JavaScript Fundamentals I

Jan 25, 2018

Q & A

Q: What is a *UTF-8 encoded character?*

A: To account for all of the different languages in the world, in the late 90s, a standardized methodology was devised that allowed virtually any type of character to be stored within 8-bit blocks on a computer. Essentially, everything is stored on a computer in binary sequences of 1s and 0s. UTF-8 helps a computer to decode these sequences and display characters on our screens in a predictable manner.

Q: What is an *operator?*

A:* An *operator* is a character that represents action.

Ex: *2* **+** *2.* + is the operator.

Q: What is an *operand?*

A: An *operand* is a character that gets operated on by an operator.

Ex: ***2 + 2.*** The *2s* are the *operands.*

Q: What is an *expression?*

A:* An *expression* is a combination of 1 or more explicit values, constants, variables, operators and/or functions that JS interprets and computes to produce another value.

Ex: ***2 + 2***

Q: What is a *statement?*

A:* A *statement* is the smallest standalone element of a program that expresses some action to be carried out. It is an instruction that commands the computer to perform a specified action. A program is formed by a sequence of 1 or more statements.

```
console.log(2 + 2);
```

Q: What are JavaScript's *built-in or primitive data structures?*

A: JS features 5 primary data structures. There is a 6th known as *symbols* that was introduced as part of ES2015. For simplicity, we omit this.

Also, by *primitive* we mean a data type that is at the lowest level. These are not even objects, and have no methods or functionality, per say. They are just 'basic' raw stored data. Most of these types are encapsulated or wrapped in *object equivalents.* This allows us to work with associated methods.

Finally, note that *primitives* are *immutable.* This means that they **cannot** be changed._ When we 'change' the value by writing to a variable, we are really creating a new instance in another memory location. This is known as 'pass by value.' This is also why when you pass a value to a function, you have to 'capture' the returning value to see a change.

Strings
A series of characters. `const str = "I'm a string.";`
Numbers
Pretty much any number...Don't use any quotes, of course, unless you want it to be a string rather than a 'number.'
Booleans
'true' or 'false.' We arrive at these values when comparing things or using logic.
Nulls
Variable has been created but has a **zero** value.
Undefined
Variable has been created but has **no** value.

From the above data types we can build more complex *composite* types like Arrays. **The act of constructing a composite type is known as composition.***

Note: *Null* and *undefined* may be difficult to grasp at this time. As we progress and do more coding, these will hopefully become more clear.

Q: What do we mean when we say that JS is a *dynamically or 'loosely' typed* language?

A: We can *dynamically* change a variable's type around after it's been declared and instantiated.

```
// 'x' is a number type
let x = 15;
console.log(x);

// See what is type of a variable with typeof operator
console.log(typeof x);

// 'x' is now a string type
x = 'Hello!';

console.log(x);

// See what is type of a variable with typeof operator
console.log(typeof x);
```

Q: Is JS an object-oriented language?

A: It *is* an object-oriented language based on *prototypes **not** classes.* Object-oriented languages promote code-reuse.

Note: Unless you have studied or are currently studying some other programming languages, this may not mean anything to you. Do not worry!

*Courtesy of Todd McLeod's Go course.

CIS 177

CIS 177
manav.misra@swic.edu

manavm1990
visionwebsoft

Ancillary course materials.