what the function does.
  - When will it return true?

```
function canDrink(person) {
  if (person.age >= 21) {
    return true;
  } else {
    return false;
  }
}
```

Assessment

- Identify the following in the function below: name, parameters, body, and return values. NOTE: The % operation returns the remainder after division.
- Describe what the function does.
  - When will it return true?

```
function isEven(number) {
  if (number % 2 === 0) {
    return true;
  } else {
    return false;
  }
}
```

Assessment

- **CHALLENGE:** Identify the following in the function below: name, parameters, body, and return values.
- Describe what the function does. When will it return true?

```
function contains(items, value) {
  for (var i = 0; i < items.length; i += 1) {
     if (items[i] === value) {
       return true;
     }
  }
  return false;
}
```

Assessment

- Use the function below to evaluate the following in the developer JS console:
  - `add(5, 2);`
  - `add(9, 7);`
  - `add(2 + 3, 2);`
  - `add(5 + 4, 7);`
  - `add(5, add(2, 3));`

```
function add(a, b) {
  console.log("Adding values:", a, b);
  return a + b;
}
```

Assessment

- Use the function below to evaluate the following:
  - `canDrink({ age: 32 })`

```
function canDrink(person) {
  if (person.age >= 21) {
    return true;
  } else {
    return false;
  }
}
```

Assessment

- Use the function below to determine the following:
  - What **argument** should you provide to `canDrink` to have it return false?

```
function canDrink(person) {
  if (person.age >= 21) {
    return true;
  } else {
    return false;
  }
}
```

Assessment

- Declare a function named `addTen` that takes a number, adds ten, and then returns the sum.
- Compute the following:
  - `addTen(24);`
  - `addTen(39);`
  - `addTen(12);`
  - `addTen(addTen(10));`

Assessment

- Declare a function named `subtract` that computes the difference between two numbers.
- Compute the following:
  - `subtract(10, 2);`
  - `subtract(2, 10);`
  - `subtract(100, 50);`
  - `subtract(100);`
  - `subtract();`

Assessment

- Declare a function named `initials` that takes the first and last name of a person and returns their initials:
  - `initials("John", "Doe") // => "JD"`
  - `initials("Delmer", "Reed") // => "DR"`
- Compute the following:
  - `initials("Jane", "Austin");`
  - `initials("Robert", "Frost");`

Assessment

- Declare a function named `isOdd` that takes a number and determines if it is odd:
- Compute the following:
  - `isOdd(1);`
  - `isOdd(21);`
  - `isOdd(2);`
  - `isOdd(34);`

Assessment

- Declare a function named `last` that takes an array and returns the last value without modifying the array.
- Compute the following:
  - `last([1, 2, 3, 4, 5]);`
  - `last([8]);`
  - `last([9, 3, 1]);`

Assessment

- Declare a function named `isLarge` that an object with a size property that can be one of the following values and returns true or false if the size is large: `"small"`, `"medium"`, or `"large"`.
  - `isLarge({ size: "small" }); // => false`
  - `IsLarge({ size: "large"}) // => true`
- Compute the following:
  - `isLarge({ color: "green", size: "small"});`
  - `isLarge({ type: "V neck", size: "large"});`
  - `isLarge({ make: "Ford", model: "Focus", size: "large"});`
  - `isLarge({ make: "Honda", model: "Accord", size: "small"});`