Object-Oriented JavaScript

Prototype: JavaScript objects inherit features from one another.

function Person(name, age) {

this.name = name;

this.age = age;

}

Person.prototype.getName = function() {

return this.name;

}

let michelle = new Person(“Michelle”, 16)

michelle.getName() //”Michelle”

Person.prototye == michelle.\_\_proto\_\_

-When calling michelle.getName() it will first check within the function Person for .getName. If not found, it will check on the michelle.\_\_proto\_\_ which is Person.prototype.

JavaScript runs in 2 main environments : browser and server-side.

JavaScript code was traditionally run by the user’s browser. Node.js is a server-side JavaScript framework that allows programmers to run JavaScript from their terminals.

npm = node package manager

ECMAScript is the standardized specification that forms the basis for JavaScript.

expression is a line of code that returns a value.

**Syntax:**

continue: skips the current iteration

break: exits the loop

String.prototype.toLowerCase

String.prototype.toUpperCase

String.prototype.indexOf

+: concatenation

Array.prototype.length

Array.prototype.pop

Array.prototype.push

Array.prototype.unshift

Array.prototype.shift

Array.prototype.indexOf

Array.prototype.slice(start, end)

Array.prototype.includes

Array.isArray

JavaScript has 6 data type: Numbers, String, Boolean, Undefined, Null, and objects.

NaN stands for Not A Number. It is a property of the global object and its not a JavaScript type.

In JS, zeros, empty strings, undefined, null, and NaN are all falsely values.

A primitive type is data that is not an object and cannot have methods. ( In Ruby, everything is an object, but in JS there are primitive data types that are not objects.

Undefined is JS default return value

**Variables and constants** in Javascript: var, let(ES6), const(ES6), window, and global.

var is a functionally-scoped variable. var is less preferred.

let is a block-scoped variable. JavaScript will raise syntaxError if let variable get to declare twice in one bock.

const is used to declare constants. Constants should be used for constructs that will not be re-declared or re-assigned

if there is no declaration. It will become a global. Unintended global variables create confusing and unpredictable errors.

Functions in JS are a special type of JS objects. (anything you can do with object applys on functions as well)

There are 3 ways to Declare functions: Function-style, Expression-style and fat arrow-style.

function name(arg1,arg2) {}

const name = function(arg1,arg2){};

const name = (arg1,arg2) => {};

function that are passed as an argument to another function are called callbacks.

object.method(arg1,arg2) => method-style

when we call let cat = new Cat(). JS creates a new blank object. JS sets a special cat.\_\_proto\_\_ property to Cat.prototype. JS runs the code in the body of the Cat function. Sets this to the blank object.

when we request cat.meow(). JS looks in cat object for meow property. if not find, it accesses the cat.\_\_protot\_\_

JS functions can take fewer arguments than expected. Unspecified arguments will have value of undefined. It also can take more arguments. All arguments are stored in a array

asynchronous function schedules work to be done in the background. (AJAX: background web request), event, timers.

The Document Object Model(DOM)

browser provides an API by which JS can access the HTML content of page.

get HTMLElements:

document.getElementById(selector) => return a single element matching the id

document.getElementByClassName(selector) => return array-like NodeList tha contains all elements matching the class name.

document.querySlectorAll(selector)

disable the input: document.getElementById(“ “).disabled = true;

Remove a p from document

const p = document.querySelector(“p”);

p.parentElement.removeChild(p);

EventTarget#addEventListener method: ask brower to store a function.

EventTarget is a parent class of HTMLElement. the function will be call when submit event happens

comment event: click, dbclick, inout, change, submit, keydown, keyup, mouseover, mousemove, mouseout, scroll, resize

AJAX request: Asynchronous JavaScript and XML (JSON). When the AJAX request completes, the browser will not load a new page; it will stay on the same page. Instead, on receipt of the HTTP response, the browser will fire a JS callback function

DOM events bubble: After an event triggers on the deepest possible element, it then triggers on its parents in nesting order

Vanilla JS: js without any library

DOM elements:

Node-the basic element representing a node on the DOM

element-inherits from node; each element on the document is a decendent of element

HTMLElement-inherits from element, but more specifically represent HTML element ( those created from an HTML tag)THT

HTMLCollection- an array like object containing a collection of HTMLElement

Document- the root element of our page, serves as an entry point to the content in our page

If we want to select something from the page, we start by calling a method on document;

it will return

Form submit:

HTML:

<form>

  <input type="text" class="input"/>

  <input type="submit" class="submit" value="Add to list"/>

</form>

<ul id="list">

</ul>

JS:

const handleFavoriteSubmit = (e) => {

e.preventDefault();

const input = document.querySelector(".input");

const data = input.value;

input.value = "";

const newListLi = document.createElement("li");

newListLi.textContent = data;

const list = document.getElementById("list");

list.appendChild(newListLi);

};

const listSubmitButton = document.querySelector(".submit");

listSubmitButton.addEventListener("click", handleFavoriteSubmit);

Show element:

HTML:

<div class="element hidden">

  <form>

    <input type="text" class="photo-url-input" placeholder="image URL"/>

    <input type="submit" class="photo-url-submit" value="Add photo"/>

  </form>

</div>

CSS:

.element {

display: block;

}

.hidden {

display: none;

}

JS:

const showElement = (e) => {

const ele = document.querySelector(".element");

if (ele.className === "element") {

ele.className = "element hidden";

} else {

ele.className = "element";

}};

const photoFormShowButton = document.querySelector(".photo-show-button");

photoFormShowButton.addEventListener("click", showElement);

//for each

document.querySelectorAll("#restaurants li").forEach((li) => {

li.addEventListener("click", toggleLi);

`});

REACT:

Single page app only has one backend route that renders HTML. To allow users to interact with the database, asynchronous AJAX request send and retrieve information to the backend. And react updates the relevant portion of the page. This brings an improvement in performance because the page isn’t entirely reloaded with every click.

NPM-Node Package Manager

package.json- lists all of an app’s JS dependencies

react components are the building blocks of a React view-layer. They are JS functions that return HTML to be rendered onto a document. Because they are typically written in JSX, components often look like HTML dropped into a JS file.

JSX is a JS syntac extension that resembles HTML and XML. React code weitten in JSX mirrors the HTML its produces, improving readability and ease of development

// JSX

const quotes = (

<div className="quotes">

<h1>I love JavaScript!</h1>

</div>

);

// plain JavaScript

const quotes = React.createElement(

"div",

{ className: "quotes" },

React.createElement("h1", {}, "I love Javascript")

);

Redux is a JS framework for managing the frontend state of a web application. It allows us to store information in an organized manner in a web app and to quickly retrieve that information from anywhere in the app.

REACT:

import React from "react";

class Form extends React.Component {

constructor(){

super();

this.state = {}

}

render(){

return()

}

}

export default Form;

* this.state.xxx.map(ele => <Item />)
* this.setState({})
* e.preventDefault();
* const input = e.target.value;