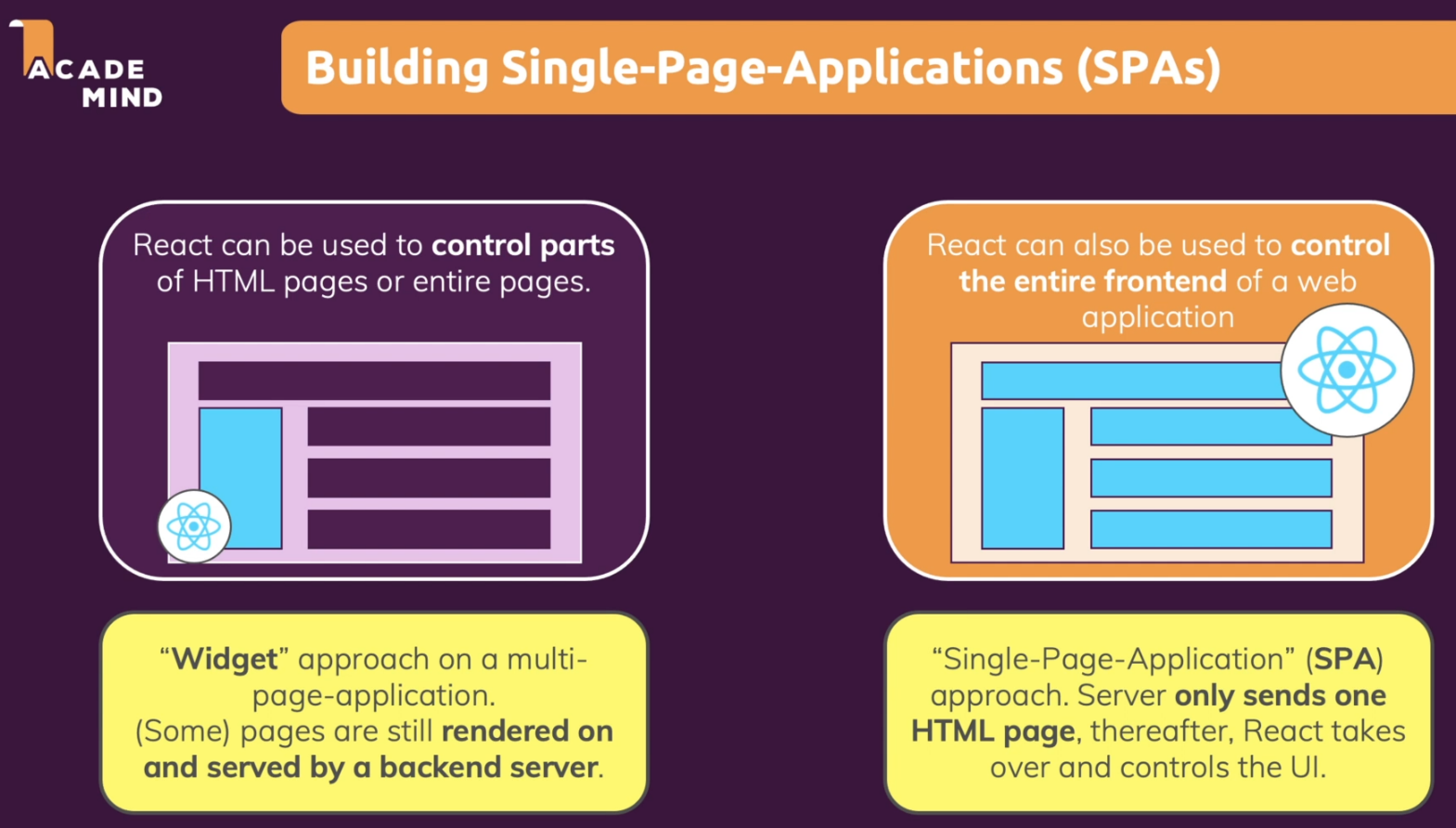
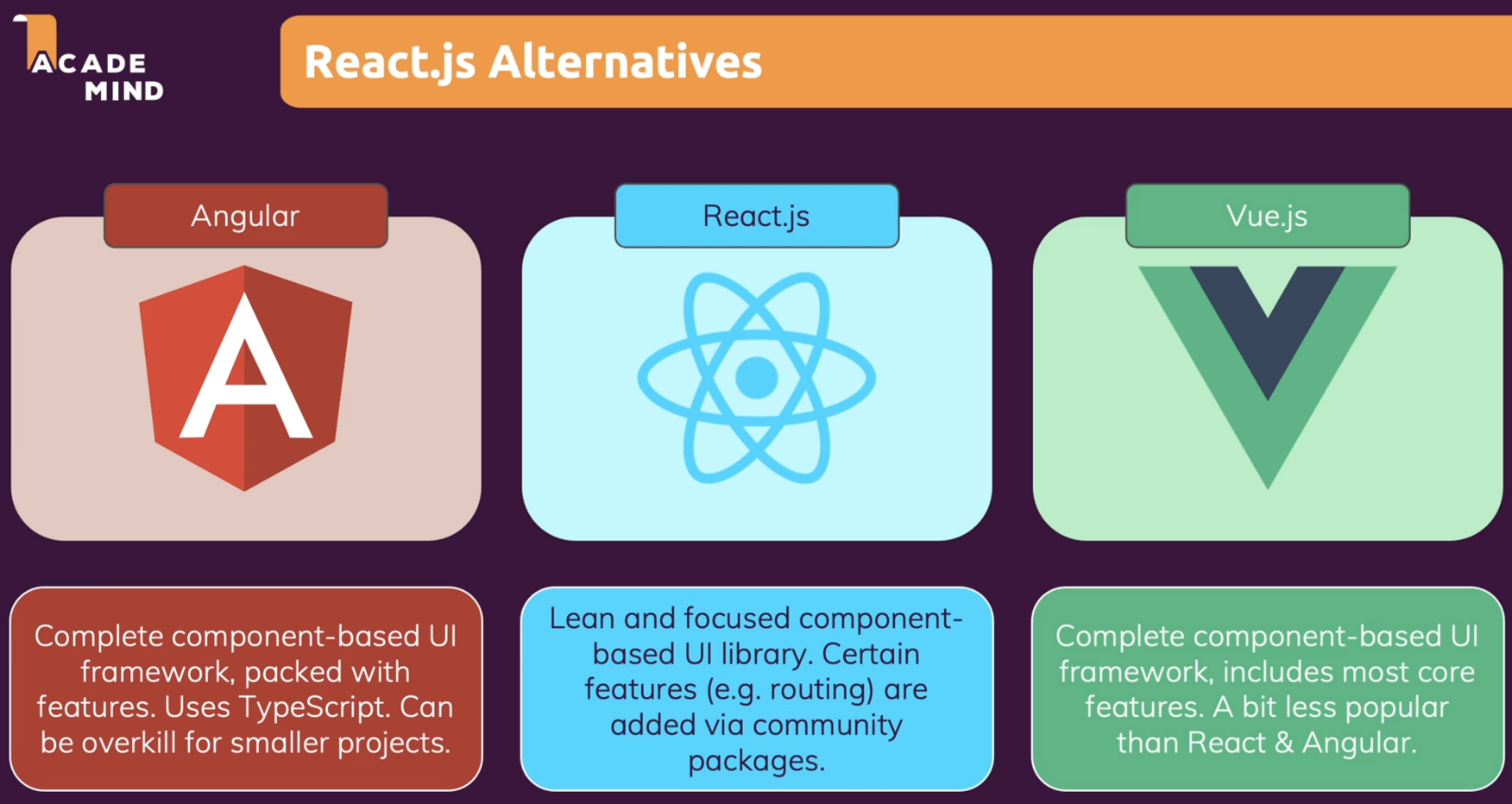
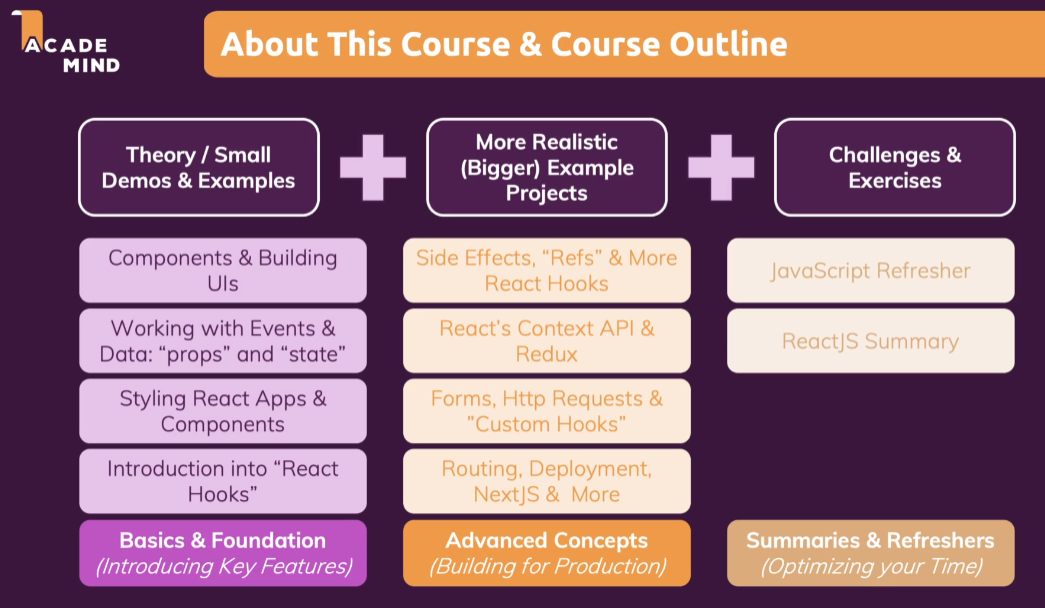
**React Notes**

React

* React is a client – side JavaScript library for building modern, reactive user interface for the Web using declarative, component – focused approach.
* JavaScript is a programming language that allows a developer to run logic in the browser.
* JavaScript runs in the browser – on the loaded page. We can manipulate the HTML Structure (DOM) of the page.
* When working with React, we often build so-called Single Page Applications.







**Imp. Notes**

**1. Understanding new keywords “let” & “const”**

* From ES6, 2 new keywords “let” & “const” are introduced that provide different ways of creating variables.
* Keyword “var” is also used to create variables but highly encouraged to use let & const.
* “let”: This keyword is used to create variable with varying values.
* “const”: This keyword is used to create variable with constant value.

**2. Arrow Functions (=>)**

* It’s a different syntax for creating JavaScript functions.

|  |  |
| --- | --- |
| const multiply = (number) => {  return number \* 2;  }  console.log(multiply (4)); // 8  const numbers = [1, 2, 3];  const doubleNumArray = numbers.map((num) => {  return num \* 2;  });  console.log(numbers);  console.log(doubleNumArray);  Output:  [1, 2, 3]  [2, 4, 6] | const printMyName = (name) => {  console.log(name); // Shivam  }  printMyName('Shivam'); |

**3. JavaScript Keywords:**

|  |  |  |
| --- | --- | --- |
| No. | Keywords | Description |
| 1. | Var, let |  |
| 2. | Const |  |
| 3. | function |  |
| 4. | class |  |
|  |  |  |
|  |  |  |

**4. Classes**

|  |  |
| --- | --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*ES6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  class Human {  constructor() {  this.gender = 'male';  }    printGender() {  console.log(this.gender);  }  }  class Person extends Human {  constructor() {  super();  this.name = 'SRV';  this.gender = 'female';  }    printMyName() {  console.log(this.name);  }  }  const person = new Person();  person.printMyName();  person.printGender(); | \*\*\*\*\*\*\*\*\*\*\*\*\*\*ES7 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  class Human {  gender = 'male';  printGender = () => {  console.log(this.gender);  }  }  class Person extends Human {  name = 'Max';  gender = 'female';    printMyName = () => {  console.log(this.name);  }    }  const person = new Person();  person.printMyName();  person.printGender(); |

**5. Spread & Rest Operators (…) 3 dots**

* Spread: Used to split up array elements or Object properties.
* Rest: Used to merge a list of function arguments into an array.

|  |  |  |
| --- | --- | --- |
| const numbers = [1, 2, 3];  const newNumbers = [...numbers, 4];  console.log(newNumbers);  **Output:**  [1,2,3,4] | const person = {  name: 'Max'  };  const newPerson = {  ...person,  age: 28  };  console.log(newPerson);  **Output:**   * [object Object] {   age: 28,   name: "Max" } | const filter = (...args) => {  return args.filter(el => el === 1);  }  console.log(filter(1, 2, 3));  **Output:**  [1] |

**6. Destructuring (Array Destructuring & Object Destructuring)**

* Destructuring allows you to pull out single element or property & store them in variables for arrays.
* Spread operator takes out all elements or properties & distribute them in a new array or object.

|  |  |
| --- | --- |
| **Array Destructuring**  const numbers = [1, 2, 3];  [num1, num2] = numbers;  console.log(num1, num2);  [num1, , num3] = numbers;  console.log(num1, num3);  Output:  1  2  1  3 | **Object Destructuring**  {name} = {name: ‘Max’, age: 28}  Console.log(name);  Console.log(age);  Output:  Max  Undefined |

**7. Reference & Primitive Types**

* For primitive types, if we reassign variable then it creates a real copy of the assigned variable.

i.e.,

|  |
| --- |
| const number = 1;  const num2 = number;  console.log(num2); // 1  Here, num2 is a real copy of number; |

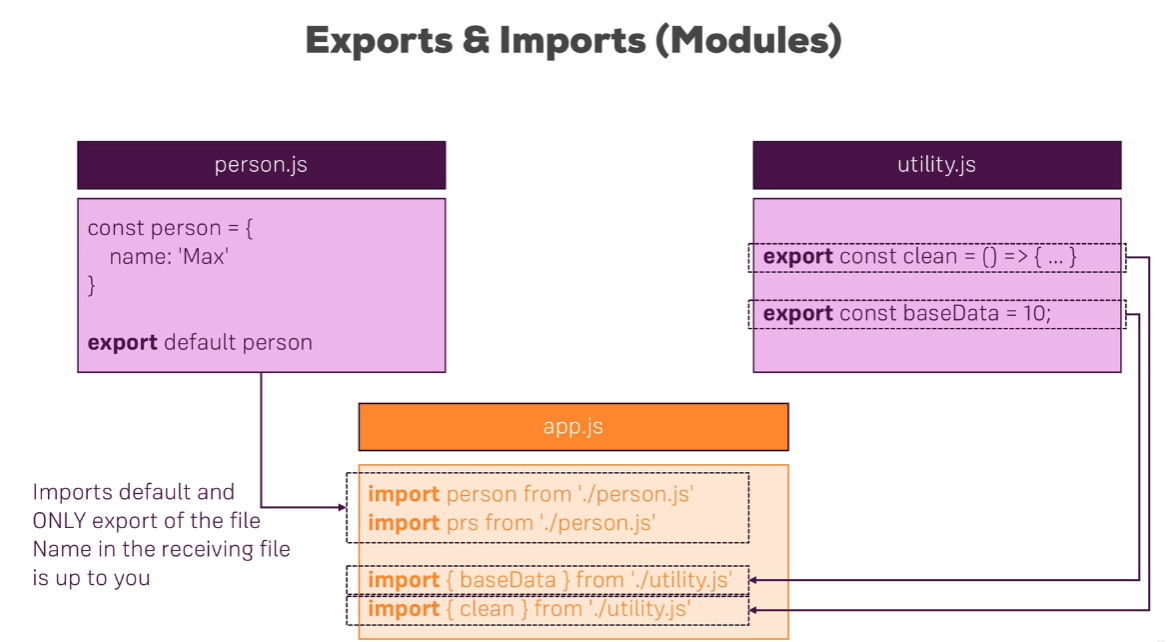
* But Objects & Arrays are reference types i.e., if we reassign object or array, only reference get changed. A new copy is not created, same object reference is reassigned.
* To create an object copy, use spread operator.

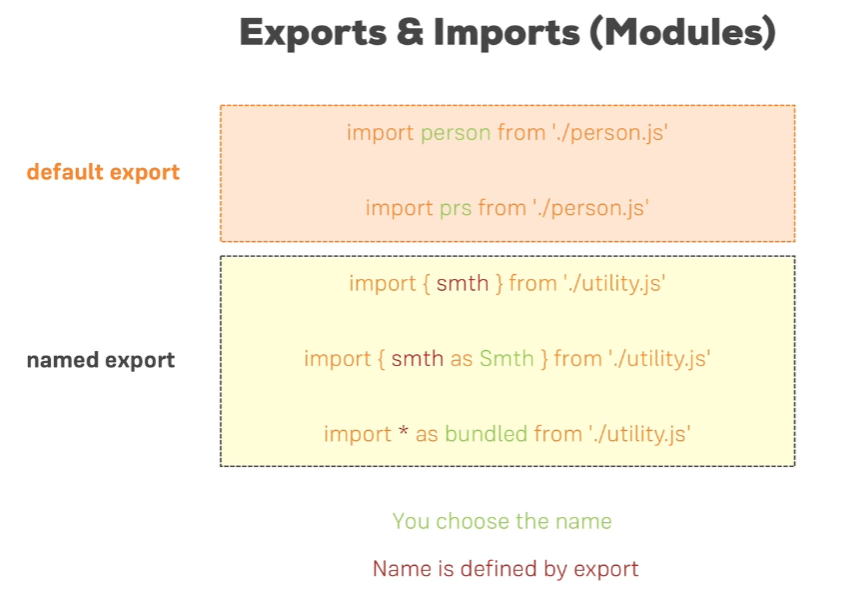
|  |
| --- |
| const person = {  name: 'Max'  };  const secondPerson = person; // Here only reference got reassigned  // Way of copying one object properties to another, Here a new object is created.  **// Using spread operator**  const thirdPerson = {  ...person  };  person.name = 'Manu';  console.log(secondPerson);  console.log(thirdPerson);  **Output:**  [object Object] {   name: "Manu" }  [object Object] {   name: "Max" } |

* **Important to keep in mind** that Objects & Arrays are reference types. If you reassign them, you’re copying the pointer not the value. Therefore, if you want to do this in a real copy way, you will have to create a new object & just copy the properties & not the entire object.

**8. Exports & Imports (Modules)**

* While doing default export, we can give name as we wish as seen in person.js
* For utility.js, the import syntax uses the curly braces to explicitly target specific things from utility.js. These are so called **Named Exports** because we import the stuff by its name.





9. Array Specific methods (ES7)

|  |  |  |
| --- | --- | --- |
| **No.** | **Methods** | **Description** |
| 1. | find () | This method will find only the first element that matches the condition & causes the callback function to fire. |
| 2. | filter () | This method allows us to iterate through an array & return all items or elements that fit the condition provided through the callback function. |
| 3. | map () | Similar to filter() method but we can do whatever we want with the content of array |
| 4. | reduce () |  |
| 5. | forEach () |  |
| 6. | some () | every (), includes (), Array.from (), Array.of (), findIndex (), fill (), values (), keys () |
|  |  |  |

|  |  |
| --- | --- |
| **Methods** |  |
| find() | let arrOfNumbers = [53, 14, 85, 66, 67, 108, 99, 10]  // Find the first even number and store it inside a variable.  let firstEvenNumber = arrOfNumbers.find((number) => number % 2 !== 1)  console.log(firstEvenNumber) // 14  // Find the first odd number and store it inside a variable.  let firstOddNumber = arrOfNumbers.find((number) => number % 2 === 1)  console.log(firstOddNumber) // 53  // Find the first number bigger than 5 and store it inside a variable.  let firstNumberBiggerThan55 = arrOfNumbers.find((number) => number > 55)  console.log(firstNumberBiggerThan55) // 85  // Find the first living person.  let firstLivingPerson = arrOfObjects.find((person) => person.living)  console.log(firstLivingPerson) // { living: true, name: 'Franklin Story Musgrave' } |