Working with Data (Concept Review)

Data in JavaScript

(Primitive) Data Types

8
“orange”
false
null

Array

[8, “orange”, false]

Object

{ name: “Leo”,
age: 8,
favoriteColor: “orange”,
present: false }

(Primitive) Data Types

```
8
“orange”
false
null
```

Array

```
[8, “orange”, false]
```

Object

```
{ name: “Leo”,
age: 8,
favoriteColor: “orange”,
present: false }
```

(Primitive) Data Types

8
“orange”
false
null

Array

[8, “orange”, false]

Object

{ name: “Leo”,
age: 8,
favoriteColor: “orange”,
present: false }

Nested Objects & Arrays

```
{ name: “Leo”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }

{ name:
    {firstName: “Leo”, lastName: “DiCaprio”},
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }
```

Nested Objects & Arrays

```
{ name: “Leo”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }
```

```
{ name:
    {firstName: “Leo”, lastName: “DiCaprio”},
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }
```

Nested Objects & Arrays

```
{ name: “Leo”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }
```

```
{ name:
    {firstName: “Leo”, lastName: “DiCaprio”},
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false }
```

```
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”
present: false }
```

Nested Objects & Arrays

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

Nested Objects & Arrays

```
let arrName = [
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

Nested Objects & Arrays

```
let arrName = [
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

arrName[0]

arrName[1]

Nested Objects & Arrays

```
let arrName = [
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

Nested Objects & Arrays

```
let arrName = [
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName[0].favoriteColors[0]

arrName[0][“favoriteColors”][0]
```

Nested Objects & Arrays

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

Nested Objects & Arrays with forEach()

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName.forEach(item =>
        console.log(item.firstName)
)
```

Nested Objects & Arrays with forEach()

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName.forEach(person =>
        console.log(person.firstName)
)
```

Nested Objects & Arrays with forEach()

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName.forEach(kdjshfk =>
        console.log(kdjshfk.firstName)
)
```

Nested Objects & Arrays

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

Nested Objects & Arrays with forEach()

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName.forEach(person => {
  person.favoriteColors.forEach(color => {
            console.log(color);
  });
});
```

Nested Objects & Arrays with map()

```
[
{ firstName: “Leo”,
lastName: “DiCaprio”,
age: 8,
favoriteColors: [“orange”, “rainbow”]
present: false},

{ firstName: “Leah”,
lastName: “DiCapricious”,
age: 10,
favoriteColors: [“blue”, “rainbow”]
present: true },
]
```

```
arrName.map(person => {
        let colors =
          person.favoriteColors.map((color) =>
          `<li>${color}</li>`).join(“ ”);
});
```

forEach()

do something with each item in an array; run the code for each item

map()

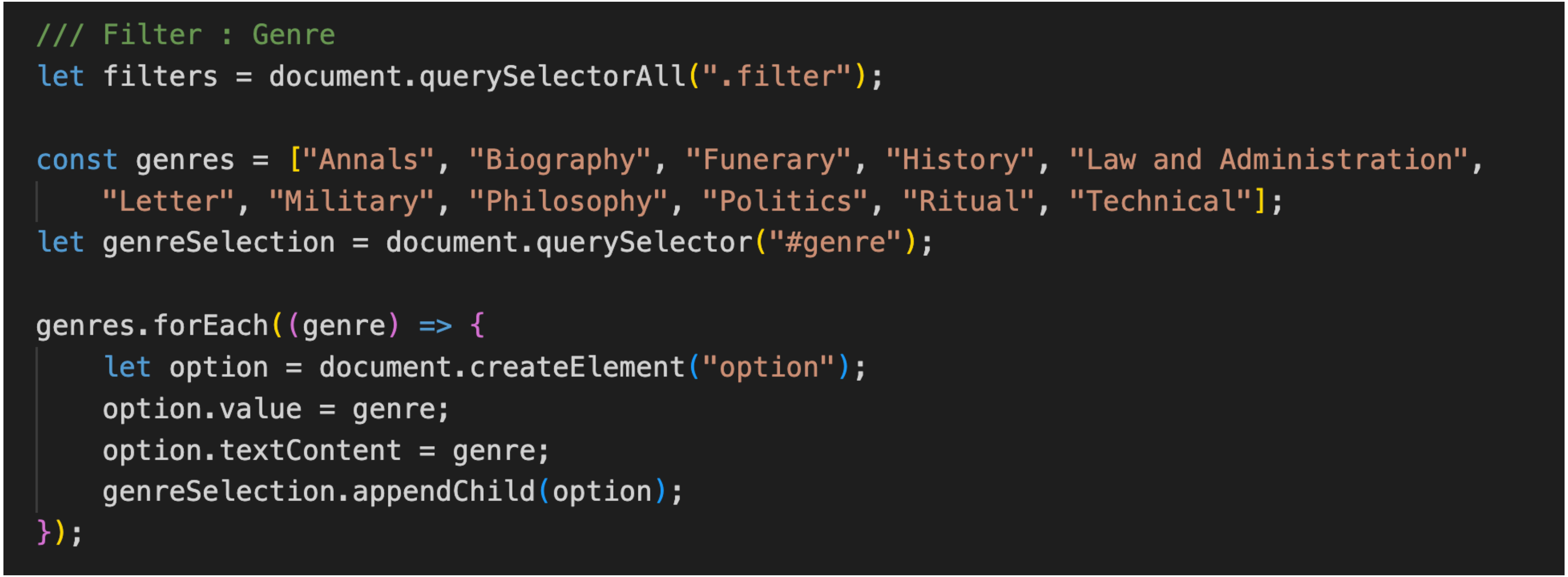
apply a transformation across the array and return a new array

forEach() and map()

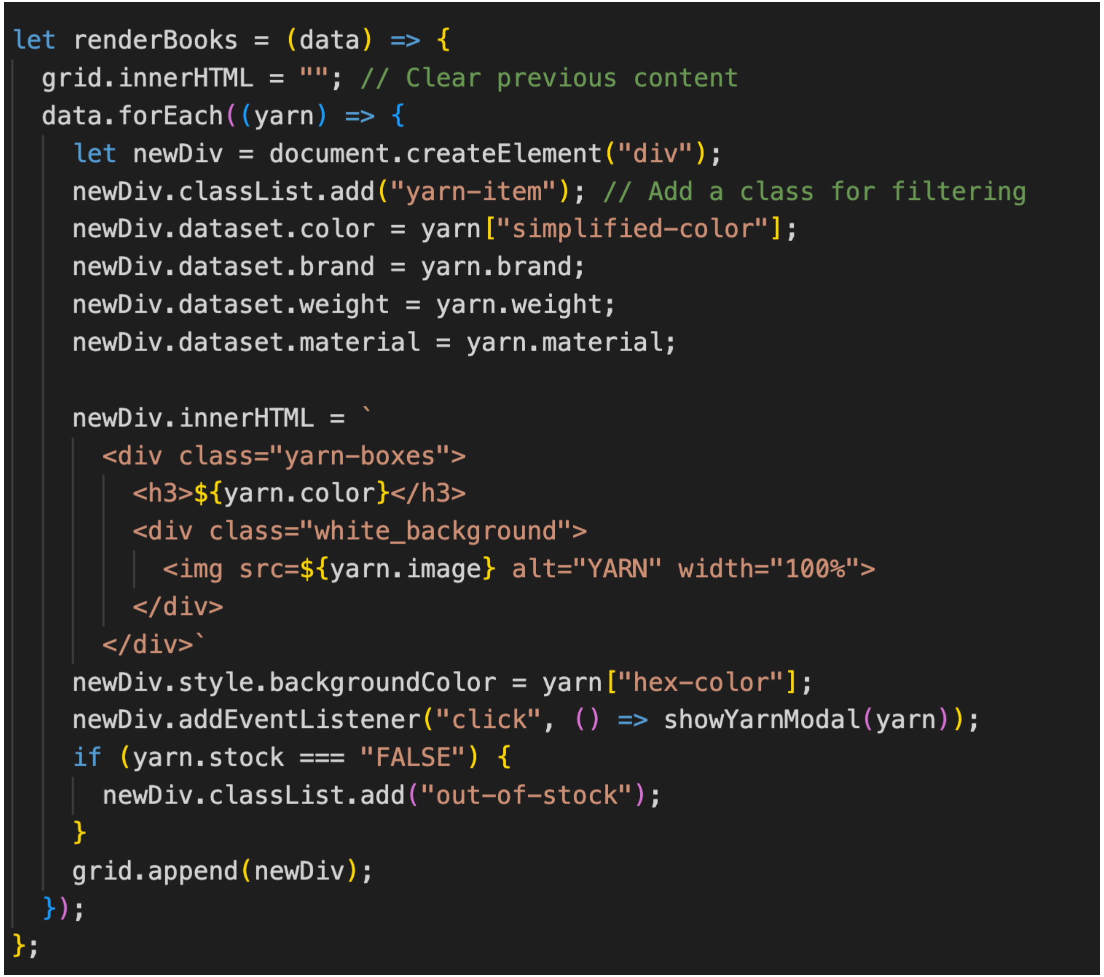
```
favoriteColors.forEach(color => {
        let content = color;
 });
```

```
favoriteColors.map((color) =>
        let content = `<li>${color}</li>`).join(“ ”);
});
```

Examples

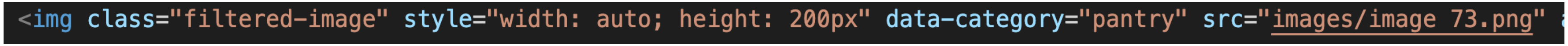


```
/// Filter : Genre
let filters = document.querySelectorAll(".filter");

const genres = ["Annals", "Biography", "Funerary", "History", "Law and Administration",
    "Letter", "Military", "Philosophy", "Politics", "Ritual", "Technical"];
let genreSelection = document.querySelector("#genre");

genres.forEach((genre) => {
    let option = document.createElement("option");
    option.value = genre;
    option.textContent = genre;
    genreSelection.appendChild(option);
});
```

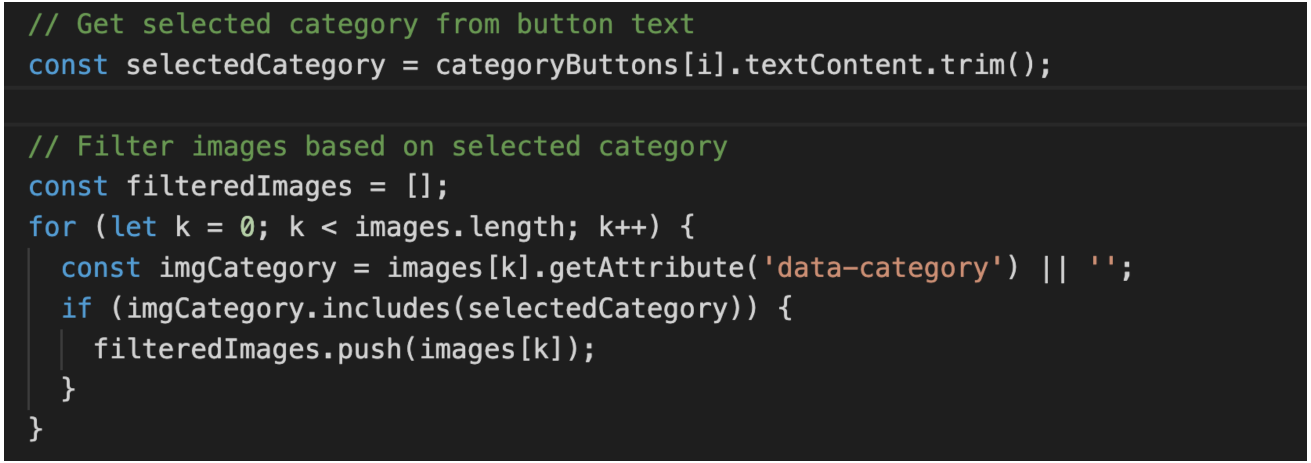


```javascript
let renderBooks = (data) => {
  grid.innerHTML = ""; // Clear previous content
  data.forEach((yarn) => {
    let newDiv = document.createElement("div");
    newDiv.classList.add("yarn-item"); // Add a class for filtering
    newDiv.dataset.color = yarn["simplified-color"];
    newDiv.dataset.brand = yarn.brand;
    newDiv.dataset.weight = yarn.weight;
    newDiv.dataset.material = yarn.material;

    newDiv.innerHTML = `
      <div class="yarn-boxes">
        <h3>${yarn.color}</h3>
        <div class="white_background">
          <img src=${yarn.image} alt="YARN" width="100%">
        </div>
      </div>`
    newDiv.style.backgroundColor = yarn["hex-color"];
    newDiv.addEventListener("click", () => showYarnModal(yarn));
    if (yarn.stock === "FALSE") {
      newDiv.classList.add("out-of-stock");
    }
    grid.append(newDiv);
  });
};
```

Christine



```html
<img class="filtered-image" style="width: auto; height: 200px" data-category="pantry" src="images/image 73.png" a
```



```javascript
// Get selected category from button text
const selectedCategory = categoryButtons[i].textContent.trim();

// Filter images based on selected category
const filteredImages = [];
for (let k = 0; k < images.length; k++) {
  const imgCategory = images[k].getAttribute('data-category') || '';
  if (imgCategory.includes(selectedCategory)) {
    filteredImages.push(images[k]);
  }
}
```

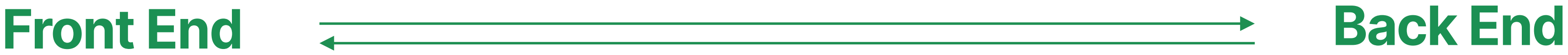
A Note on the Ends

Front End

the user-facing part of a website or application, encompassing the visual elements and interactive components that users see and interact with, built using technologies like HTML, CSS, and JavaScript (frameworks like React, a combination of HTML & JS called JSX, also fall into this bucket)

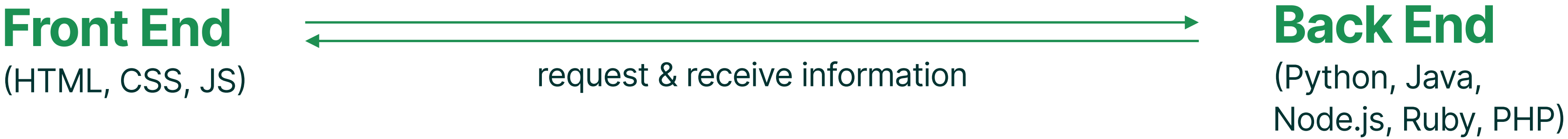
Back End

the server-side logic and infrastructure that handles data processing, storage, and application functionality



Front End

Back End



Front End

(HTML, CSS, JS)

request & receive information

Back End

(Python, Java,
Node.js, Ruby, PHP)

API

An API (Application Programming Interface) is a way for different apps or websites to talk to each other and share information.

It's like ordering something online—you send a request, and the website sends you what you asked for.

API

Because you don’t know how long it’s going to take for this data to arrive, you ask JavaScript to return a promise instead.

If you’ve used “fetch” then you’ve asked a promise.

API

JavaScript is a synchronous language by default. That means it runs code line by line, in order. So for things like API calls or timers, which might take time, we tell JavaScript to return a promise, and come back to it when the data is available.

API

Examples of asynchronous functions in JavaScript:

fetch() used to get data from an API or CSV
setTimeout() waits a certain amount of time, then runs code
async/await a cleaner way to wait for asynchronous things to finish