JS Functions

What are functions?

- A reusable section of code that has a purpose and a name
- The bread and butter of JS
- They can associate names with subprograms

What are functions?

Creating new words is normally bad practice, though fun

It is essential in programming!

- By giving a name to a part of our program
    - We make it flexible, reusable and more readable
- They become **executable** sub-programs

How do they work?

- We **define** a function
- We **call** (or **execute**) it
    - When we want the code within the function to run

What can functions do?

Anything!

- Calculations
- Animations
- Change CSS
- Change, add, or delete elements on the page
- Speak to a server (e.g. an API)
- ...

Declaring Functions

```
// A Function Declaration

function sayHello () {
  console.log( "Hello" );
}

// A Function Expression

var sayHi = function () {
  console.log( "Hi" );
};
```

Calling Functions

```
function sayHello () {
  console.log( "Hello!" );
}

sayHello(); // The callsite
```

Calling Functions

```
var sayHello = function () {
  console.log( "Hello!" );
}

sayHello(); // The callsite
```

Exercise

Create a Roll Virtual Dice Function

Optional: Receive a parameter to decide how many sides the Dice actually has (e.g. a 12-side dice)

Parameters | | Arguments

They aren't dynamic... yet! This brings us to parameters or arguments

Parameters (or **arguments**) allow us to provide a function with extra data or information

This is what makes a function flexible!

Parameters || Arguments

```
function sayHello ( name ) {
  var greeting = "Hello " + name;
  console.log( greeting );
}

sayHello( "Bill" );

sayHello( "Jane" );

sayHello(); // ???
```

Parameters || Arguments

```
function multiply (x, y) {
  console.log( x * y );
}

multiply( 5, 4 );

multiply( 10, -2 );

multiply( 100, 0.12 );
```

Some Pseudocode

```
changeTheme function
    RECEIVE a themeChoice ("light" or "dark")
    IF themeChoice === "light"
        CHANGE the body background to "white"
        CHANGE the text color to "black"
    ELSE
        CHANGE the body background to "black"
        CHANGE the text color to "white"

moveToLeft function
    RECEIVE an element to animate
    STORE the current left position as currentLeft
    STORE the new left position, as desiredLeft, by adding 100px to currentLeft
    UPDATE the left position of the provided element to be desiredLeft
```

Passing in Variables

```
var addTwoNumbers = function (x, y) {
  return x + y;
};

var firstNumber = 10;

addTwoNumbers( firstNumber, 4 );
addTwoNumbers( firstNumber, 6 );
```

Parameters vs. Arguments

- A parameter is where the data is received
- An argument is where the data is passed in (the actual data)

```
function doSomething (parameter) {}

doSomething(argument);
```

Return Values

Sometimes your function calculates something and you want the result!

Return values allow us to do just that

We can store the result of calculations with return values

Return Values

```
function squareNumber ( x ) {
  var square = x * x;
  return square;
};

var squareOfFour = squareNumber( 4 );
```

Return Values

```
function squareNumber ( x ) {
  var square = x * x;
  return square;
};

var squareOfFour = squareNumber( 4 );

var squareOfTwelve = squareNumber( 12 );

squareNumber(8) + squareNumber(11);

squareOfFour + squareOfTwelve;
```

Return Values

```
function square ( x ) {
  return x * x;
};

function double ( x ) {
  return x * 2;
}

var result = double( square( 5 ) );
```

Return Values

```
function sayHello () {
  return "No.";
  console.log( "Hi!" );
};

sayHello();
```

- A return value means that a function has a result
- It is **always** the **last line** that executes

Return Values

```
var addTwoNumbers = function ( x, y ) {
  return x + y;
};

var firstNumber = 10;

addTwoNumbers( firstNumber, 4 );
addTwoNumbers( firstNumber, 6 );
```

Return Values - Callbacks

```
function runCallback ( cb ) {
  // Wait a second
  cb();
}

function delayedFunction () {
  console.log("I was delayed");
}

runCallback( delayedFunction );
```

Function Guidelines

Follow the **F.I.R.S.T** Principle:

- **F**-ocussed
- **I**-ndependent
- **R**-eusable
- **S**-mall
- **T**-estable

*Also, make it error-tolerant

In-class Exercise / Homework

Do the exercises found here

Resources

- Speaking JavaScript
- JavaScript.info
- Eloquent JavaScript
- CodeBurst
- MDN