Javascript Notes

Traversy Media - JS Crash Course For Beginners - 13/03/2019

Basics

- You add the javascript right before </body> tag in your html;
    - <script src="./js/main.js"></script>
- You have a Js console in DevTools - you can also run js code there

    - you output here with “console.log(’text’);
    - you clear it with “clear();”
- mdn developer - best doc for js = https://developer.mozilla.org/en-US/
- console → .log(), .error(), .warn() etc.
- comments: // = single line and /* … … … */ = multi-line comments

---

Variables

How to initialise and assign

- var - globally scoped
- let - block lvl scope; you can reassign values;
- const - block lvl scope; you can’t reassign values;
- example: let age = 30; age = 31; OR const gender = ‘male’;
- with const, you must assing a value, with let, you can do: let age; (no assign needed)

Data Types

- String, Numbers, Boolean, null, undefined, Symbols
- const name = ‘John’; - string
- const age = 30;
- const rating = 4.7;
- const myBool = true;
- const x = null; const y = undefined;
- GET TYPE: console.log(typeof name); ⇒ string;
- let z; console.log(typeof z) ⇒ undefined

Strings

- concatenation
    - console.log(”My name is “ + name + “ and i am “ + age);

- Template String
    - console.log(`My name is ${name} and I am ${age}`);
- Properties:
    - string.length = length
    - string.toUpperCase()
    - string.toLowerCase()
    - string.substring(0,5) (string='Hello World! ⇒ 'Hello')
    - you can chain string functions: string.substring(0,5).toUpperCase()
    - string.split('') - splits by the given character; example: string.split(', ') / string.split(' ') etc.

Arrays

- const fruits = ['apple', 'orange', 'pear'];
- you can have any data type in each array (ex.: fruits=[1,'apple', 4];
- fruits[1] ⇒ orange
- add: fruits[3] = 'pineapple' - not recommended
- fruits.push('thisIsPushedToTheEnd') ;
- fruits.unshift('thisIsAddedToTheBeginning');
- fruits.pop() ⇒ removes the last element
- Array.isArray(fruits) = true (check if an array)
- Get index of a value: fruits.indexOf('pear'); ⇒ 2
- Array of objects:
  todos = [
  { id: 1, text: 'Take out the trash', isCompleted: true },
  { id: 2, text: 'Clean your room', isCompleted: false }]
  - to access fields: console.log(todos[1].text) - for example
  - you can transform this to a JSON format:
  const todoJSON = JSON.stringify(todos)

Array Methods

- todos.forEach(function(todo){
  console.log(todo.id);
  } ⇒ this parses through each obj in the array and calls a function each object as a parameter
- let todoText = todos.map(function(todo) {
  return todo.text; }); ⇒ todoText = ['Take out the trash', 'Clean your room']
  ↔
  array.map retuns an array of the result of all the functions
- filter method:
  let todoCompleted = todos.filter(function(todo) {
  return todo.isCompleted === true; });
  ⇒ returns an array of todo objects
- You can use array methods together todos.filter(...).map(...);

Date:

let myDate = new Date('5-3-1970');
myDate.getFullYear(); ⇒ 1970

Object Literals

- basically a dictionary (a key ↔ value pair)
- let person = { firstName: 'Tudor', lastName: 'Gulin', age: 21, hobbies= ['sports', 'video games'],
  address: { street: 'ananasului', city: 'Cluj-Napoca' }
  }
  - you access this by: person.firstName
  - person.address.city
- To get the fields as variables (in a way)
  const { firstName, lastName, address: { city} } = person;
  console.log(city) ⇒ 'Cluj-Napoca'
- ADD PROPERTIES:
  person.newField = newValue;

LOOPS

For Loop

- for(let i = 0; i<10; i++){
  //do smth
  console.log('For Loop Number: ${i}');
  }

While Loop

- let i =0;
  while(i<0){
  //do smth
  i++;
  }

Array Loop

- you can do: for(let i = 0; i < todos.length;i++){//do smth}
- for(let todo=ObjectInArray of todos=nameOfTheArray){ //do smth }

If Statement

- if(x === 10 || y < 10){
  //do smth }
  else if(x > 10 && y ===10){
  //do smth }else {
  // do smth else }
  === → matches data type
  == → doesn't, so '10' = 10
  && - and
  || - or
- const color = x > 10 ? 'red' : 'blue';
  if x > 10 ⇒ color is red, else it's blue

Switch

- switch(color){
      case 'red':
          //do smth

```
            break;
        case ‘blue’:
            // do smth
            break;
        default:
             //default
              break;
     }
```

FUNCTIONS

```
function addNumbers(num1 = 1, num2 = 1){ // num1=1 ⇒ default value is 1
   const result = num1+num2;
   console.log(result);
   return result;
}
addNumbers(1, 7); // call the function
```

Arrow function

```
const addNumbers = (num1 =1, num2 = 1) ⇒ {
      return num1 + num2;
}
console.log(addNumbers(4,3)); ⇒ 7
```

OOP Basics

Construction Functions

```
function Person( firstName, lastName, email, dateOfBirth){
       this.firstName = firstName;
       this.lastName = lastName;
       this.dateOfBirth = new Date(dateOfBirth);

      this.getBirthYear = function () { // create functions inside a person
           return this.dateOfBirth.getFullYear;
       }
      this.getFullName = function(){
        return `${this.firstName} ${this.lastName}`;
     }
}
// To Instantiate an Object:
const person1 = new Person('Petrisor', 'Cornelache', '4-20-1969');
------------------------------------------
Person.prototype.getBirthDay = function(){
    return this.dateOfBirth.getDay();
}
```

Classes

it's the same as the above, but it's similar to OOP in other pr languages

```
class  Person {

     constructor(firstName, lastName, dateOfBirth){

          this.firstName = firstName;
          this.lastName = lastName;
          this.dateOfBirth = new Date(dateOfBirth);
       }
     getBirthYear(){
    return this.dateOfBirth.getFullYear(); }
    getFullName(){ return `${this.firstName} ${this.lastName}`; }
    }
```

Subclasses

```
class Parent extends Person{
      constructor( firstName, lastName, dob, noKids){
          super(firstName, lastName, dob);
          this.noKids = noKids;
}
getBirthYear(){ super.getBirthYear();}
getNoKids(){ return this.noKids;}
}
```

THE DOM

window object - the parent f the browser; the very top level!!!

document - to select

Selection

use querySelector and querySelectorAll mainly

Single element selectors

const myElement = document.getElementById('enter-id'); ⇒ you get the element in your html
with the given ID

querySelector:
const myContainer = doccument.querySelector('.container');
const myHeader = document.querySelector(h1');
- if there are more than 1 h1 or .container(tag w the class container)
it will select only the first one

Multiple element selectors

const allItems = document.querySelectorAll('.item');
⇒ all elements with the class of item

(older ones: document.getElementsByClassName('item');
document.getElementsByTagName('h3');)

Manipulating the DOM

const ul = document.querySelector('.items');
REMOVES:
ul.remove(); ⇒ the ul is removed
ul.lastElementChild.remove(); ⇒ the last element is removed
EDIT:
ul.firstElementChild.textContent ='New Text';
ul.children[1].innerText = 'New Inner Text'; (this gets the element with the index of 1(2nd)
ul.lastElementChild.innerHTML = '<h2>New Text</h2>' ⇒
you edit the html of whatever

EDIT THE CSS:
const btn = documentQuerySelectorAll('.btn');

btn.style.background = ‘red’; // this changes the background (css) to red of the btn class

EVENTS

example:
const btn = document.querySelectorAll(’.btn’);
btn.addEventListener(’click’, (e) ⇒ {
    console.log(’clicked’);
     e.preventDefault(); ⇒ this prevents the default of that event
});

the event object (e in the above case)
has a few usefull properties:
e.target.className ⇒ gets you the class clicked obj
e.target.id ⇒ gets you the id of the clicked obj

- btn.addEventListener(’click’, (e) ⇒{
  e.preventDefault();
  document.querySelector(’body’).classList.add(’newClass’);
  document.querySelector(’.items’).lastElementChild.innerHTML=’<h2>this changes when
  you click the btn</h2>’;
- hover event = mouseover
- mouseout - you enter the obj and then you leave the obj(with the cursor)

Others

- to see if a field is empty: nameInput (input html).value===’’
- to make smth only last a certain amount of time:
  setTimeout(() ⇒ //remove what you want to remove, 3000 = 3 seconds);
- let age = prompt(’Enter your age: ‘);
- Restructuring:
  let name = ...;
  let height =...;
  let myF = function(){...}
  let person = {name, height, myF}; ⇒ person is an obj w the 2 fields and the function

- ... Operator:
  let l1=[1,2,3,4];
  let l2=[5,6];
  let l3 = [...l1, ...l2]; ⇒ l3 = [1,2,3,4,5,6];
  !!! This works with objects as well !!!
- rest Operator:
  let l1 = [1,2,3,4];
  let [first, second, ...rest] = l1;
  first =1 , second = 2, rest = [3,4]
  - works for objects as well