Session_2 (Javascript + React)

> Var

The var statement declares a function-scoped or globally-scoped variable.

```
var x = 1;

function abc() {
  var x = 2;

  console.log(x);
  // Expected output: 2
}

abc()
console.log(x);
// Expected output: 1
```

```
var x = 1;

{
  var x = 2;
  console.log(x);
  // Expected output: 2
}

console.log(x);
  // Expected output: 2
```

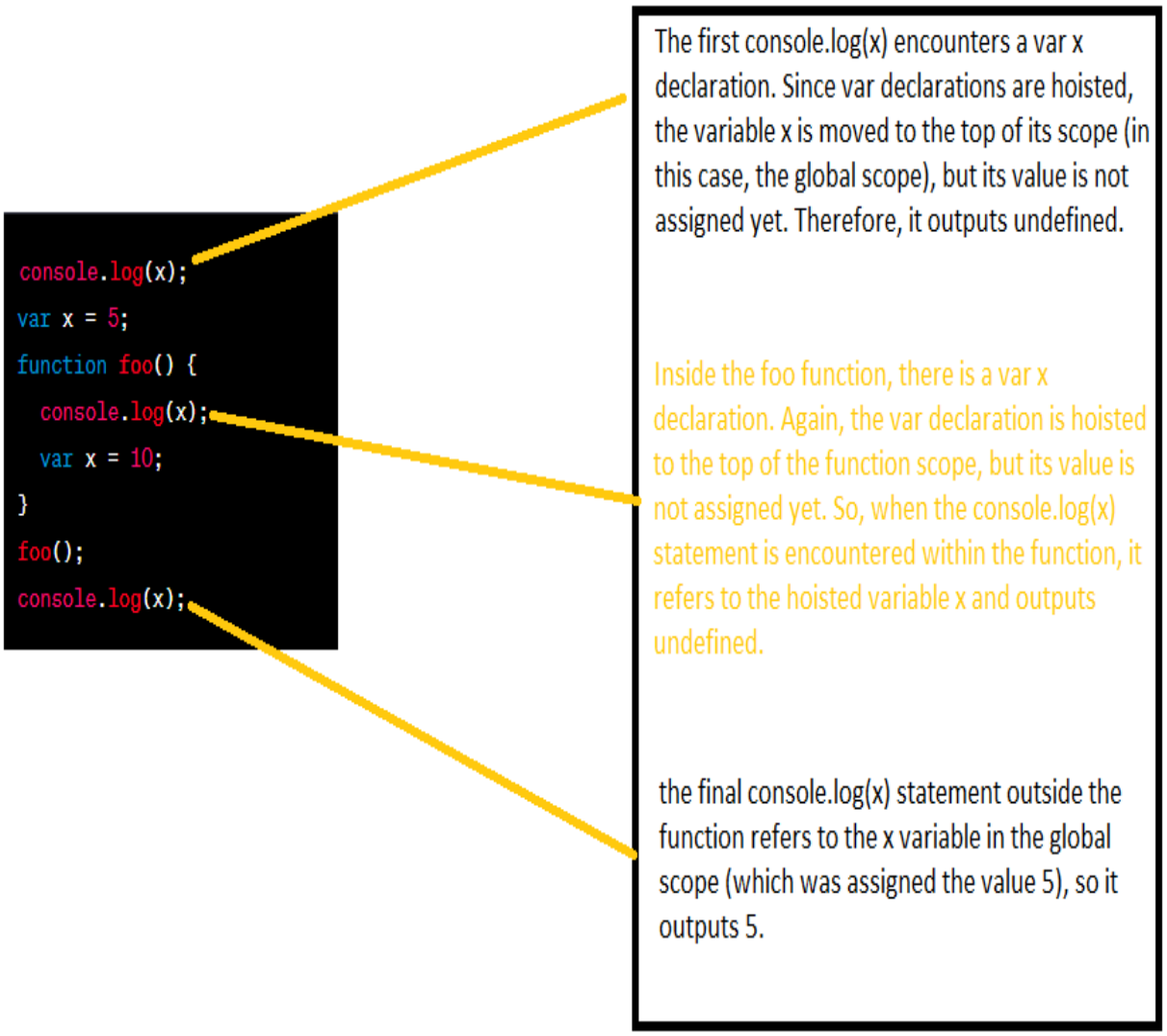
```
var x = 1;

if (x === 1) {
  var x = 2;

  console.log(x);
  // Expected output: 2
}

console.log(x);
// Expected output: 2
```

Explanation of below code from assignment 1:

```
console.log(x);
var x = 5;
function foo() {
  console.log(x);
  var x = 10;
}
foo();
console.log(x);
```

Output

```
undefined
undefined
5
```



> Let

The let declaration declares a block-scoped local variable.

```
let x = 1;

if (x === 1) {
  let x = 2;

  console.log(x);
  // Expected output: 2
}

console.log(x);
// Expected output: 1
```

- Block scope: The scope created with a pair of curly braces.

Variables declared by `let` have their scope in the block for which they are declared, as well as in any contained sub-blocks. In this way, `let` works very much like `var`. The main difference is that the scope of a `var` variable is the entire enclosing function:

```
function varTest() {
  var x = 1;
  {
    var x = 2; // same variable!
    console.log(x); // 2
  }
  console.log(x); // 2
}
```

```
function letTest() {
  let x = 1;
  {
    let x = 2; // different variable
    console.log(x); // 2
  }
  console.log(x); // 1
}
```

>Const

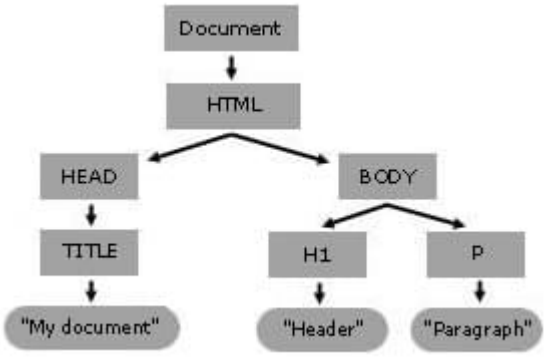
The const declaration creates block-scoped constants, much like variables declared using the let keyword. The value of a constant can't be changed through reassignment (i.e. by using the assignment operator), and it can't be redeclared (i.e. through a variable declaration). However, if a constant is an object or array its properties or items can be updated or removed.

>What is a DOM tree?

A DOM tree is a tree structure whose nodes represent an HTML or XML document's contents. Each HTML or XML document has a DOM tree representation. For example, consider the following document:

```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
</html>
```

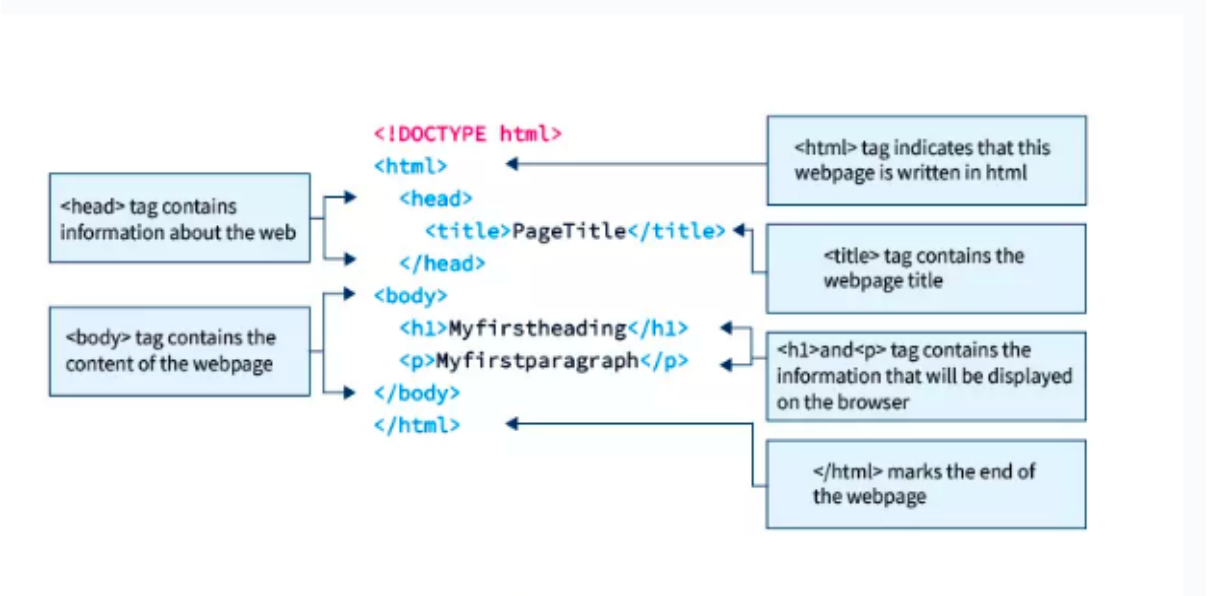
It has a DOM tree that looks like this:



When a web browser parses an HTML document, it builds a DOM tree and then uses it to display the document.

>HTML

Basic structure of a HTML document :



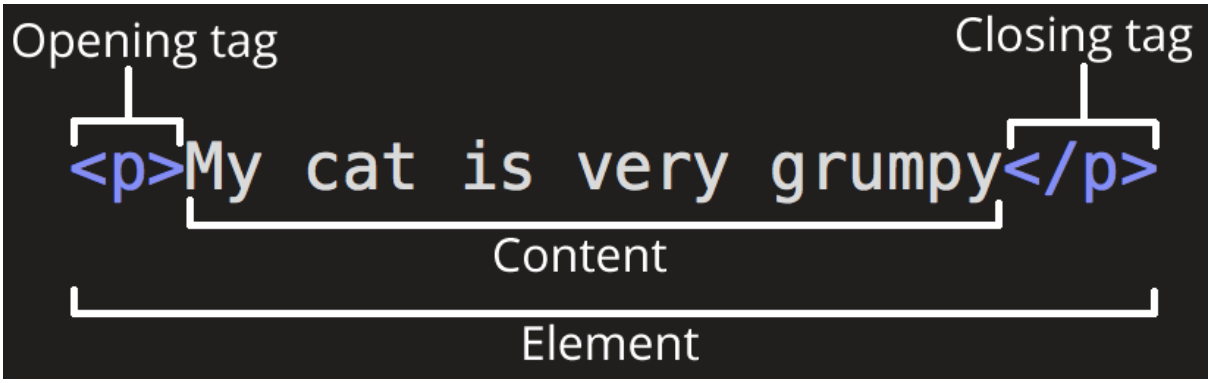
<!DOCTYPE>

The tag in HTML is used to inform the browser about the HTML version used in the web page. It is referred to as the document type declaration (DTD). It is not really a tag/element but rather an instruction to the browser regarding the document type.

NOTE: *DOCTYPEs are required for legacy reasons. When omitted, browsers tend to use a different rendering mode that is incompatible with some specifications. Including the DOCTYPE in a document ensures that the browser makes a best-effort attempt at following the relevant specifications.*

HTML is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way.

Example:
My cat is very grumpy



The main parts of our element are as follows:

1. The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and closing angle brackets. This states where the element begins or starts to take effect — in this case where the paragraph begins.
2. The closing tag: This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
3. The content: This is the content of the element, which in this case, is just text.
4. The element: The opening tag, the closing tag, and the content together comprise the element.