Making things click together - RL project development

adapt

Before starting development on new project (0)

Communication paths

- Knowing skills of team members
- Who handles what
- How to gain access to external tools
- Information storage and accessibility

adapt

Before starting development on new project (1)

Defining project goal

- **Short term** - identify project's critical parts and estimate their technical difficulties and possible challenges
- **Long term** - gain understanding of what waits ahead

adapt

Before starting development on new project (2)

Choosing framework based on:

- Project scope
- Existing solutions
- Team experiences
- Internal research
- Accessibility

adapt

Before starting development on new project (3)

Analyzing project design/wireframes

- Defining what is section/block/element
- Determine breakpoints
- Identifying anomalies

Define base style (CSS)

- Basic style for basic HTML elements
- Defining what is block/element/section
- Extensions (SASS/SCSS/PostCSS/...)
- Methodology (BEM/OOCSS/SMACSS/...)

adapt

Before starting development on new project (4)

Style guide is a set of standards for the writing and design of documents, either for general use or for a specific publication, organization, or field. (It is often called a style sheet, though that term has other meanings.) A style guide establishes and enforces style to improve communication.

- Do we need it?
- Advantages?
- Disadvantages?

adapt

Before starting development on new project (5)

Code coverage: can we do it and should we do it?

Functional testing types:

- Unit testing
- Integration testing
- System testing
- Sanity testing
- Smoke testing
- Interface testing
- Regression testing

adapt

Before starting development on new project (6)

Data structure

- What data do we need?
- How it should be structured?
- What can we get it?
- Should we normalize data and if so - where?
- Data overhead problem

adapt

Proof of concept

Proof of concept (**PoC**) is a realization of a certain method or idea in order to demonstrate its feasibility, or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. A proof of concept is usually small and may or may not be complete.

Coding rules

adapt

Coding rules: naming conventions (0)

adapt

Coding rules: naming conventions (1)

Properties/variables/functions

- camelCase vs under_score
- Underscore before the property name
- UPPERCASE Constants
- Single letter prefixes
- Capital first letter
- Single letter variable
- Meaningless variable name (foo/bar/...)
- Scope of variable

Coding rules: naming conventions (2)

Project structure and it’s folders

- Basically whatever works for you
- Group assets in graceful/meaningful way
- Understand usage of content inside it
- Filename should reflect its location and usage
- Don’t be afraid of having deep folder “depth” as long as it make sense
- camelCase all the way except “classes”
- Prevent filename collisions

adapt

Passive exercise no. 0

adapt

Coding rules: style (3)

adapt

Coding rules: style (3)

ESLint - linter for JavaScript, which allow to create your own rule and rule sets and plug them in as you see fit. It helps to find problematic patterns or code that doesn't adhere to certain style guidelines.

```
{
    "rules": {
        "semi": ["error", "always"],
        "quotes": ["error", "double"]
    }
}
```

adapt

Coding rules: style (4)

Setting up ESLint:

1. *npm i -g eslint*
2. *Eslint --ext .js,.jsx -c ~/my-eslint.json file.js or “/**”*

Active exercise no. 0

adapt

Coding rules: style (5)

Stylelint - A mighty, modern linter that helps you avoid errors and enforce conventions in your styles.

```
{
    "rules": {
        "color-hex-case": "lower",
        "color-hex-length": "short",
        "color-no-invalid-hex": true
    }
}
```

adapt

What is Babel and what it does?

Babel is a **JavaScript** compiler/transpiler. Its allows developers to write source code in a preferred programming language or markup language and have it translated into JavaScript, a language understood by modern web browsers.

```
const presets = [
  [
    "@babel/env",
    {
      targets: {
        edge: "17",
        firefox: "60",
        chrome: "67",
        safari: "11.1",
      },
      useBuiltIns: "usage",
    },
  ],
];

module.exports = { presets };
```

adapt

What is Webpack and what it does?

Webpack is a **module bundler**. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'bundle.js'
  }
};
```

Style guide

Style guide targets frontend features implemented with JavaScript, CSS, and HTML.

Benefits:

- Reduce code overhead.
- Clear and reusable components.
- Style consistency.
- Saves time.

Example: https://react-styleguidist.js.org/examples/basic/

Static typing in JS (0)

JavaScript does not have static types. Therefore there is no way to do a static type check that you can rely on in JavaScript.

You cannot rely on **instanceof** or **typeof**. It lies and leads to unpredictive behavior.

((() => 0) instanceof Object) === true

Nominal types (type checking based on the name or identity of a single class/interface) will never work well in JavaScript.

Static typing in JS (1)

Benefits:

- Real-time detection of bugs/errors.
- Tells purpose of the function.
- Scales down complex error handling.
- Reduce runtime type errors.
- Clarifies data flow.

Static typing in JS (2)

Two main options:

- TypeScript

```ts
interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person: Person) {
    return "Hello, " + person.firstName + " " + person.lastName;
}
```

- Flow

```ts
type User = {|
  userId: number,
  name: string
|}

export const fetchUserInvestments = ({user, name}: User) => {
  // ...
}
```

adapt

Questions?