Program structure

Friday, March 29, 2024

11:26 AM

**Bindings**

To catch and hold values, JavaScript provides a thing called a **binding, or variable**:

*let caught = 5 \* 5;*

The special word (keyword) **let** indicates that this sentence is going to define a binding.

You should imagine bindings as tentacles, rather than boxes. They do not contain values; they grasp them.

If you ask for the value of an empty binding, you’ll get the value undefined.

The words **var** and **const** can also be used to create bindings.

The first, **var** (short for “variable”), is the way bindings were declared in pre-2015 JavaScript.

The word **const** stands for constant. It defines a constant binding, which points at the same value for as long as it lives. This is useful for bindings that give a name to a value so that you can easily refer to it later.

*var name = "Ayda";*

*const greeting = "Hello ";*

*console.log(greeting + name);*

*// → Hello Ayda*

**Functions**

**Control flow**

Conditional execution

Conditional execution is created with the **if** keyword in JavaScript.

*let theNumber = Number(prompt("Pick a number"));*

*if (!Number.isNaN(theNumber)) {*

*console.log("Your number is the square root of " + theNumber \* theNumber);*

*}*

The statement after the if is wrapped in braces ({ and }) in this example. The braces can be used to group any number of statements into a single statement, called a ***block***. You could also have omitted them in this case, since they hold only a single statement, but to avoid having to think about whether they are needed, most JavaScript programmers use them in every wrapped statement.

You can use the **else** keyword, together with **if**, to create two separate, alternative execution paths.

You can use the **else** keyword, together with if, to create two separate, alternative execution paths.

If you have more than two paths to choose from, you can “chain” multiple **if/else** pairs together

While and Do loops

A statement starting with the keyword **while** creates a loop. The word while is followed by an expression in parentheses and then a statement, much like if.

*let result = 1;*

*let counter = 0;*

*while (counter < 10) {*

*result = result \* 2; counter = counter + 1;*

*}*

*console.log(result);*

*// → 1024*

A **do** loop is a control structure similar to a while loop. It differs only on one point: a do loop always executes its body at least once, and it starts testing whether it should stop only after that first execution.

*let yourName;*

*do {*

*yourName = prompt("Who are you?");*

*} while (!yourName);*

*console.log(yourName);*

Indenting code

The role of this indentation inside blocks is to make the structure of the code stand out.

For loops

*for (let number = 0; number <= 12; number = number + 2) { console.log(number); }*

The parentheses after a **for** keyword must contain two semicolons. The part before the first semicolon initializes the loop, usually by defining a binding. The second part is the expression that checks whether the loop must continue. The final part updates the state of the loop after every iteration.

Breaking out of a loop

There is a special statement called **break** that has the effect of immediately jumping out of the enclosing loop.

The **continue** keyword is similar to break, in that it influences the progress of a loop. When continue is encountered in a loop body, control jumps out of the body and continues with the loop’s next iteration.

Dispatching on a value with Switch

There is a construct called switch that is intended to express such a “dispatch” in a more direct way than if/else. Unfortunately, the syntax JavaScript uses for this (which it inherited from the C/Java line of programming languages) is somewhat awkward—a chain of if statements may look better.

*switch (prompt("What is the weather like?")) {*

*case "rainy":*

*console.log("Remember to bring an umbrella.");*

*break;*

*case "sunny":*

*console.log("Dress lightly.");*

*break;*

*case "cloudy":*

*console.log("Go outside.");*

*break;*

*default:*

*console.log("Unknown weather type!");*

*break;*

*}*

**Capitalization**

Binding names may not contain spaces, yet it is often helpful to use multiple words to clearly describe what the binding represents.

The standard JavaScript functions, and most JavaScript programmers, follow the style convention: they capitalize every word except the first. It is not hard to get used to little things like that, and code with mixed naming styles can be jarring to read.

**Comments**

JavaScript has two ways of writing comments. To write a single-line comment, you can use two slash characters (//).

A section of text between /\* and \*/ will be ignored in its entirety, regardless of whether it contains line breaks.