JS FUNCTIONS

Objects and Functions

Objectives

- Discuss and use function declarations
- Identify the parts of a function: name, arguments, parameters and returns.
- Review and apply objects for problem solving.

WHY?

- Why create functions?
  - They help us wrap up common procedures we've written into readable and reusable pieces
  - They force us to give a name to refer to something we are doing.
    - createUser: might create a user for an application
    - authenticateUser: might attempt to verify a users credentials
    - renderPosts: might attempt to display facebook posts
    - validateBlogPost: might check that post has a title and body for a blog.

Functions in the wild

- 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

- 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)
- Since each tweet is very similar in their structure there might be one rendering function that renders a tweet
  - renderTweet(tweetData)
- As you write a new tweet there might be function check the count of the number of characters you've written.
  - countCharacters(tweetText);

Functions

To create a function we start by declaring it using a `function` keyword. This communicates that we are attempting to define a new function.

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

Functions

To create a function we start by declaring it using a `function` keyword. This communicates that we are attempting to define a new function.

Function name

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

Functions

To create a function we start by declaring it using a `function` keyword. This communicates that we are attempting to define a new function.

Function name

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

parameters

Functions

To create a function we start by declaring it using a `function` keyword. This communicates that we are attempting to define a new function.

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

Everything between curly's is part of the body

Functions

To create a function we start by declaring it using a `function` keyword. This communicates that we are attempting to define a new function.

Function name

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

The return value

parameters

Functions

Our first function declaration

- It has no parameters
- It just runs a series of statements we've defined

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

Calling Functions

To call our function

- We must first declare it

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

- Then we can call it

```
greet();
```

Function Params

Lets modify our greet function take a name to greet

- We must first declare it

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

- Then we can call it

```
greetUser(“Jane”);
greetUser(“John”);
```

Function Scope

Lets modify our greet function to use variables outside the body

- We redefine greetUser with the following

```
var salutation = “Welcome, ”;

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

- Then we can call it

```
greetUser(“Jane”);
```

Exercise

- Without changing anything else set the salutation to “Hello”
- What do you see when you run greetUser(“Jane”);
- Bonus: Move the “`Nice to meet you, ”` to a variable called `compliment` outside the function, and redfine greetUser to use the `compliment`.
- Bonus: Change compliment to “You’re awesome, “.

Function: Returns

- Let’s write a function that computes something and returns the value.

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

- This function takes the first name and last name adds them together

Function: Using Returns

- Let’s write a function that computes something and returns the value.

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

- Now when we call the function we can use the return value in our program!

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

Exercise: Using Returns

- Now when we call the function we can use the return value in our program!

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

- Try the above using our fullName function
  - What is the value of myName
  - What is the value of friendName

Function: Arguments vs Params

- When we define a function the names of the values are called parameters.

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

- When we call the function the values provided are called arguments

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

Function Inputs

- When we define a function with two to three parameters this is fine to use positional parameters.

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

Function Inputs

- If our inputs are going have more than 3 arguments you might want to consider rewriting the function to use one parameter object with attributes

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

Function Inputs

- If our inputs are going have more than 3 arguments you might want to consider rewriting the function to use one parameter object with attributes

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


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

Assessment

- Identify the following in the function below: name, parameters, body, and return values.
  - Hint: unless a function has at least one return it will always return `undefined`.
  - Describe what the function does.

```
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 })`
  -
    ```
    var john = { age: 22 };
    canDrink(john);
    ```

```
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 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"});`