Javascript: Variables & Objects

var

- The variable statement declares a variable, optionally initializing it to a value.

```
// String
var greeting = “hello";

// Number
var favoriteNum = 33;

// Boolean
var isAwesome = true;

// undefined
var foo;
var setToUndefined = undefined;

// null
var empty = null;
```

const

- Similar to the var statement*
- However, the value cannot be redeclared or reassigned.
- It is thus CONSTANT

```
// String
const greeting = 'hello';
// Number
const favoriteNum = 33;
// Boolean
const isAwesome = true;
```

** but block scoped. More on this later…*

const Errors

```
// Number
const favoriteNum = 33;

favoriteNum = 23;
```

- Cannot change your mind once const initialised
- Reassignment prohibited - error if attempted.

```
> const favoriteNum = 33;
  favoriteNum = 23;
  ▶ Uncaught TypeError: Assignment to constant variable.
        at <anonymous>:3:13
>
```

let

- The let statement declares a variable, optionally initializing it to a value.

```
// Number
let favoriteNum = 33;

favoriteNum = 23;
```

- The variable may be assigned a different value at any time

Always use **const** or **let**

Never use **var** - it can be considered obsolete for our purposes

Primitive Data Types

- 6 Primitive Data Types
- JavaScript is known as a "weakly" typed language.
- This means is that when you create variables and assign them to values, you do not have to specify the type of data you are working with.

```
// String
const greeting = “hello";

// Number
let favoriteNum = 33;

// Boolean
const isAwesome = true;

// undefined
let foo;
let setToUndefined = undefined;

// null
let empty = null;
```

Object Data Types

- Whereas primitive data typed variables hold individual values. e.g:
  - numbers
  - strings
  - boolean etc…
- Object types can hold *more than one value. e.g.:*
  - a number AND a string.
  - 2 numbers and a boolean and a string
  - 3 strings and 2 numbers
- Objects are central to creating interesting and powerful programs

Creating an Object

- Introduces singe variable called ‘homer’.

- This is an object with two fields
  - firstName, containing ‘homer’
  - lastName, containing ‘simpson’

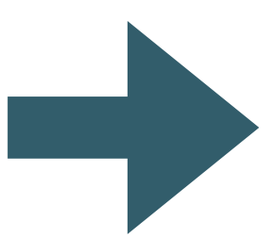
```
const homer = {
  firstName: 'homer',
  lastName: 'simpson',
};
```

Objects with Strings & Numbers

```
const bart = {
  firstName: 'bart',
  lastName: 'simpson',
  age: 10,
};

console.log(bart);
```

- An object containing 2 strings and a number.



{ firstName: 'homer', lastName: 'simpson' }

Anatomy of an Object

name of the object

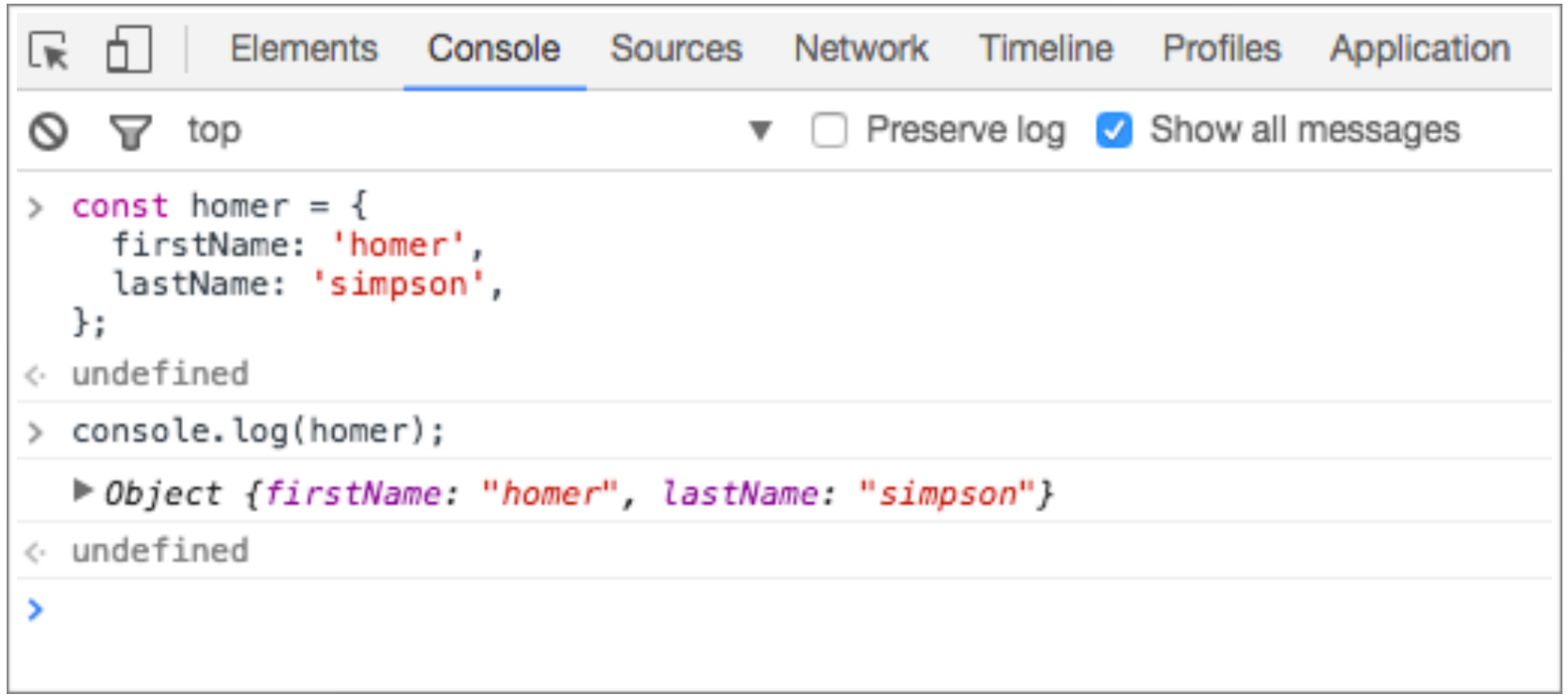
attributes (fields) of the object

```
const homer = {
  firstName: 'homer',
  lastName: 'simpson',
  age: 50,
};
```

attribute (field) values for the homer object

a specific attribute - called ‘age’

Objects in the Console



- We can paste code directly in the console for experimentation purposes
- Can be useful when learning or to clarify your understanding about some syntax/feature

Objects with Functions

```
const marge = {
  firstName: 'marge',
  lastName: 'simpson',
  age: 10,
  sayHello() {
    console.log('Hello from me!');
  },
};

marge.sayHello();
```

name of the object

data attributes (fields) of the object

attribute values for the object

a function attribute of the object

```
const marge = {
  firstName: 'marge',
  lastName: 'simpson',
  age: 45,
  sayHello() {
    console.log('Hello from me!');
  },
};

console.log(marge);
console.log(marge.firstName);
console.log(marge.age);

marge.sayHello();
```

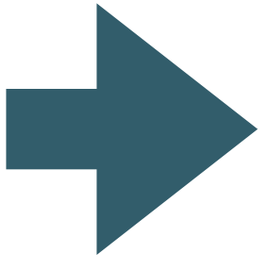
accessing marge's fields

calling the function within the marge object.

this refers to the 'current' object. Ned in this case

```
const ned = {
  firstName: 'ned',
  lastName: 'flanders',
  age: 45,
  speak() {
    console.log('How diddley do? says ' + this.firstName);
  },
};

ned.speak();
```



```
How diddley do? says ned
```