Chapter 2: Program Structure

Variables
variable rules:
- can't be a reserved word
- may not include spaces
- digits can be included, eg hugs22, but cannot start with a digit
- cannot include punctuation, except $ and _

= to change where variable (tentacle) points
variables don't contain values, they grasp them
two variables can refer to the same value
value of an empty variable = undefined
can define multiple vars in one statement, eg. var one = 1, two =2;

Keywords and reserved words
reserved words:
break case catch class const continue debugger default delete do else enum export extends false finally for function if implements import in instanceof interface let new null package private protected public return static super switch this throw true try typeof var void while with yield

The environment
The collection of variables and their values that exist at a given time is called the **environment**.

Functions
A **function** is a piece of program wrapped in a value.
Executing a function is called **invoking**, **calling**, or **applying** it.
Values given to functions are called **arguments**.

The console.log function

Return values
Showing a dialog box or writing text to the screen is a **side effect**.
When a function produces a value, it is said to **return** that value.
Anything that produces a value is an **expression** in JavaScript, which means function calls can be used within larger expressions

prompt and confirm
You can ask the user an OK/Cancel question using **confirm**. This returns a Boolean: true if the user clicks OK and false if the user clicks Cancel.

confirm("Shall we, then?");

The **prompt** function can be used to ask an “open” question.

prompt("Tell me everything you know.", "...");

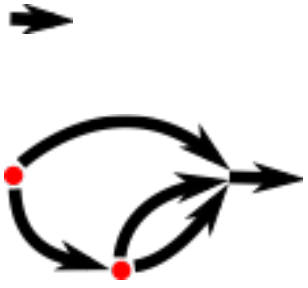
Control Flow
straight control flow

Conditional execution

conditional execution, where we choose between two different routes based on a Boolean value

if

if/else/if



while and do loops

Indenting Code

for loops

for (var counter = 0; counter < 10; counter = counter + 1)

The part before the first semicolon **initializes** the loop, usually by **defining a variable**. 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
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.

Updating variables succinctly

```
counter++;      count up by 1
counter—;       count down by 1
counter += 3;  count up by n
result *= 2 ;     multiply by n
counter -= 3    count down by n
```

Dispatching on a value with switch

```
if (variable == "value1") action1();
else if (variable == "value2") action2();
else if (variable == "value3") action3();
else defaultAction();
```

There is a construct called **switch** that is intended to solve such a "dis- patch" in a more direct way

```
switch (prompt("What is the weather like?")) {
case "rainy":
        console.log("Remember to bring an umbrella.");
        break;
case "sunny":
        console.log("Dress lightly.");
case "cloudy":
        console.log("Go outside.");
        break;
default:
```

```
        console.log("Unknown weather type!");
        break;
}
```

It starts executing statements there, even if they’re under another label, until it reaches a break statement.
In some cases, such as the "sunny" case in the example, this can be used to share some code between cases

Capitalization

fuzzyLittleTurtle

Comments
// single-line comment
/* */ section of text

Summary

conditional (if, else, and switch) and looping (while, do, and for) statements.

Functions are special values that encapsulate a piece of program. You can invoke them by writing functionName(argument1, argument2).