JavaScript First Steps

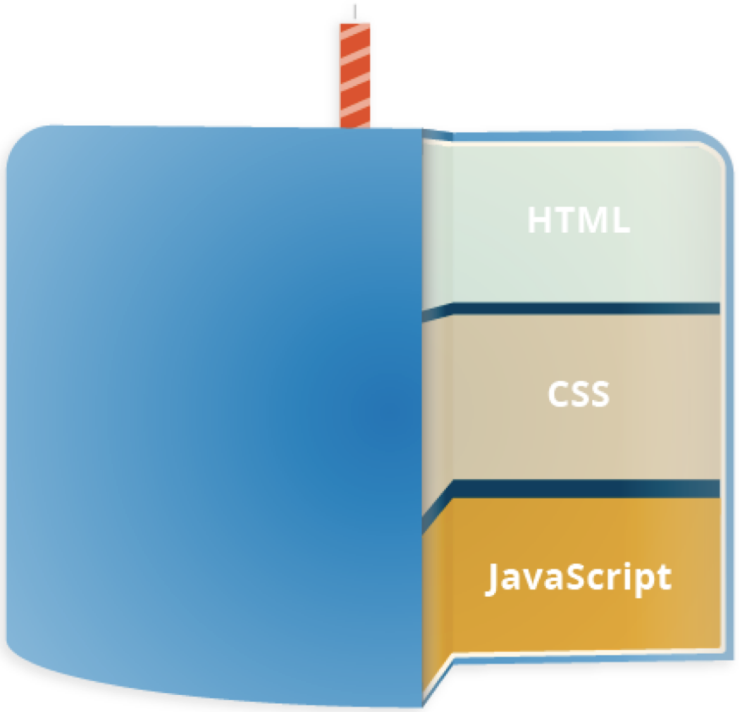
Assoc. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Dept. of Computer Engineering
Khon Kaen University

Agenda

- What is JavaScript?
- A first splash into JavaScript
- What went wrong? Troubleshooting JavaScript
- Storing the information you need – Variables
- Basic in JavaScript
- Handling text
- Useful string methods
- Arrays

A high-level definition

- JavaScript is a scripting or programming language that allows you to implement complex things on web pages



Example with only HTML

- The three layers build on top of one another nicely
- Let's take a simple text label as an example
- We can mark it up using HTML to give it structure and purpose

```
1 | <p>Player 1: Chris</p>
```

Player 1: Chris

Example with HTML and CSS

```css
p {
  font-family: 'helvetica neue', helvetica, sans-serif;
  letter-spacing: 1px;
  text-transform: uppercase;
  text-align: center;
  border: 2px solid rgba(0,0,200,0.6);
  background: rgba(0,0,200,0.3);
  color: rgba(0,0,200,0.6);
  box-shadow: 1px 1px 2px rgba(0,0,200,0.4);
  border-radius: 10px;
  padding: 3px 10px;
  display: inline-block;
  cursor: pointer;
}
```

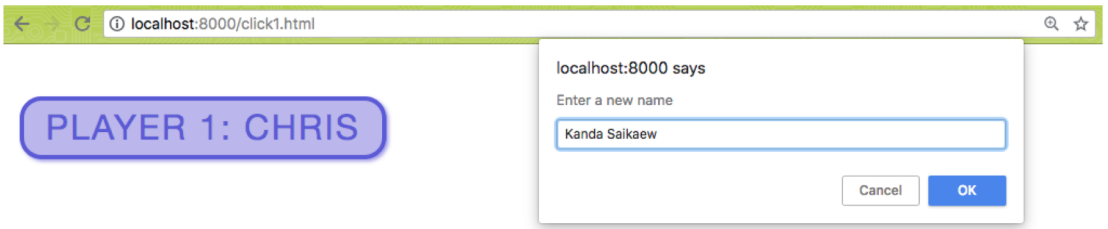
PLAYER 1: CHRIS

Example with HTML, CSS, and JavaScript

```
1  var para = document.querySelector('p');
2
3  para.addEventListener('click', updateName);
4
5  function updateName() {
6    var name = prompt('Enter a new name');
7    para.textContent = 'Player 1: ' + name;
8  }
```

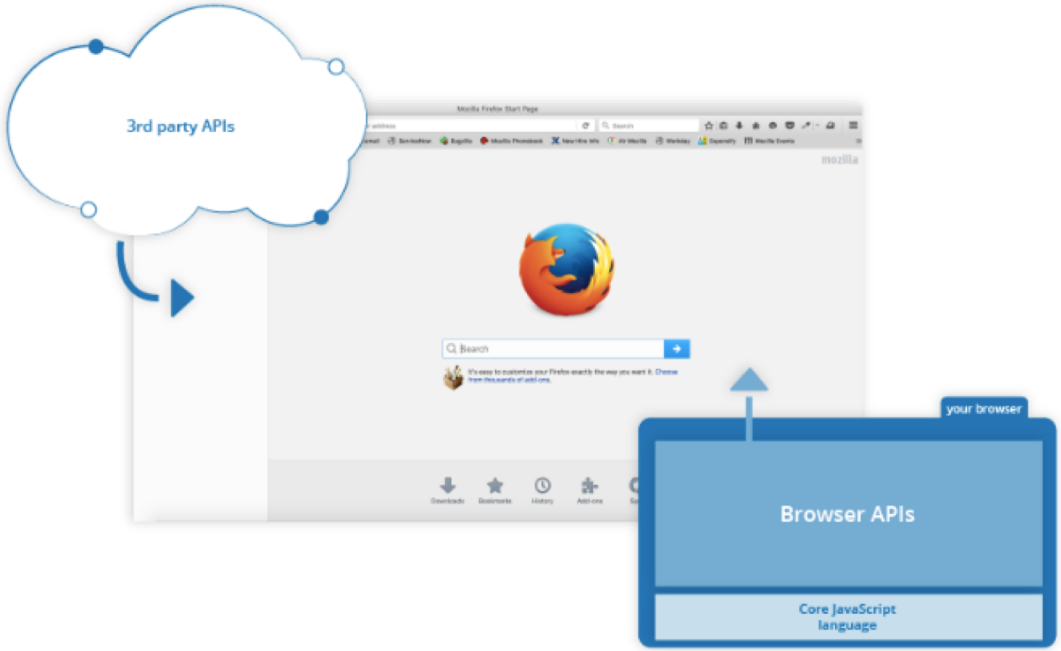
JavaScript Sample1

- Develop the web app that accepts the input name and then update the paragraph with the entered name



Application Programming Interfaces (APIs)

- APIs are ready-made sets of code building blocks that allow a developer to implement programs that would otherwise be hard or impossible to implement



Browser APIs (1/2)

- **Browser APIs** are built into your web browser, and are able to expose data from the surrounding computer environment, or do useful complex things
  - The DOM (Document Object Model) API allows you to manipulate HTML and CSS, creating, removing and changing HTML, dynamically applying new styles to your page, etc.

Browser APIs (2/2)

- The Geolocation API retrieves geographical information. This is how Google Maps is able to find your location, and plot it on a map.
- The Canvas and WebGL APIs allow you to create animated 2D and 3D graphics. People are doing some amazing things using these web technologies
- Audio and Video APIs like HTMLMediaElement and WebRTC allow you to do really interesting things with multimedia, such as play audio and video right in a web page, or grab video from your web camera and display it on someone else's computer

Third party APIS

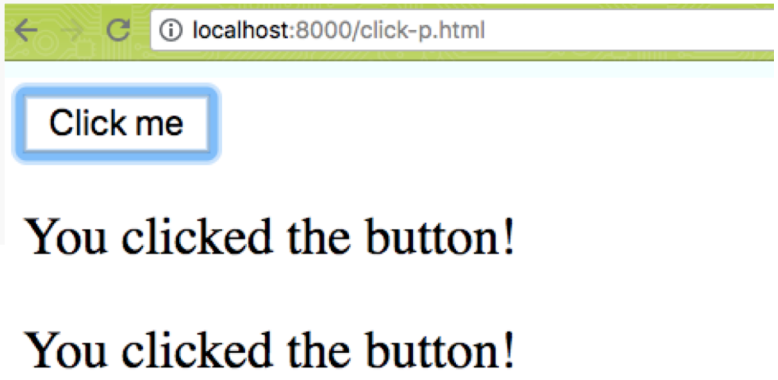
- Third party APIs are not built into the browser by default. For example:
  - The Twitter API allows you to do things like displaying your latest tweets on your websiste
  - The Google Maps API allows you to embed custom maps into your website, and other such functionality

How do you add JavaScript to your page?

- JavaScript is applied to your HTML page in a similar manner to CSS
- Whereas CSS uses <link> elements to apply external stylesheets and <style> elements to apply internal stylesheets to HTML, JavaScript only needs one friend in the world of HTML — the <script> element
- Interval JavaScript
- External JavaScript

Sample Internal JavaScript

```html
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <title>Apply JavaScript example</title>
    <script>
      document.addEventListener("DOMContentLoaded", function() {
        function createParagraph() {
          var para = document.createElement('p');
          para.textContent = 'You clicked the button!';
          document.body.appendChild(para);
        }
        var buttons = document.querySelectorAll('button');
        for(var i = 0; i < buttons.length ; i++) {
          buttons[i].addEventListener('click', createParagraph);
        }
      });
    </script>
  </head>
  <body>
    <button>Click me</button>
  </body>
</html>
```



Exercise1: JavaScript Click and Count

Develop the web app that counts and displays the number of clicks on the button

External JavaScript File

javascript/click-external.html (test-site) — Brackets

```
1  <!DOCTYPE html>
2  <html lang="en-US">
3    <head>
4      <meta charset="utf-8">
5      <title>Apply JavaScript
       example</title>
6      <script src="click-script.js" async>
       </script>
7    </head>
8    <body>
9      <button>Click me</button>
10   </body>
11 </html>
```

```
1  function createParagraph() {
2    var para =
     document.createElement('p');
3    para.textContent = 'You clicked the
     button!';
4    document.body.appendChild(para);
5  }
6
7  var buttons =
   document.querySelectorAll('button');
8
9  for(var i = 0; i < buttons.length ; i++)
   {
10   buttons[i].addEventListener('click',
     createParagraph);
11 }
```

Inline JavaScript handlers Example

```
1 | function createParagraph() {
2 |   var para = document.createElement('p');
3 |   para.textContent = 'You clicked the button!';
4 |   document.body.appendChild(para);
5 | }
```

```
1 | <button onclick="createParagraph()">Click me!</button>
```

Inline JavaScript handlers

- In the example, the <button> element includes an inline onclick handler to make the function run when the button is pressed
- It is bad practice to pollute your HTML with JavaScript
- It is inefficient since you have to include the onclick="createParagraph()" attribute on every button you wanted the JavaScript to apply to

Event Listener

```
document.addEventListener("DOMContentLoaded", function() {
  ...
});
```

- This is an event listener, which listens for the browser's "DOMContentLoaded" event, which signifies that the HTML body is completely loaded and parse
- The JavaScript inside this block will not run until after that event is fired, therefore the error is avoided

The async attribute

- In the external example, we use a more modern JavaScript feature to solve the problem, the **async** attribute, which tells the browser to continue downloading the HTML content once the <script> tag has been reached
- In this case both the script and the HTML will load simultaneously and the code will work

An old-fashioned solution

- Put your script element right at the bottom of the body (e.g. just before the </body> tag), so that the script would load after all the HTML has been parsed
- Problem:
  - Loading/parsing of the script is completely blocked until the HTML DOM has been loaded
  - On larger sites with lots of JavaScript, this can cause a major performance issue, slowing down your site.

async and defer

- There are actually two ways we can bypass the problem of the blocking script
  - **async** and **defer**
- Async scripts will download the script without blocking rendering the page and will execute it as soon as the script finishes downloading
- You get no guarantee that scripts will run in any specific order
- It's best to use async when the scripts in the page run independently

Sample code with async

```
1 | <script async src="js/vendor/jquery.js"></script>
2 |
3 | <script async src="js/script2.js"></script>
4 |
5 | <script async src="js/script3.js"></script>
```

- You can’t reply on the order the scripts will load in
- Jquery.js may load before or after script2.js and script3.js

Sample code with defer

- Defer will run the scripts in the order they appear in the page and execute them as soon as the script and content are downloaded

```
1 | <script defer src="js/vendor/jquery.js"></script>
2 |
3 | <script defer src="js/script2.js"></script>
4 |
5 | <script defer src="js/script3.js"></script>
```

Async vs defer

- If your scripts can run independently without dependencies then use async.
- If your scripts depend on other scripts load them using defer and put their corresponding <script> elements in the order you want the browser to execute them.

Comments

- There are two types
- A single line comment is written after a double forward slash (//)

```
1 | // I am a comment
```

- A multi-line comment is written between the strings /* and */

```
1 | /*
2 |   I am also
3 |   a comment
4 | */
```

Agenda

- What is JavaScript?
- **A first splash into JavaScript**
- What went wrong? Troubleshooting JavaScript
- Storing the information you need – Variables
- Basic in JavaScript
- Handling text
- Useful string methods
- Arrays

Guess the number game

- We'll learn how to build up the simple game
- I want you to create a simple guess the number type game. It should choose a random number between 1 and 100, then challenge the player to guess the number in 10 turns. After each turn the player should be told if they are right or wrong — and, if they are wrong, whether the guess was too low or too high. It should also tell the player what numbers they previously guessed. The game will end once the player guesses correctly, or once they run out of turns. When the game ends, the player should be given an option to start playing again.

Sample run of the game #1

Number guessing game

We have selected a random number between 1 and 100. See if you can guess it in 10 turns or fewer. We'll tell you if your guess was too high or too low.

Enter a guess: Submit guess

Previous guesses: 30

Wrong!

Last guess was too high!

Number guessing game

We have selected a random number between 1 and 100. See if you can guess it in 10 turns or fewer. We'll tell you if your guess was too high or too low.

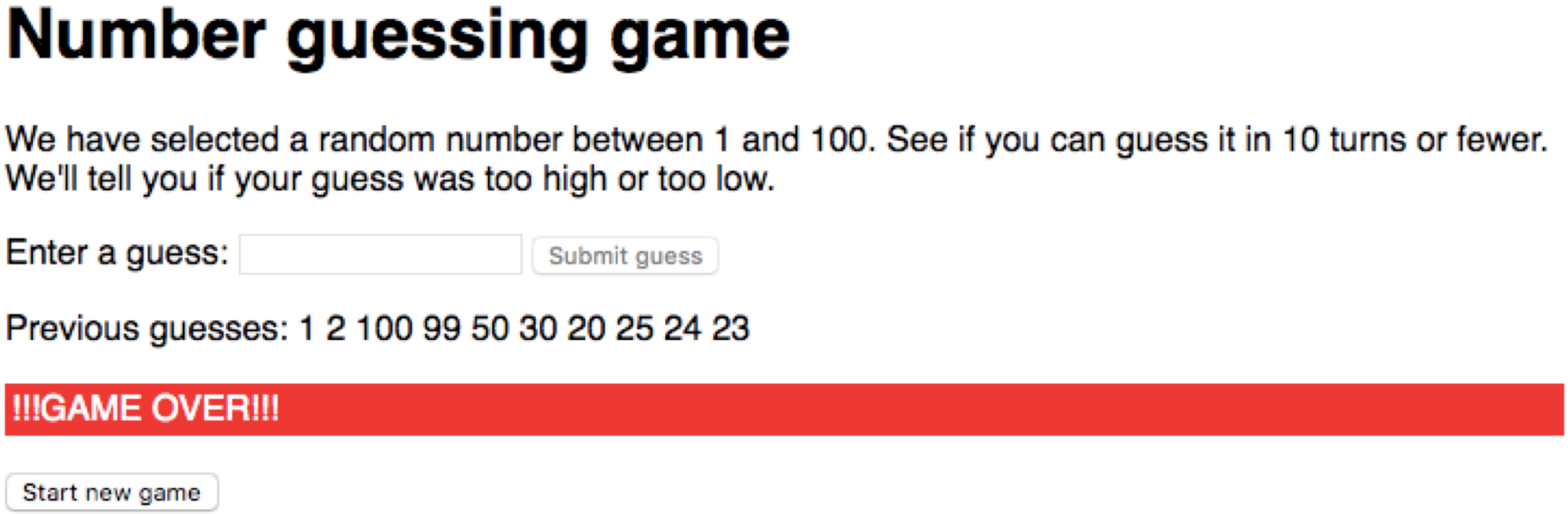
Enter a guess: Submit guess

Previous guesses: 30 20 10 15 18 17

Congratulations! You got it right!

Start new game

Sample run of the game #2



Algorithms (1/3)

1. Generate a random number between 1 and 100.
2. Record the turn number the player is on. Start it on 1.
3. Provide the player with a way to guess what the number is.
4. Once a guess has been submitted first record it somewhere so the user can see their previous guesses.
5. Next, check whether it is the correct number.

Algorithms (2/3)

6. If it is correct:
   - Display congratulations message.
   - Stop the player from being able to enter more guesses (this would mess the game up).
   - Display control allowing the player to restart the game
7. If it is wrong and the player has turns left:
   - Tell the player they are wrong.
   - Allow them to enter another guess.
   - Increment the turn number by 1.

Algorithms (3/3)

8. If it is wrong and the player has no turns left:
   - Tell the player it is game over.
   - Stop the player from being able to enter more guesses (this would mess the game up).
   - Display control allowing the player to restart the game.
9. Once the game restarts, make sure the game logic and UI are completely reset, then go back to step 1.

Initial setup

- To begin this game, download a copy of the html file at https://github.com/krunapon/krunapon.github.io/blob/master/number-guessing-game-start.html
- The place where will be adding all our code inside the <script> element

```
1  <script>
2
3    // Your JavaScript goes here
4
5  </script>
```

Adding variables to store our data

```
1  var randomNumber = Math.floor(Math.random() * 100) + 1;
2
3  var guesses = document.querySelector('.guesses');
4  var lastResult = document.querySelector('.lastResult');
5  var lowOrHi = document.querySelector('.lowOrHi');
6
7  var guessSubmit = document.querySelector('.guessSubmit');
8  var guessField = document.querySelector('.guessField');
9
10 var guessCount = 1;
11 var resetButton;
```

References to the web page

```
27    <body>
28        <h1>Number guessing game</h1>
29
30        <p>We have selected a random number between 1 and 100. See if you can guess it in 10
          turns or fewer. We'll tell you if your guess was too high or too low.</p>
31
32        <div class="form">
33            <label for="guessField">Enter a guess: </label><input type="text" id="guessField"
              class="guessField">
34            <input type="submit" value="Submit guess" class="guessSubmit">
35        </div>
36
37        <div class="resultParas">
38            <p class="guesses"></p>
39            <p class="lastResult"></p>
40            <p class="lowOrHi"></p>
41        </div>
42
```

Functions and Conditionals

```
function checkGuess() {
  var userGuess = Number(guessField.value);
  if (guessCount === 1) {
    guesses.textContent = 'Previous guesses: ';
  }
  guesses.textContent += userGuess + ' ';

  if (userGuess === randomNumber) {
    lastResult.textContent = 'Congratulations! You got it right!';
    lastResult.style.backgroundColor = 'green';
    lowOrHi.textContent = '';
    setGameOver();
  } else if (guessCount === 10) {
    lastResult.textContent = '!!!GAME OVER!!!';
    setGameOver();
  } else {
    lastResult.textContent = 'Wrong!';
    lastResult.style.backgroundColor = 'red';
    if(userGuess < randomNumber) {
      lowOrHi.textContent = 'Last guess was too low!';
    } else if(userGuess > randomNumber) {
      lowOrHi.textContent = 'Last guess was too high!';
    }
  }

  guessCount++;
  guessField.value = '';
  guessField.focus();
}
```

Events

- The constructs that listen out for the event happening are called **event listeners**, and the blocks of code that run in response to the event firing are called **event handlers**

```
1 | guessSubmit.addEventListener('click', checkGuess);
```

Event Listener

```
1 | guessSubmit.addEventListener('click', checkGuess);
```

- This is an event listener to the guessSubmit button
- This is a method that takes two input values (called *arguments*)
  - the type of event we are listening out for (in this case click) as a string
  - the code we want to run when the event occurs (in this case the checkGuess() function)

Creating Elements in Document

```
1  function setGameOver() {
2    guessField.disabled = true;
3    guessSubmit.disabled = true;
4    resetButton = document.createElement('button');
5    resetButton.textContent = 'Start new game';
6    document.body.appendChild(resetButton);
7    resetButton.addEventListener('click', resetGame);
8  }
```

Functions and loops

```
function resetGame() {
  guessCount = 1;

  var resetParas = document.querySelectorAll('.resultParas p');
  for (var i = 0 ; i < resetParas.length ; i++) {
    resetParas[i].textContent = '';
  }

  resetButton.parentNode.removeChild(resetButton);

  guessField.disabled = false;
  guessSubmit.disabled = false;
  guessField.value = '';
  guessField.focus();

  lastResult.style.backgroundColor = 'white';

  randomNumber = Math.floor(Math.random() * 100) + 1;
}
```

A small discussion on objects

```
1 | guessField.focus();
```

- The line uses the focus() method to automatically put the text cursor into the <input> text field as soon as the page loads
- The user can start typing their first guess right away, without having to click the form field first

JavaScript objects

- In JavaScript, everything is an object
- An object is a collection of related functionality stored in a single grouping
- We first created a guessField variable that stores a reference to the the text input form field in our HTML

```
1 | var guessField = document.querySelector('.guessField');
```

Playing with browser objects

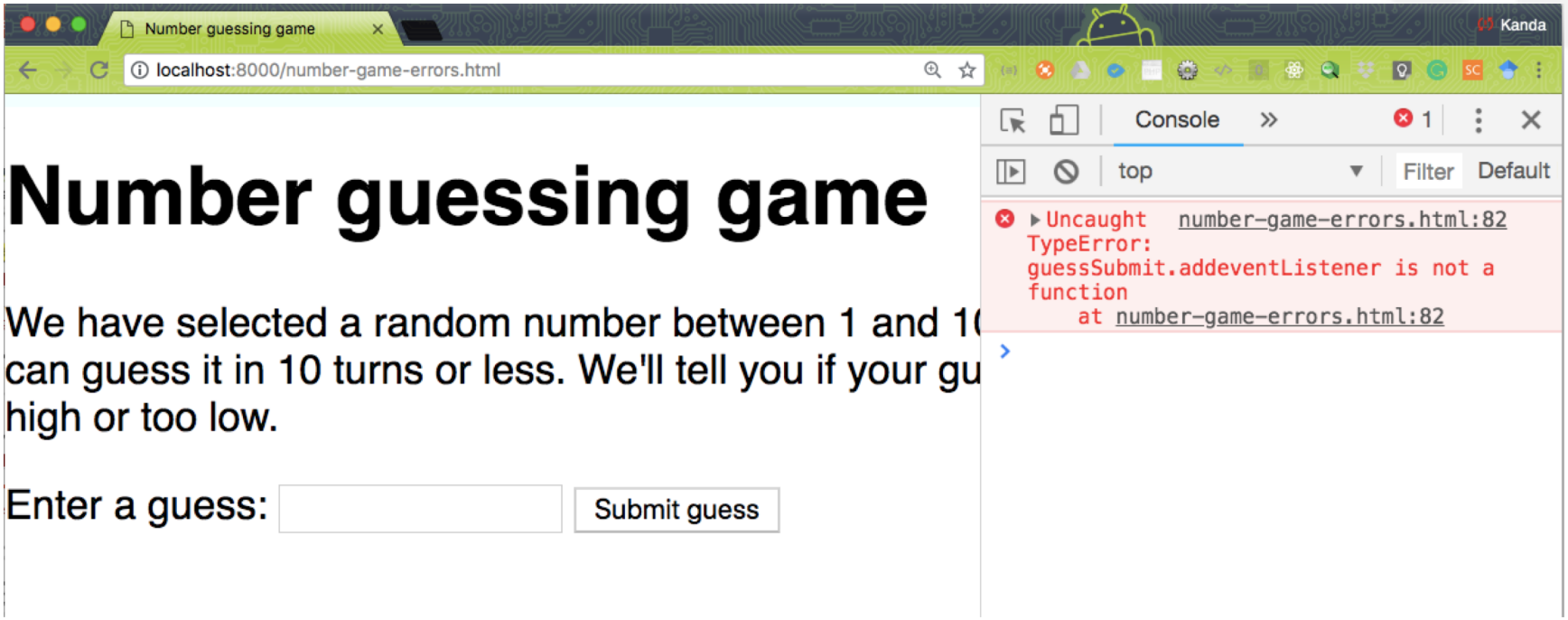
Agenda

- What is JavaScript?
- A first splash into JavaScript
- **What went wrong? Troubleshooting JavaScript**
- Storing the information you need – Variables
- Basic in JavaScript
- Handling text
- Useful string methods
- Arrays

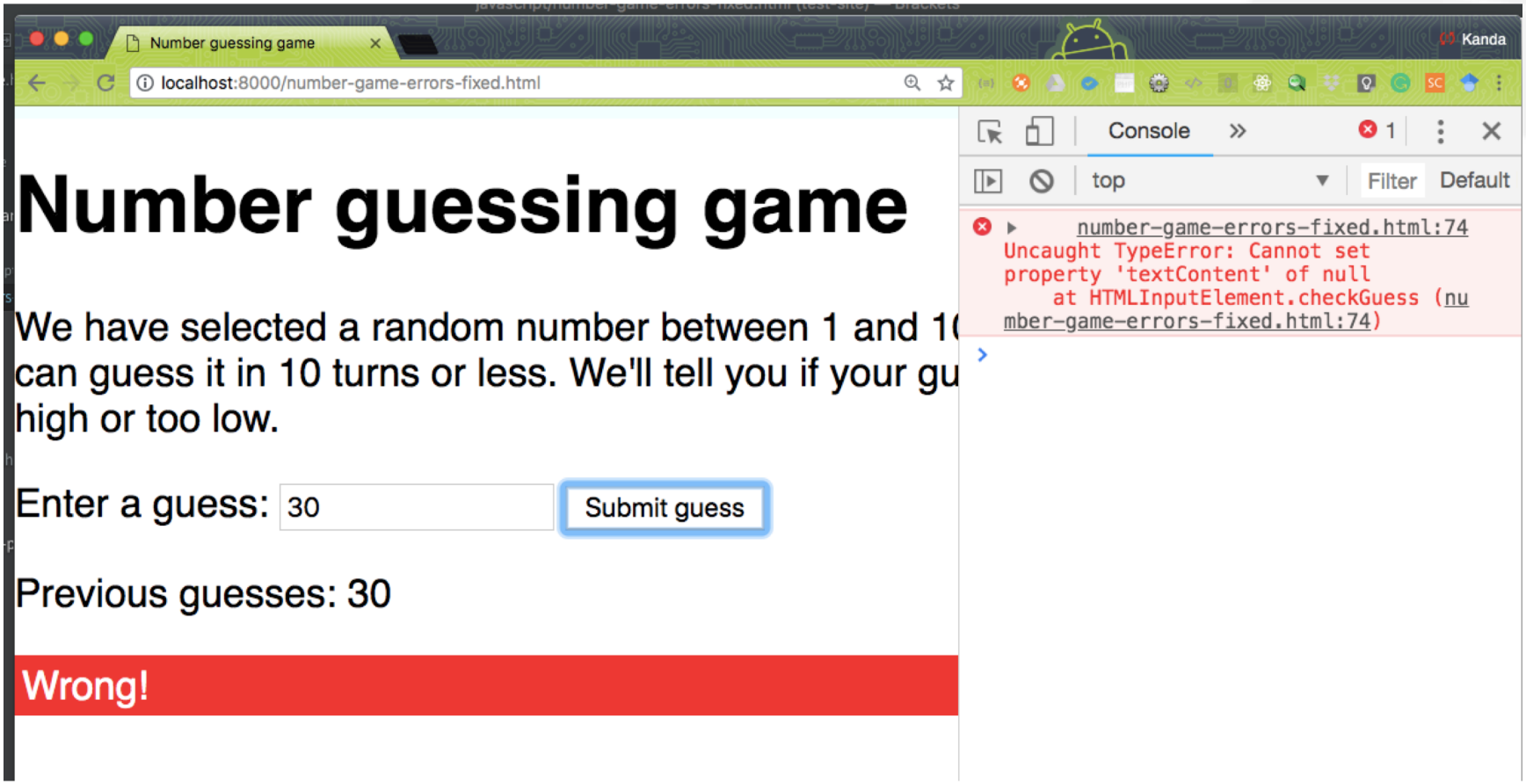
Types of error

- Syntax errors
  - There are spelling errors in your code that actually cause the program not to run at all
- Logic errors
  - There are errors where the syntax is actually correct but the code is not what you intended it to be, meaning that program runs successfully but gives incorrect results

Using developer tools JavaScript console



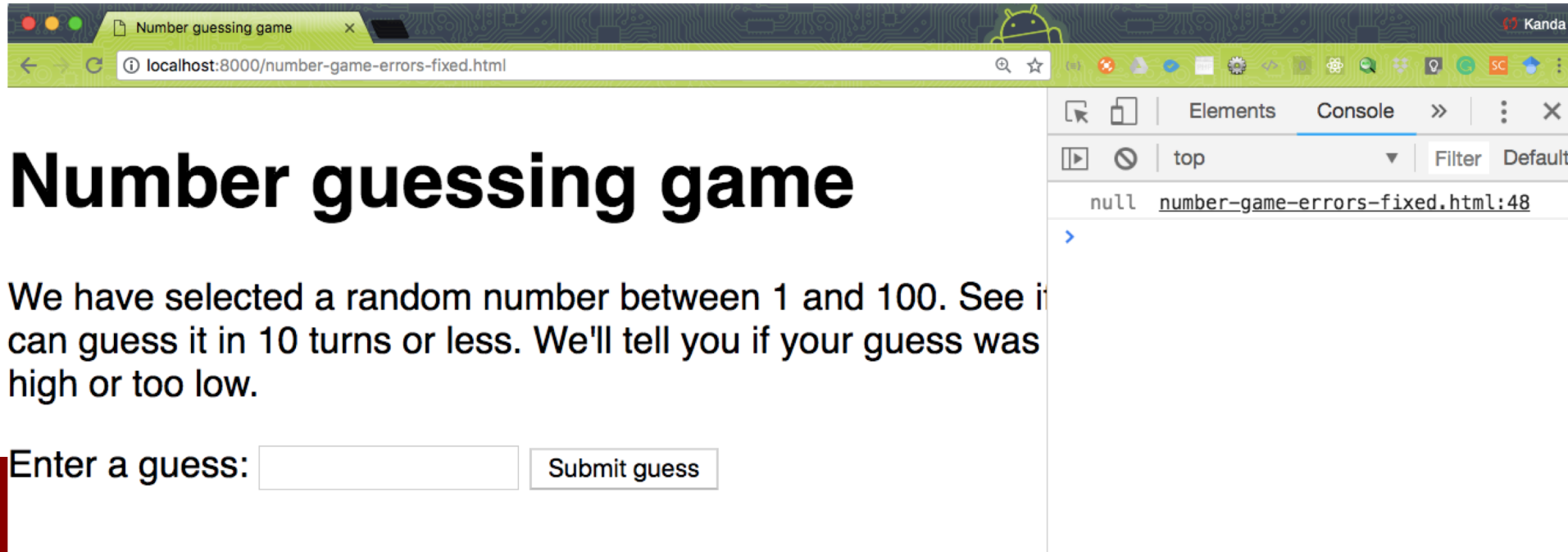
Syntax errors round two



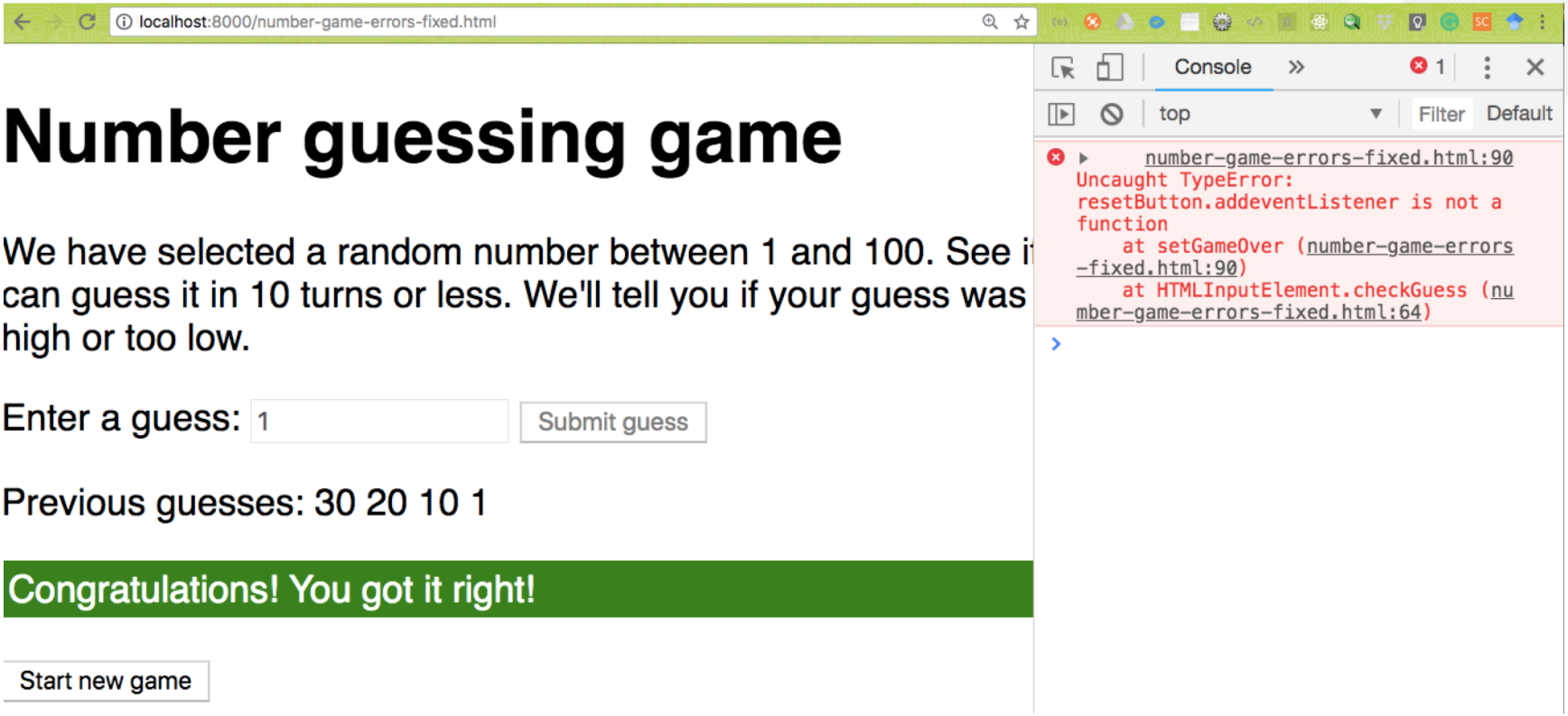
Using console.log()

```
1 | lowOrHi.textContent = 'Last guess was too high!';
```

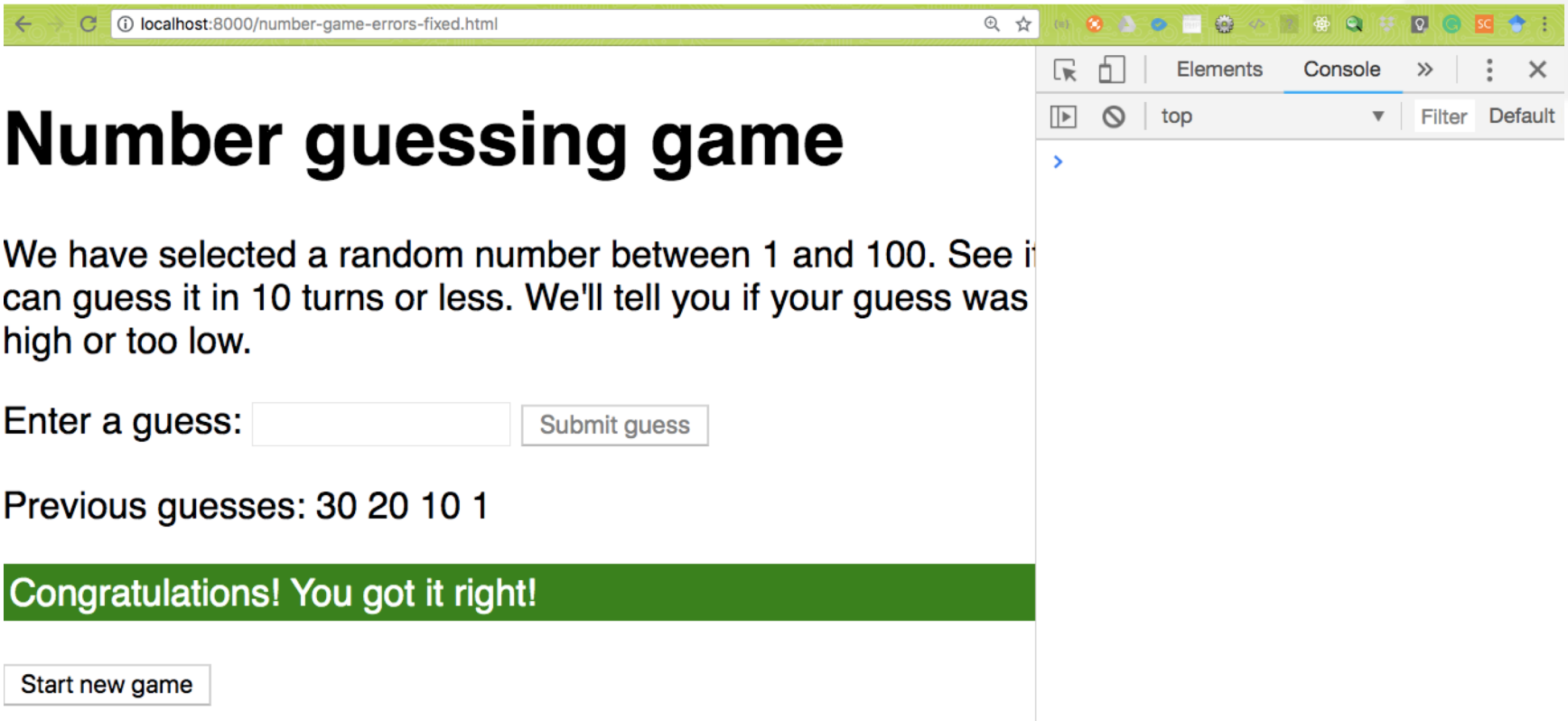
This line is trying to set the textContent property of the lowOrHi variable to a text string, but it's not working because lowOrHi does not contain what it's supposed to



Syntax errors round three



Logic error


localhost:8000/number-game-errors-fixed.html
Elements
Console
top
Filter
Default
Number guessing game
We have selected a random number between 1 and 100. See i
can guess it in 10 turns or less. We'll tell you if your guess was
high or too low.
Enter a guess:
Submit guess
Previous guesses: 30 20 10 1
Congratulations! You got it right!
Start new game

Agenda

- What is JavaScript?
- A first splash into JavaScript
- What went wrong? Troubleshooting JavaScript
- **Storing the information you need – Variables**
- Basic in JavaScript
- Handling text
- Useful string methods
- Arrays

Arrays

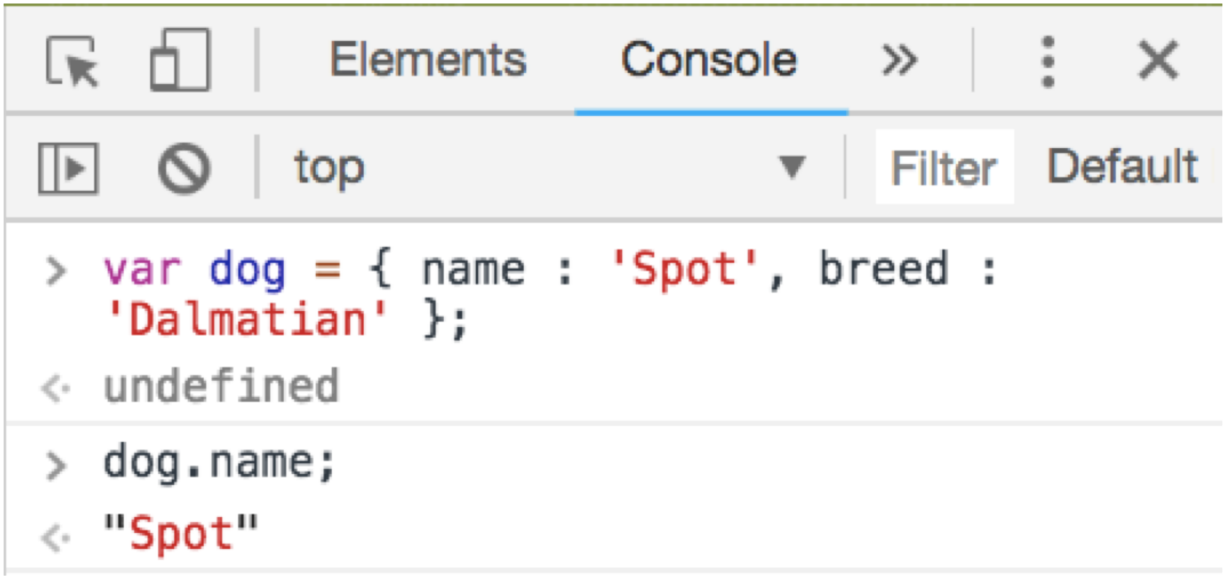
- An array is a single object that contains multiple values enclosed in square brackets and separated by commas

```
1 | var myNameArray = ['Chris', 'Bob', 'Jim'];
2 | var myNumberArray = [10,15,40];
```

```
1 | myNameArray[0]; // should return 'Chris'
2 | myNumberArray[2]; // should return 40
```

Objects

- An object is a structure of the code that models a real-life object.



Arithmetic Operators

Operator	Name	Purpose	Example
`+`	Addition	Adds two numbers together.	`6 + 9`
`-`	Subtraction	Subtracts the right number from the left.	`20 - 15`
`*`	Multiplication	Multiplies two numbers together.	`3 * 7`
`/`	Division	Divides the left number by the right.	`10 / 5`
`%`	Remainder (sometimes called modulo)	Returns the remainder left over after you've divided the left number into a number of integer portions equal to the right number.	`8 % 3` (returns 2, as three goes into 8 twice, leaving 2 left over.)

Comparison operators

Operator	Name	Purpose	Example
`===`	Strict equality	Tests whether the left and right values are identical to one another	`5 === 2 + 4`
`!==`	Strict-non-equality	Tests whether the left and right values **not** identical to one another	`5 !== 2 + 3`
`<`	Less than	Tests whether the left value is smaller than the right one.	`10 < 6`
`>`	Greater than	Tests whether the left value is greater than the right one.	`10 > 20`
`<=`	Less than or equal to	Tests whether the left value is smaller than or equal to the right one.	`3 <= 2`
`>=`	Greater than or equal to	Tests whether the left value is greater than or equal to the right one.	`5 >= 4`

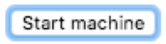
Functions and Conditionals

```html
<button>Start machine</button>
<p>The machine is stopped.</p>
```

```javascript
var btn = document.querySelector('button');
var txt = document.querySelector('p');

btn.addEventListener('click', updateBtn);

function updateBtn() {
  if (btn.textContent === 'Start machine') {
    btn.textContent = 'Stop machine';
    txt.textContent = 'The machine has started!';
  } else {
    btn.textContent = 'Start machine';
    txt.textContent = 'The machine is stopped.';
  }
}
```



The machine is stopped.

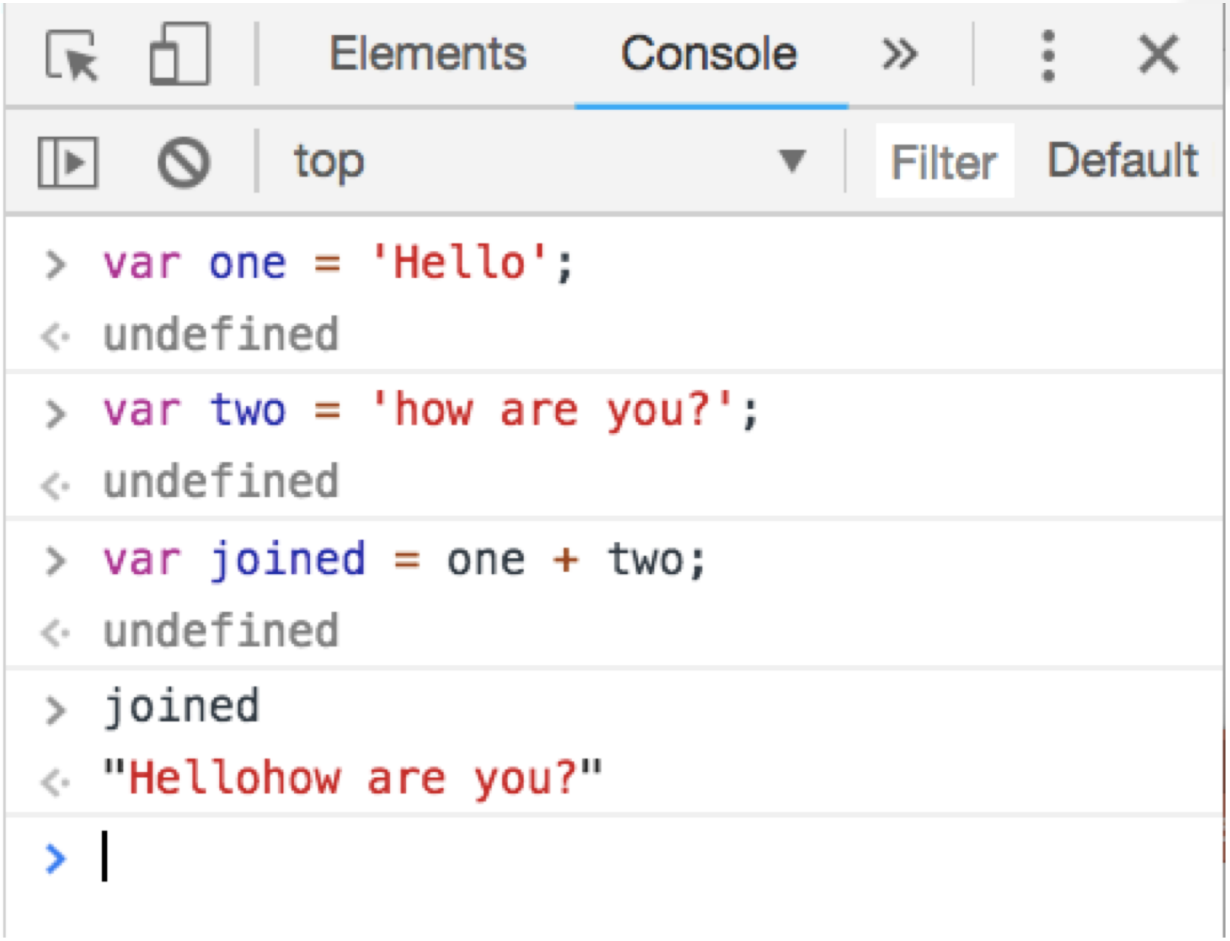
Agenda

- What is JavaScript?
- A first splash into JavaScript
- What went wrong? Troubleshooting JavaScript
- Storing the information you need – Variables
- Basic in JavaScript
- **Handling text**
- Useful string methods
- Arrays

Single quotes vs. double quotes

```
> var sglDbl = 'Would you eat a "fish
  supper"?';
  var dblSgl = "I'm feeling blue.";
  sglDbl;
<· "Would you eat a "fish supper"?"
> sglDbl
<· "Would you eat a "fish supper"?"
> dblSgl
<· "I'm feeling blue."
> var bigmouth = 'I\'ve got no right to
  take my place ...';
<· undefined
> bigmouth
<· "I've got no right to take my place
   ..."
```

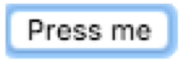
Concatenating strings



Concatenation in context

```
1 | <button>Press me</button>
```

```
1 | var button = document.querySelector('button');
2 |
3 | button.onclick = function() {
4 |   var name = prompt('What is your name?');
5 |   alert('Hello ' + name + ', nice to see you!');
6 | }
```



Numbers vs Strings

```
> var myString = '123';
<· undefined
> var myNumber = Number(myString);
<· undefined
> myNumber;
<· 123
> var myNum = 123;
<· undefined
> var myString = myNum.toString();
<· undefined
> typeof myString;
<· "string"
>
```

Useful String methods

- Finding the length of a string and finding a substring inside a string and extracting it

```
> var browserType = 'mozilla';
<· undefined
> browserType.length;
<· 7
> browserType.indexOf('zilla');
<· 2
> browserType.indexOf('vanilla');
<· -1
> browserType.slice(0,3);
<· "moz"
```

Converting between strings and arrays

```
> var myData = 'KKU, CMU, CU, PSU, TU';
< undefined
> var myArray = myData.split(',');
< undefined
> myArray
< ▶ (5) ["KKU", " CMU", " CU", " PSU", " TU"]
> myArray.length
< 5
> myArray[0];
< "KKU"
> var myNewString = myArray.join(',');
< undefined
> myNewString
< "KKU, CMU, CU, PSU, TU"
> var names = ['Red', 'Green', 'Blue'];
< undefined
> names.toString();
< "Red,Green,Blue"
```

Adding array items

- To add at the end of an array we can use push()

```
> myArray.push('STU');
<- 6
> myArray
<- ▶ (6) ["KKU", " CMU", " CU", " PSU", "
     TU", "STU"]
> myArray.push('MU', 'KMUTT');
<- 8
> myArray
<- ▶ (8) ["KKU", " CMU", " CU", " PSU", "
     TU", "STU", "MU", "KMUTT"]
```

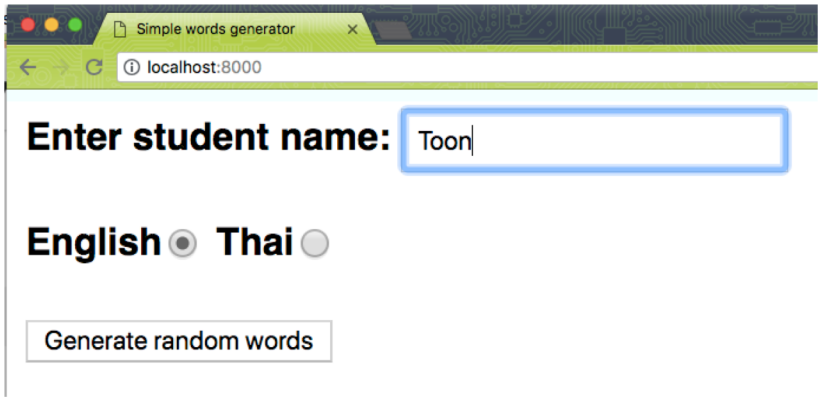
Removing an array item

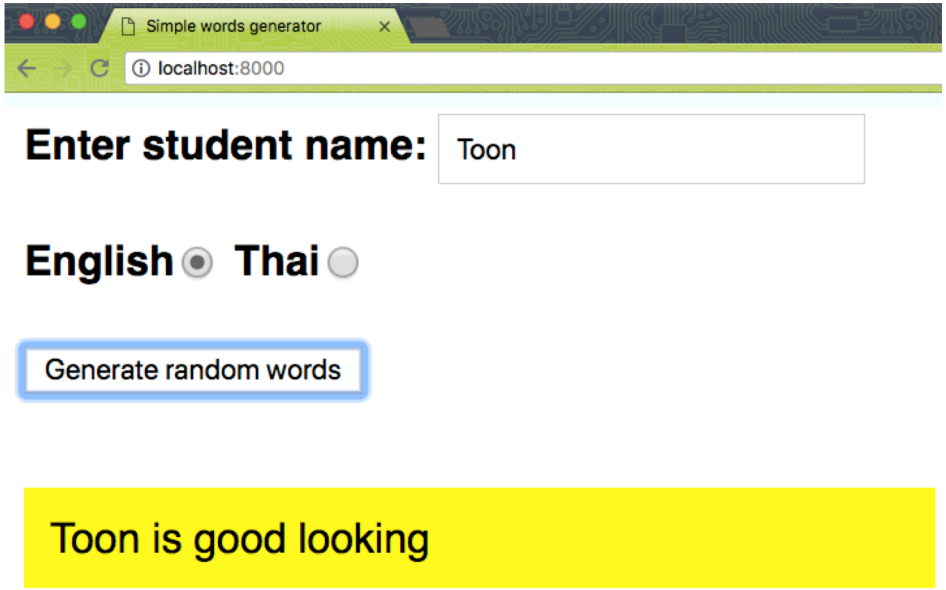
- To remove an item at the end of an array we can use pop()

```
> var removedItem = myArray.pop();
<· undefined
> myArray
<· ▶ (7) ["KKU", " CMU", " CU", " PSU", " TU", "STU", "MU"]
> removedItem
<· "KMUTT"
```

Exercise (1/2)

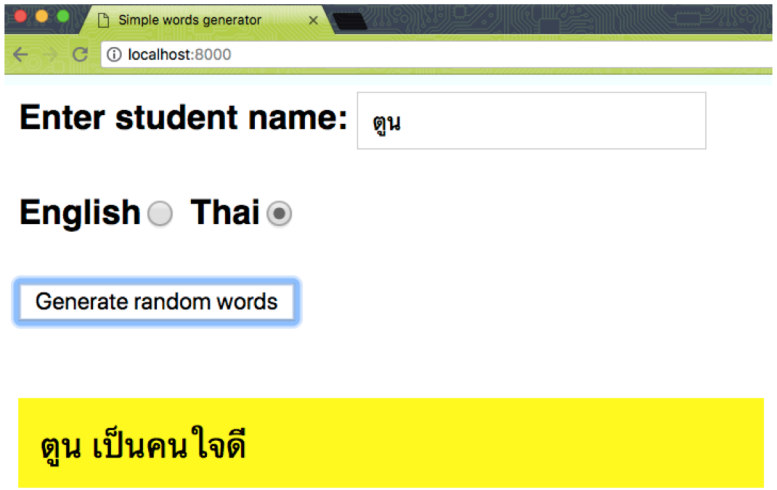
- Develop the web app that generates random words appending with the entered name





Exercise (2/2)

- Only one radio button can be selected at a time



References

- https://medium.freecodecamp.org/how-to-build-an-html-calculator-app-from-scratch-using-javascript-4454b8714b98
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/A_first_splash
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Silly_story_generator