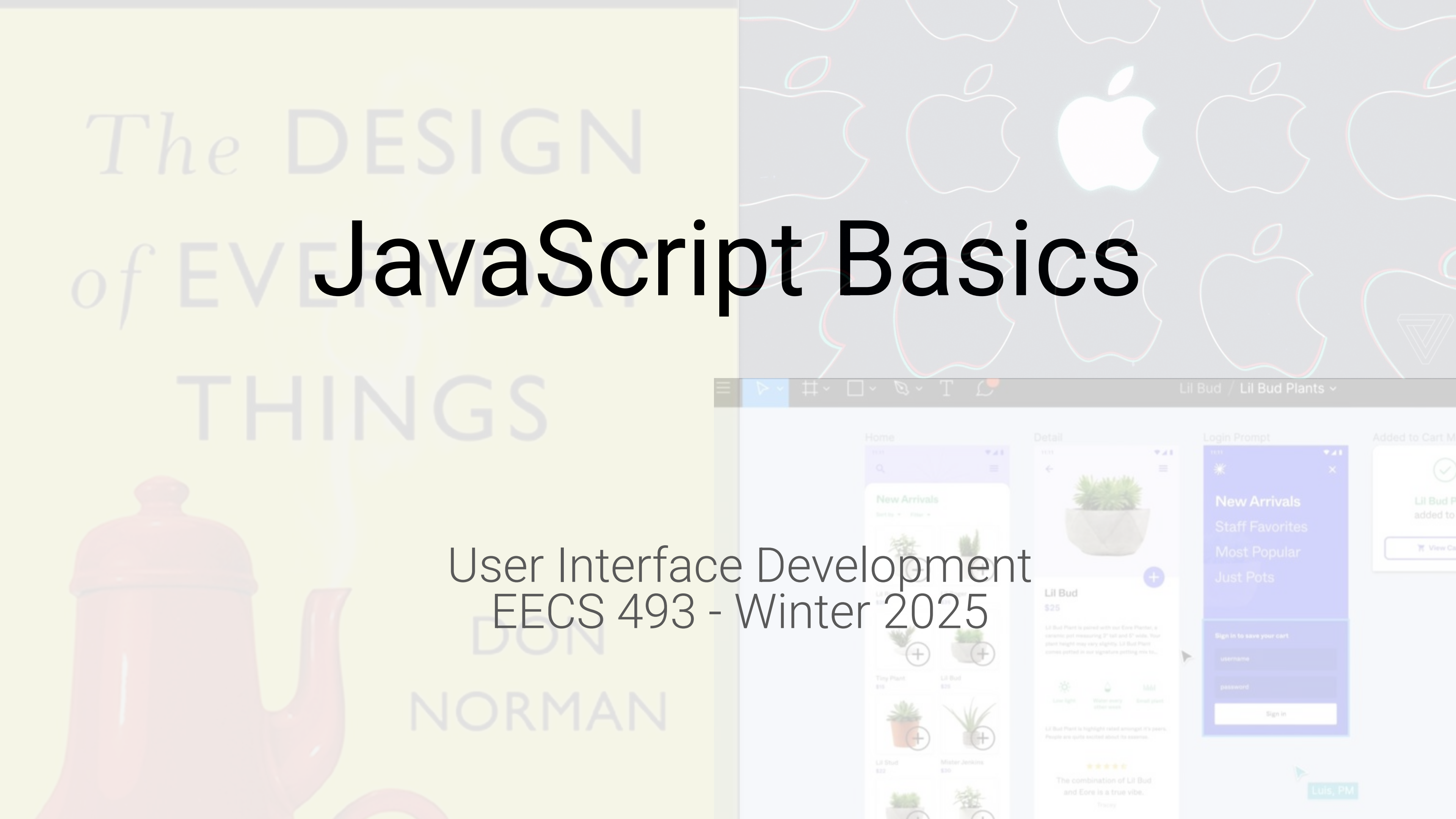
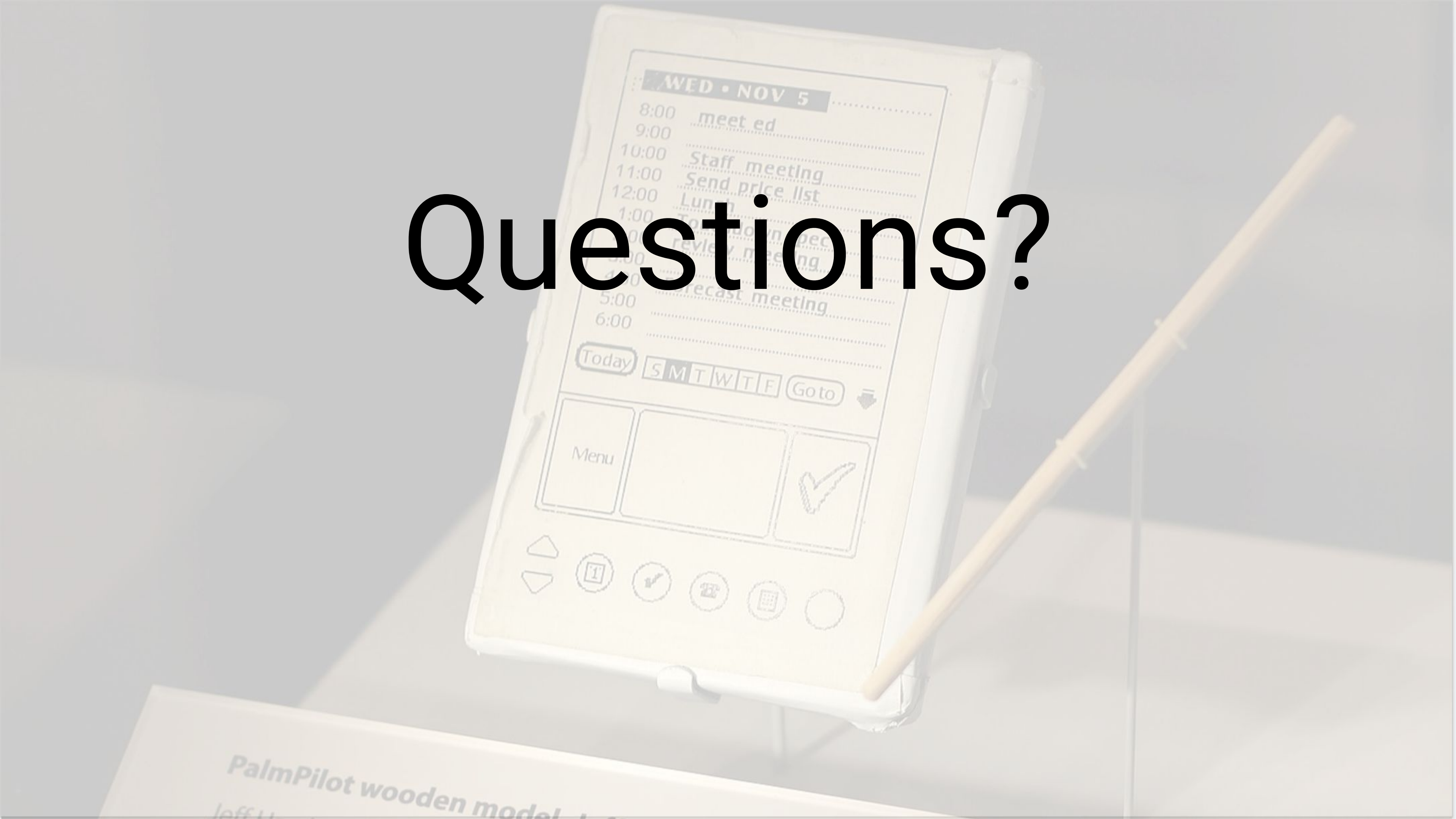
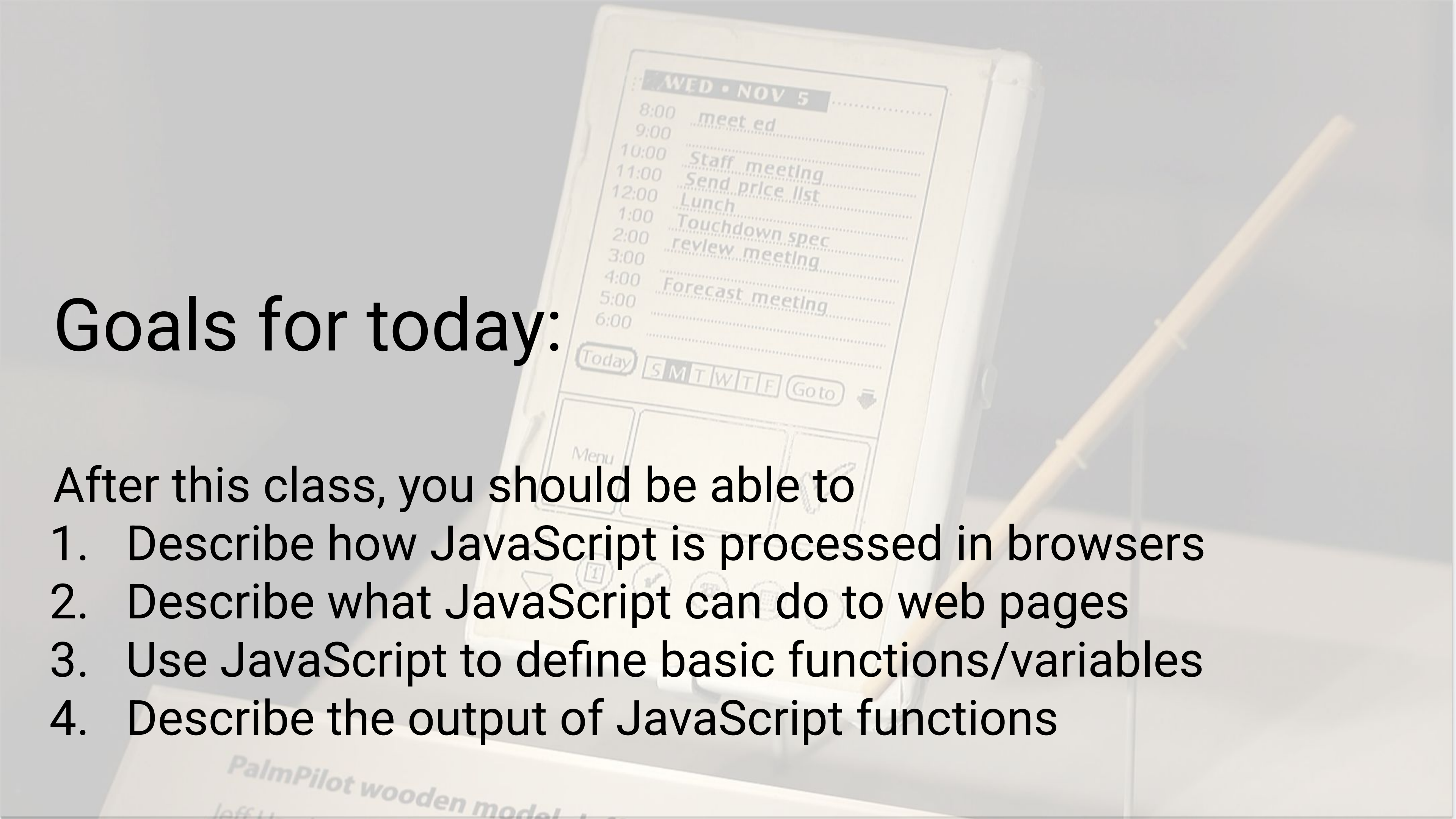
JavaScript Basics

User Interface Development
EECS 493 - Winter 2025



Questions?


WED • NOV 5
8:00 meet ed
9:00
10:00 Staff meeting
11:00 Send price list
12:00 Lunch
5:00
6:00
Today
S M T W T F
Go to
Menu
PalmPilot wooden model



Goals for today:

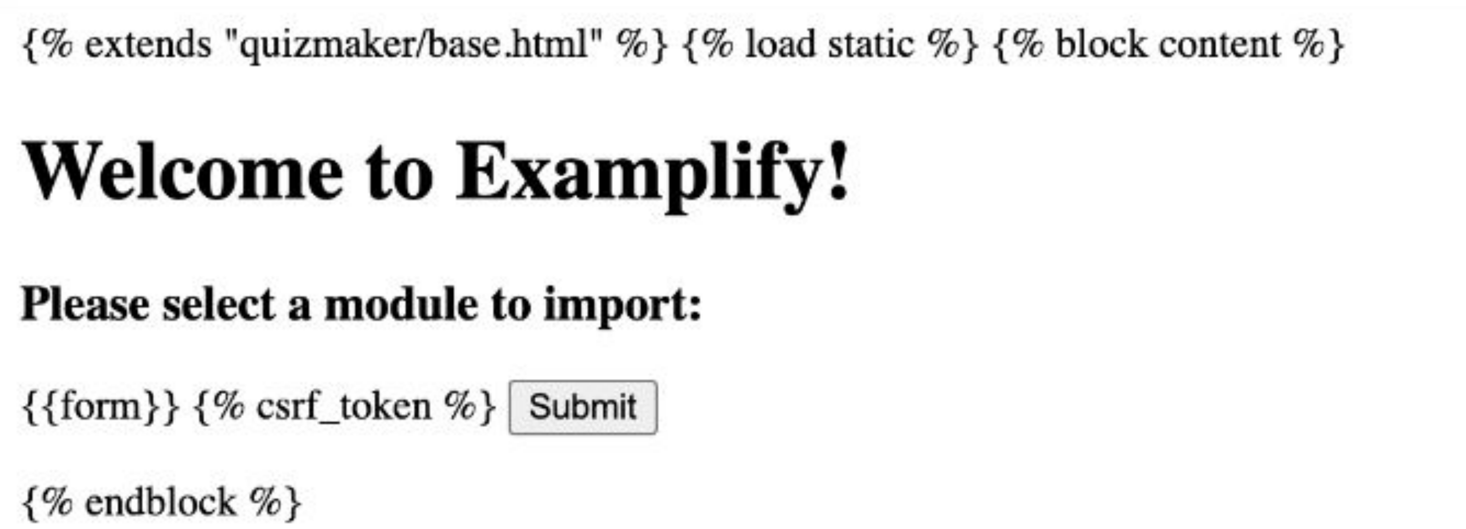
After this class, you should be able to

1. Describe how JavaScript is processed in browsers
2. Describe what JavaScript can do to web pages
3. Use JavaScript to define basic functions/variables
4. Describe the output of JavaScript functions

Why do we need JavaScript?

- HTML defines what is on our page (content)
- CSS defines how our content is laid out(layout)
- JavaScript defines how it all works (interactivity)

HTML only



{% extends "quizmaker/base.html" %} {% load static %} {% block content %}

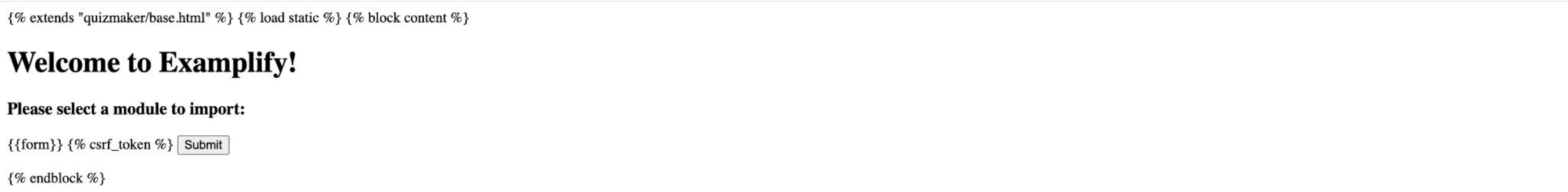
Welcome to Examplify!

Please select a module to import:

{{form}} {% csrf_token %}

{% endblock %}

HTML+ CSS



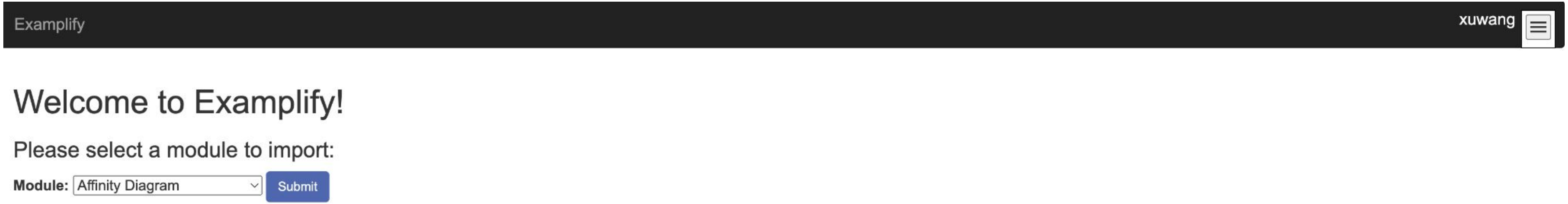
{% extends "quizmaker/base.html" %} {% load static %} {% block content %}

Welcome to Examplify!

Please select a module to import:

{{form}} {% csrf_token %} Submit

{% endblock %}

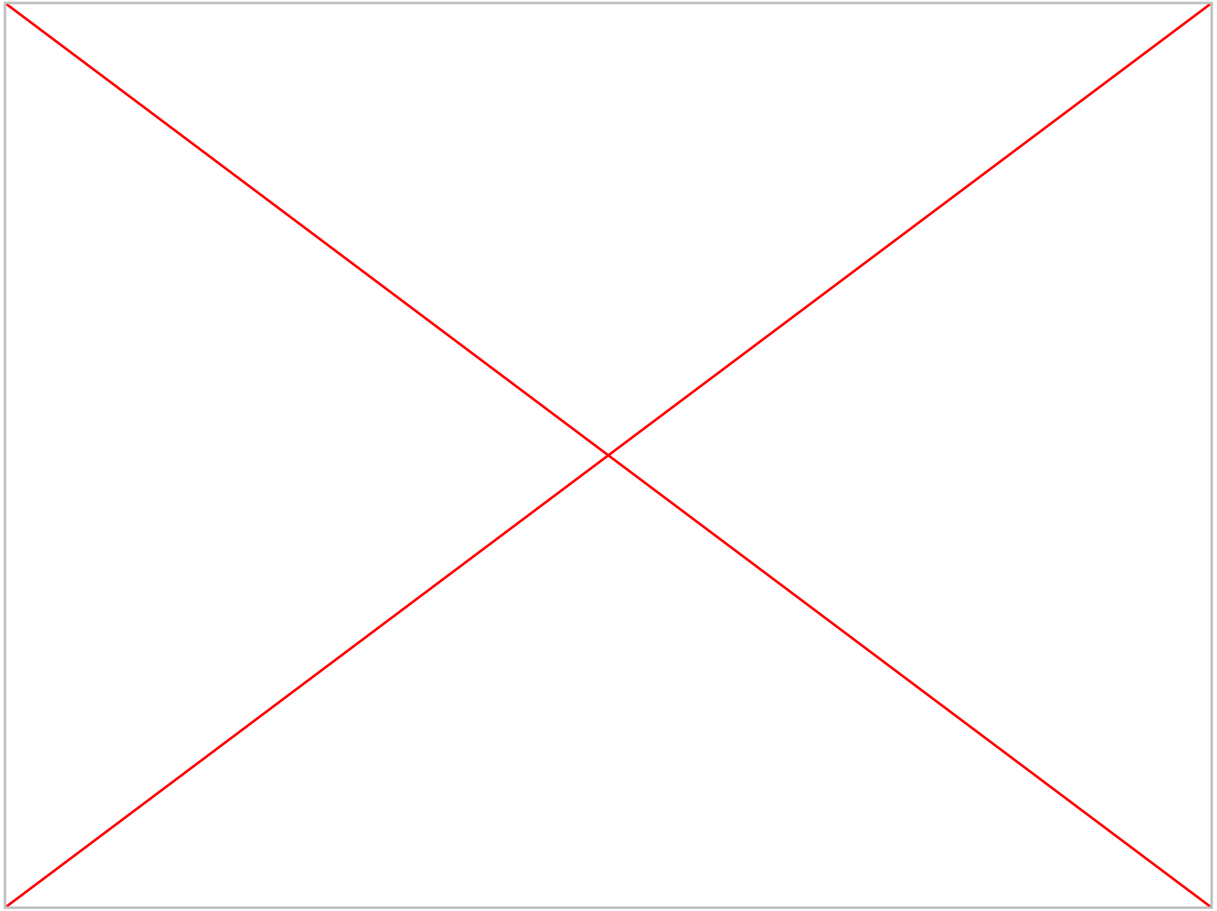


Welcome to Examplify!

Please select a module to import:

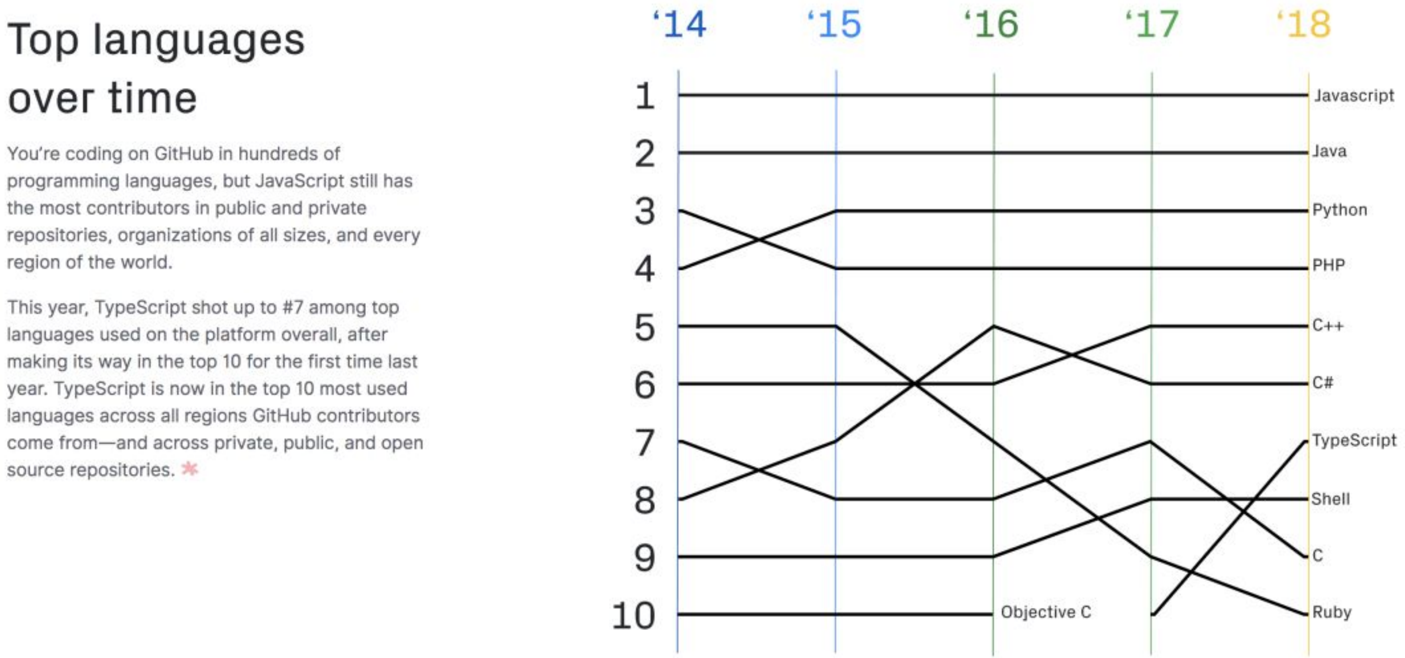
Module: Affinity Diagram Submit

HTML+ CSS + **JavaScript**



A bit of history of Javascript

Invented by Brendan Eich (who co-founded Mozilla) in 1995.

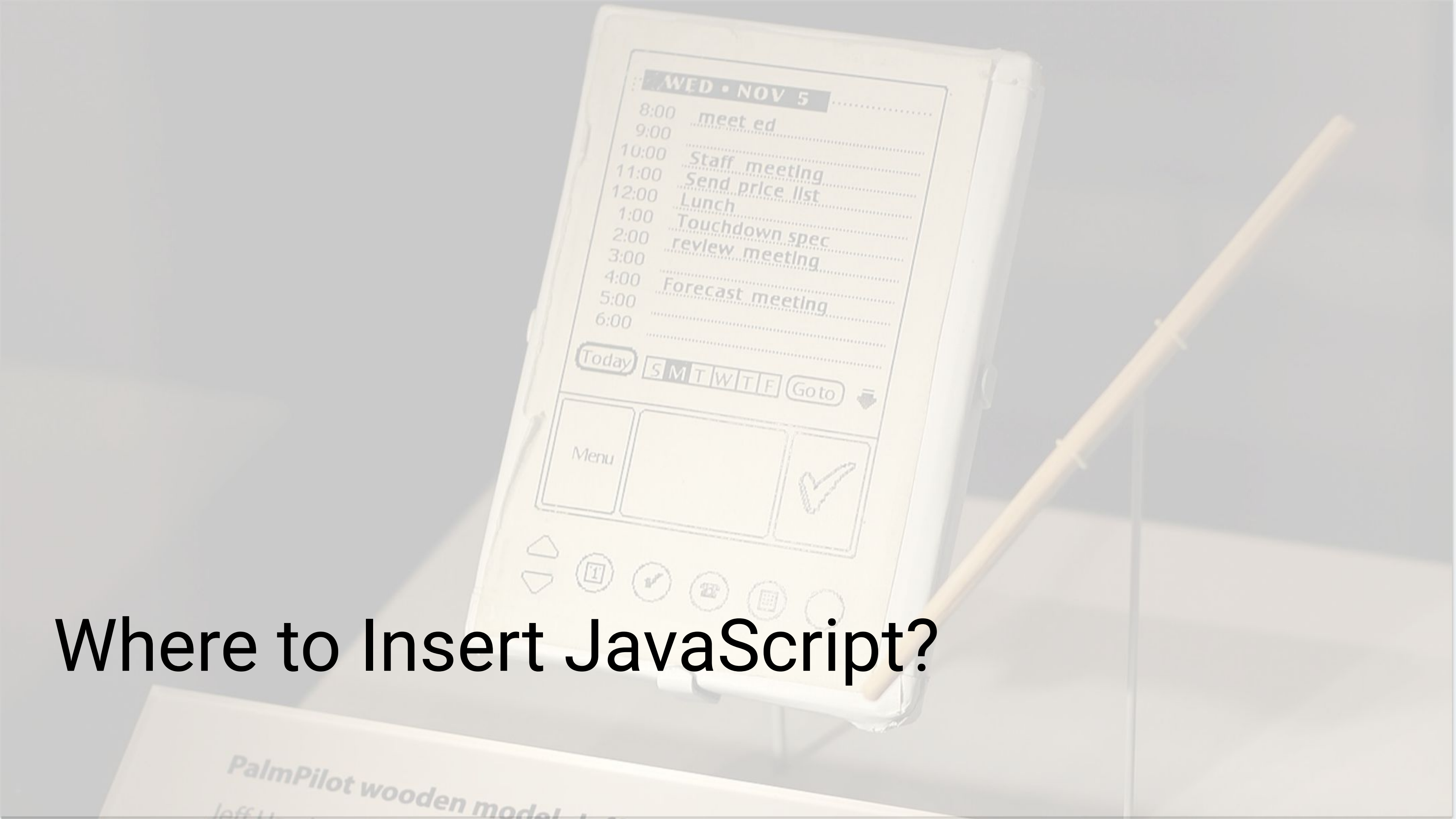


Top languages over time

You're coding on GitHub in hundreds of programming languages, but JavaScript still has the most contributors in public and private repositories, organizations of all sizes, and every region of the world.

This year, TypeScript shot up to #7 among top languages used on the platform overall, after making its way in the top 10 for the first time last year. TypeScript is now in the top 10 most used languages across all regions GitHub contributors come from—and across private, public, and open source repositories. *

Where to Insert JavaScript?

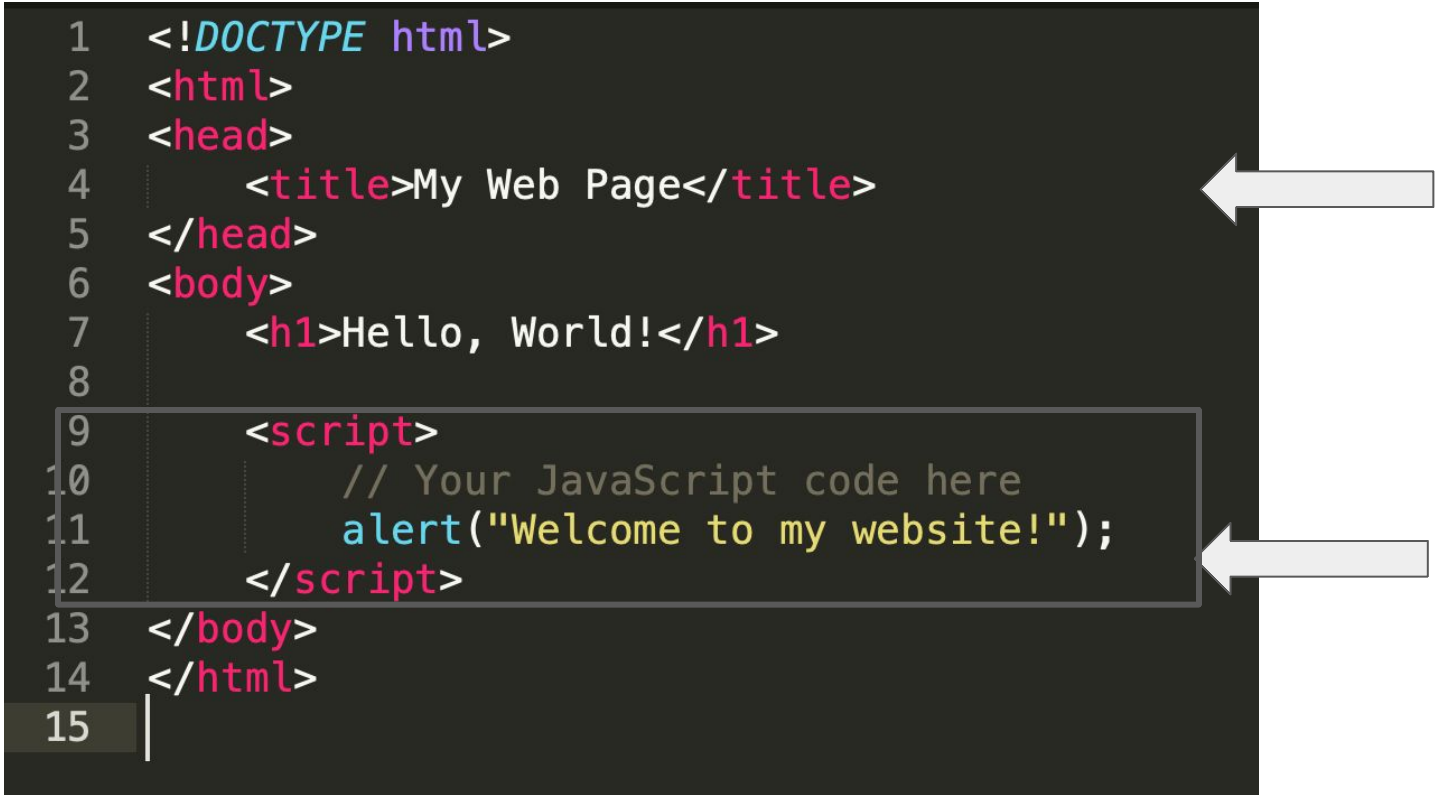


Where to insert JavaScript

- JavaScript in the HTML file
- JavaScript in an external file

JavaScript in HTML file: including your JavaScript code between <script> tags

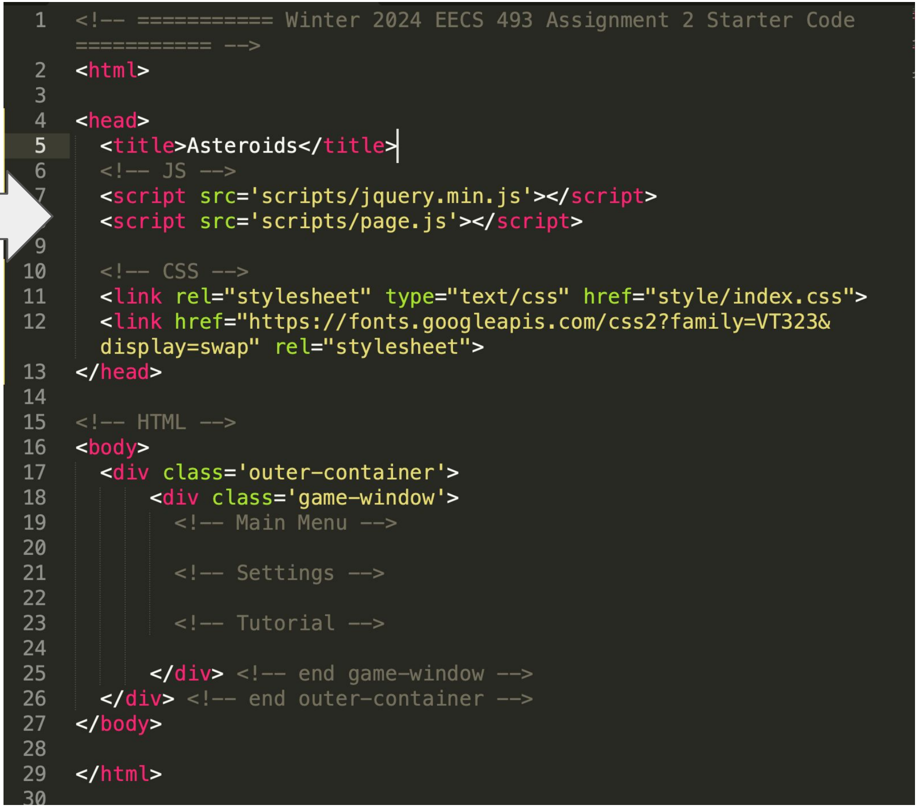
- In <head>
- Right before </body>



```html
<!DOCTYPE html>
<html>
<head>
    <title>My Web Page</title>
</head>
<body>
    <h1>Hello, World!</h1>

    <script>
        // Your JavaScript code here
        alert("Welcome to my website!");
    </script>
</body>
</html>

```

Show what this page does

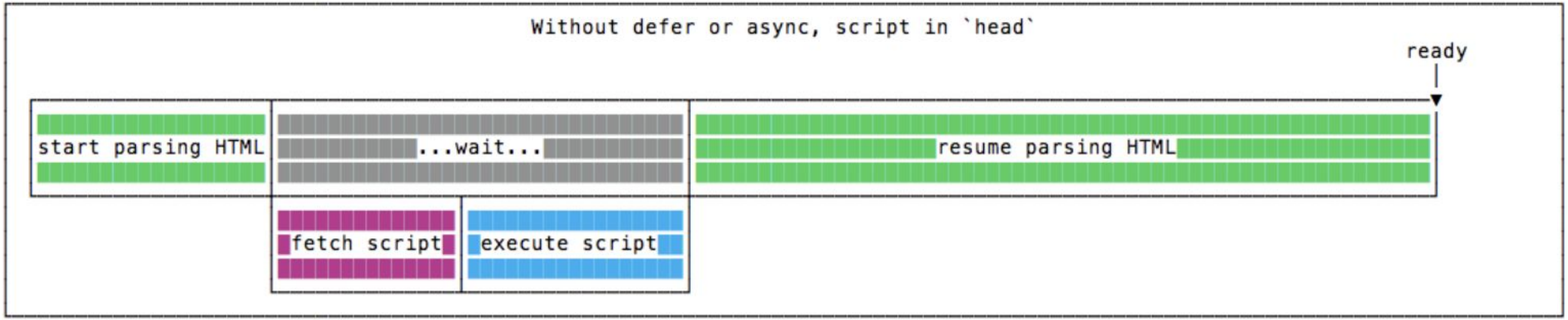
Including JS in an external .js file (recommended)



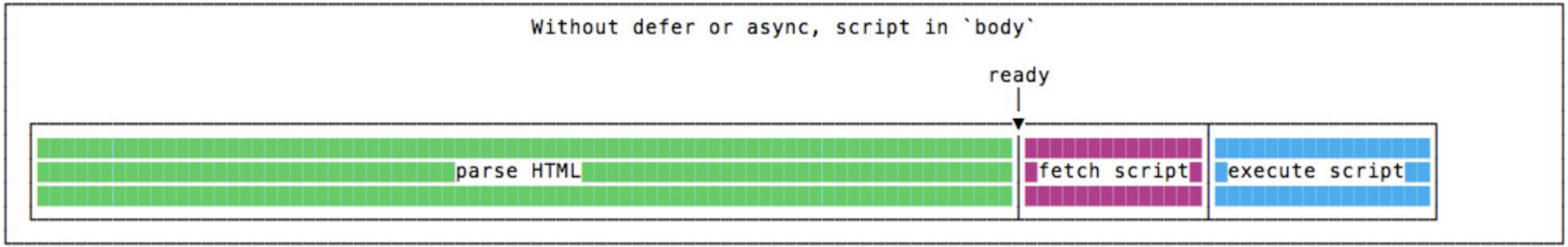
```
<!-- =========== Winter 2024 EECS 493 Assignment 2 Starter Code =========== -->
<html>

<head>
  <title>Asteroids</title>
  <!-- JS -->
  <script src='scripts/jquery.min.js'></script>
  <script src='scripts/page.js'></script>

  <!-- CSS -->
  <link rel="stylesheet" type="text/css" href="style/index.css">
  <link href="https://fonts.googleapis.com/css2?family=VT323&display=swap" rel="stylesheet">
</head>

<!-- HTML -->
<body>
  <div class='outer-container'>
    <div class='game-window'>
      <!-- Main Menu -->

      <!-- Settings -->

      <!-- Tutorial -->

    </div> <!-- end game-window -->
  </div> <!-- end outer-container -->
</body>

</html>

```

When is JavaScript executed

JavaScript are commands to the browser



Without defer or async, script in `head`



Without defer or async, script in `body`

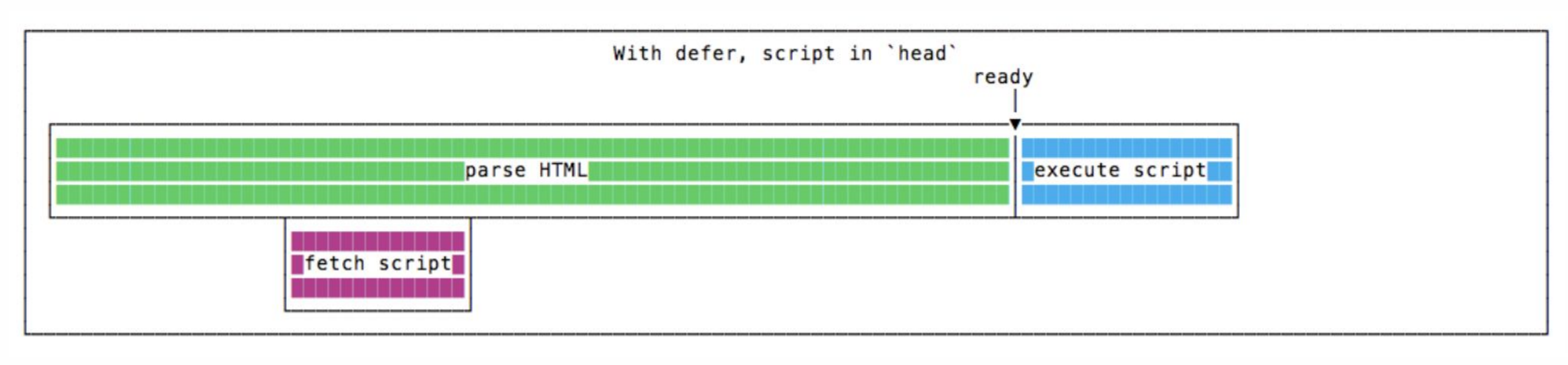
JavaScript in <head> vs <body>

- It’s recommended to include scripts at the bottom of <body> to ensure your HTML content loads first.
- The scripts may not apply, when they depend on the HTML content.

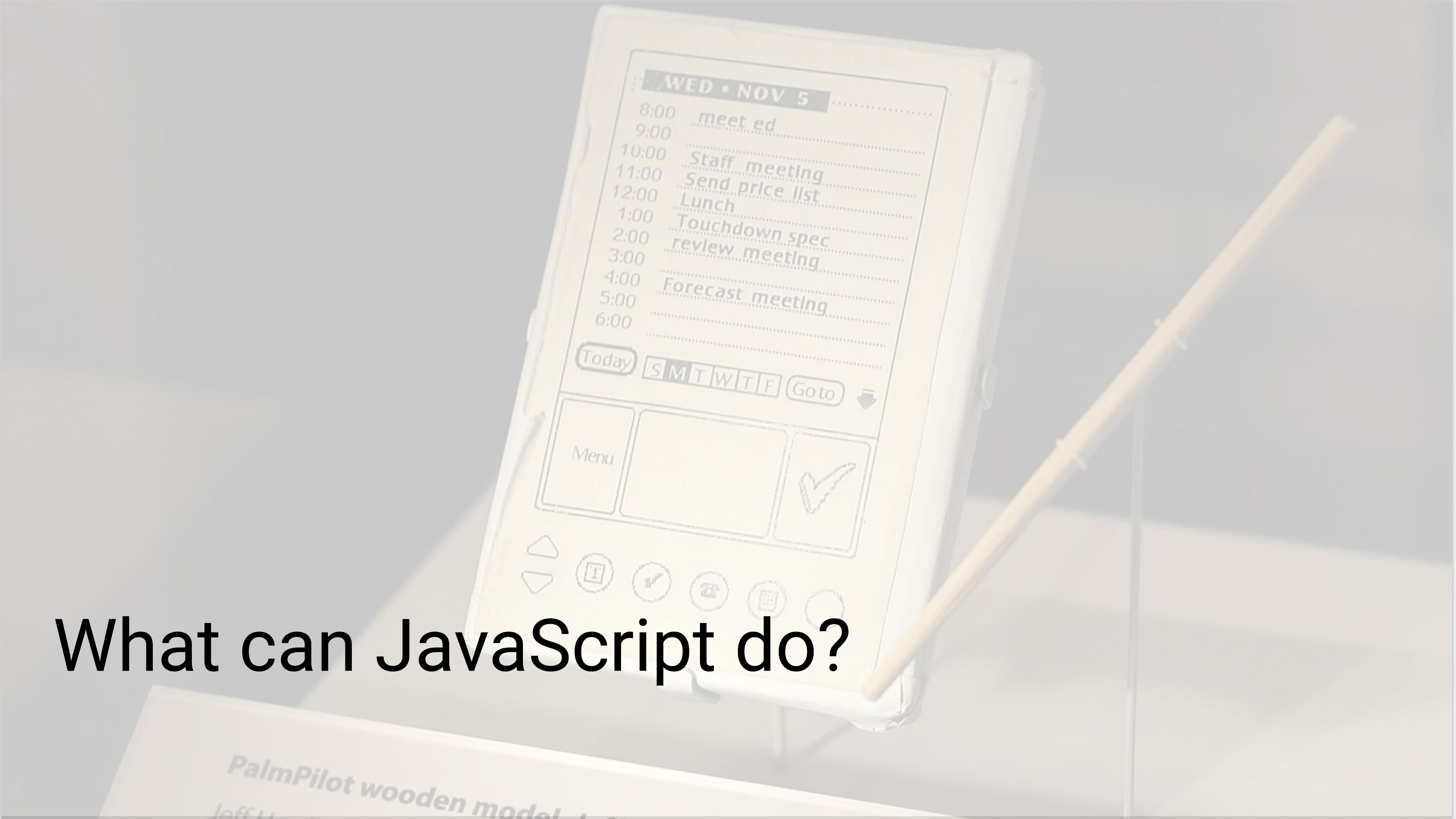
Livecoding example 1

script-defer

JavaScript in <head> with defer



What can JavaScript do?



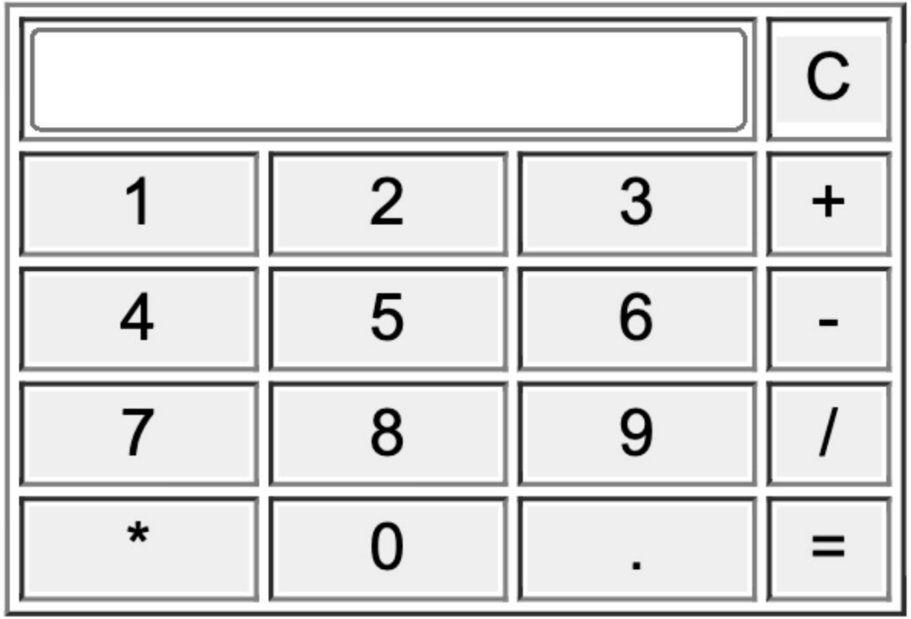
Livecoding example 2

jsbasics/jsdemo.html

Use JavaScript to

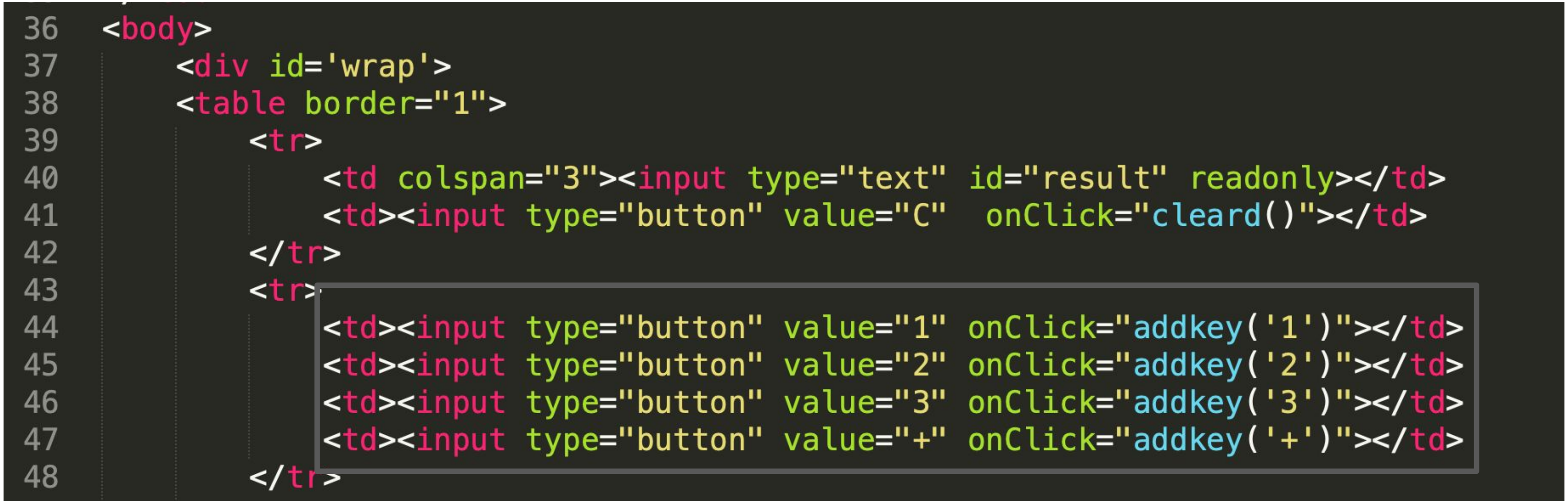
- Change HTML content
- Change value of objects
- Change style
- Change layout
- …

Let’s build a calculator (calculator.html)



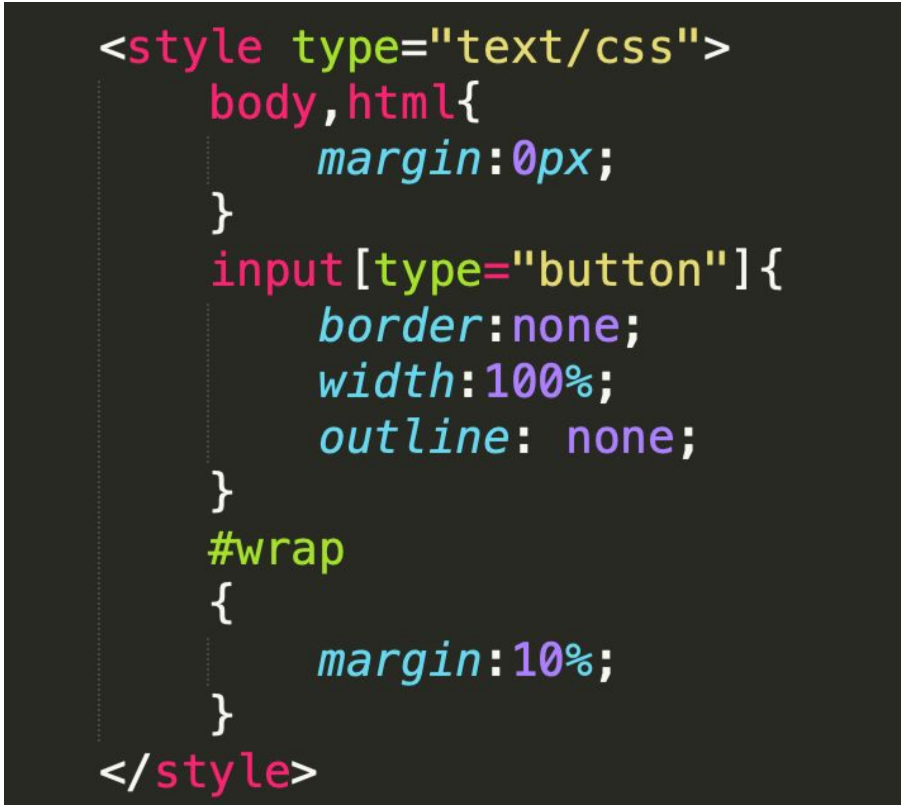
HTML: Buttons in Tables

Each button calls a JavaScript function when it gets a click event



```
<body>
    <div id='wrap'>
    <table border="1">
        <tr>
            <td colspan="3"><input type="text" id="result" readonly></td>
            <td><input type="button" value="C"  onClick="cleard()"></td>
        </tr>
        <tr>
            <td><input type="button" value="1" onClick="addkey('1')"></td>
            <td><input type="button" value="2" onClick="addkey('2')"></td>
            <td><input type="button" value="3" onClick="addkey('3')"></td>
            <td><input type="button" value="+" onClick="addkey('+')"></td>
        </tr>
```

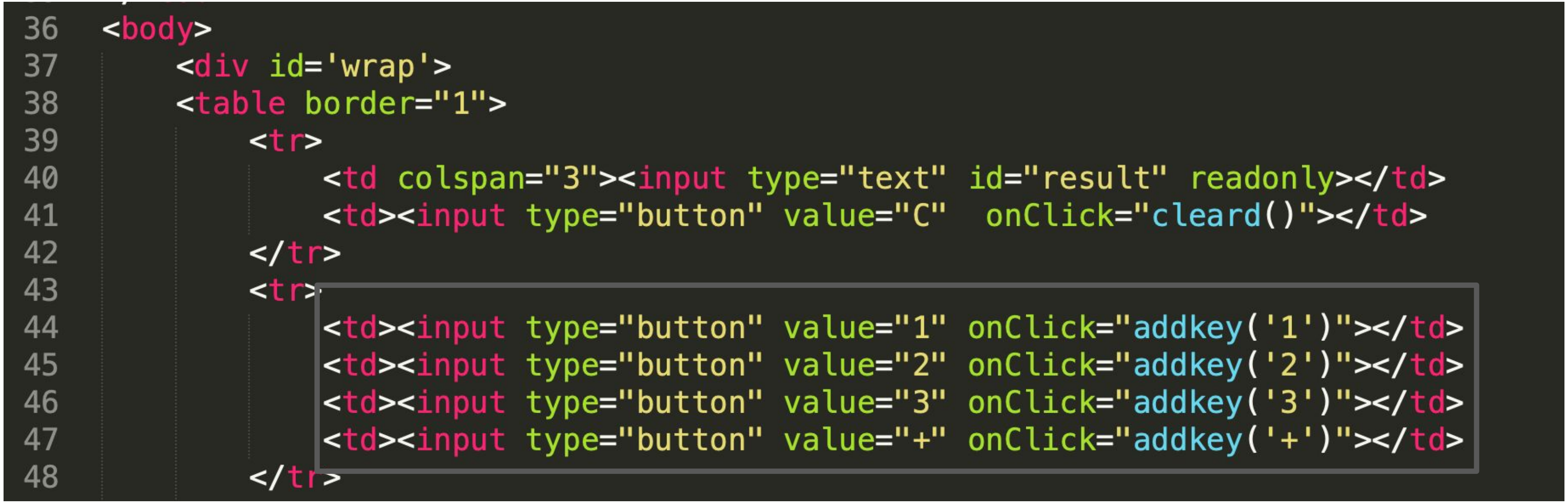
CSS: Pretty minimal



```
<style type="text/css">
    body,html{
        margin:0px;
    }
    input[type="button"]{
        border:none;
        width:100%;
        outline: none;
    }
    #wrap
    {
        margin:10%;
    }
</style>
```

JavaScript: Handling Keypresses

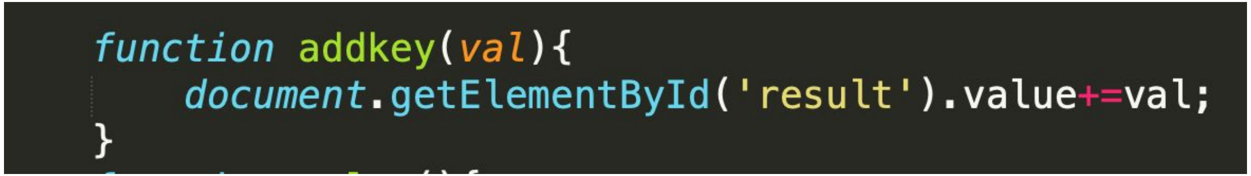
Each keypress just adds a character to the results field.



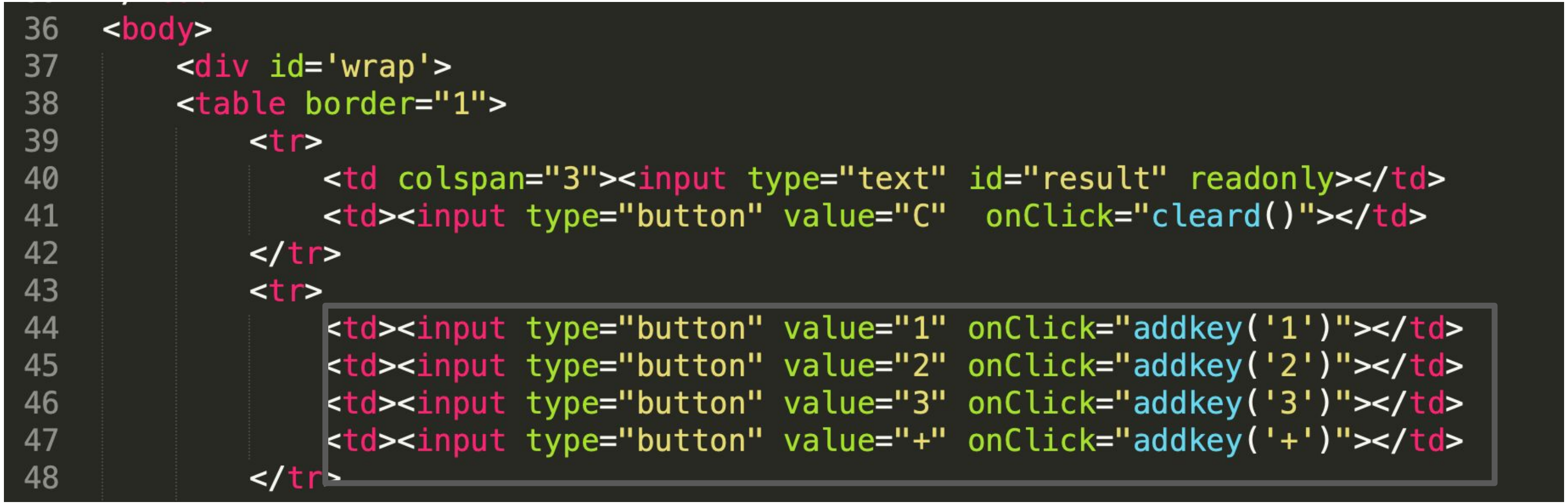
```
<body>
    <div id='wrap'>
    <table border="1">
        <tr>
            <td colspan="3"><input type="text" id="result" readonly></td>
            <td><input type="button" value="C"  onClick="cleard()"></td>
        </tr>
        <tr>
            <td><input type="button" value="1" onClick="addkey('1')"></td>
            <td><input type="button" value="2" onClick="addkey('2')"></td>
            <td><input type="button" value="3" onClick="addkey('3')"></td>
            <td><input type="button" value="+" onClick="addkey('+')"></td>
        </tr>
```

JavaScript: Handling Keypresses

Each keypress just adds a character to the results field.



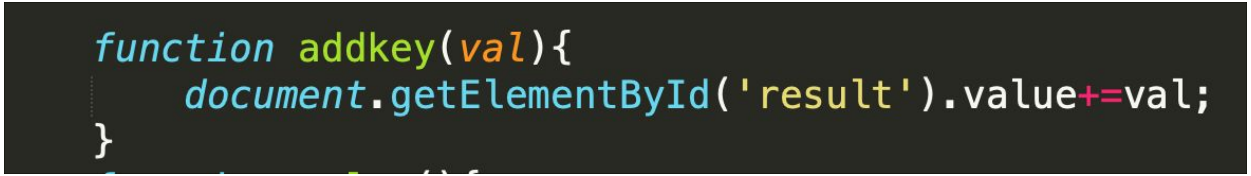
```
function addkey(val){
    document.getElementById('result').value+=val;
}
```



```
<body>
    <div id='wrap'>
    <table border="1">
        <tr>
            <td colspan="3"><input type="text" id="result" readonly></td>
            <td><input type="button" value="C"  onClick="cleard()"></td>
        </tr>
        <tr>
            <td><input type="button" value="1" onClick="addkey('1')"></td>
            <td><input type="button" value="2" onClick="addkey('2')"></td>
            <td><input type="button" value="3" onClick="addkey('3')"></td>
            <td><input type="button" value="+" onClick="addkey('+')"></td>
        </tr>
```

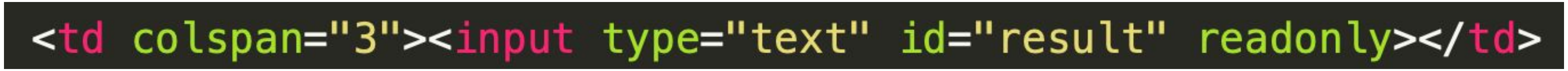
JavaScript: Handling Keypresses

Each keypress just adds a character to the results field.



```
function addkey(val){
    document.getElementById('result').value+=val;
}
```

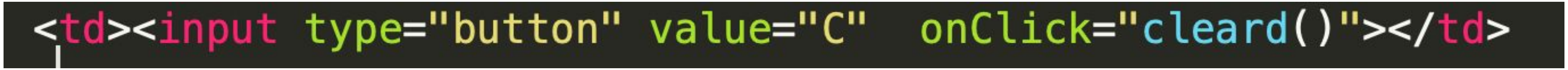
- document.getElementById(‘result’) finds the text area named ‘result’ in the HTML



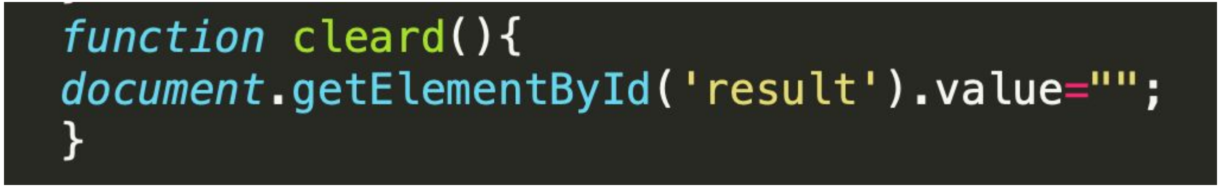
```
<td colspan="3"><input type="text" id="result" readonly></td>
```

- The value is the string in the text area
- += does string concatenation

The clear button erases the text

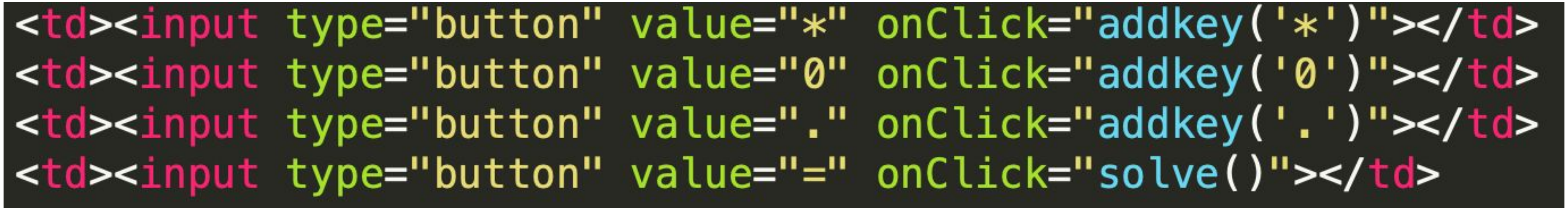


```
<td><input type="button" value="C"  onClick="cleard()"></td>
```

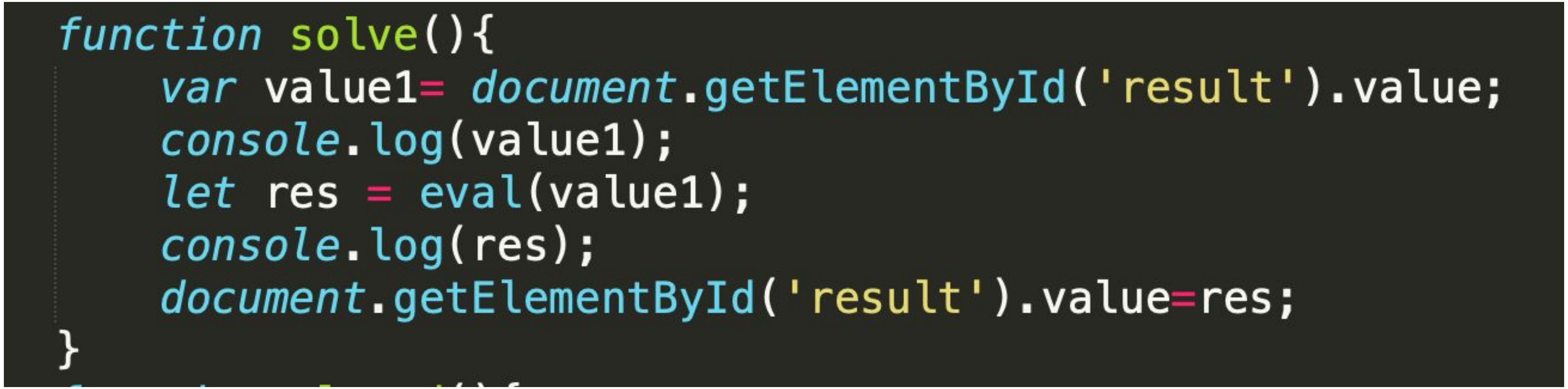


```
function cleard(){
document.getElementById('result').value="";
}
```

= executes the equation

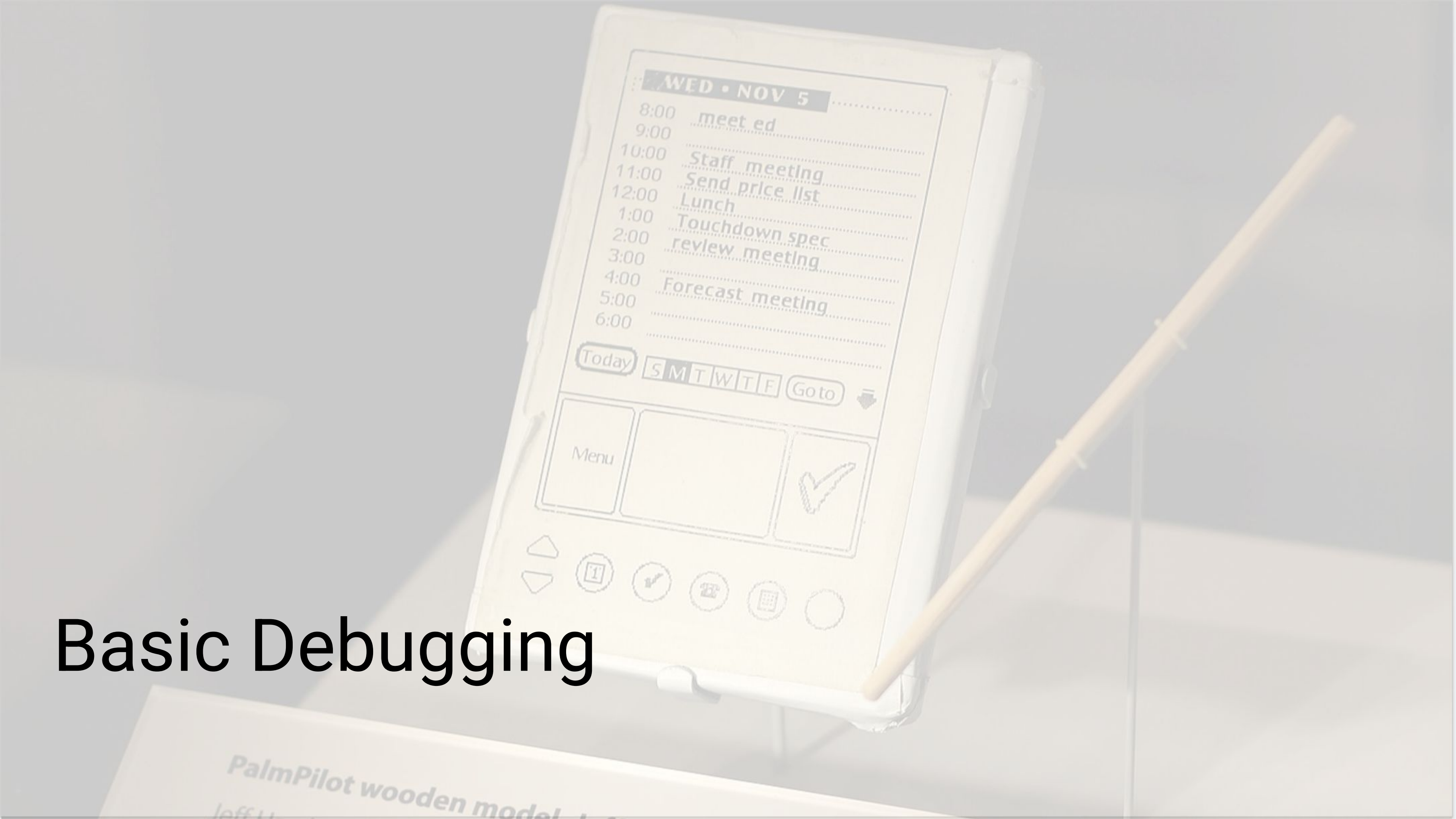


```
<td><input type="button" value="*" onClick="addkey('*')"></td>
<td><input type="button" value="0" onClick="addkey('0')"></td>
<td><input type="button" value="." onClick="addkey('.')"></td>
<td><input type="button" value="=" onClick="solve()"></td>
```



```
function solve(){
    var value1= document.getElementById('result').value;
    console.log(value1);
    let res = eval(value1);
    console.log(res);
    document.getElementById('result').value=res;
}
```

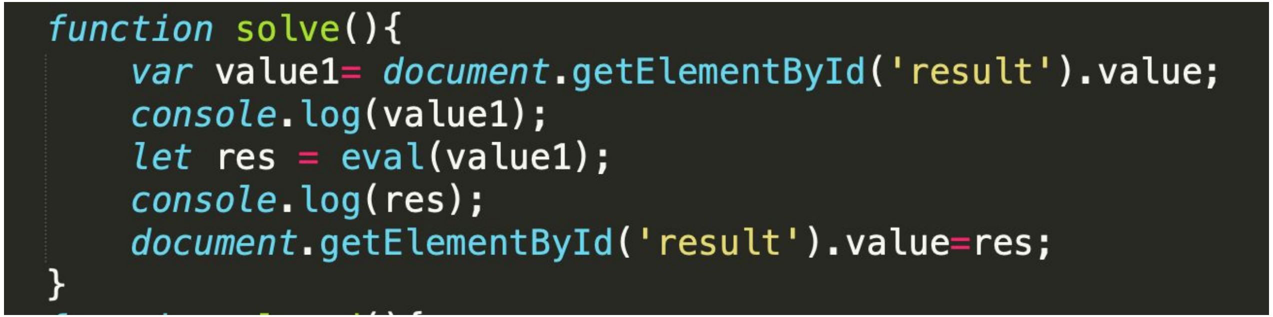
Basic Debugging



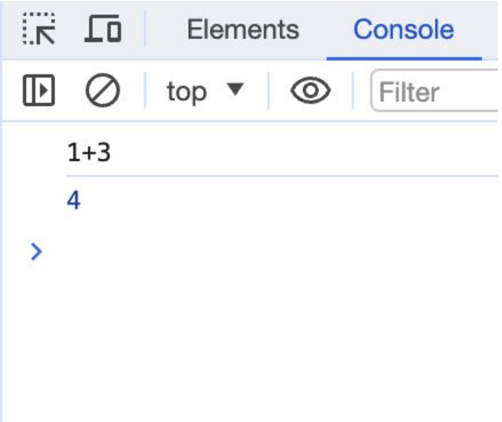
Getting output

Calculator example again (calculator.html)

- console.log()

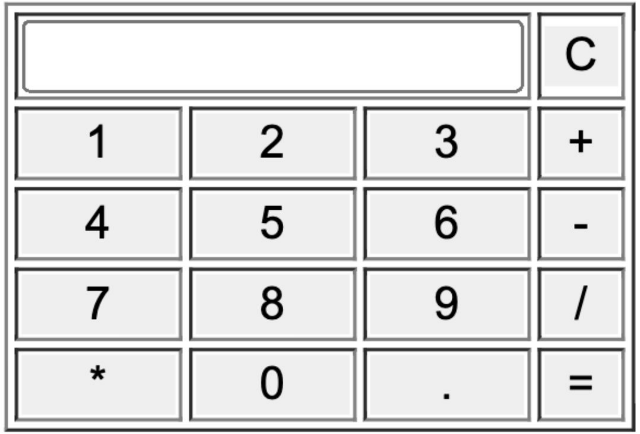


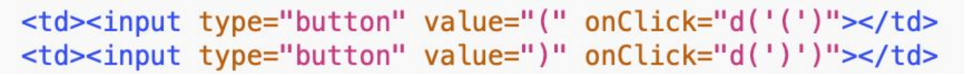
```
function solve(){
    var value1= document.getElementById('result').value;
    console.log(value1);
    let res = eval(value1);
    console.log(res);
    document.getElementById('result').value=res;
}
```



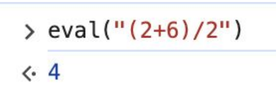
```
1+3
4
>
```

For this calculator, if I want to add parentheses to the calculator, so that it can compute formulas, such as (2+6)/2=4, what do I need to do? *





```
<td><input type="button" value="(" onClick="d('(')"></td>
<td><input type="button" value=")" onClick="d(')')"></td>
```



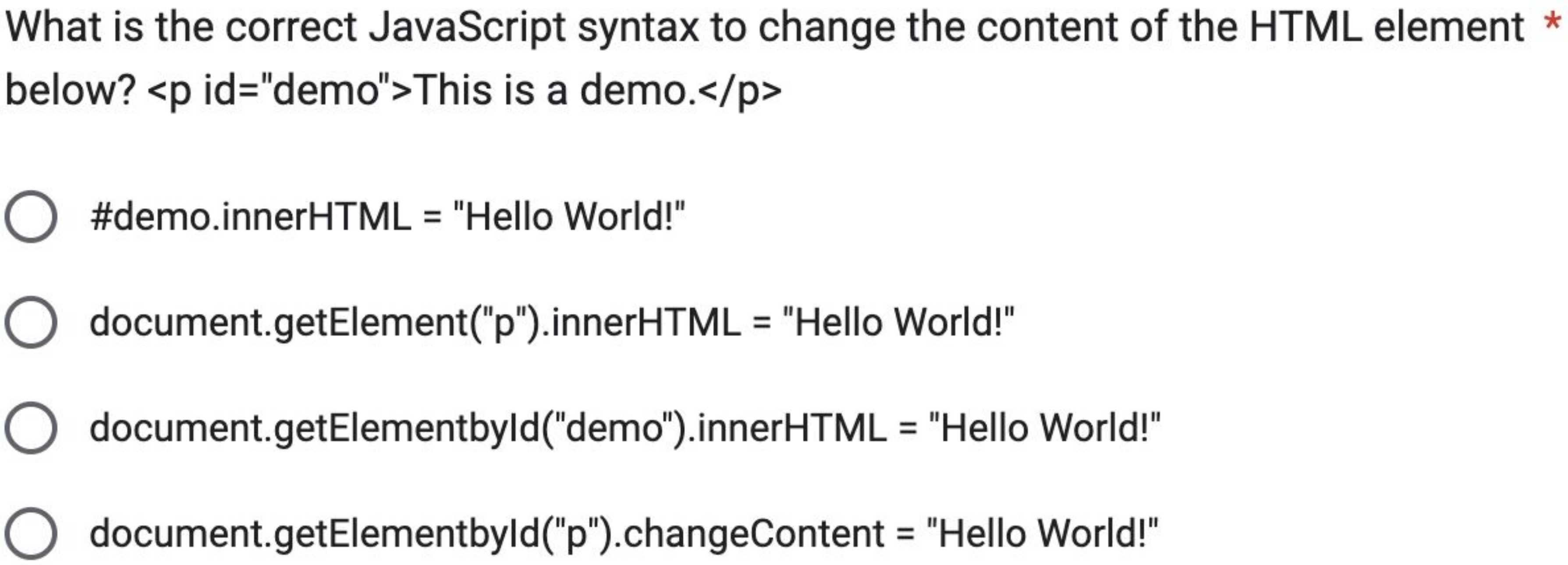
```
> eval("(2+6)/2")
<· 4
```

○ Add two buttons that take inputs on "(" and ")", do not need additional changes.

○ Add two buttons that take inputs on "(" and ")", change the solve function so that it handles parentheses.

○ Make the result text area a textbox that takes keyboard inputs from users

○ Add two buttons that take inputs on "(" and ")", and implement a new function that saves user input into a queue.



What is the correct JavaScript syntax to change the content of the HTML element below? <p id="demo">This is a demo.</p> *

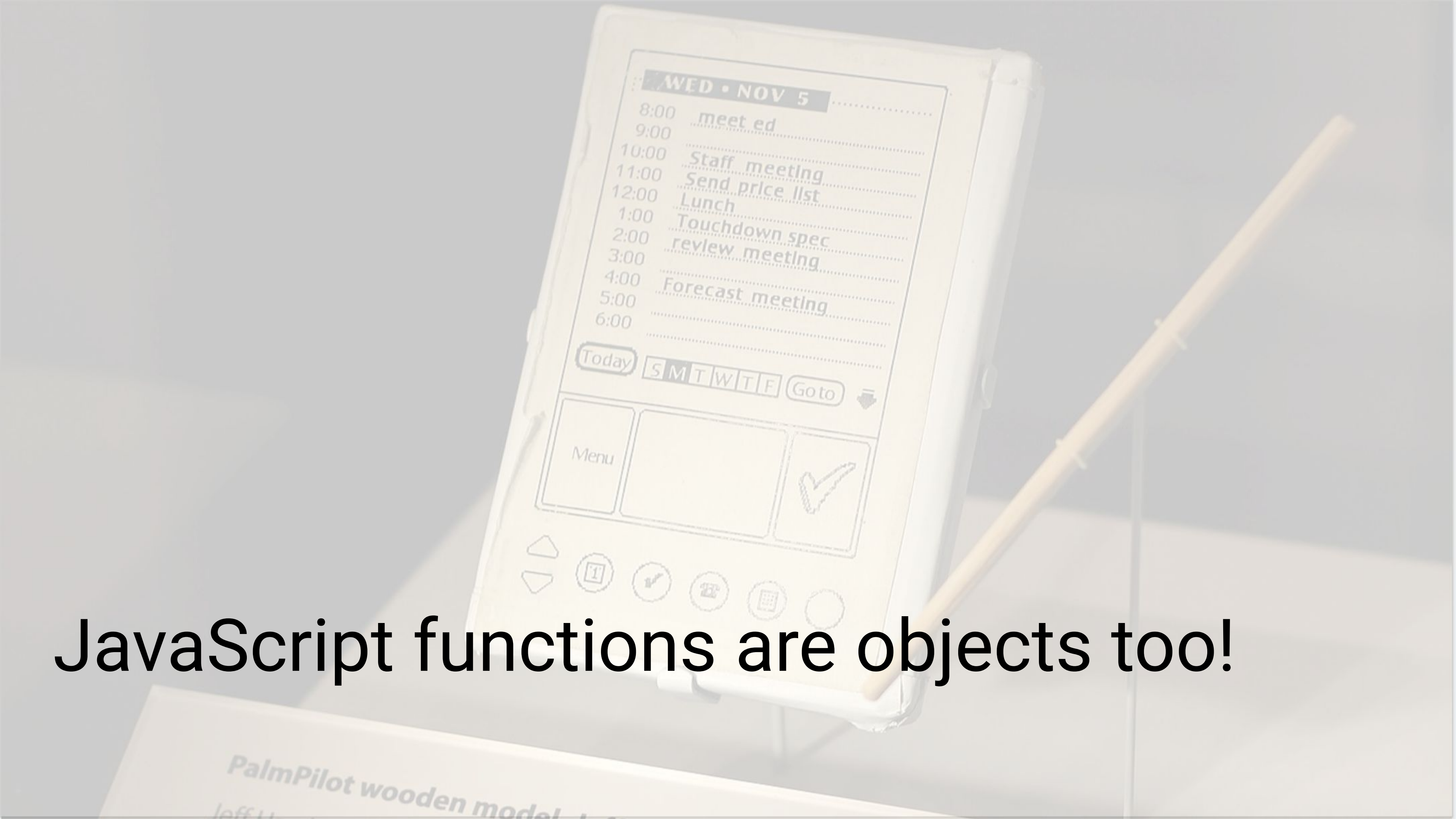
○ #demo.innerHTML = "Hello World!"

○ document.getElement("p").innerHTML = "Hello World!"

○ document.getElementbyId("demo").innerHTML = "Hello World!"

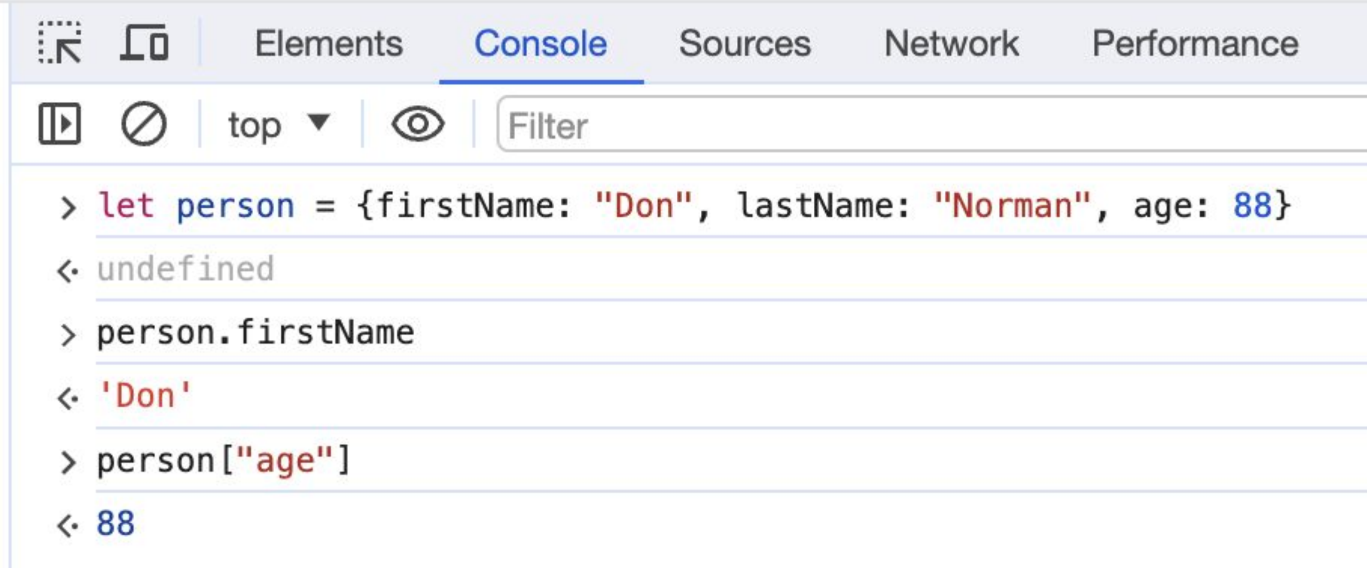
○ document.getElementbyId("p").changeContent = "Hello World!"

JavaScript functions are objects too!



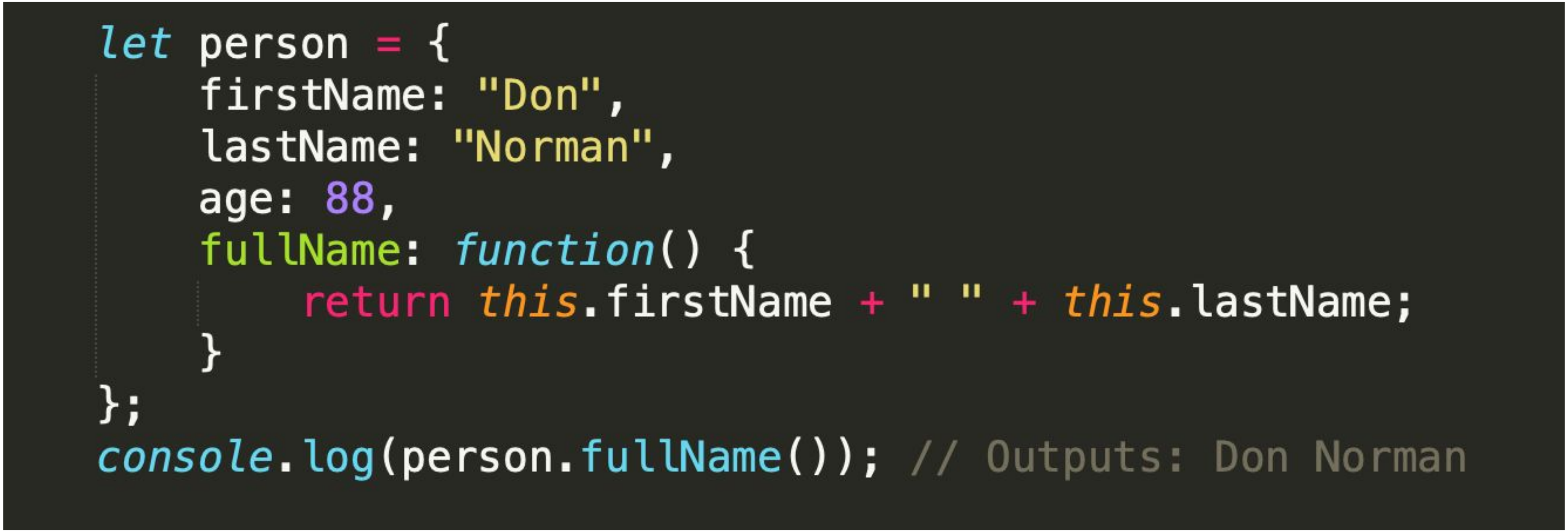
Objects

- The simplest way to create an object is to define it as a list of name-value pairs enclosed in curly braces {}.
- Access the elements with dot notation or square brackets.



```
> let person = {firstName: "Don", lastName: "Norman", age: 88}
<· undefined
> person.firstName
<· 'Don'
> person["age"]
<· 88
```

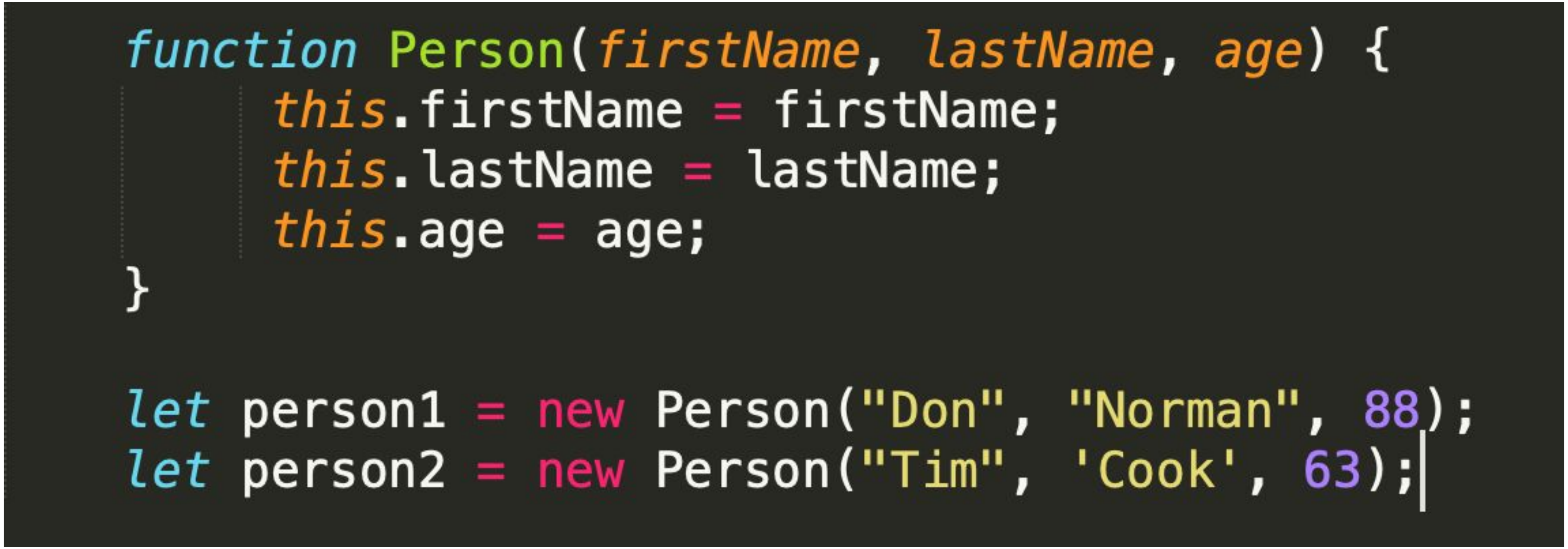
Objects can have properties and methods



```
let person = {
    firstName: "Don",
    lastName: "Norman",
    age: 88,
    fullName: function() {
        return this.firstName + " " + this.lastName;
    }
};
console.log(person.fullName()); // Outputs: Don Norman
```

Creating objects using constructor function

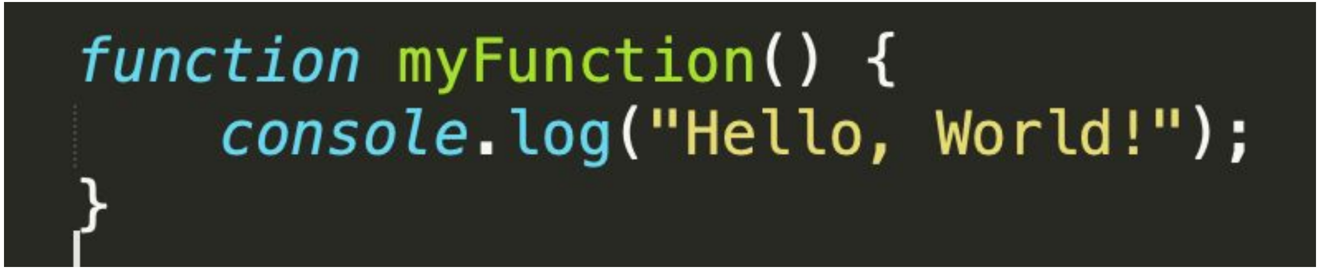
When you need to create multiple objects with the same properties and methods



```
function Person(firstName, lastName, age) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
}

let person1 = new Person("Don", "Norman", 88);
let person2 = new Person("Tim", 'Cook', 63);
```

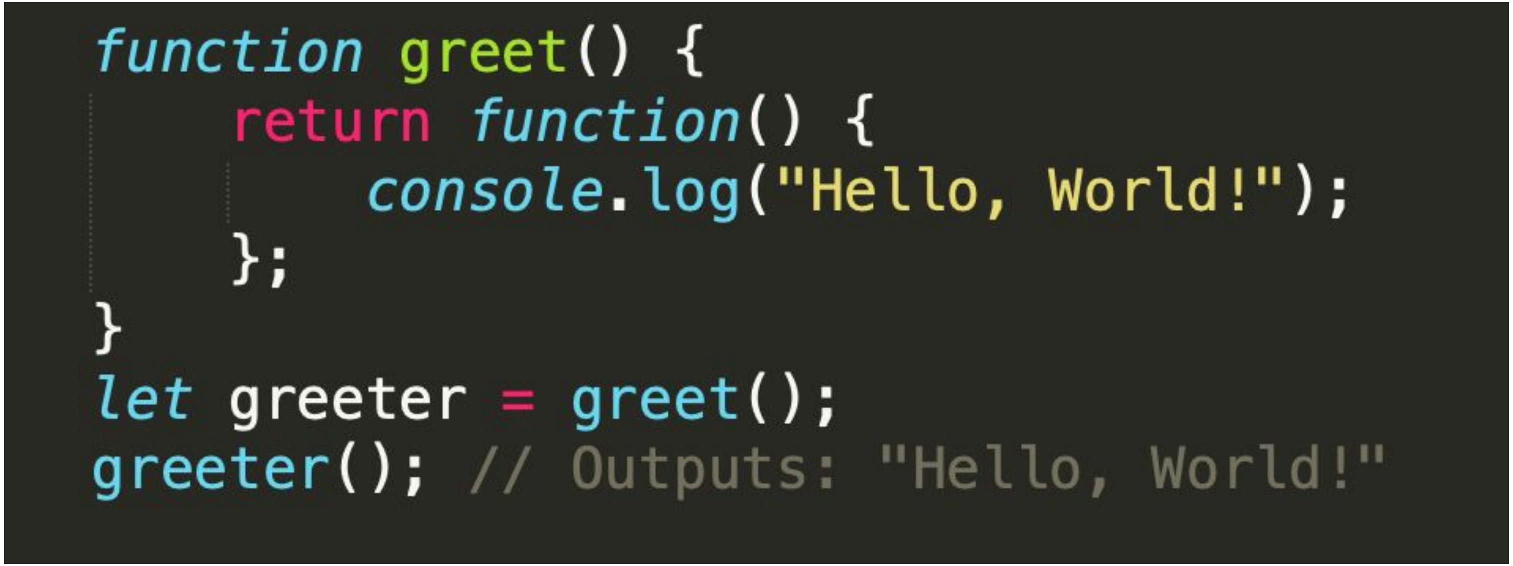
Functions are objects too!

- In JavaScript, functions are first-class objects. This means they can be
    - Stored in variables, arrays or other objects
    - Passed as arguments to other functions
    - Returned as values from functions



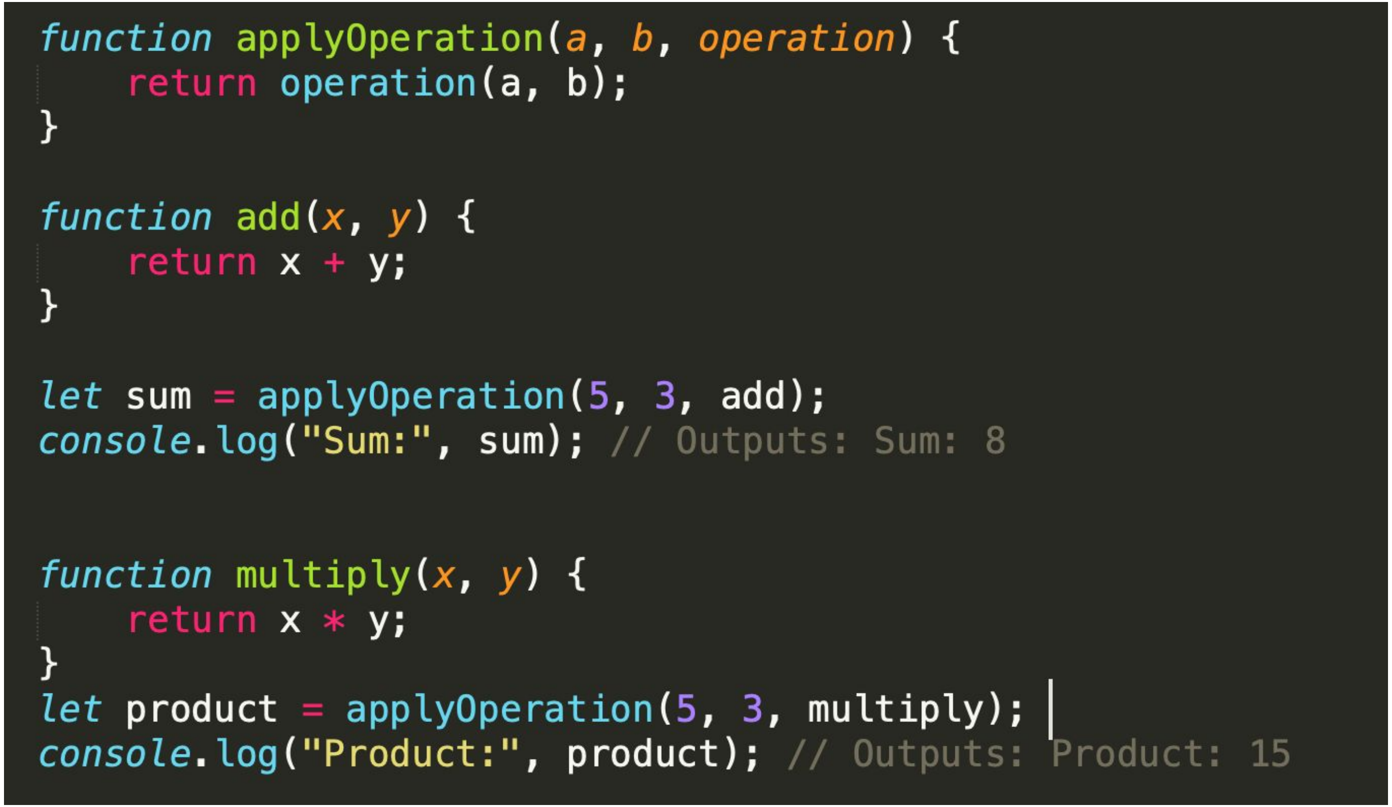
```
function myFunction() {
    console.log("Hello, World!");
}
```

Returned as values from other functions



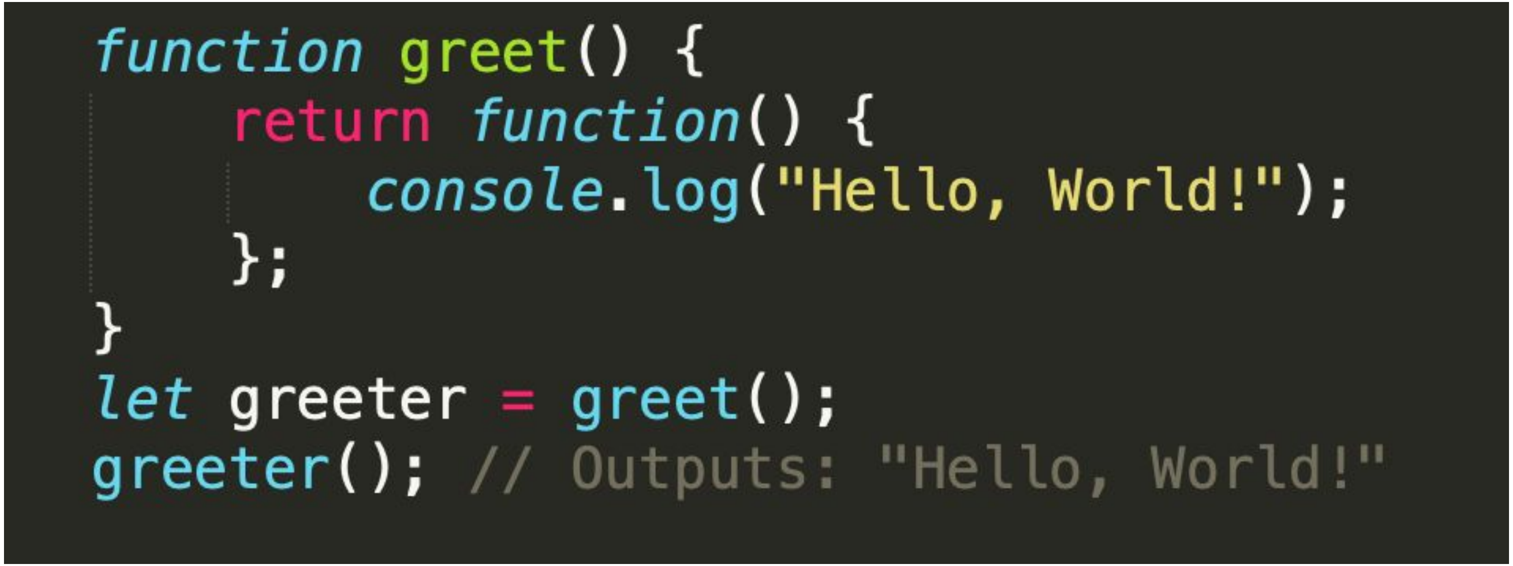
```
function greet() {
    return function() {
        console.log("Hello, World!");
    };
}
let greeter = greet();
greeter(); // Outputs: "Hello, World!"
```

Passed as arguments to other functions



```
function applyOperation(a, b, operation) {
    return operation(a, b);
}

function add(x, y) {
    return x + y;
}

let sum = applyOperation(5, 3, add);
console.log("Sum:", sum); // Outputs: Sum: 8


function multiply(x, y) {
    return x * y;
}
let product = applyOperation(5, 3, multiply);
console.log("Product:", product); // Outputs: Product: 15
```

Functions can be anonymous

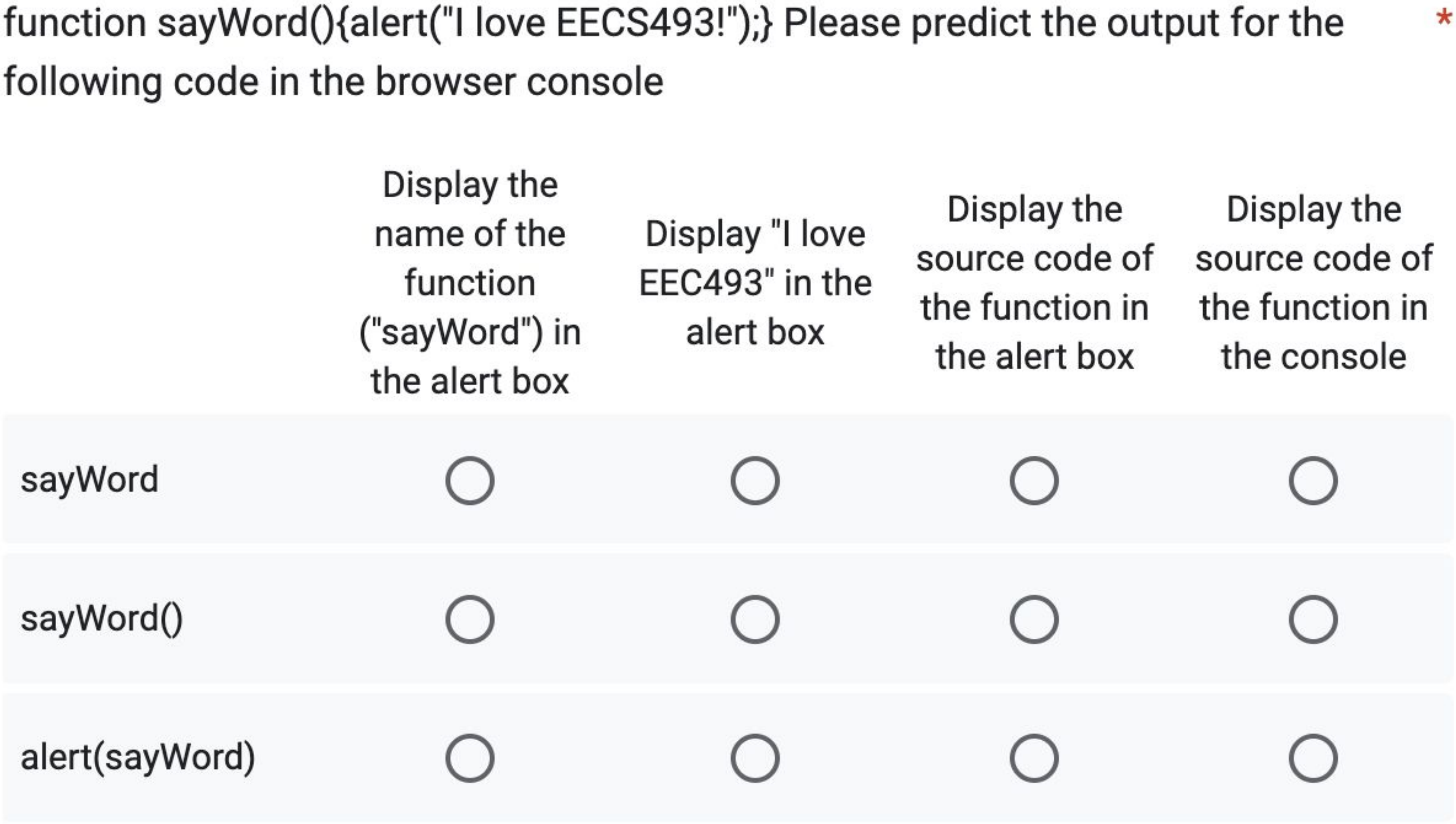


```
function greet() {
    return function() {
        console.log("Hello, World!");
    };
}
let greeter = greet();
greeter(); // Outputs: "Hello, World!"
```

Used as one-off function

Livecoding example 3

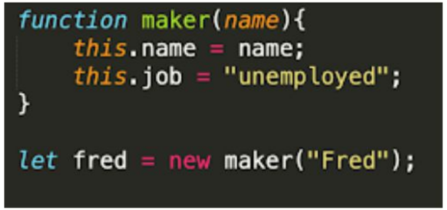
jsobjects.html



function sayWord(){alert("I love EECS493!");} Please predict the output for the following code in the browser console *

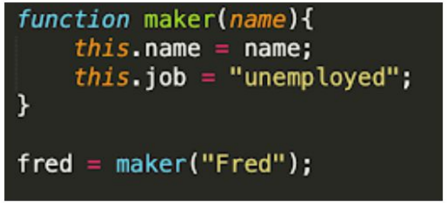
	Display the name of the function ("sayWord") in the alert box	Display "I love EEC493" in the alert box	Display the source code of the function in the alert box	Display the source code of the function in the console
sayWord	○	○	○	○
sayWord()	○	○	○	○
alert(sayWord)	○	○	○	○

I want to create a maker instance with name "Fred" and job "unemployed", which of the following would be right? *



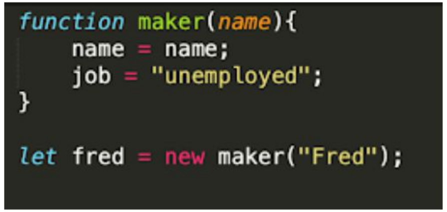
```
function maker(name){
    this.name = name;
    this.job = "unemployed";
}

let fred = new maker("Fred");
```

○ Option 1



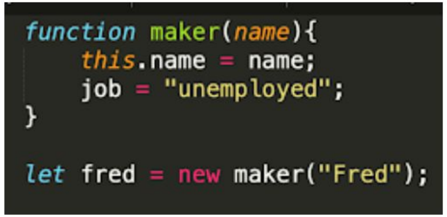
```
function maker(name){
    this.name = name;
    this.job = "unemployed";
}

fred = maker("Fred");
```

○ Option 2



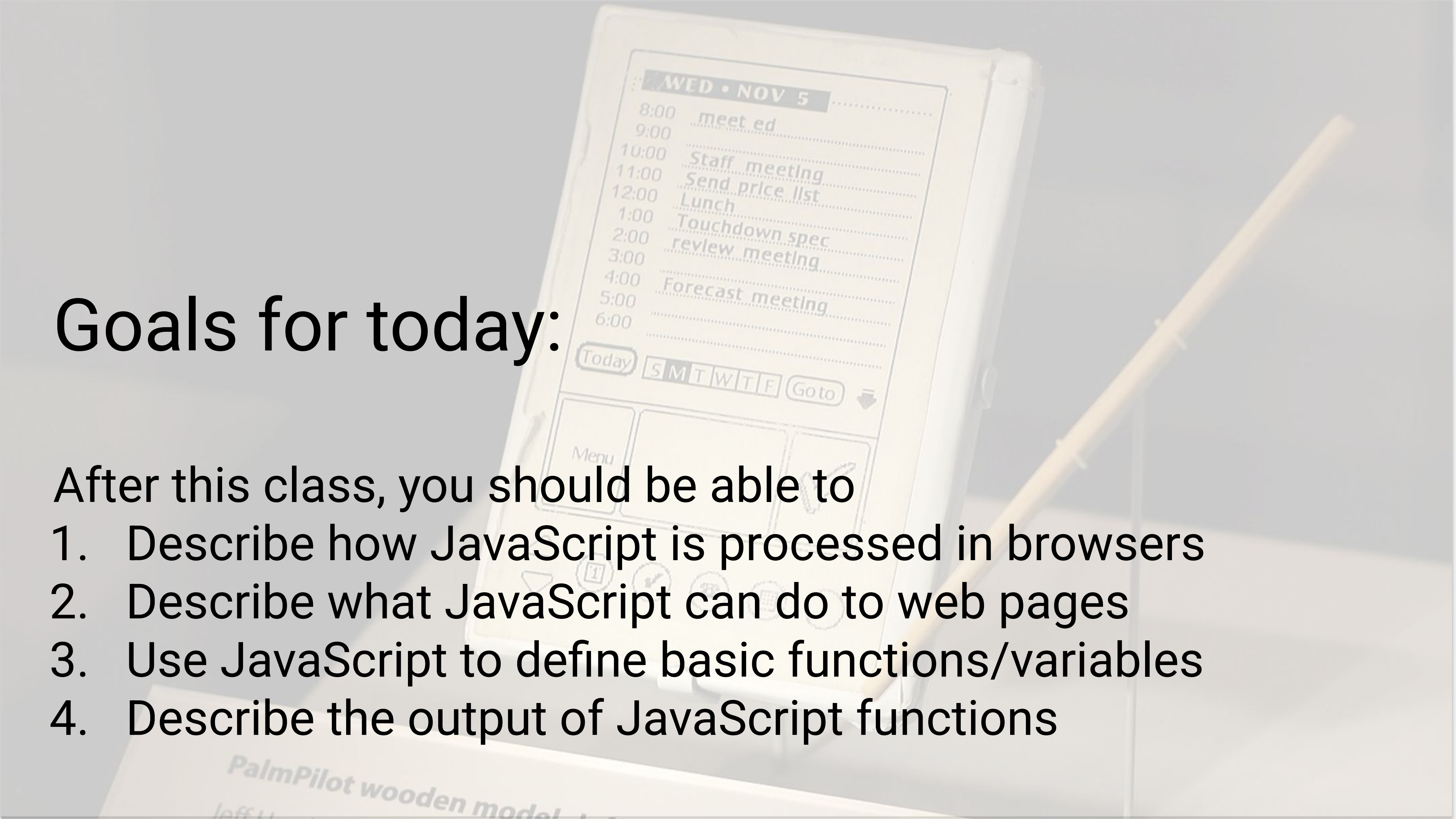
```
function maker(name){
    name = name;
    job = "unemployed";
}

let fred = new maker("Fred");
```

○ Option 3



```
function maker(name){
    this.name = name;
    job = "unemployed";
}

let fred = new maker("Fred");
```

○ Option 4



Goals for today:

After this class, you should be able to

1. Describe how JavaScript is processed in browsers
2. Describe what JavaScript can do to web pages
3. Use JavaScript to define basic functions/variables
4. Describe the output of JavaScript functions