1. Install nodejs and npm, show 2503 and 2502 error.

Solution:

Go to C:\Windows\temp, right click the folder. Select security, click edit, select users(….), click full control.

A screenshot of a social media post

Description automatically generated

1. Fix a vulnerable npm package in package-lock.json:

npm audit fix, npm will install latest version of package automatically.

不要隨便用，npm start后，會出現下面的error,

*npm ERR! code ELIFECYCLE*

*npm ERR! errno 1*

*npm ERR! react-demo@0.1.0 start: `webpack-dev-server --mode development --hot --open --history-api-fallback`*

*npm ERR! Exit status 1*

*npm ERR!*

*npm ERR! Failed at the react-demo@0.1.0 start script.*

*npm ERR! This is probably not a problem with npm. There is likely additional logging output above.*

*npm ERR! A complete log of this run can be found in:*

*npm ERR! C:\Users\fphwp\AppData\Roaming\npm-cache\\_logs\2019-11-16T20\_17\_37\_176Z-debug.log*

可以用npm update ... (e.g. npm update react)

1. SetState is not a function: 原因是没有bind function. 比如，原来是onClick={handleLogout}, 变成{this.handleLogout}即可解决
2. Indent: Ctrl k + ctrl f
3. ComponentDidMount is after render. The order is constructor, render,componentDidMount.
4. Difference between Component and pureComponent: component doesn’t implement *shouldComponentUpdate()* by default. On the other hand, purecomponent does implement it and perform a shallow comparison on react state and props values.

It increases app performance. Shallow comparison is comparing scala values and references when comparing object.

1. Name Is missing in props validation:

class Greeting extends React.PureComponent {

  render() {

    return <h1>Hi there, my name is {this.props.name}!</h1>;

  }

}

Add following code:

Greeting.propTypes = {

  name: PropTypes.string

};

Greeting後面的是小寫p開頭,name後面的是大寫P開頭

1. Sass: you can use variables and other sass function
2. Promise: A good rule of thumb is to always either return or terminate promise chains, and as soon as you get a new promise, return it immediately, to flatten things.

From <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises>

JS is single threaded, meaning that two bits of script cannot run at the same time. They have to run one after another.

<https://developers.google.com/web/fundamentals/primers/promises>

1. React-scroll: 随着scroll down, navigation menu的标题随着变色
2. React-parallax-tilt: when mouse move on the image, animation spears.
3. State an props: state is equivalent to local variables in a function. Props, on the other hand, is equivalent to function parameters.

Class SampleFunc extends React.Component{

Render(){

Return <div>Hello {this.props.name}</div>

}

}

<SampleFunc name = “Joni ” />

1. Async function always return a promise, whether you use await or not. That promise resolves with whatever the async function returns, or rejects with whatever the async function throws.

<https://developers.google.com/web/fundamentals/primers/async-functions>

fetch(url).then… equals to await fetch(url).

1. Hook: they are functions that let you hook into React state and lifecycle features from function components. With Hook, you can use state in function, which previously you had to convert it to a class.

useEffect: similar to componentDidMount and componentDidUpdate.

1. Bem: [https://codeburst.io/understanding-css-bem-naming-convention-a8cca116d252#targetText=BEM%20stands%20for%20Block%2C%20Element](https://codeburst.io/understanding-css-bem-naming-convention-a8cca116d252" \l "targetText=BEM stands for Block, Element)

<https://medium.com/fed-or-dead/battling-bem-5-common-problems-and-how-to-avoid-them-5bbd23dee319>

B: block, E: elements, M:modifier, cannot name element inside another element.

1. Sass: Mix are sass functions that group css declarations together. We can use it in any class with @include command. SCSS is Sass version 3. In sass, & take the place of the parent ‘s name.
2. Update set-value and meme in package-lock.json: use

npm install set-value and npm install mem

1. React template: <https://nndou.netlify.com/>
2. Multi lingual react: <https://medium.com/@ricklee_10931/react-multi-lingual-with-react-i18next-57879f986168>

<https://github.com/i18next/react-i18next>

1. Bootstrap: navbar-expand-lg讓nav menu排在一行
2. Bootstrap：mr-auto to force sibling columns away from one another.
3. Country Flag: <https://cdnjs.com/libraries/flag-icon-css>
4. Cannot display background image: set position in css and use style in js file
5. npm ERR! code ELIFECYCLE

npm ERR! errno 1

npm ERR! react-website@0.1.0 start: `webpack-dev-server --mode development --hot --open --history-api-fallback`

npm ERR! Exit status 1

npm ERR!

npm ERR! Failed at the react-website@0.1.0 start script.

npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:

npm ERR! C:\Users\Administrator\AppData\Roaming\npm-cache\\_logs\2019-12-17T05\_47\_36\_515Z-debug.log

解决方法：去webpack.config.js里删除

const ExtractTextPlugin = require(“extract-text-webpack-plugin”);

const extractTextPlugin = new ExtractTextPlugin(“css/app.css”);

1. Position in css: default is static, absolute is positioned to its first no static ancestor element, relative is positioned relative to its normal position.
2. … spread syntax:

<https://stackoverflow.com/questions/40124680/what-does-in-react-native-mean>

1. Z-index: stack order of element.
2. Vh: relative to 1% of the height of the viewport (viewport: the browser window size)
3. Server IP Address in cloudflare:

<https://support.cloudways.com/creating-a-record-cloudflare/>

<https://stackoverflow.com/questions/46764113/how-to-get-the-ip-of-aws-s3-hosted-site>

Goto s3, properties->static website hosting, copy jw.flypenguins.net.s3-website-ap-southeast-2.amazonaws.com,

Goto cmd->ping jw.flypenguins.net.s3-website-ap-southeast-2.amazonaws.com

1. Where to change in aws for cloudflare:



右面Name Servers: 把原來的default值先保存到一個word裏，以備不時之需。然後把cloudflare裏的值複製過來。



把第一個flypenguins.net選中，在右面value的地方輸入cloudflare裏的 值

1. Breakpoint in scss:

- wide (1281px, 1680px)

- normal (981px, 1280px)

-narrow (841px, 980px)

- narrower (737px, 840px)

- mobile (481px, 736px)

- mobile portrait ( null, 480px)

From html5up directive

1. Dasharray in css: 6,6 第一个6代表线的长度，第二个6代表间隔，以此类推下去
2. Cubic-bezier in transition: set your own speed from start to end,it is belonged to transition-timing-function.
3. 522 cannot connect. 按照item 29方式,ping一下，得到32 bytes data后去cloudflare修改即可。
4. Potential vulnerable only in package-lock.json (git not provide merge)
5. npm install <dep>
6. npm uninstall <dep>

Step 2 remove <dep> in package.json where is not appeared before

<https://stackoverflow.com/questions/49582891/proper-way-to-fix-potential-security-vulnerability-in-a-dependency-defined-in-pa>

1. Router reference links:

<https://stackoverflow.com/questions/52859834/redirecting-after-deleting-page-route-in-react-and-redux>

<https://stackoverflow.com/questions/30915173/react-router-go-back-a-page-how-do-you-configure-history>

<https://javamastermind.com/2020/01/16/react-navigation-manage-session-history/>

<https://stackoverflow.com/questions/44121069/how-to-pass-params-with-history-push-link-redirect-in-react-router-v4>

1. Query link to:

<https://spin.atomicobject.com/2019/10/01/a-quick-start-guide-to-query-strings-with-react-router/>

<https://reacttraining.com/react-router/web/api/Link>

1. Promise

<https://medium.com/@theflyingmantis/async-await-react-promise-testing-a0d454b5461b>

<https://medium.com/@theflyingmantis/callbacks-vs-promises-and-basics-of-js-80d3d1515e81>

promise is an object that links the producing and consuming together.

Let promise = new Promise(function(resolve, reject)){}. The function passed to promise is an executor (it is singer).resolve and reject are callbacks. The executor should call only one resolve or one reject. All further calls of resolve and reject are ignored.

1. Closure:

Inner function has access to outer function’s variable.

1. currying:

<https://developer.huawei.com/consumer/cn/forum/topicview?tid=0201233940793580151&fid=23>

1. callback

<https://javascript.info/callbacks>

the second argument is as function that runs when the action is completed.

1. Chain

Promise.then(f1).catch(f2) == promise.then(f1,f2)

The answer is no, first one is chain, when error occurs in f1, can be catched by f2. the second one is unhandled.

<https://javascript.info/promise-chaining>

1. Error handling with promise

new Promise(function(resolve, reject) {

setTimeout(() => {

throw new Error("Whoops!");

}, 1000);}).catch(alert);

Won’t trigger catch. Because only synchronous errors are handled.error is generated not while the executor is running, but later.

new Promise(function(resolve, reject) {  
setTimeout(() => {  
reject(new Error("Whoops!"))  
}, 1000);  
}).catch(alert);

This works.

1. Promise.all

Promise.all([

new Promise(resolve => setTimeout(() => resolve(1), 3000)), // 1

new Promise(resolve => setTimeout(() => resolve(2), 2000)), // 2

new Promise(resolve => setTimeout(() => resolve(3), 1000)) // 3]).then(alert);

The result is [1,2,3].

Even though the first promise takes the longest time to resolve, it’s still first in the array of results.

45 anonymous

They have no name, so called anonymous. They are not accessible outside of ask.

ask(

"Do you agree?",

function() { alert("You agreed."); },

function() { alert("You canceled the execution."); });

46 async and await

Async ensures that the function returns a promise. Await makes JS wait until that promise settles and returns its value. That does not cost any CPU resources, because the engine can do other jobs in the meantime: execute other scripts, handle events, etc.

1. Function expression vs. Function declaration

A function declaration can be called earlier than it is defined. A function expression is created when the execution reaches it and is usable only from that moment.

<https://javascript.info/function-expressions>

1. Variable name

The name must contain only letters, digits, or the $ and \_ .The first character must not be a digit.each word except first starting with a capital letter: myVeryLongName. Capital-named constants are only used as aliases for hard-coded values.(e.g. const COLOR\_RED = “#F00”)

Vs C#

Must Start with letter or \_. May contain characters, decimal digit. Interface start with capital I. Attribute types end with Attribute.Enum use singular for non-flag and plural for flag.

<https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-1.1/x2dbyw72%28v%3dvs.71%29>

49. data type

NaN, BigInt: > 253-1 or <–(-253-1), null, undefined, type of null is not object. <https://javascript.info/types>

50 type conversions

Number(undefined) – NaN

Number(null) – 0

1. Array

Let arr = new Array(5);

Let arr = [];

(vs. C#:

Int[] arr = new int[5];

)

Push/pop(stack) run fast vs. Shift/unshift(queue) are slow.

52.

When set global parameters, got to .eslintrc, e.g.

  "globals":{

    "SERVICE\_ID": true,

    "TEMPLATE\_ID": true,

    "USER\_ID": true

  }

When set id secret, got to webpack, set DefinePlugin

const definePlugin = new webpack.DefinePlugin({

  SERVICE\_ID: JSON.stringify("your\_service\_id"),

  TEMPLATE\_ID: JSON.stringify("your\_template\_id"),

  USER\_ID: JSON.stringify("your\_user\_id"),

});

After add id value, refresh website does not work, ctrl c, redo npm start, it will works.

53 difference between PureComponent and Component is that React.Component doesn’t implement shouldComponentUpdate(), but React.PureComponent implements it with a shallow prop and state comparison.

54 shallow comparison

Shallow compare does check for equality. When comparing scalar values (numbers, strings) it compares their values. When comparing objects, it does not compare their attributes - only their references are compared

<https://stackoverflow.com/questions/36084515/how-does-shallow-compare-work-in-react>

55 lifecycle

A screenshot of a social media post

Description automatically generated

<https://medium.com/@aravishack/react-js-life-cycle-of-components-6267eb79b564>

1. TypeError: this.getResolve is not a function at object.loader安裝minicssextractplugin:

因为当前sass的版本太高, 直接用npm install sass-loader@7.3.1 --save-dev 就可以了，同名的第三库会自动更换

1. Entrypoint mini-css-extract-plugin = \*

When using miniCssExtractPlugin, it appears the above yellow text. To remove it, uninstall miniCssExtractPlugin, only use style-loader.

<https://github.com/webpack-contrib/mini-css-extract-plugin/issues/23>

58 if image size is too large, use apple preview to choose quality.

59

* Attributes – is what’s written in HTML.
* Properties – is what’s in DOM objects.

A small comparison:

|  | **Properties** | |
| --- | --- | --- |
| Type | Any value, standard properties have types described in the spec | |
| Name | Name is case-sensitive | |
| **Attributes** | |
| A string | |
| Name is not case-sensitive | |

1. querySelector(css) returns the first element for the given CSS selector.
2. document.createElement(tag) – creates an element with the given tag,
3. Why DOCTYPE is important:

top-level geometry properties may work a little bit differently when there’s no <!DOCTYPE HTML> in HTML. Odd things are possible.

In modern HTML we should always write DOCTYPE.

1. Add error information and remove it in 5 seconds

<https://javascript.info/coordinates>

1. Relative vs absolute:

Relative to its current position,if you set relative but no other attributes (top, left,bottom, right), it will have no effect on its position at all.

Absolute: based on its parent position

1. Event handlers:

// right

button.onclick = sayThanks;

<input type="button" id="button" onclick="sayThanks()">

button.onclick = function() {

sayThanks(); // <-- the attribute content goes here};

}

// wrong

button.onclick = sayThanks();

1. Perfect center in css:

display: flex;

Justify-content: center;

Align-items:center;

1. useContext: context provides a container containing some data and making it available to the entire hierarchy of components below.
2. Event.target vs this

For instance, if we have a single handler form.onclick, then it can “catch” all clicks inside the form. No matter where the click happened, it bubbles up to <form> and runs the handler. In form.onclick handler:

* this (=event.currentTarget) is the <form> element, because the handler runs on it.
* event.target is the actual element inside the form that was clicked.

1. data attribute in html5: data-\* (<article data-column=”3”>…</article>)

Js access: article.dataset.column

Css access: article[data-column=”3”] {}

1. Browser default actions

There are two ways to tell the browser we don’t want it to act:

* The main way is to use the event object. There’s a method event.preventDefault().
* If the handler is assigned using on<event> (not by addEventListener), then returning false also works the same.

1. Custom events

The generic Event(name, options) constructor accepts an arbitrary event name and the options object with two properties:

* bubbles: true if the event should bubble.
* cancelable: true if the event.preventDefault() should work.

1. 函数内部声明变量的时候，一定要使用var命令。如果不用的话，你实际上声明了一个全局变量

我们有时候需要得到函数内的局部变量。但是，前面已经说过了，正常情况下，这是办不到的，只有通过变通方法才能实现。

那就是在函数的内部，再定义一个函数。

1. Oncopy: If we want to disable selection to protect our page content from copy-pasting, then we can use another event: oncopy.
2. For JS-code it means that we should check if (event.ctrlKey || event.metaKey). ctrl in windows and cmd on Mac.
3. Drag n’s drop with mouse events
4. Events flow: ball.mousedown → document.mousemove → ball.mouseup (don’t forget to cancel native ondragstart).
5. At the drag start – remember the initial shift of the pointer relative to the element: shiftX/shiftY and keep it during the dragging.
6. Detect droppable elements under the pointer using document.elementFromPoint.
7. Moving the mouse:

Events mouseover/out trigger even when we go from the parent element to a child element. The browser assumes that the mouse can be only over one element at one time – the deepest one.

Events mouseenter/leave are different in that aspect: they only trigger when the mouse comes in and out the element as a whole. Also they do not bubble.

1. Pointer events

Remember to set touch-events: none in CSS for elements that we engage, otherwise the browser will hijack many types of touch interactions, and pointer events won’t be generated.

Unless you code for Internet Explorer 10, or for Safari 12 or below, there’s no point in using mouse or touch events any more – we can switch to pointer events.

1. Scrolling

We can’t prevent scrolling by using event.preventDefault() in onscroll listener, because it triggers after the scroll has already happened.

But we can prevent scrolling by event.preventDefault() on an event that causes the scroll, for instance keydown event for pageUp and pageDown.

1. **Case matters: "KeyZ", not "keyZ"**

To reliably track layout-dependent characters, event.key may be a better way.

On the other hand, event.code has the benefit of staying always the same, bound to the physical key location, even if the visitor changes languages. So hotkeys that rely on it work well even in case of a language switch.

1. Preload image in js

function isVisible(elem) {

let coords = elem.getBoundingClientRect();

let windowHeight = document.documentElement.clientHeight; // top elem edge is visible?

let topVisible = coords.top > 0 && coords.top < windowHeight; // bottom elem edge is visible?

let bottomVisible = coords.bottom < windowHeight && coords.bottom > 0;

return topVisible || bottomVisible;

}

function showVisible() {

for (let img of document.querySelectorAll('img')) {

let realSrc = img.dataset.src;

if (!realSrc) continue;

if (isVisible(img)) {

img.src = realSrc; img.dataset.src = '';

}

}

}

showVisible(); w

indow.onscroll = showVisible;

1. Stale closure

The stale closure problem occurs when a closure captures outdated variables. An efficient way to solve stale closures is to correctly set the dependencies of React hooks. Or, in case of stale state, use a functional way to update the state.

1. Oninput

The input event occurs after the value is modified.

So we can’t use event.preventDefault() there – it’s just too late, there would be no effect.

1. Cut,copy, paste

We also can use event.preventDefault() to abort the action, then nothing gets copied/pasted/cut.

1. Submit

The handler can check the data, and if there are errors, show them and call event.preventDefault(), then the form won’t be sent to the server.

1. DOMContentLoaded

External style sheets don’t affect DOM, so DOMContentLoaded does not wait for them.

But there’s a pitfall. If we have a script after the style, then that script must wait until the stylesheet loads:

<link type="text/css" rel="stylesheet" href="style.css">

<script> // the script doesn't not execute until the stylesheet is loaded alert(getComputedStyle(document.body).marginTop);

</script>

1. Built-in browser autofill

Firefox, Chrome and Opera autofill forms on DOMContentLoaded.

For instance, if the page has a form with login and password, and the browser remembered the values, then on DOMContentLoaded it may try to autofill them (if approved by the user).

So if DOMContentLoaded is postponed by long-loading scripts, then autofill also awaits. You probably saw that on some sites (if you use browser autofill) – the login/password fields don’t get autofilled immediately, but there’s a delay till the page fully loads. That’s actually the delay until the DOMContentLoaded event.

1. Script issue
2. Scripts can’t see DOM elements below them, so they can’t add handlers etc.
3. If there’s a bulky script at the top of the page, it “blocks the page”. Users can’t see the page content till it downloads and runs:
4. Defer

The defer attribute tells the browser that it should go on working with the page, and load the script “in background”, then run the script when it loads.

* Scripts with defer never block the page.
* Scripts with defer always execute when the DOM is ready, but before DOMContentLoaded event.

The defer attribute is ignored if the <script> tag has no src.

1. Async

* The page doesn’t wait for async scripts, the contents are processed and displayed.
* DOMContentLoaded and async scripts don’t wait for each other

**Dynamic scripts behave as “async” by default.**

1. **CORS (cross-origin policy)**

**To allow cross-origin access, the**<script>**tag needs to have the**crossorigin**attribute, plus the remote server must provide special headers.**

There are three levels of cross-origin access:

1. **No**crossorigin**attribute** – access prohibited.
2. crossorigin="anonymous" – access allowed if the server responds with the header Access-Control-Allow-Origin with \* or our origin. Browser does not send authorization information and cookies to remote server.
3. crossorigin="use-credentials" – access allowed if the server sends back the header Access-Control-Allow-Origin with our origin and Access-Control-Allow-Credentials: true. Browser sends authorization information and cookies to remote server.
4. **Make unselectable:**
5. Use CSS property user-select: none. (elem cannot be copied)
6. Prevent default action in onselectstart or mousedown events. (elem still be copied)

<div>Selectable <div id="elem">Unselectable</div> Selectable</div> <script> elem.onselectstart = () => false; </script>

1. removeAllRanges()

If the selection already exists, empty it first with removeAllRanges(). And then add ranges. Otherwise, all browsers except Firefox ignore new ranges.

1. 4ms

here’s the in-browser minimal delay of 4ms for many nested setTimeout calls. Even if we set 0, it’s 4ms (or a bit more).

1. Macrotasks and microtasks

setTimeout(() => alert("timeout")); Promise.resolve() .then(() => alert("promise")); alert("code");

1. code shows first, because it’s a regular synchronous call.
2. promise shows second, because .then passes through the microtask queue, and runs after the current code.
3. timeout shows last, because it’s a macrotask.
4. Firefox vs chrome

The difference is that Firefox treats a timeout of 2000ms or less are acceptable, but after it – removes the “trust”, assuming that now it’s “outside of the user action”. chrome is not.

1. Iframe

When an iframe comes from the same origin, and we may access its document, there’s a pitfall. It’s not related to cross-origin things, but important to know.

1. IE support strings

 IE supports only strings, so we should JSON.stringify complex objects to support that browser.

1. Same origin

“same origin” if they have the same protocol, domain and port.

http://site.com/ and **https://**site.com, they are not same origin

http://site.com/ and http://site.com:**8080,** they are not same origin

* http://site.com
* http://site.com/
* http://site.com/my/page.html

They are same

1. The clickjacking

Here’s how clickjacking was done with Facebook:

1. A visitor is lured to the evil page. It doesn’t matter how.
2. The page has a harmless-looking link on it (like “get rich now” or “click here, very funny”).
3. Over that link the evil page positions a transparent <iframe> with src from facebook.com, in such a way that the “Like” button is right above that link. Usually that’s done with z-index.
4. In attempting to click the link, the visitor in fact clicks the button.

How to fix:

* It is recommended to use X-Frame-Options: SAMEORIGIN on pages (or whole websites) which are not intended to be viewed inside frames.
* Use a covering <div> if we want to allow our pages to be shown in iframes, but still stay safe.

1. ArrayBuffer

ArrayBuffer has nothing in common with Array:

* It has a fixed length, we can’t increase or decrease it.
* It takes exactly that much space in the memory.
* To access individual bytes, another “view” object is needed, not buffer[index].

1. createObjectURL vs readAsDataURL

Both ways of making an URL of a Blob are usable. But usually URL.createObjectURL(blob) is simpler and faster.

**URL.createObjectURL(blob)**

* We need to revoke them if care about memory.
* Direct access to blob, no “encoding/decoding”

**Blob to data url**

* No need to revoke anything.
* Performance and memory losses on big Blob objects for encoding.

1. Fetch

A typical fetch request consists of two await calls:

let response = await fetch(url, options); // resolves with response headers

let result = await response.json(); // read body as json

1. Append and set in formData

The set method removes fields with the same name, append doesn’t. That’s the only difference between them.

1. abortController

Use built-in object AbortController to cancel ongoing fetch and other asynchronous tasks.

1. CORS for simple requests

The server can inspect the Origin and, if it agrees to accept such a request, adds a special header Access-Control-Allow-Origin to the response. That header should contain the allowed origin (in our case https://javascript.info), or a star \*. Then the response is successful, otherwise an error.

1. Search and hash in URL

* search – a string of parameters, starts with the question mark ?
* hash starts with the hash character #

1. difference between encodeURIComponent and encodeURI

* encodeURI encodes only characters that are totally forbidden in URL.
* encodeURIComponent encodes same characters, and, in addition to them, characters #, $, &, +, ,, /, :, ;, =, ? and @.

for a whole URL we can use encodeURI,While for URL parameters we should use encodeURIComponent instead:

1. XMLHttpRequest

In modern web-development XMLHttpRequest is used for three reasons:

1. Historical reasons: we need to support existing scripts with XMLHttpRequest.
2. We need to support old browsers, and don’t want polyfills (e.g. to keep scripts tiny).
3. We need something that fetch can’t do yet, e.g. to track upload progress.

Otherwise using fetch.

1. Resumable file upload
2. First, create a file id, to uniquely identify the file we’re going to upload
3. Send a request to the server, asking how many bytes it already has
4. use Blob method slice to send the file from startByte

https://javascript.info/resume-upload

1. WebSocket vs EventSource

WebSocket objects are cross-origin by nature.

| **WebSocket** | **EventSource** |
| --- | --- |
| Bi-directional: both client and server can exchange messages | One-directional: only server sends data |
| Binary and text data | Only text |
| WebSocket protocol | Regular HTTP |

1. Long polling

Long polling is the simplest way of having persistent connection with server, that doesn’t use any specific protocol like WebSocket or Server Side Events.

1. Cookies

Cookies are small strings of data that are stored directly in the browser.

1. LocalStorage vs sessionStorage

| **localStorage** | **sessionStorage** |
| --- | --- |
| Shared between all tabs and windows with the same origin | Visible within a browser tab, including iframes from the same origin |
| Survives browser restart | Survives page refresh (but not tab close) |

The sessionStorage exists only within the current browser tab.

1. Cross-Site Request Forgery (XSRF) attack

you are logged into the site bank.com. That is: you have an authentication cookie from that site. Your browser sends it to bank.com with every request, so that it recognizes you and performs all sensitive financial operations.

Now, while browsing the web in another window, you accidentally come to another site evil.com. That site has JavaScript code that submits a form <form action="https://bank.com/pay"> to bank.com with fields that initiate a transaction to the hacker’s account.

The browser sends cookies every time you visit the site bank.com, even if the form was submitted from evil.com. So the bank recognizes you and actually performs the payment.

samesite forbids the browser to send the cookie with requests coming from outside the site, helps to prevent XSRF attacks.

1. IndexDB

It’s a simple key-value database, powerful enough for offline apps.

Difference with other database:

Unlike server-side databases, IndexedDB is client-side, the data is stored in the browser, so we, developers, don’t have “any time” access to it. An object store is a core concept of IndexedDB. Counterparts in other databases are called “tables” or “collections”. It’s where the data is stored

1. Css animation vs JavaScript animation

Limitations of CSS animations compared to JavaScript animations:

**Merits**

* Simple things done simply.
* Fast and lightweight for CPU.

**Demerits**

* JavaScript animations are flexible. They can implement any animation logic, like an “explosion” of an element.
* Not just property changes. We can create new elements in JavaScript for purposes of animation.

1. Make animation smooth

For instance, changing style.left from 0px to 100px moves the element. If we increase it in setInterval, changing by 2px with a tiny delay, like 50 times per second, then it looks smooth. That’s the same principle as in the cinema: 24 frames per second is enough to make it look smooth.

The pseudo-code can look like this:

let timer = setInterval(function() {

if (animation complete) clearInterval(timer);

else increase style.left by 2px }, 20);

// change by 2px every 20ms, about 50 frames per second

1. Custom element name

Custom element name must have a hyphen -, e.g. my-element and super-button are valid names, but myelement is not.

1. Customized built in elements

Requires one more .define argument, and is="..." in HTML

<script>

// The button that says "hello" on click

class HelloButton extends HTMLButtonElement {

constructor() {

super(); this.addEventListener('click', () => alert("Hello!"));

}

}

customElements.define('hello-button', HelloButton, {extends: 'button'}); </script>

<button is="hello-button">Click me</button>

1. Iterable

map.keys(), .values(),.entries()

set.keys(), .values(),.entries()

String,

Array

vs. IEnumerable in C#

Stack, List, Queue, Dictionary, Set, Array implements IEnumerable<T> can call any LINQ method.

1. Two ways to create a regular expression
2. regexp = new RegExp("pattern", "flags");
3. regexp = /pattern/gmi; // with flags g,m and I

The main difference between these two syntaxes is that pattern using slashes /.../ does not allow for expressions to be inserted (like string template literals with ${...}). They are fully static.

While new RegExp, is more often used when we need to create a regexp “on the fly” from a dynamically generated string.

1. Access object properties
2. Dot property accessor: object.property
3. Square brackets property access: object['property']
4. Object destructuring: const { property } = object
5. Unicode

With Unicode properties we can look for words in given languages(e.g. Chinese), special characters (quotes, currencies). Flag u and class \p{… } enable the support of Unicode in regular expressions.

1. String start ^ and end $

^...$ are often used to test whether or not a string fully matches the pattern.

1. Word boundary : \b

alert( "Hello, Java!".match(/\bJava\b/) ); // Java

alert( "Hello, JavaScript!".match(/\bJava\b/) ); // null

alert( "1 23 456 78".match(/\b\d\d\b/g) ); // 23,78

1. Call back in jQuery

A callback function is executed after the current effect is finished.

1. Escaping

Use backslash \ on special characters [ \ ^ $ . | ? \* + ( )

1. Double backslash in new RexExp()

we need double backslashes \\, cause string quotes consume one of them.

1. Ranges

* **\d** – is the same as [0-9],
* **\w** – is the same as [a-zA-Z0-9\_],
* **\s** – is the same as [\t\n\v\f\r ], plus few other rare unicode space characters.

1. Multi-language

let regexp = /[\p{Alpha}\p{M}\p{Nd}\p{Pc}\p{Join\_C}]/gu;

Which looks for wordly characters in any language.

1. Test vs. Match

*regexObject*.**[test](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/test)**( *String* )

*string*.**[match](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match)**( *RegExp* )

Use .test if you want a faster boolean check. Use .match to retrieve all matches when using the g global flag.

1. Lazy quantifiers

usually a question mark ? is a quantifier by itself (zero or one), but if added after another quantifier (or even itself) it gets another meaning – it switches the matching mode from greedy to lazy.

1. Check email

let regexp = /[-.\w]+@([\w-]+\.)+[\w-]+/g

Same as /[-\.\w]+@([\w-]+\.)+[\w-]+/g

1. Domain check

let regexp = ([\w-]+\.)+\w+/g;

1. year-month-day

let dateRegexp = /(?<year>[0-9]{4})-(?<month>[0-9]{2})-(?<day>[0-9]{2})/;

1. Quotes

let regexp = /(?<quote>['"])(.\*?)\k<quote>/g;

// let str = `He said: "She's the one!".`;

// alert( str.match(regexp) );

// "She's the one!"

1. Regexp for time

let regexp = /([01]\d|2[0-3]):[0-5]\d/g;

1. Lookahead and lookbehind

| **Pattern** | **type** | **matches** |
| --- | --- | --- |
| X(?=Y) | Positive lookahead | X if followed by Y |
| X(?!Y) | Negative lookahead | X if not followed by Y |
| (?<=Y)X | Positive lookbehind | X if after Y |
| (?<!Y)X | Negative lookbehind | X if not after Y |

1. Extra parenthesis in capturing groups

let regexp = /\d+(?=(€|kr))/; // extra parentheses around €|kr

1. atomic capturing groups

Use lookahead to rescue;

let regexp = /^(\w+\s?)\*$/

=>

let regexp = /^((?=(?<word>\w+))\k<word>\s?)\*$/;

// parentheses are named ?<word>, referenced as \k<word>

1. Flag y

The flag y allows to perform the search at the given position in the source string.

**The flag**y**makes**regexp.exec**to search exactly at position**lastIndex**, not “starting from” it.**

let str = 'let varName = "value"';

let regexp = /\w+/y;

regexp.lastIndex = 3;

alert( regexp.exec(str) ); // null (there's a space at position 3, not a word) regexp.lastIndex = 4;

alert( regexp.exec(str) ); // varName (word at position 4)

1. Match

If we want the result to be an array, we can write like this:

let result = str.match(regexp) || [];

1. Match vs. matchAll

There are 3 differences from match:

1. It returns an iterable object with matches instead of an array. We can make a regular array from it using Array.from.
2. Every match is returned as an array with capturing groups (the same format as str.match without flag g).
3. If there are no results, it returns not null, but an empty iterable object.
4. Split

alert('12, 34, 56'.split(/,\s\*/)) // array of [12, 34, 56]

1. Search

**The important limitation:**search**only finds the first match.**

Stock location System need to download:

* visual studio community 2019
* visual studio code (code.visualstudio.com, user installer)
* ssms (click link **Download SQL Server Management Studio** on the ssms page)
* node-v…-x64.msi (this already includes npm)

Stock Location system code:

Just copy projects from VS and VS Code, respectively (Including all packages)

Stock Location system database:

* right click the database (StockLocation) and select tasks/ backup
* make sure that the backup type is full
* click add and specify the location and backup name
* copy the created backup file to another computer
* in ssms, right click the sql server instance and select Restore database
* select device and click the ellipsis button to navigate to the copied backup file

When backup database show error:



Fix:

Just remove the existing .bak file and re-run.