Things_to_learn_before_learning_a_Javascript_Framework.md

Javascript fundamentals

- Basic syntax
- Variables
- Arrays & Object Literals
- Events
- Functions, loops, conditionals

Modules

- ES6 modules, Typescript (Angular)
- Parcel, Webpack & Babel
- Export & Export Default

Classes

- Structuring a class
- Constructors
- Methods & properties
- Instantiation
- Extending classes

Arrow functions

- scope and "lexical this"

```
setTimeout(() => {
        console.log("Arrow function");
}, 1000);
```

Promises/asynchronous requests

- Create & receive requests
- Standard .then() and catch() syntax
- Async/await is optional but recommended
- Fetch API for making HTTP requests

```
function createPost(post) {
        return new Promise((resolve, reject) => {
                setTimeout(() => {
                        posts.push(post);
                        const error = false;
                        if(!error) {
                                resolve();
                        } else {
                                reject('Error');
                        }
                }, 2000);
        })
}

createPost({ title: "post 3", body: "post 3" })
.then(getPosts)
.catch(err => console.log(err));
```

Promise.all:

```
const promise1 = Promise.resolve('Hello World');
const promise2 = 10;
const promise3 = new Promise((resolve, reject) =>
setTimeout(resolve,2000,'Goodbye'));
const promise4 = fetch('https://jsonplaceholder.typicode.com/users')
.then(res => res.json());

Promise.all([promise1, promise2, promise3, promise 4])
.then(values =>
        console.log(values)
);
```

Async/Await:

```
async function init(){
        await createPost({ title: 'Post 3', body: 'Post 3'});
        getPosts();
}
init();
```

Async/Await/Fetch:

```
async function fetchUsers(){
        const res = await fetch('https://jsonplaceholder.typicode.com/users');
        const data = await res.json();
        console.log(data);
}
```

Destructuring

- const { name, email } = user;
- const { name, email, adddress: { city } } = user;

Concepts of Components & State

- Each component can have it's own data & state of being
- Nested components (Parent & children)

Spread operator

State is usually immutable, we cannot simply change it, we need to make a copy. The spread operator(...) allows us to do that

```
const userState = {
        name: 'John'
}
const newUserState = {
        ...userState,
        age: 30
}
```

Higher order array functions

- forEach(): iteration/looping

```
companies.forEach(function(company){
        console.log(company);
});
```

- map(): manipulating data to create new array

```
const companyNames = companies.map(function(company){
        return company.name + '  ' + company.category;
});
```

OR

```
const companyNames = companies.map(() => company.name + '  ' + company.category);
```

- filter(): used to filter out certain pieces of data

```
const test = ages.filter(function(age){
        if(age >= 21){
                return true;
        }
});
```

OR

```
const test = ages.filter(age => age >= 21);
```

- sort(): used to sort based on object property

```
const sortedCompanies = companies.sort(function(c1, c2){
        if(c1.start > c2.start) return 1;
        else return -1;
});
```

OR

```
const sortedCompanies = companies.sort((a,b) => (a.start > b.start ? 1 :-1));
```

OR

```
const sortedCompanies = companies.sort((a,b) => (a.start - b.start));
```

- reduce(): reduces the array to a single value

```
const agesSum = ages.reduce(function(total, age){
        return total + age;
},0);
```

OR

```
const agesSum = ages.reduce((total, age) => total + age, 0);
```