FUNCTIONS

FUNCTIONS

IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

RECAP

FUNCTIONS IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

YOU CAN STORE A FUNCTION IN A VARIABLE

RECAP

FUNCTIONS IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

YOU CAN STORE A FUNCTION IN A VARIABLE

YOU CAN HAVE A FUNCTION RETURN A FUNCTION

RECAP

FUNCTIONS IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

YOU CAN STORE A FUNCTION IN A VARIABLE

YOU CAN HAVE A FUNCTION RETURN A FUNCTION

YOU CAN HAVE A FUNCTION TAKE IN A FUNCTION AS AN ARGUMENT

RECAP

FUNCTIONS IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

1. YOU CAN STORE A FUNCTION IN A VARIABLE
2. YOU CAN HAVE A FUNCTION RETURN A FUNCTION
3. YOU CAN HAVE A FUNCTION TAKE IN A FUNCTION AS AN ARGUMENT

THESE 3 PROPERTIES COLLECTIVELY ARE CALLED "FIRST CLASS FUNCTIONS"

RECAP

FUNCTIONS IN JAVASCRIPT CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

JAVASCRIPT IS A PROGRAMMING LANGUAGE THAT SUPPORTS FUNCTIONS

1

YOU CAN FUNCTION IN A VARIABLE

2

YOU CAN FUNCTION AS FUNCTION

3

YOU CAN TAKE IN A FUNCTION AS AN ARGUMENT

THESE 3 PROPERTIES COLLECTIVELY ARE CALLED "

RECAP

JAVASCRIPT IS A PROGRAMMING LANGUAGE THAT SUPPORTS

THESE 3 PROPERTIES COLLECTIVELY ARE CALLED "

FUNCTIONS

IN THIS WAY, JAVASCRIPT HAS A STRONG FUNCTIONAL PROGRAMMING FLAVOUR TO IT.

RECAP

OBJECTS

OBJECTS

"AN OBJECT IS A SET OF KEY-VALUE PAIRS, WHERE THE VALUES CAN ALSO BE FUNCTIONS"

EACH KEY-VALUE PAIR IS CALLED A PROPERTY OF THE OBJECT

RECAP

EXAMPLE 38: HOISTING, AND FUNCTION LITERALS VERSUS DECLARED FUNCTIONS

"DECLARED FUNCTIONS" ARE JUST REGULAR FUNCTIONS CREATED LIKE THIS

```
function declaredFunction() {
  console.log("Declared function..I exist");
}
```

"FUNCTION LITERALS" ARE THE OTHER TYPE OF FUNCTIONS IN JAVASCRIPT

"FUNCTION LITERALS" ARE THE OTHER TYPE OF FUNCTIONS IN JAVASCRIPT

```
var someFunction = function() {
    console.log("Inside a function literal - I ex
}
```

"FUNCTION LITERAL" IS JUST A FANCY NAME FOR A FUNCTION THAT IS CREATED LIKE THIS

"FUNCTION EXPRESSION" IS WHAT THE RIGHT HAND SIDE OF THIS ASSIGNMENT IS CALLED.

"FUNCTION LITERALS" ARE THE OTHER TYPE OF FUNCTIONS IN JAVASCRIPT

```
var someFunction = function() {
  console.log("Inside a function literal - I ex
}
```

"FUNCTION LITERAL" IS JUST A FANCY NAME FOR A FUNCTION THAT IS

"FUNCTION EXPRESSION" IS WHAT THE RIGHT HAND SIDE OF THIS ASSIGNMENT IS

(BTW, THIS IS ALSO AN EXAMPLE OF AN "ANONYMOUS FUNCTION" BECAUSE IT DOES NOT HAVE AN EXPLICIT NAME)

"DECLARED FUNCTIONS" ARE JUST REGULAR FUNCTIONS CREATED LIKE THIS

```
function declaredFunction() {
  console.log("Declared function..I exist");
}
```

"FUNCTION LITERALS" ARE THE OTHER TYPE OF FUNCTIONS IN JAVASCRIPT

```
var someFunction = function() {
  console.log("Inside a function literal - I exis
}
```

THE TWO TYPES OF FUNCTIONS ARE ALMOST IDENTICAL, EXCEPT IN ONE RESPECT..

THE TWO TYPES OF FUNCTIONS ARE ALMOST IDENTICAL, EXCEPT

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

THE JAVASCRIPT INTERPRETER PROCESSES ALL DECLARED FUNCTIONS AS THE PAGE IS LOADING

THE JAVASCRIPT INTERPRETER PROCESSES ALL DECLARED FUNCTIONS AS THE PAGE IS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

THIS IS BEFORE ANY JAVASCRIPT CODE HAS ACTUALLY RUN

THE JAVASCRIPT INTERPRETER PROCESSES ALL DECLARED FUNCTIONS AS THE PAGE IS

THIS IS BEFORE ANY JAVASCRIPT CODE HAS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY **FUNCTION LITERALS COME INTO EXISTENCE.**

THE JAVASCRIPT INTERPRETER PROCESSES ALL DECLARED FUNCTIONS AS THE PAGE IS

THIS IS BEFORE ANY JAVASCRIPT CODE HAS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

FUNCTION LITERALS ONLY COME INTO EXISTENCE WHEN THE CORRESPONDING FUNCTION EXPRESSION IS EVALUATED

THE JAVASCRIPT INTERPRETER PROCESSES ALL DECLARED FUNCTIONS AS THE PAGE IS

THIS IS BEFORE ANY JAVASCRIPT CODE HAS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

FUNCTION LITERALS ONLY COME INTO

AND FUNCTION EXPRESSIONS CAN ONLY BE EVALUATED WHEN THE JAVASCRIPT CODE RUNS!

THE JAVASCRIPT INTERPRETER PROCESSES
ALL DECLARED FUNCTIONS AS THE PAGE IS

THIS IS BEFORE ANY JAVASCRIPT CODE HAS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE
ANY **FUNCTION LITERALS COME INTO EXISTENCE.**

FUNCTION LITERALS ONLY COME INTO

AND FUNCTION EXPRESSIONS CAN ONLY BE
EVALUATED **WHEN THE JAVASCRIPT CODE RUNS!**

THE JAVASCRIPT INTERPRETER PROCESSES
ALL DECLARED FUNCTIONS AS THE PAGE IS

THIS IS BEFORE ANY JAVASCRIPT CODE HAS

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY FUNCTION LITERALS COME INTO EXISTENCE.

FUNCTION LITERALS ONLY COME INTO

AND FUNCTION EXPRESSIONS CAN ONLY BE
EVALUATED WHEN THE JAVASCRIPT CODE RUNS!

DECLARED FUNCTIONS ALL COME INTO EXISTENCE BEFORE ANY

THIS IS EQUIVALENT TO ALL OF THE DECLARED FUNCTIONS BEING "HOISTED" TO THE TOP OF THE JAVASCRIPT CODE

THIS IS EQUIVALENT TO ALL OF THE

DECLARED FUNCTIONS BEING "HOISTED"

TO THE TOP OF THE JAVASCRIPT CODE

ALL DECLARED FUNCTIONS, NO MATTER WHERE THEY APPEAR IN THE CODE, ARE "HOISTED" TO THE TOP

ALL DECLARED FUNCTIONS, NO MATTER WHERE THEY APPEAR

DECLARED FUNCTIONS CAN BE USED ANYWHERE IN CODE

FUNCTION LITERALS CAN BE USED ONLY AFTER BEING EVALUATED

DECLARED FUNCTIONS

THIS IS THE ONLY SIGNIFICANT DIFFERENCE BETWEEN THE TWO

FUNCTION LITERALS

HOISTING IN ACTION

```
function functionExample() {
  declaredFunction();

  try {
    someFunction();
  }
  catch(error) {
   console.log(error.message);
  }

    var someFunction = function() {
      console.log("Inside a function literal - I exist");
    }
    someFunction();

  }


  function declaredFunction() {
    console.log("Declared function..I exist");
  }
```

HOISTING IN ACTION

```
function functionExample() {
  declaredFunction();

  try {
    someFunction();
  }
  catch(error) {
   console.log(error.message);
  }

    var someFunction = function() {
      console.log("Inside a function literal - I exist");
    }
    someFunction();

  }


  function declaredFunction() {
    console.log("Declared function..I exist");
  }
```

HOISTING IN ACTION

A DECLARED FUNCTION RIGHT AT THE END OF THE CODE

```
function declaredFunction() {
  console.log("Declared function..I exist");
}
```

HOISTING IN ACTION

SOME OTHER FUNCTION THAT APPEARS BEFORE THAT DECLARED FUNCTION..

```
function functionExample() {
  declaredFunction();

  try {
    someFunction();
  }
  catch(error) {
    console.log(error.message);
  }

  var someFunction = function() {
    console.log("Inside a function literal - I exist");
  }
  someFunction();

}


function declaredFunction() {
  console.log("Declared function..I exist");
}
```

HOISTING IN ACTION

```
declaredFunction();
```

..IS ABLE TO SUCCESSFULLY CALL THE DECLARED FUNCTION

```
Declared function..I exist
```

```
function declaredFunction() {
  console.log("Declared function..I exist");
}
```

HOISTING IN ACTION

OUR FUNCTION ALSO CREATES A FUNCTION LITERAL..

```
var someFunction = function() {
  console.log("Inside a function literal - I exist");
}
```

HOISTING IN ACTION

```
try {
  someFunction();
}
catch(error) {
 console.log(error.message);
}
```

X

BUT THIS FUNCTION LITERAL CAN ONLY BE CALLED AFTER ITS FUNCTION EXPRESSION HAS BEEN

```
someFunction is not a function
```

HOISTING IN ACTION

```
try {
  someFunction();
}
catch(error) {
 console.log(error.message);
}
```

X

WE NEED A TRY/CATCH BLOCK TO HANDLE THIS ERROR (MORE LATER:-))

```
var someFunction = function() {
  console.log("Invoked function literal - I exist");
}
someFunction();
```

BUT THIS FUNCTION LITERAL CAN ONLY BE CALLED AFTER ITS FUNCTION EXPRESSION HAS BEEN

```
someFunction is not a function
```

```
function declaredFunction() {
  console.log("Declared function..I exist");
}
```

HOISTING IN ACTION

BUT AFTER THE FUNCTION EXPRESSION HAS BEEN EVALUATED, CALLING THE FUNCTION WORKS OUT FINE!

```
someFunction();

function declaredFunction() {
  console.log("Declared function..I exist");
}

try {
  someFunction();
}
catch(error) {
  console.log(error.message)
}

var someFunction = function() {
  console.log("Inside a function literal - I exist");
}
someFunction();
```

```
Inside a function literal - I exist
```

ALL DECLARED FUNCTIONS, NO MATTER WHERE THEY APPEAR

DECLARED FUNCTIONS CAN BE USED ANYWHERE IN CODE

FUNCTION LITERALS CAN BE USED ONLY AFTER BEING EVALUATED

DECLARED FUNCTIONS

THIS IS THE ONLY SIGNIFICANT DIFFERENCE BETWEEN THE TWO

FUNCTION LITERALS