JAVASCRIPT

JAVASCRIPT

BROWSERS HAVE A WAY TO “INTERPRET” JAVASCRIPT, I.E. TO EXECUTE JAVASCRIPT CODE

BROWSERS HAVE A WAY TO "INTERPRET" JAVASCRIPT, I.E. TO EXECUTE JAVASCRIPT CODE

ASIDE, IT IS NO LONGER COMPLETELY CLEAR WHETHER JAVASCRIPT IS COMPILED OR INTERPRETED (JIT COMPILERS BLUR THE LINE BETWEEN THE TWO)

EITHER WAY, JAVASCRIPT IS "EXECUTED" BY THE BROWSER

BROWSERS HAVE A WAY TO "INTERPRET" JAVASCRIPT, I.E. TO EXECUTE JAVASCRIPT CODE

ASIDE, IT IS NO LONGER COMPLETELY CLEAR WHETHER JAVASCRIPT IS COMPILED OR INTERPRETED (JIT COMPILERS BLUR THE LINE

EITHER WAY, JAVASCRIPT IS "EXECUTED" BY THE BROWSER

BROWSERS TYPICALLY MAKE 2 PASSES OVER THE CODE, ALLOWING FORWARD REFERENCES

BROWSERS TYPICALLY MAKE 2 PASSES OVER THE CODE, ALLOWING FORWARD REFERENCES

(THIS IS WHY ITS OK TO USE A FUNCTION BEFORE THAT FUNCTION HAS BEEN DEFINED)

BROWSERS TYPICALLY MAKE 2 PASSES OVER THE CODE, ALLOWING FORWARD REFERENCES

ITS OK TO USE A FUNCTION BEFORE THAT FUNCTION HAS BEEN DEFINED

BROWSERS TYPICALLY MAKE 2 PASSES OVER THE CODE, ALLOWING FORWARD REFERENCES

(THIS IS WHY ITS OK TO USE A FUNCTION BEFORE THAT FUNCTION HAS BEEN DEFINED)

FUNCTIONS AND VARIABLES COME INTO EXISTENCE WHEN THE PAGE LOADS..

FUNCTIONS AND VARIABLES COME INTO EXISTENCE WHEN THE PAGE LOADS..

AND CEASE TO EXIST WHEN THE BROWSER LOADS A DIFFERENT PAGE (OR EVEN RELOADS THIS PAGE)

FUNCTIONS AND VARIABLES COME INTO EXISTENCE WHEN THE PAGE LOADS..

FUNCTIONS AND VARIABLES COME INTO EXISTENCE WHEN THE PAGE LOADS..

AND CEASE TO EXIST WHEN THE BROWSER LOADS A DIFFERENT PAGE (OR EVEN RELOADS THIS PAGE)

AND CEASE TO EXIST WHEN THE BROWSER LOADS A DIFFERENT PAGE (OR EVEN RELOADS THIS PAGE)

AND CEASE TO EXIST WHEN THE BROWSER LOADS A DIFFERENT PAGE (OR EVEN RELOADS THIS PAGE)

AND CEASE TO EXIST WHEN THE BROWSER LOADS A DIFFERENT PAGE (OR EVEN RELOADS THIS PAGE)

SCOPE

SCOPE WILL BECOME A COMPLICATED TOPIC IN JAVASCRIPT

BUT IT STARTS SIMPLE ENOUGH :-)

GLOBAL LOCAL

SCOPE

GLOBAL

GLOBAL VARIABLES EXIST FOR AS LONG AS THE PAGE. THEY COME INTO EXISTENCE WHEN THE PAGE LOADS, AND CEASE TO EXIST WHEN THE PAGE UNLOADS

LOCAL

LOCAL VARIABLES EXIST INSIDE A FUNCTION, OR SOME SPECIFIC SCOPE

SCOPE

GLOBAL

GLOBAL VARIABLES EXIST FOR AS LONG AS THE PAGE. THEY COME INTO EXISTENCE WHEN THE PAGE LOADS, AND CEASE TO EXIST WHEN THE PAGE UNLOADS

LOCAL

LOCAL VARIABLES EXIST INSIDE A FUNCTION, OR SOME SPECIFIC SCOPE

SCOPE IN JAVASCRIPT IS FASCINATING AND VERY DIFFERENT FROM JAVA/C++, WE WE WILL SEE WHEN WE GET TO CLOSURES

EXAMPLE 5: LOCAL AND GLOBAL VARIABLES

EXAMPLE 5: LOCAL AND GLOBAL VARIABLES

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

OUR JAVASCRIPT IS ENCLOSED BETWEEN SCRIPT TAGS, AS USUAL

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

WE SPECIFY THAT THE AS SOON AS THE PAGE LOADS, THE BROWSER SHOULD EXECUTE THE FUNCTION `printX`

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

HERE IS A VARIABLE FLOATING ABOUT OUTSIDE ANY FUNCTION - ITS A GLOBAL VARIABLE

IT WILL EXIST FOR AS LONG AS THE PAGE IS LOADED IN THE BROWSER

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

HERE INSIDE THE FUNCTION `printX` IS ANOTHER VARIABLE OF THE SAME NAME - THIS

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

THE FUNCTION `printX` CALLS ANOTHER FUNCTION CALLED `printAnotherX`

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " + x);
  console.log("Inside printX: pi = " + pi);
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

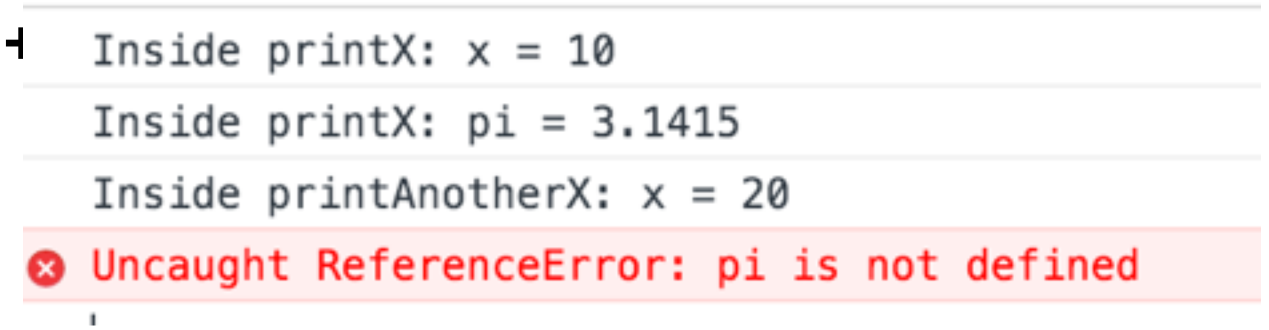
THE FUNCTION `printX` CALLS ANOTHER FUNCTION CALLED `printAnotherX`

HERE INSIDE THE FUNCTION `printAnotherX` IS YET ANOTHER VARIABLE OF THE SAME NAME

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " +
  console.log("Inside printX: pi = "
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```

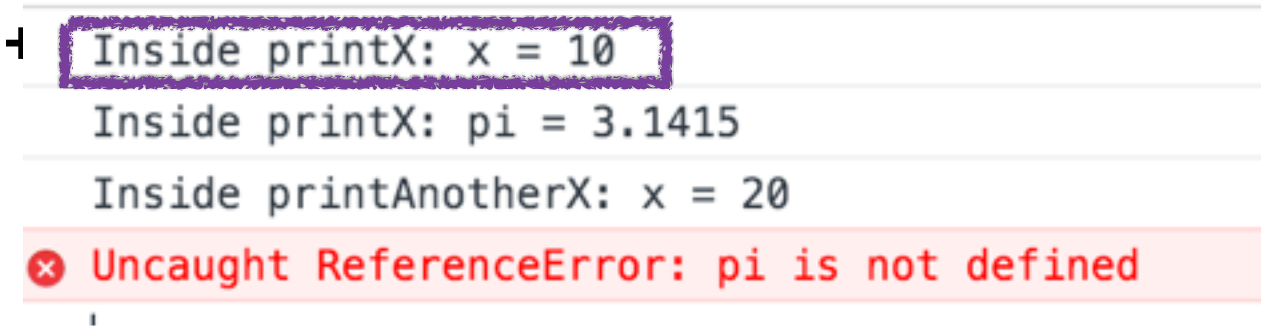


```
Inside printX: x = 10
Inside printX: pi = 3.1415
Inside printAnotherX: x = 20
Uncaught ReferenceError: pi is not defined
```

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " +
  console.log("Inside printX: pi = "
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```



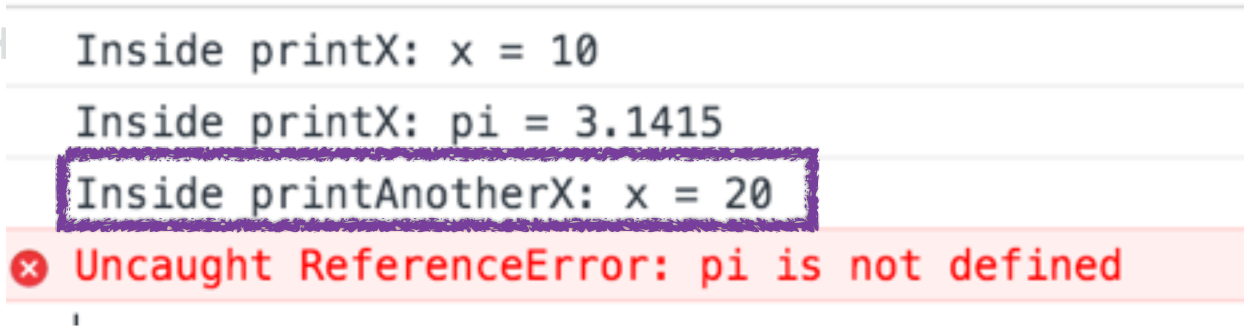
```
Inside printX: x = 10
Inside printX: pi = 3.1415
Inside printAnotherX: x = 20
Uncaught ReferenceError: pi is not defined
```

WHEN A LOCAL AND A GLOBAL VARIABLE BOTH EXIST IN A SCOPE, THE LOCAL VERSION TAKES

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " +
  console.log("Inside printX: pi = "
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```



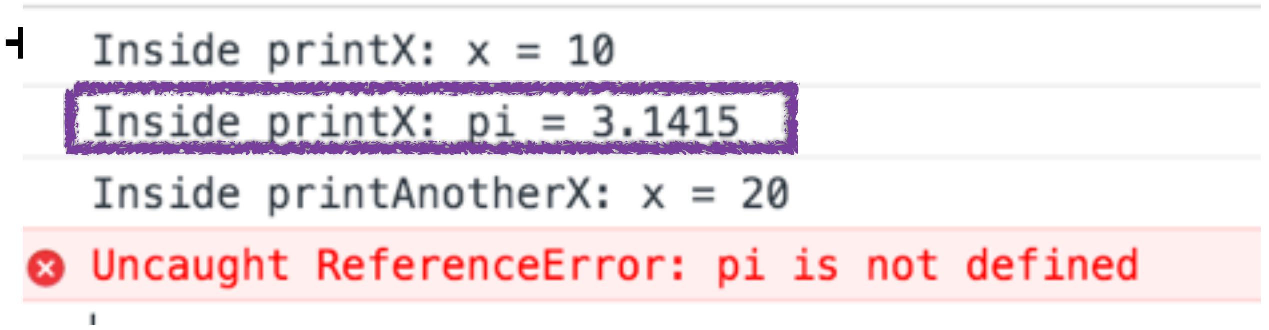
```
Inside printX: x = 10
Inside printX: pi = 3.1415
Inside printAnotherX: x = 20
Uncaught ReferenceError: pi is not defined
```

WHEN WE PRINT THE SAME VALUE FROM INSIDE **printAnotherX** THE LOCAL VARIABLE AGAIN

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " +
  console.log("Inside printX: pi = "
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```



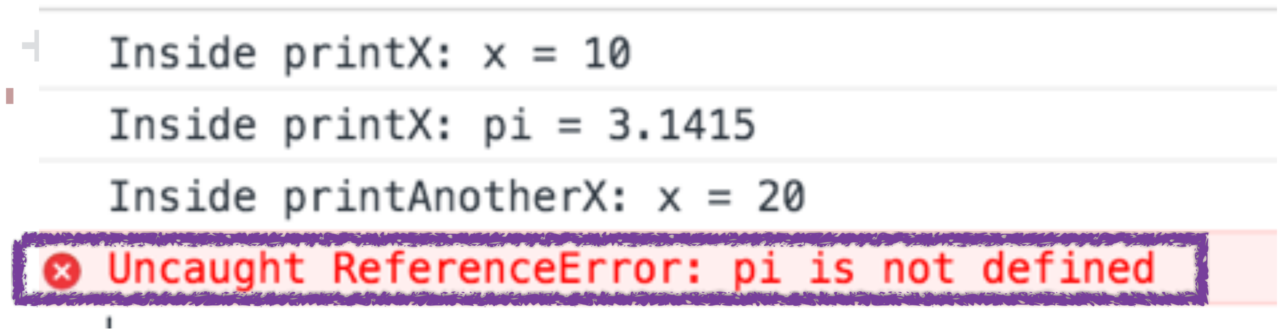
```
Inside printX: x = 10
Inside printX: pi = 3.1415
Inside printAnotherX: x = 20
Uncaught ReferenceError: pi is not defined
```

THE VARIABLE PI IS LOCAL TO `printX`, ATTEMPTING TO PRINT IT FROM

LET'S UNDERSTAND HOW GLOBAL AND LOCAL VARIABLES LOOK - AND WORK

```
<script>
window.onload = printX;
var x = 5;
function printX() {
  var x = 10;
  var pi = 3.1415;
  console.log("Inside printX: x = " +
  console.log("Inside printX: pi = "
  printAnotherX();
}

function printAnotherX() {
  var x = 20;
  console.log("Inside printAnotherX: x = " + x);
  console.log("Inside printAnotherX: pi = " + pi);
}
</script>
```



```
Inside printX: x = 10
Inside printX: pi = 3.1415
Inside printAnotherX: x = 20
Uncaught ReferenceError: pi is not defined
```

THE VARIABLE PI IS LOCAL TO `printX`, ATTEMPTING TO PRINT IT FROM