

# WEB230: JavaScript 1

## Module 1B: Program Structure

### Expressions

- Expressions are bits of code that evaluate to something
- Expressions can be combined

```
1
3+4
(3+4) * (5-2)
```

### Statements

- Statements should end in a semicolon (;)

```
1;
3+4;
(3+4) * (5-2);
```

### Bindings

- A variable is a named place to store a value
  - It is "Bound" to the name
- The `let` keyword creates a new variable
- Only use `let` once when creating the variable

```
let dog = "Fido";
dog = "Rex";
```

### `var` and `const`

- `var` is an older way to declare variables
  - now `let` is preferred
- `const` declares variables that are "constant" meaning they cannot be assigned new values
- more later

### Binding Names

- JavaScript has words that cannot be used for variable names
- Some of these are keywords in JavaScript, some are reserved for future versions

`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

- collection of variables and their values that already exist
- they are variables that are part of the language standard

- allow access to the surrounding system

## Functions

- A function is a piece of program wrapped in a value
- These variables have the type `function` and can be run:

```
alert("Good morning!");
```

- Executing a function is called *invoking* or *calling* it

## The `console.log` Function

- `console.log` is a function that will display values
- In web browsers there is a console where you can see these messages

```
console.log("JavaScript is fun so far!");
```

- The period is not part of the name
  - more in chapter 4

## Return Values

- Writing text to the screen is a *side effect*
- Some functions produce a value
- When a function produces a value, it is said to *return* that value

```
Math.min(4,6,3,1); // returns 1
```

- We will write functions in chapter 3

## Prompt and Confirm

- We can also ask the user to approve something
- Returns a boolean

```
confirm("Are you sure?");
```

- Or provide a value
- Returns a string

```
prompt("How many would you like?", "4");
```

## Control Flow

- Statements can execute one after the other

```
let num = Number(prompt("Pick a number", ""));  
alert ("Your number is the square root of " + num * num);
```

- Each line runs in turn

## Conditional Execution

- Programs don't have to run in a linear fashion
- An alternative is *conditional execution*

```
let num = Number(prompt("Pick a number", ""));

if (isNaN(num)) {
  alert (" Your number is the square root of " + num * num);
}
```

## While and do Loops

- `while` and `do` loops repeat statements

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

## Indenting Code

- Indent code inside of blocks (`{...}`)
- Makes programs easier to read
- The computer doesn't care
- Tabs vs spaces!!! The never ending argument!

## For Loops

- Counter, test, and increment all in one

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

## Breaking out of a Loop

- use the `break` statement to end a loop prematurely

```
for (let num = 20; ; num++) {
  if (num % 7 == 0)
    break;
}
console.log(num);
```

## Updating Bindings Succinctly

- We often update a variable base on its current value

```
counter = counter + 1;
```

- JavaScript provides a shortcut:

```
counter += 1;
```

- Or even:

```
counter++;
```

## Dispatching on a Value With switch

```
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;  
}
```

## Capitalization

- Variable names may not contain spaces
- Often have more than one word

```
fuzzylittleturtle  
fuzzy_little_turtle  
FuzzyLittleTurtle  
fuzzyLittleTurtle
```

- Last one preferred in JavaScript
- Called *Camel Case*

## Comments

- Two types of comments
- Single line:

```
// Some note about the program
```

- Multi-line:

```
/*
```

```
A longer note about the program  
that goes on and on
```

```
*/
```