JavaScript Essential Training

1.JavaScript: A Brief Introduction

JavaScript: First contact

- Modern JavaScript and JavaScript tooling
- JSX and TypeScript
- Frameworks

JSX
A syntax extension of JavaScript created for the React JavaScript framework.

Components

- Objects
- Methods
- Function s
- Template literals
- Arrays

Navigating the JS landscape

- JavaScript is constantly evolving.
- The core languages; sometimes referred to as vanilla JavaScript or Vanilla JS

ECMAScript
The browser specifications of the JavaScript language

ES6, ES2015, ES2017, ES 2020, Etc.
Refers to the user of features defined in ECMAScript, but not necessarily supported by modern browser.
Using ECMAScript usually means also using Babel.js to make it work in current browser implementations

TypeScript
Variation, dialect, or flavor of JavaScript introducing features like strong typing

React, Vue, Angular
JavaScript frameworks allowing us to write JavaScript based front end applications
Introduce new coding conventions like JSX, and reliance on tools like Babel, WebPack, and Node.js

npm, WebPack, Gulp
Build tools and infrastructure to automate the process of optimizing human readable JavaScript for best browser performance

Node.js
JavaScript server runtime used to run JavaScript everywhere; used to run npm, WebPack, Babel, and more on your computer.

Tools for working with JavaScript

- Modern Browser – ideally all browsers for testing
- Code editor – Visual Code Studio is becoming the industry standard

- Live server environment – extension to Visual Studio Code
- The browser console – available in most browsers

Linting and Formatting

- Prettier helps automatically clean up your formatting
- ESLint helps automatically detect coding errors and can-do basic cleanup automatically
- Both require Node.js

Learning JavaScript backward

- Objects and methods
- Data Types and the DOM
- Methods, functions, and events

Focus on finding ways that makes the JavaScript language make sense to you

Chapter Quiz

1. What does it mean when we say JavaScript is an "object-oriented" language?

Answer: JavaScript is modeled around objects with properties and methods which can be handled in a modular fashion.

2. What happens to the website and the code when you write code in the browser console?

Answer: Code in the browser console impacts the current browser instance only. It exists in the console for as long as the window is open.

3. What is an indicator of someone being a good JavaScript developer?

Answer: They follow standards, invest in learning, use formatting and linting tools for consistency, and write accessible code.

4. What is the natural environment for JavaScript?

Answer: The browser, server environments, and your computer.

5. What is ECMAScript?

Answer: The specification describing how browsers should implement and interpret JavaScript.

6. Where should you develop and test your JavaScript code?

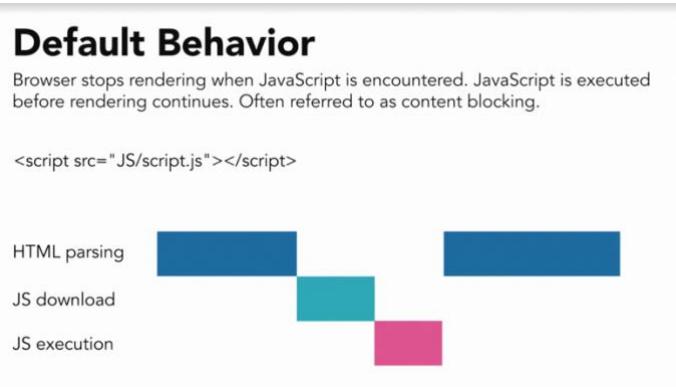
Answer: Develop in a code editor, test in as many browsers as you can get your hands on.

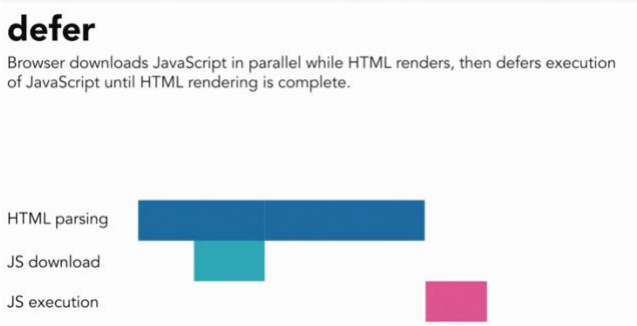
7. Why have command line and automation tools become popular in JavaScript development?

Answer: They simplify complex processes and introduce features to help developers write better code.

2.Up and Running with JS

Modern JavaScript loading





async/defer Should be Standard
Only use render blocking when you have specific reason. Loading JavaScript in the footer is now an anti-pattern

Chapter Quiz

1. When does the browser execute JavaScript?

Answer: By default: When the script is encountered. If the script is set to "async", when the script is fully loaded. If the script is set to "defer" when the entire HTML page is rendered.

2. What is the correct markup for adding an external JavaScript file to an HTML document?

Answer: <script src="javascript.js" async></script>

3. What happens when you defer JavaScript?

Answer: The browser loads the JavaScript asynchronously when it is encountered, then waits until all HTML is rendered before executing the script.

4. JavaScript modules are heavily used in frameworks like React and Vue. What is the advantage of using modules?

Answer: Modules enable modularization of code where individual functions, components, data objects, and other parts can be separated into individual files.

3.Objects

JavaScript Objects: The code version

Define an object:
const backpack; (variable holds data)

const backpack = {}; (curly brackets define data as an object)

const backpack = { name: “Everyday Backpack”, }; (properties defined using key pairs.)

```
const backpack = {
name: “Everyday Backpack ”,
volume: 30,
color: grey,
strapLength: {.     (Properties can nest sub-objects with their own properties)
left: 26,
right: 26,
},
lipOpen: false;
};
```

this. Example:

```
lidOpen:false,
toggleLid: function (lid status) {
```

this.lipOpen = lipStatus
“this” keyword refers to the current object

Object Containers

Objects are typically Constants

- We can change the properties of the object inside the container. We can’t remove or replace the object from the container

Object Properties

Example:
const backpack = {
name: “Everyday Backpack ”, **(name is key and Everyday Backpack is the value)**
volume: 30,
color: grey,
pocketNum: 15, **(camelCase property names to avoid issues)**
strapLength: {.
left: 26,
right: 26,
},
lipOpen: false;
};

3.Sidebar: Strings Output
Mix text variables with template literal

How to display JavaScript in HTML Example:
Script.js

```
import Backpack from "./Backpack.js";

const everydayPack = new Backpack(
  "Everyday Pack",
  30,
  "grey",
  15,
  26,
  26,
  false,
  "December 5, 2018 15:00:00 PST"
);

const content = `
  <main>
```

```
    <article>
      <h1>${everydayPack.name}</h1>
      <ul>
        <li>Volume: ${everydayPack.volume}</li>
        <li>Color: ${everydayPack.color}</li>
        <li>Age: ${everydayPack.backpackAge()}</li>
        <li>Number of pockets: ${everydayPack.pocketNum}</li>
        <li>Left strap length: ${everydayPack.strapLength.left}</li>
        <li>Right strap length: ${everydayPack.strapLength.right}</li>
        <li>Lid status: ${everydayPack.lidOpen}</li>
      </ul>
    </article>
  </main>
`;

document.body.innerHTML = content;

console.log("The everydayPack object:", everydayPack);
console.log("The pocketNum value:", everydayPack.pocketNum);
console.log("Days since aquired:", everydayPack.backpackAge());
```

Backpack.js

```
class Backpack {
  constructor(
    name,
    volume,
    color,
    pocketNum,
    strapLengthL,
    strapLengthR,
    lidOpen,
    dateAcquired
  ) {
    this.name = name;
    this.volume = volume;
    this.color = color;
    this.pocketNum = pocketNum;
    this.strapLength = {
      left: strapLengthL,
      right: strapLengthR,
    };
```

```
    this.lidOpen = lidOpen;
    this.dateAcquired = dateAcquired;
  }
  toggleLid(lidStatus) {
    this.lidOpen = lidStatus;
  }
  newStrapLength(lengthLeft, lengthRight) {
    this.strapLength.left = lengthLeft;
    this.strapLength.right = lengthRight;
  }
  backpackAge() {
    let now = new Date();
    let acquired = new Date(this.dateAcquired);
    let elapsed = now - acquired; // elapsed time in milliseconds
    let daysSinceAcquired = Math.floor(elapsed / (1000 * 3600 * 24));
    return daysSinceAcquired;
  }
}

export default Backpack;
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>String output</title>
    <script type="module" src="Backpack.js"></script>
    <script type="module" src="script.js"></script>
  </head>
  <body></body>
</html>
```

Traditional String Output

Example:

const content = “<h1> + everydayPack.name + “</h1>”; **(Will use name of everyday backpack for header)**

6.Variables: Containers for everything

Var

Var statement declares function scoped or globally scoped variables

Example:

```
var container = 5
```

Scope

Variables that are globally scoped can be re assigned a value.

Let

The let statement declares a block scoped local variable.

Example:

```
let color = “blue”;
```

Const

Constants are block scoped much like variables declared using ley keyword. The value of a constant can’t be changed through reassignment and can’t be redeclared.

Example:

```
const color = “purple”;
```

Data Types

Examples:

String:

```
let stringDemo = "A string of text.";
console.log("String:", stringDemo);
```

Number:

```
let integerDemo = 4;
console.log("Integer:", integerDemo);

let floatDemo = 5.6;
console.log("Floating point number:", floatDemo);
```

Boolean:

```
let booleanDemo = true;
console.log("Boolean value:", booleanDemo);
```

Null Values (nothing):

```
let nullDemo = null;
console.log("Null value:", nullDemo);
```

Undefined:

```
let undefinedDemo;
console.log("Undefined:", undefinedDemo);
```

```
let undefinedAssignedDemo = undefined;
console.log("Undefined assigned:", undefinedAssignedDemo);
```

Object:

```
const objectDemo = {
  dance: "Mambo",
  number: 5,
};
console.log("Object:", objectDemo);
```

Array:

```
const arrayDemo = ["tango", "foxtrot", "waltz", "rumba", "bolero"];
console.log("Array:", arrayDemo);
```

7.Arrays

Arrays Explained

Functions and Arrays

- Add or remove objects
- Find objects
- Reorder objects

Example:

```
/**
 * Solution: Build and modify an array
 * - Build an array with 8 items
 * - Remove the last item
 * - Add the last item as the first item on the array
 * - Sort the items by alphabetical order
 * - Use the find() method to find a specific item in the array
 * - Remove the item you found using the find method from the array.
 */

const deskArray = [
  "pen",
  "camera",
  "phone",
  "notebook",
  "headphones",
  "lightbulb",
  "usb drive",
];
console.log("Original array:", deskArray);

// Remove the last item:
// @link https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop
// deskArray.pop();
// console.log("New array:", deskArray);

// Add last item as the first item on the array:
// https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift
// deskArray.unshift(deskArray.pop());
// console.log("Last item is now first:", deskArray);
```

```
// Sort items by alphabetical order:
// @link https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort
// deskArray.sort();
// console.log("Sorted array:", deskArray);

// Find "notebook":
// @link https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/find
// const foundItem = deskArray.find((item) => item === "notebook");
// console.log("Found item:", foundItem);

// Find and remove an item:
// @link https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice
// @link https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf
// let remove = "notebook";
// deskArray.splice(deskArray.indexOf(remove), 1);
// console.log(`Array with "${remove}" removed:`, deskArray);
```

8.Functions and Methods

Examples of expression and Immediately Invoked Function Expression:

```
/**
 * Working with functions
 * @link https://developer.mozilla.org/en-US/docs/Glossary/Function
 */

// Function declaration:
function doSomeMath(a, b) {
  let c = a + b;
  return c;
}

// Function expression:
const doMoreMath = function (a = 3, b = 2) {
  let c = a * b;
  return c;
};

console.log("Do some math:", doSomeMath(5, 6));
console.log("Do more math:", doMoreMath(5, 6));

// Immediately Invoked Function Expression (IIFE)
// (function () {
//   let a = 4;
//   let b = 6;
//   let c = doSomeMath(a, b);
//   console.log(`The sum of a and b is: ${c}`);
// })();
```

A standard function

Example:

```
/**
 * A standard function
 * @link https://developer.mozilla.org/en-US/docs/Glossary/Function
 */

const greenPack = {
  name: "Frog Pack",
```

```
  color: "green",
  volume: 8,
  pocketNum: 3,
};

const addPack = function (currentPack) {
  const newArticle = document.createElement("article");
  newArticle.innerHTML = `
    <h1>${currentPack.name}</h1>
    <ul>
      <li>Volume: ${currentPack.volume}</li>
      <li>Color: ${currentPack.color}</li>
      <li>Number of pockets: ${currentPack.pocketNum}</li>
    </ul>
  `;
  return newArticle;
};

const main = document.querySelector("main");
main.append(addPack(greenPack));
```

Arrow Function

Example:

Traditional Function:

```
function (a) {
return a + 100;
}
```

Traditional Function turned into Arrow Function:

```
a => a + 100;
```

Callback

Example:

```
callback(finalTip)
```

Switch

Example:

```
switch (true) {
case age < 30:
description = “new”
break;

case age >= 30 && age < 365:
description = “lightly used”
break;

case age >= 3365 && age < 1095:
description = “used”
break;
```

```
case age  >= 1095:
description = “old”
break;

default:
console.log(‘There is no description for ${age} . ‘)
}
```

9.Events

Typical Dom Events

Dom events are sent to notify code of interesting things that have taken place.

Event Listeners

addEventListener()

target.addEventListener (target, callback [, options]);
(Options typically “false” or left blank)

Example:

```
const button = document.querySelector(“button”)

button.addEventListener{
“click”,
( e ) => {
console.log (‘Event fired: ${e}’)
}
);
```