MISSION READY

DARE TO DEVELOP

ES6, Regex, jQuery

Ewan Zhang

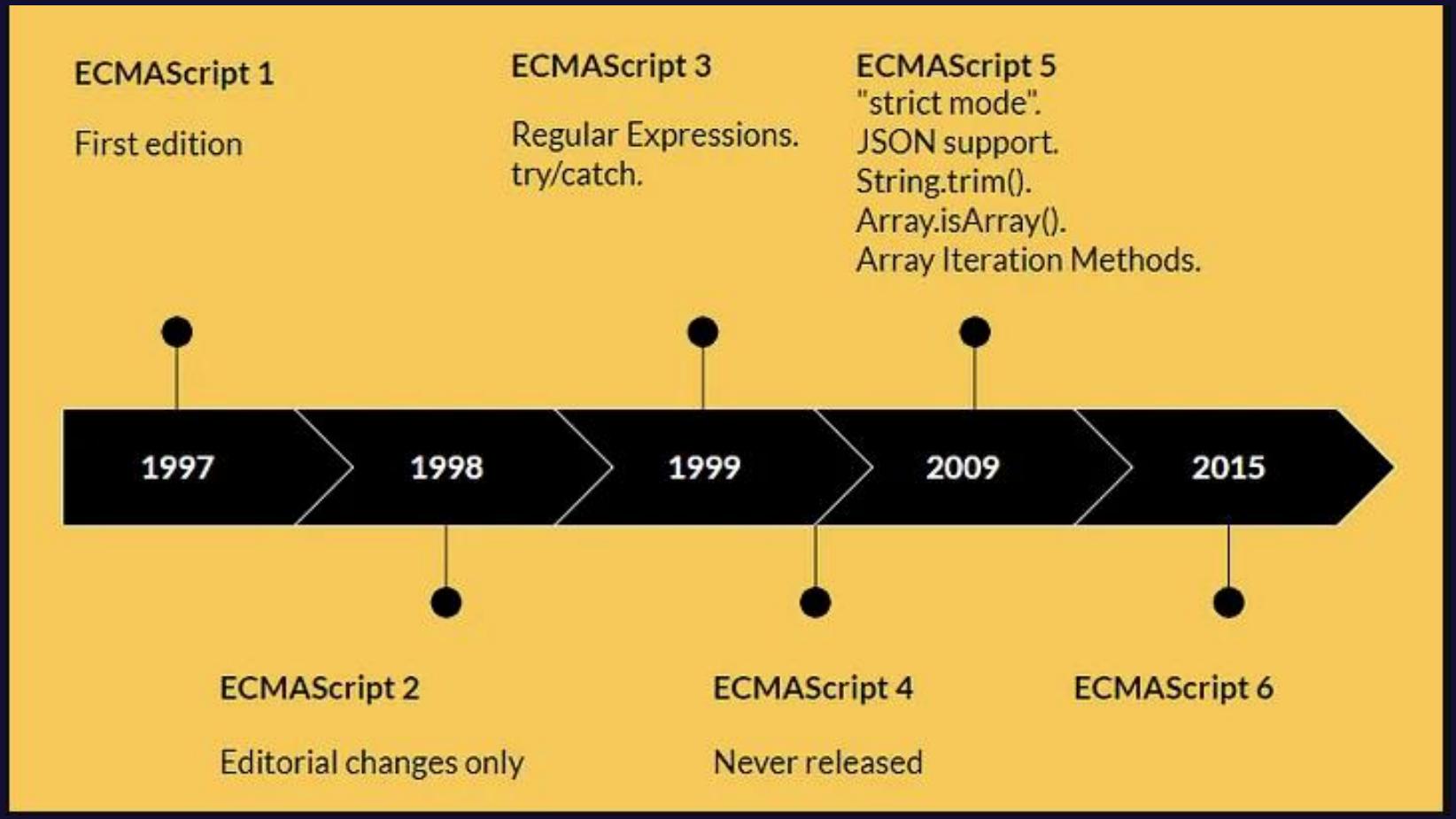
JavaScript Versions

Evolution of JavaScript

JavaScript Versions

- JavaScript was invented by Brendan Eich (1995), and in 1997 became an ECMA standard.
  - ECMA stands for European Computer Manufacturer's Association.
- ECMA is the standards organization responsible for developing and maintaining the ECMAScript specification.
- ***ECMAScript*** is a ***standard*** that web browsers follow while interpreting JavaScript.
- ***ECMAScript (ES) === JavaScript* versions**

https://stackoverflow.com/a/30113184/5544970


ECMAScript 1
First edition
ECMAScript 3
Regular Expressions.
try/catch.
ECMAScript 5
"strict mode".
JSON support.
String.trim().
Array.isArray().
Array Iteration Methods.
1997
1998
1999
2009
2015
ECMAScript 2
Editorial changes only
ECMAScript 4
Never released
ECMAScript 6

ECMAScript update

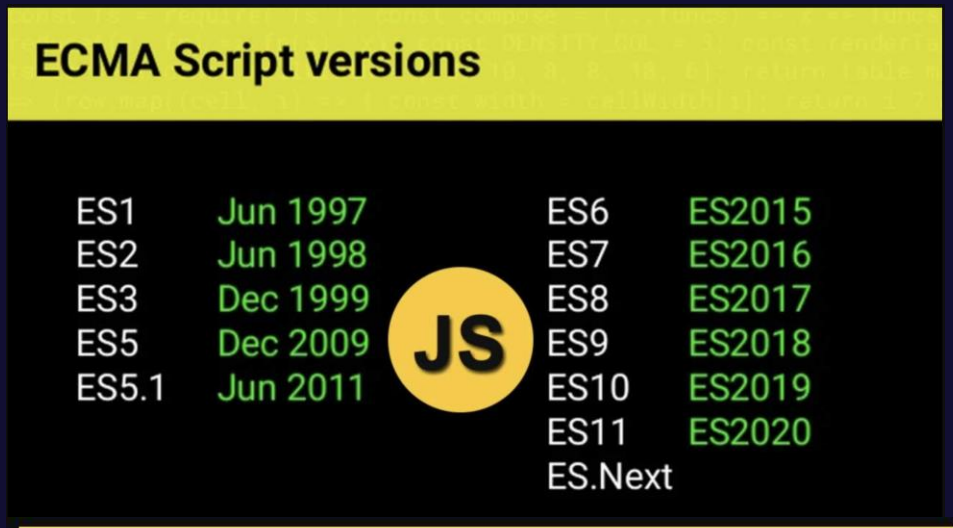
- **Language Improvement/Evolution**
  These updates help make programming languages more powerful, efficient, and developer-friendly.
- **Compatibility**
  This will allow developers to write code that works consistently across different browsers, devices, and environments.
- **Security Enhancements**
  As new threats and attack vectors emerge, it becomes necessary to update programming languages to provide stronger security measures.
- **Performance Optimization**
  This can lead to improved execution speed, reduced memory usage, and overall better performance of JavaScript applications.

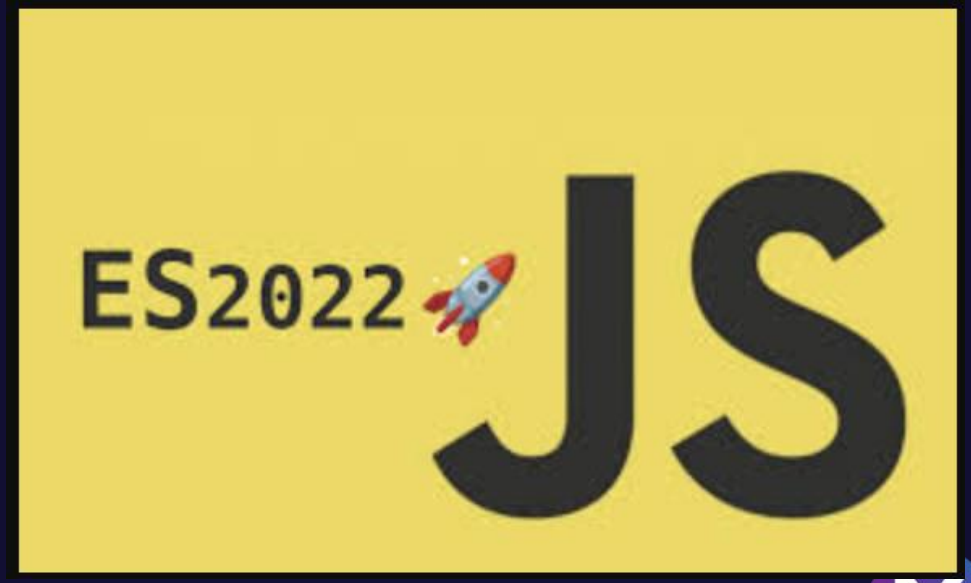
Which version are we on currently?

ES2022 (ES13)

- In June 2022, The ECMA International approved the latest version of the official specification ES13 aka ECMAScript 2022.

- Example for new feature in ES13:
  at() function for Indexing.





```
const fruits = ['apple', 'banana', 'cherry'];

console.log(fruits.at(0));   // 'apple'
console.log(fruits.at(2));   // 'cherry'
console.log(fruits.at(-1));  // 'cherry'
console.log(fruits.at(5));   // undefined
```

at() Method is not available before ES2022.

You can use at() method in String && Array

JavaScript built-in: Array: at

Usage % of all users ?

Global 90.82%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE *	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	B
4-91	12-91	3.1-15.3	2-89	10-77			3.2-15.3	4-15.0								
92-112	92-112	15.4-16.4	90-112	78-97	6-10		15.4-16.4	16.0-19.0		12-12.1		2.1-4.4.4				
113	113	16.5	113	98	11	113	16.5	20	all	73	13.4	113	113	13.1	13.18	
114-116		16.6-TP	114-115													

Notes | Test on a real browser | Feedback

See full reference on MDN Web Docs.

Support data for this feature provided by:

MDN browser-compat-data

https://caniuse.com/mdn-javascript_builtins_array_at

ES6 features

Arrow functions and more

ES6 (2015)

- ES6 is all about reducing amount of code you need to write
- You have seen some ES6 features already
  - let and const variables
  - For.. of

```
//loop thru an array to display all items
const arr1 = [1, 2, 3];
for (let item1 of arr1) {
    console.log(item1);
}
```

Arrow Functions

- A compact alternative to a traditional function, with a few limitations.

- Very convenient for simple one-line actions, when we're just too lazy to write many words ☺

```
() => { }
```

```
// Traditional Function
function doubleNum (num) {
  return num * 2;
}
```

```
// Arrow Function
let doubleNum = (num) => {
   return num * 2;
}
```

Single params do not require parentheses

```
// Traditional Function
function doubleNum (num) {
  return num * 2;
}

// Arrow Function
let doubleNum = num => {
   return num * 2;
}
```

```
// Traditional Function
function sum (first, second) {
  return first + second;
}

// Arrow Function
let sum = (first, second) => {
  return first + second;
};
```

Single line statements do not require the body braces **{}** and **return**

```
// Traditional Function
function double (num) {
  return num * 2;
}
```

```
// Arrow Function
let double = num => num * 2;
```

```
// Traditional Function
function double(num) {
  console.log(num);
  return num * 2;
}
```

```
// Arrow Function
let double = (num) => {
  console.log(num);
  return num * 2;
};
```

Arrow Function Exercise

1. Write an arrow function that takes in two parameters and divides the first with the second and returns the result.
   - Example, `console.log(divideNum(6, 2)); //should print out 3`

```
const divideNum = (num1, num2) => num1 / num2;
```

2. Write an arrow function that checks if a number is even and returns true if it is. [Hint: use the % operator]
   - Example,

```
console.log(isEven(6)); //should print true
console.log(isEven(3)); //should print false
```

```
const isEven = num1 => num1 % 2 === 0;
```

Default Parameters

- ES6 allows named parameters to be initialised with default values if no value or undefined is passed.

```
function multiply(a, b) {
  return a * b;
}

multiply(5, 2); // 10
multiply(5); // NaN - ERROR
```

WITH DEFAULT PARAMETERS

```
function multiply(a, b = 1) {
  return a * b;
}

multiply(5, 2); // 10
multiply(5); // 5
```

Default Parameters Exercise

- Add a default value of 1 to the second parameter in Exercise 1, so that: `console.log(divideNum(3)); //should print out 3`

```
const divideNum = (num1, num2 = 1) => {
        return num1 / num2;
    };
console.log(divideNum(6, 3));
console.log(divideNum(10)); // Should print the same number
```

```
const divideNum = (num1, num2 = 1) => num1 / num2;
```

Template Literals

- Instead of using + to concatenate String variables
  - write the full string inside backticks `
  - use variables directly inside the backticks using ${variableName}
  - You can have multi-line Strings inside of backticks

```
let word1 = 'Hello';
let word2 = 'world';

let combined = word1 + ' ' + word2;

console.log(combined);
```

```
let word1 = 'Hello';
let word2 = 'world';

let combined = `${word1} ${word2}`;

console.log(combined);
```

Spread Operator

- A way to create a copy of an array
  - It doesn't just pass the reference

```
let arr1 = [1,2,3,4,5];
let arr2 = [0,...arr1,6,7,8,9];

console.log(arr2); //[0,1,2,3,4,5,6,7,8,9]
console.log(arr1); //[1,2,3,4,5]
```

Spread Operator Exercise

- Combine two array below together by using Spread Operator
- Store the new array in a variable.
- Console this new array and get the result below

```
let arr1 = [1, 2, 3];
let arr2 = ["A", "B", "C"];

//[1,2,3,"A","B","C"]
```

JavaScript shorthands

```
if (x > 10) {
    answer = "greater than 10";
} else {
    answer =  "less than 10";
}
```

```
const answer = x > 10 ? "greater than 10" : "less than 10";
```

JavaScript shorthands

```
if (likeJavaScript === true)
```

```
if (likeJavaScript != true)
```

```
if (likeJavaScript)
```

```
if (!likeJavaScript)
```

JavaScript shorthands

We can use this shorthand to check if some value exist

```
let userName;
let userInput= ''; //can be changed if user set up their name


if (userInput) {
   userName  =  userInput;
} else {
   userName  = 'please enter your name';
}
const  userName  =  userInput ||  'please enter your name';
```

JavaScript shorthands

```
const num1 = Number("100");
const num2 =  Number("100.01");
```

```
const num1 = +"100"; // converts from string to integer number
const num2 =  +"100.01"; // converts from string to decimal number
```

Regular Expressions

A smart way to match strings

Regular Expressions (Regex)

- There are times when you need to test whether a String matches a pattern. For example,
  - Mobile numbers needs to fit into 10 digits with the first digit being 0
  - Driver license IDs need to fit into 2 letters and 6 digits
  - Password must have at least 8 characters, at least one upper case letter, one lowercase letter and a number

Defining a Regex and Testing

- A regex can be defined using a RegExp object or a literal between 2 "/" (ES6)

```
let myRegex = new RegExp("a|b");
```

```
let  myRegex  = /a|b/;
```

- Testing using a regex (returns a Boolean. True if matched, false if not)

```
let answer1 =  myRegex.test("mission");//false
let answer2 =  myRegex.test("ready");  //true
```

String.match(regex)

```
const paragraph = 'The quick brown fox jumps over the lazy dog. It barked.';
const myRegex = /a|b/g;
const found = paragraph.match(myRegex);

console.log(found);
// Expected output: Array ['b', 'a', 'b', 'a']
```

1. It will return the matched results in an Array.

2. We usually add “g” as global at the end of regex.
   It tells JS that we want to return all the matched results in that string.

3. Without the “g”, it will return the first matched result only.

Regex bracket expressions

- [a-z] matches a character from lowercase a to lowercase z
- [A-G] matches a character from uppercase A to uppercase G
- [0-7] matches a character from 0 to 7
- [a-zA-Z0-9] matches any letter and digits (not symbols)

- Mobile Number: 10 digits starting with 0

```
/0[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]/
```

- Driver license: 2 letters and 6 digits

```
/[a-zA-Z][a-zA-Z][0-9][0-9][0-9][0-9][0-9][0-9]/
```

Regex character classes

- \w matches a character from a-z, A-Z, 0-9 and underscore _
- \d matches a character from 0 to 9
- \s matches a white space, line break or tab

- Mobile Number: 10 digits starting with 0

```
/0\d\d\d\d\d\d\d\d\d/
```

- Driver license: 2 letters and 6 digits

```
/[a-zA-Z][a-zA-Z]\d\d\d\d\d\d/
```

Regex Quantifier

- * matches previous item 0 or more times
- + matches previous item 1 or more times
- ? matches previous item 0 or 1 time
- {3} matches previous item exactly 3 times
- ^ matches the beginning of a String
- $ matches the end of a String

- Mobile Number: 10 digits starting with 0

```
/^0\d{9}$/
```

- Driver license: 2 letters and 6 digits

```
/^[a-zA-Z]{2}\d{6}$/
```

Creating a simple email regex

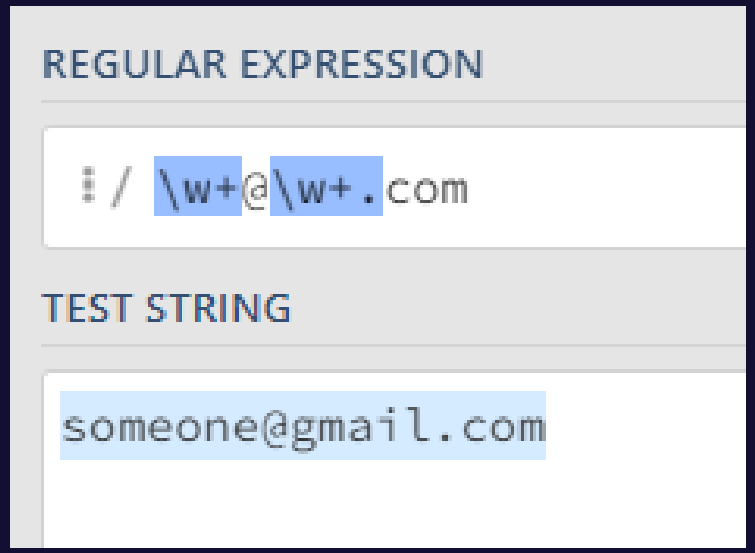
- Let's breakdown what an email contains

```
someone@gmail.com
```

Some words | @ symbol | Some more words | .com

Email regex

- Regex for word \w
- Could be multiple words (add a +)
- @ symbol
- Some more words \w+
- .com



A more proper implementation: https://regexr.com/3e48o

Exercise

- Create a regular expression that validates a password.
- The password must have at least 8 characters
- Bonus1: the password should start with an uppercase letter
- Bonus2: the password should end with number

https://regex101.com/


HOW TO REGEX
STEP 1: OPEN YOUR FAVORITE EDITOR
@GARABATOKID
STEP 2: LET YOUR CAT PLAY ON YOUR KEYBOARD
/^([A-Z0-9_\.-

More Regex patterns and Practices

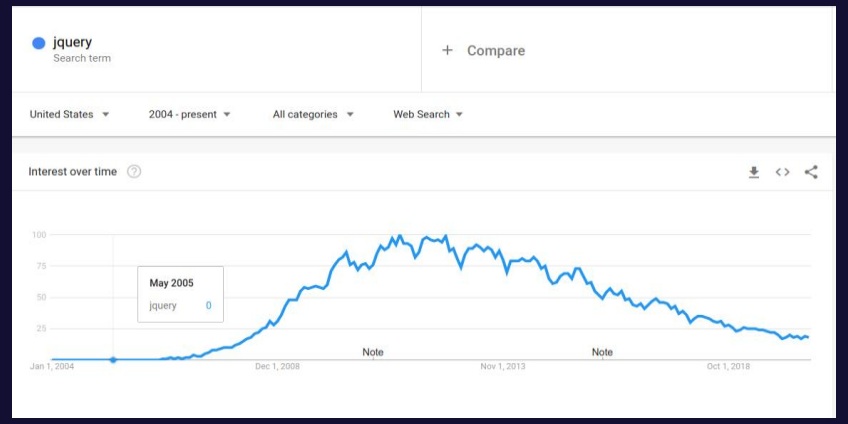
- Regex explanations
  - https://code.tutsplus.com/tutorials/a-simple-regex-cheat-sheet--cms-31278
- Regex practices / game
  - https://regexone.com/

jQuery

Historic HTML function shortcuts – Good to know

jQuery

- jQuery is a library that helps to simplify some JavaScript code
- Effectively make these tasks simpler:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX



- Access by download or link (recommended)

```
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
```

jQuery Syntax

- Access JQuery using $
- Select a HTML element and then perform Action on it
- Inside <script></script>
- Wrapped in $(document).ready(function() { }
- Syntax is **$(selector).action()**

```html
<!DOCTYPE html>
<html>

<head>
    <script src="https://ajax.googleapis.com/ajax/libs/
jquery/3.6.0/jquery.min.js"></script>
    <script>
        $(document).ready(function () {
            $("button").click(function () {
                $("p").hide();
            });
        });
    </script>
</head>

<body>

    <h2>Heading</h2>

    <p>This is paragraph 1.</p>
    <p>This is paragraph 2.</p>

    <button>Click me to hide paragraphs</button>

</body>

</html>
```

jQuery Selectors

- You can choose which HTML element by specifying the following (same convention as the selectors in CSS:
- $(“p”)
- $(“#id”)
- $(“.class”)
- $(“*”)
- More selectors https://www.w3schools.com/jquery/jquery_selectors.asp

```
$("button").click(function () {
    $("p").hide();
});
```

jQuery Methods

- text() (gives access to the text inside our element)
- click() (applies a click event listener)
- dblclick() (applies a double click event listener)
- mouseenter() (applies a mouseover event listener)
- mouseleave() (applies a mouseout event listener)
- hide() (hides the element)
- show() (unhides the element)
- toggle() (toggles between hidden and shown)
- animate() (sets some animation property on the element)
- css() (adds some css to our element)

```
$("button").click(function () {
    $("p").hide();
});
```

Use jQuery to change HTML

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

```html
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
    <script>
        //JQuery goes inside document.ready()
        $(document).ready(function () {

            //when click on button, alert the html content of id="test"
            $("button").click(function(){
                alert("Old HTML: " + $("#test").html());
            });

            //when click on button, change the content of id="test" to html.
            $("button").click(function(){
                $("#test").html("<b>Hello world!</b>");
            });
          }
    </script>
</head>

<body>
    <h2 id="test">This is Heading</h2>
    <button>Click me to set heading</button>
</body>
</html>
```

Use jQuery to change style

- css() – used to get or set style of an element
- To get a css property
  - css("propertyname");
- To change a css property
  - css("propertyname","value");
- To change many properties
  - css({"propertyname":"value","propertyname":"value",...});

- More about JQuery HTML/CSS
  - https://www.w3schools.com/jquery/jquery_ref_html.asp

```html
<html>

<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jque
ry/3.6.0/jquery.min.js"></script>
    <script>
        //JQuery goes inside document.ready()
        $(document).ready(function () {

            //onclick, alert the color of id="test"
            $("button").click(function(){
                alert("color: " + $("#test").css("color"));
            });

            //onclick, change css of id="test" to html
            $("button").click(function(){
                $("#test").css("color","red");
            });
        }
    </script>
</head>

<body>
    <h2 id="test">This is Heading</h2>
    <button>Click me to set heading</button>
</body>

</html>
```

More jQuery

- JQuery Cheatsheet
  - https://oscarotero.com/jquery/

- JQuery practices
  - Write document.getElementById(“display”).innerHTML = “Message” in Jquery

```
$("#display").text("Message")
```

  - Write document.getElementsByClassName("example“).style.background-color='blue' in JQuery

```
$(".example").css("background-color", "blue")
```

  - https://www.w3resource.com/jquery-exercises/part1/index.php

MISSION READY

www.missionreadyhq.com

DARE TO DEVELOP

Thank you

Ewan Zhang