12.7.

A JavaScript object literal is a comma-separated list of name-value pairs wrapped in curly braces. Object literals encapsulate data, enclosing it in a tidy package. This minimizes the use of global variables which can cause problems when combining code.

John object

var john = {

firstName: 'John',

lastName: 'Smith',

birthYear: 1990,

family: ['Jane','Mark','Bob','Emily'],

job: 'teacher',

isMarried: false;

};

var jane = new Object();

jane.firstName = 'Jane';

jane.birthYear = 1994;

jane['lastName'] = 'Smith';

var john = {

firstName: 'John',

lastName: 'Smith',

birthYear: 1990,

family: ['Jane','Mark','Bob','Emily'],

job: 'teacher',

isMarried: false,

calcAge: function(){

return 2019 - this.birthYear;

}

};

console.log(john.calcAge(1990));

**in this case –this- refers to the john object, and can be used**

var john = {

firstName: 'John',

lastName: 'Smith',

birthYear: 1990,

family: ['Jane','Mark','Bob','Emily'],

job: 'teacher',

isMarried: false,

calcAge: function(){

this.age = 2019 - this.birthYear;

}

};

john.calcAge();

console.log(john);

**storin functions in the method declaration – this points to itsself**

11.7.

There are two ways to add javascript to your file – the first being inline scripts, and the other by addin an external JS file where the code is stored.

<script>

console.log('Hello World!');

</script>

-inline script, writing in the console – logging values to the developer console

<script src="script.js"></script>

-how to link the js file to the html markup – is written after the body tag

console.log('Hello World!!!');

-the same code but in the external file (wont work without the link)

* JavaScript is a lightweight, cross-platform, object-oriented computer programming language
* JavaScript is one of the three core technologies of web development
* Today, JavaScript can be used in different places
  + **Client-side: JavaScript was traditionally only used in the browser**
  + Server-side: Thanks to node.js, we can use JavaScript on the server as well
* JavaScript is what made modern web development possible
  + Dynamic effects and interactivity
  + Modern web applications that we can interact with
* **Frameworks/libraries like React and Angular are 100% based on JavaScript: you need to master JavaScript in order to use them!**

Variable – like a container in which we can store a value and use it over and over again instead of writing it every time.

var firstName = 'John';

console.log(firstName);

Different data types in JS:

* **Number:** Floating point numbers, for decimals and integers
* **String:** Sequence of characters, used for text
* **Boolean:** Logical data type that can only be **true** or **false**
* **Undefined:** Data type of a variable that does not have a value yet
* **Null:** Also means ‘non-existent’

Variables cant start with anything but letters, $ and \_

Comments – single line // and multi line /\* \*/

var job, isMarried;

job = 'teacher';

isMarried = false;

Multiple variables can be declared on the same lines and then receive their assigned value later

With **type coercion** – the browser transforms everything into a string in order for it to be written on screen (in the console in this case)

alert(firstName + ' is a ' + age + ' year old '+job+ '. Is he married? '+isMarried);

We can use alert to have it written on a pop-up window

var lastName = prompt('What is his last Name?');

console.log(firstName + ' ' + lastName);

When you want to store a user input

var firstName = 'John';

var civilStatus = 'single';

if (civilStatus === 'married'){

console.log(firstName + ' is married!');

}

else{

console.log(firstName + ' is not married!');

}

Elif statements

AND (&&) => true if ALL are true

OR (||) => true if ONE is true

NOT (!) => inverts true/false value

var firstName = 'John';

var age = 16;

age >= 18 ? console.log(firstName+' drinks beer.')

: console.log(firstName+' drinks juice');

THE THERNARY OPERATOR

var job = 'instructor';

switch(job){

case 'teacher':

case 'instructor':

console.log(firstName + ' teaches kids how to code.');

break;

case 'driver':

console.log(firstName + ' drives an uber in Lisbon.');

break;

case 'designer':

console.log(firstName+ ' designs beautiful websites.');

break;

default:

console.log(firstName + ' does something else.');

}

Truthy and falsy values ‘

Falsy values: undefined, nul, 0, ‘ ’, NaN

Truthy values: not falsy ones

The difference between equality operators (=== and ==) – is that == uses type coercion (meaning that it will change between datatypes to compare (string and int for example)) – while === is more specific > should be used more oftenly to avoid bugs

function calculateAge(birthYear) {

return 2019 - birthYear;

}

var ageJohn = calculateAge(1990);

console.log(ageJohn);

define to be able to reuse

function yearsUntilRetirement(year, firstName){

var age = calculateAge(year);

var retirement = 65 - age;

if (retirement>0){

console.log(firstName + ' retires in '+ retirement + ' years.');

}

else{

console.log(firstName + ' is already retired.')

}

}

Function statements and expressions

In switch, return immediately closes, so there’s no need to put in a break

Arrays!

var names = ['John', 'Mark', 'Jane'];

var years = new Array(1990, 1969, 1948); //will not use - use the 1st

console.log(names);

console.log(names.length);

.length used to tell us how many items there are in an array

//Mutate array data

names[1] = 'Ben';

names[names.length] = 'Mary'

console.log(names);

the order of reading matters!

-adding new arrays with names.length

var john = ['John', 'Smith', 1990, 'teacher', false];

arrays can store different data types

john.push('blue');

used to add an item to the end of an array

john.unshift('Mr.');

adds an item to the beginning

john.pop();

remove last

john.shift();

remove first

console.log(john.indexOf(1990));

lets us know the array position of an element, if the element we’re testing for isnt in the array, it returns -1;