# React Intro

Monday, 26 October 2020 22:45

React Team Definition -> A Javascript Library for building UI's
- "A Javascript Library" -> Runs in the browser, not on a server! (Speed)
- "building UI's" -> Splittable into manageable components! (Dependencies)

**Some Key Thoughts:**
Definition of React: Writing manageable, maintainable and reuseable code
Essence of React: Think of it like writing "custom HTML elements"
Solution for: Having to write complex UI, cherrypicking from HTML and JS features

**Defining the Benefit of a General framework or library:**
Provides the Ability to Update, Maintain and Develop according to modern practices.

**Meet the Accomplices:**
**React DOM** -> as opposed to logically coding components, covers rendering to HTML DOM
**JS Preprocessor Babel -> TODO: research this (allows html-like JSx in JS syntax)**

**The most basic teamplay of React and ReactDOM explained:**
A react component is just a function that returns to-be DOM code using JSx in a JS file.
Function call "ReactDOM.render()" renders the React-parsed JS(x) into the actual DOM at runtime.

Important to remember is that the seemingly HTML in JSx is just a close match custom JS syntax.
For example, the HTML keyword "class" already exists in JS so JSx matched it by "className".

The previously referenced function call takes argument "<ElemName />" and a HTML element.
An element is provided using, for example, "document.querySelector('#p1')" as the selector.

Like that, it points to existing DOM elements and adds in a block element containing our JSx code.

**Tailoring the contents of a component to the caller's wishes:**
The React "props" argument references (and contains) all attributes the ReactDOM might use.
Like so, bare text between the <p> tags could display attr "name" using "{props.name}".

**Evolving ReactDOM's inline custom elem to a variable:**
Instead of calling multiple times for rendering, we could use the custom elems inside a variable.
By doing this, provided we wrap it by a div (1 root elem allowed), we render multiline code!

**Why React?**
Huge Ecosystem of active community-support, driving high-performing sourcecode.
A focus on business logic as opposed to catching up to individual code-support of every component.
Solution to a UI-state-problem where vanilla JS requires manual focus-setting when editing the DOM.

**Alternatives to React, providing Context:**
Backbone and Amber are alternatives, greater ones are Angular and Vue.
Not so much an alternative is jQuery, it focuses on the "how" and merely assists in traversing the DOM.

**Differing SPA's and MPA's:**
Single Page App -> returns on page which morphs into (virtual) new pages using runtime-logic.
One single "ReactDOM.render()" call roots all else into one elem, mounted on the DOM.

Multi Page App -> pages are bound to their routing, React provides widgets etc BUT there's server comm.
The "ReactDOM.render()" calls render instances of React-code-context into their DOM-mounting points.