Sorin's JavaScript course in a nutshell

A humble try to demistify some important JavaScript concepts.

The curriculum is based on Gordon Zhu "Practical JavaScript" Todo APP Versions.

JavaScript **DATA TYPES**

Primitives

Number (25, 23.2)

String ("Hello World!", 'My name is Bob')

Boolean (true, false)

Null (the value is 'nothing' → var bob = null;)

Undefined (the value is not been set → var bob;)

Objects

[ ] **Arrays** ['Bob', 23]

( ) **Functions** (age) => 2019 - 1966

Classes class Hero {...}

APP VERSION **1** - **ARRAYS**

- ✓ It should have a place to **store** TODOs
- ✓ It should have a way to **display** TODOs
- ✓ It should have a way to **add new** TODOs
- ✓ It should have a way to **change** a TODO
- ✓ It should have a way to **delete** a TODO

THE ANATOMY OF AN **ARRAY**

[ ]

An array is a data structure that stores one or more similar type of values in a single value


```
//Arrays can contain ANYTHING !
[
  "Hello World",                    // Strings
  256,                              // Numbers
  true,                             // Booleans
  {color: "#fc0", name: "Tada"},    // Objects
  ["Yes", "Other", "Arrays"]        // Arrays !
];
```

APP VERSION **1** - **ARRAYS**

```
//Use var arrName = [ …, …, ... ] to store items
var todos = ["item 1", "item 2", "item 3"];

//Use console.log() to display / output
console.log(todos); //display array items
console.log(todos.length); //display the n° of items
```

```
//Use array.push() to add items
todo.push("item 4");

//Use array[index] to assign values
todos[3] = "item 4 updated";

//Use array.splice() to delete values
todos.splice(0, 1);
```

APP VERSION **2** - **FUNCTIONS**

- ✓ It should have **a function to display** TODOs
- ✓ It should have **a function to add** TODOs
- ✓ It should have **a function to change** TODOs
- ✓ It should have **a function to delete** TODOs

THE ANATOMY OF A **FUNCTION**

( )

In computing, a function is a sequence of instructions within a larger computer program

JS Keyword | Function Name | Parameter 1 | Parameter 2 | Parameter N

```
function changeTodo(position, newText, ...)
{
  todos[position] = newText;
}

//Calling the function with "real" values
changeTodo(1, "Other text");
changeTodo(3, "Yet another text");
```

START → `{`

BODY → `todos[position] = newText;`

END → `}`

APP VERSION 2 - FUNCTIONS

```
var todos = ['item 1', 'item 2', 'item 3'];

function displayTodos() {
  console.log('My Todos: ', todos);
}

function addTodo(todo) {
  todos.push(todo);
  displayTodos();
}

function changeTodo(position, newText) {
  todos[position] = newText;
  displayTodos();
}

function deleteTodo(position) {
  todos.splice(position, 1);
  displayTodos();
}
```

APP VERSION **3** - **OBJECTS**

- ✓ It should store TODOs array in **a object**
- ✓ It should have a todos **property**
- ✓ It should have a `displayTodos()` **method**
- ✓ It should have a `addTodo()` **method**
- ✓ It should have a `changeTodo()` **method**
- ✓ It should have a `deleteTodo()` **method**

THE ANATOMY OF AN **OBJECT**

{ }

An object is a collection of properties, and a property is an association between a name (or key) and a value

KEY (can be a key, a string or a number)

VALUE (can be ANYTHING !)

two dots
(separate key/value)

START

```
{
    key1 :  "Tada!" ,
    key2 :  3.14 ,
    key3 :  ["item 1", 56] ,
    key4 :  function() {return 5;} ,
    key5 :  {key1: "Bob", key2: 2}
}
```

comma
(separate items)

PROPERTIES

END

3 - OBJECTS

APP VERSION

```
var todoList = {

  todos: ['item 1', 'item 2', 'item 3'],

  displayTodos: function () {
    console.log('My Todos: ', this.todos);
  },

  addTodo: function (todo) {
    this.todos.push(todo);
    this.displayTodos();
  },

  changeTodo: function (position, newValue) {
    this.todos[position] = newValue;
    this.displayTodos();
  },

  deleteTodo: function (position) {
    this.todos.splice(position, 1);
    this.displayTodos();
  }

};
```

APP VERSION **4** - **BOOLEANS**

✓ `todoList.addTodo()` should **add objects**

✓ `todoList.changeTodo()` should **change the todo Text property**

✓ `todoList.toggleCompleted()` should **change the completed property**

HOW **BOOLEANS** WORK

true / **false**

TWO POSSIBLE VALUES (same principle of the binary system: on / off)

true / **false** or **1** / **0**

EXAMPLES WHEN **TESTING CONDITIONS**

```
var completed = false;
var y = 3;

     is false !
if(completed) { //False, skip this block! }

     is true !
if(y === 3)   { //True, run this block! }
```

APP VERSION 4 - BOOLEANS
(only relevant changes)

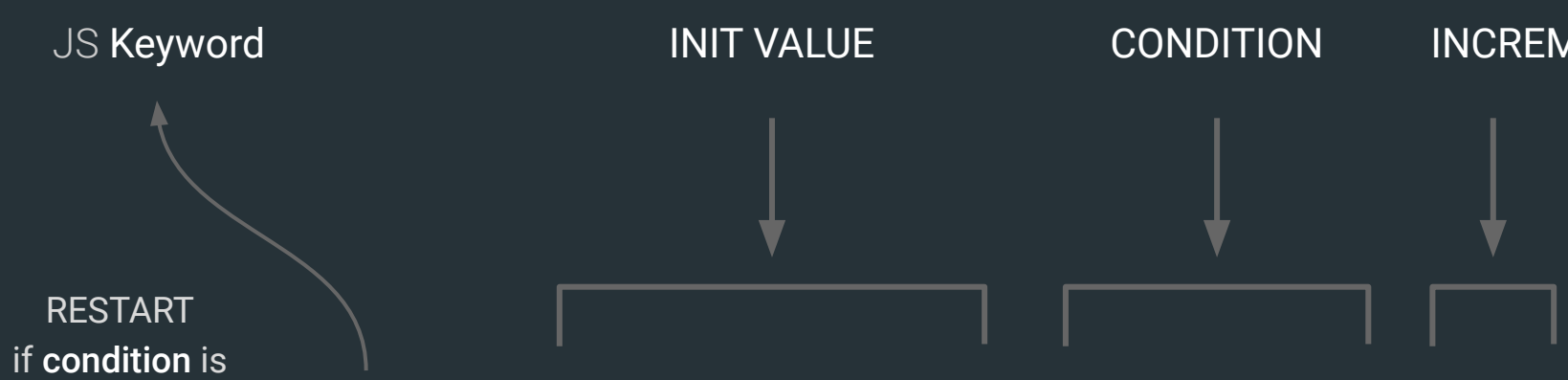
```
var todoList = {

    todos: [
        {todoText: "Item 1", completed: false},
        {todoText: "Item 2", completed: false},
        {todoText: "Item 3", completed: true}
    ],

    addTodo: function(todoText) {
        this.todos.push({
            todoText: todoText,
            completed: false
        });
        this.displayTodos();
    },

    toggleCompleted: function(position) {
        var todo = this.todos[position];
        todo.completed = !todo.completed;
        this.displayTodo();
    }

};
```

APP VERSION **5** - **LOOPS & CONDITIONS**

- ✓ `displayTodos()` should **show todoText**
- ✓ `displayTodos()` should **tell you if todos is empty**
- ✓ `displayTodos()` should **show completed**

THE ANATOMY OF A **LOOP**

Loops in are used to execute the same block of code a specified number of times or while a specified condition is true

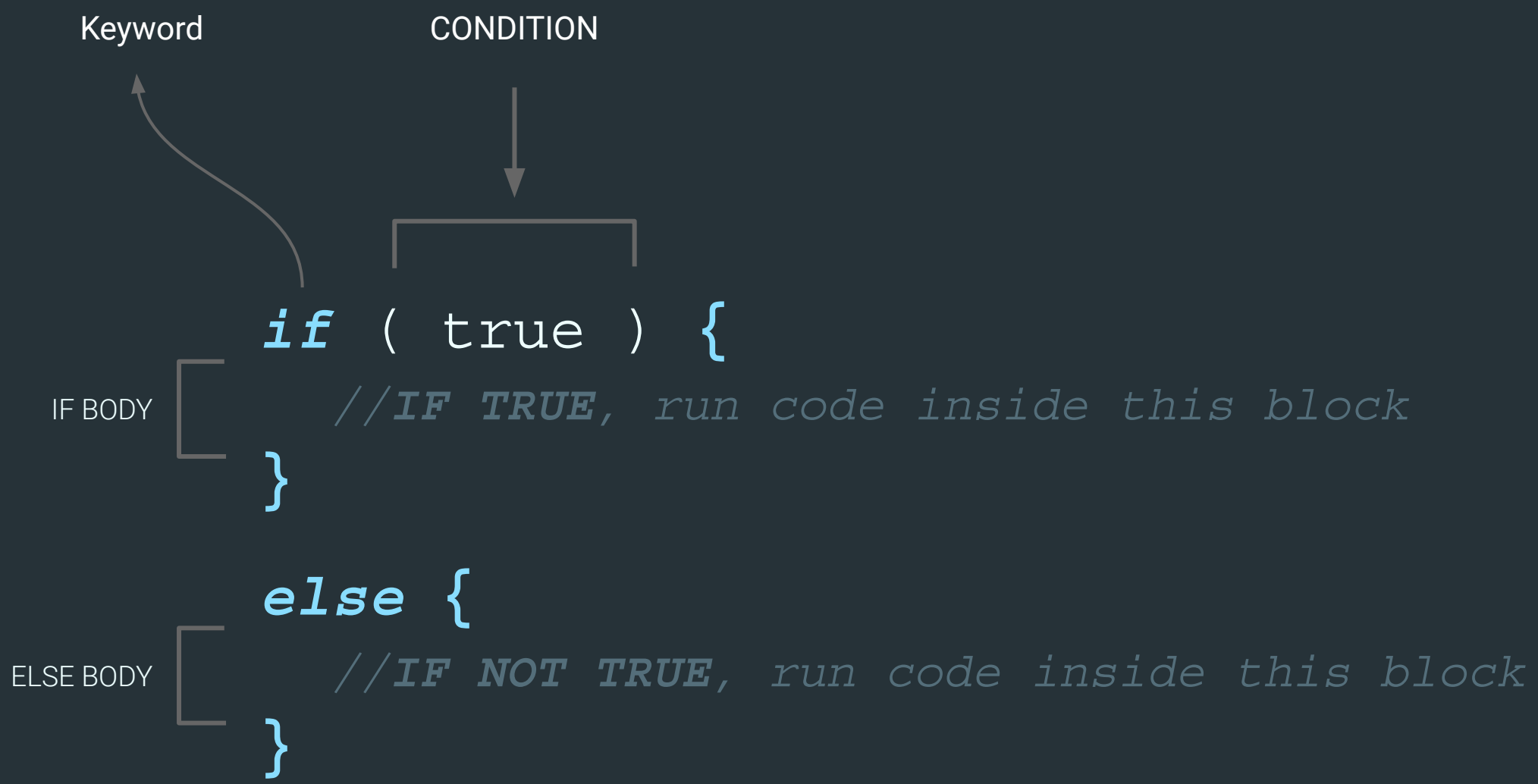



```
for ( let i = 0; i < 10; i++ ) {
  console.log("the value of i is: ", i);
}

//Loop into todos array
for (let i = 0; i < todos.length; i++) {
  console.log("todo item: ", todos[i].todoText);
}
//todo item: "Item 1"
//todo item: "Item 2"
//etc.
```

THE ANATOMY OF A **CONDITION**

?

In computer science, conditional statements perform different computations depending on whether a boolean condition is true or false



APP VERSION 5 - LOOP & CONDITIONS

(only relevant changes)

```
var todoList = {

  displayTodo: function() {

    if(this.todos.length === 0) {
      console.log('Your todo list is empty!');
    } //End if

    else {
      for(var i = 0; i < this.todos.length; i++) {
        if(this.todos[i].completed === true) {
          console.log('(x)', this.todos[i].todoText);
        } //End if
        else {
          console.log('( )', this.todos[i].todoText);
        } //End else
      } //End for loop
    } //End else

  }, //End displayTodo() method

};
```

APP VERSION **6** - **TOGGLE ALL**

- ✓ `toggleAll()`:
  If everything's true, make everything false
- ✓ `toggleAll()`:
  Else make everything true

APP VERSION 6 - TOGGLE ALL

(only relevant changes)

```
var todoList = {
  toggleAll: function () {
    var totalTodos = this.todos.length;
    var completedTodos = 0;
    // Get the number of completed todos
    for (var i = 0; i < totalTodos; i++) {
      if (this.todos[i].completed === true) {
        completedTodos++;
      }
    }
    // If everything is true, make everything false.
    if (completedTodos === totalTodos) {
      for (var i = 0; i < totalTodos; i++) {
        this.todos[i].completed = false;
      }
      // Otherwise make everthing true.
    } else {
      for (var i = 0; i < totalTodos; i++) {
        this.todos[i].completed = true;
      }
    }
  }, // End toggleAll() method
}; // End todoList{} object
```

APP VERSION **7** - **HTML** and the **DOM**

- ✓ There should be a button “Display todos”
- ✓ There should be a button “Toggle all”
- ✓ Clicking “Display todos” button should run `todoList.displayTodos()`
- ✓ Clicking “Toggle All” button should run `todoList.toggleAll()`

VERSION 7 - HTML & THE DOM

The Document Object Model (DOM) is a programming API for HTML and XML documents.
It defines the logical structure of documents and the way a document is accessed and manipulated.

```
<!-- index.html -> HTML BUTTONS -->
<button id="displayTodosBtn">Display Todos</button>
<button id="toggleAllBtn">Toggle All</button>
```

```
//scripts.js -> JS addEventListener() method
var displayTodosBtn = document.getElementById('displayTodosBtn');
var toggleAllBtn = document.getElementById('toggleAllBtn');

displayTodosBtn.addEventListener('click', function() {
  todoList.displayTodos();
});

toggleAllBtn.addEventListener('click', function() {
  todoList.toggleAll();
});
```

APP VERSION **8** - **Get data from inputs**

- ✓ Code Refactoring
- ✓ I should have working controls for `.addTodo()`
- ✓ I should have working controls for `.changeTodo()`
- ✓ I should have working controls for `.deleteTodo()`
- ✓ I should have working controls for `.toggleCompleted()`

VERSION **8** - **CODE REFACTORING: HTML**

```html
<!-- Global buttons -->
<button onclick="handlers.displayTodos()">Display Todos</button>
<button onclick="handlers.toggleAll()">Toggle All</button>

<!-- Toggle completed input + button-->
<input id="toggleCompletedPositionInput" type="number" min="0">
<button onclick="handlers.toggleCompleted()">Toggle Completed</button>

<!-- ADD todo input + button -->
<input id="addTodoTextInput" type="text">
<button onclick="handlers.addTodo()">Add</button>

<!-- CHANGE todo input + input + button -->
<input id="changeTodoPositionInput" type="number" min="0">
<input id="changeTodoTextInput" type="text">
<button onclick="handlers.changeTodo()">Change Todo</button>

<!-- DELETE todo input + button -->
<input id="deleteTodoPositionInput" type="number" min="0">
<button onclick="handlers.deleteTodo()">Delete</button>
```

VERSION **8** - **CODE REFACTORING: JS**

```
/**
* HANDLERS OBJECT - "specialized" in DOM events
* ---------------------------------------------
* In programming it's adviced to use "specialized" objects and methods
  for a more flexible application.
* Here, the handlers object is "specialized" in DOM event-handlers
* Please refer to "app-versions-curriculum/v8" to see the code
*/
var handlers = {
  //ADD TODO using the DOM
  addTodo: function() { //code... }

  //CHANGE TODO using the DOM
  changeTodo: function() { //code... }

  //DELETE TODO using the DOM
  deleteTodo: function() { //code... }

  //TOGGLE ALL using the DOM
  toggleCompleted: function() { //code... }
};
```

APP VERSION **9** - **Escape from the console**

- ✓ There should be an <li> element for each todo
- ✓ Each <li> element should contain `.todoText`
- ✓ Each <li> element should show `.completed` status

VERSION 9 - ESCAPE FROM THE CONSOLE

```
//VIEW OBJECT - CREATE LI ITEMS ON THE FLY
var view = {

  displayTodos: function () {

    const todoUl = document.querySelector('ul');

    //Start Loop
    for (let i = 0; i < todoList.todos.length; i++) {
      let todo = todoList.todos[i];
      let x = '( ) ';
      if (todo.completed === true) {
        x = '(x) ';
      }

      let todoTextWithCompletion = x + todo.todoText;
      let todoLi = document.createElement('li');
      todoLi.textContent = todoTextWithCompletion;
      todoUl.appendChild(todoLi);
    } //End Loop

  } //End displayTodos() method

};
```

APP VERSION **10** - **Click to delete**

- ✓ There should be a way to create delete buttons
- ✓ There should be a delete button for each todo
- ✓ Each <li> should have an id with the todo index
- ✓ Delete buttons should have access to the **todo id**
- ✓ Clicking delete should update `todoList.todos[]` and the **DOM**

APP VERSION **11** - **Destroy all for() loops**

✓ `todoList.toggleAll()` should use `forEach()`

✓ `view.displayTodos()` should use `forEach()`